tower_defence

1.0

Generated by Doxygen 1.9.5

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

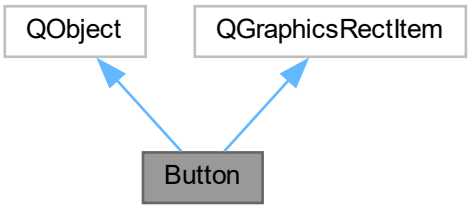
Class Documentation

4.1 Button Class Reference

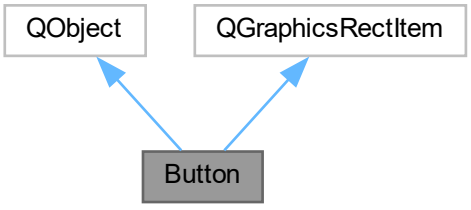
This is the button object.

`#include <button.h>`

Inheritance diagram for Button:



Collaboration diagram for Button:



Signals

- void **clicked** ()

Public Member Functions

- **Button** (QString name, int w, int h, QColor backgroundColor=Qt::green, QGraphicsItem ∗parent=NULL)

  *Construct a new **Button** (p. 7):: **Button** (p. 7) object. Creates a button.*
- void **mousePressEvent** (QGraphicsSceneMouseEvent ∗event)
- void **hoverEnterEvent** (QGraphicsSceneHoverEvent ∗event)

  Changes the button color when mouse hovers over the button.
- void **hoverLeaveEvent** (QGraphicsSceneHoverEvent ∗event)

  Changes the button color back to original color, when mouse no longer hovers over it.

4.1.1 Detailed Description

This is the button object.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Button()

```
Button::Button (
            QString name,
            int w,
            int h,
            QColor backgroundColor = Qt::green,
            QGraphicsItem * parent = NULL )
```

Construct a new **Button** (p. 7):: **Button** (p. 7) object. Creates a button.

Parameters

name	Text shown on the button.
w	Width of the button.
h	Height of the button.
backgroundColor	Color of the button.
parent	The parent of the button.

4.1.3 Member Function Documentation

4.1.3.1 hoverEnterEvent()

```
void Button::hoverEnterEvent (
            QGraphicsSceneHoverEvent * event )
```

Changes the button color when mouse hovers over the button.

4.1.3.2 hoverLeaveEvent()

```
void Button::hoverLeaveEvent (
            QGraphicsSceneHoverEvent * event )
```

Changes the button color back to original color, when mouse no longer hovers over it.

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

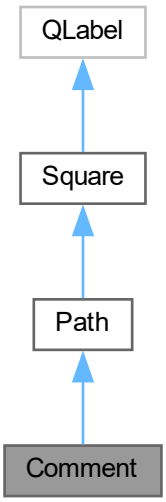
- **button.h**
- **button.cpp**

4.2 Comment Class Reference

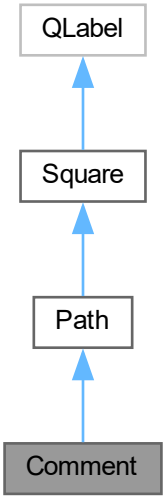
The **Comment** (p. 9) class. Represents the comment tower.

```
#include <comment.h>
```

Inheritance diagram for Comment:



Collaboration diagram for Comment:



Public Slots

- void **breakComment** ()

  Comment::breakComment (p. *10*) *Deletes the* ***Comment*** (p. *9*) *tower.*

Public Member Functions

- **Comment** (int x, int y, int duration, **Path** ∗old, QWidget ∗parent)

  Comment::Comment (p. *11*) *Constructor function for* ***Comment*** (p. *9*) *towers.*
- QPointF **towerCenter** ()

  Comment::towerCenter (p. *12*) *Gets the absolute position of the* ***Comment*** (p. *9*) *tower.*
- **Path** ∗ **getOld** ()

  Comment::getOld (p. *11*) *Returns the old path which was replaced when the* ***Comment*** (p. *9*) *tower was built.*
- ∼**Comment** ()

  Comment::∼Comment (p. *10*) *The destructor for the* ***Comment*** (p. *9*) *tower.*
- bool **isTimerActive** ()
- void **startTimer** ()

  Comment::startTimer (p. *10*) *Starts the breakdown timer of the* ***Comment*** (p. *9*) *tower.*

Additional Inherited Members

4.2.1 Detailed Description

The **Comment** (p. 9) class. Represents the comment tower.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Comment()

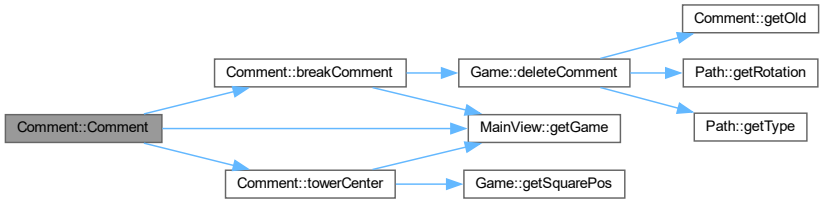
```
Comment::Comment (
            int x,
            int y,
            int duration,
             Path * old,
            QWidget * parent )
```

Comment::Comment (p. 11) Constructor function for **Comment** (p. 9) towers.

Parameters

x	The x-coordinate of the comment to be created.
y	The y-coordinate of the comment to be created.
duration	The amount of time enemies need to break down the **Comment** (p. 9) tower.
old	The path which the comment tower will be placed over.
parent	The parent widget of the **Comment** (p. 9) tower

Here is the call graph for this function:



4.2.3 Member Function Documentation

4.2.3.1 getOld()

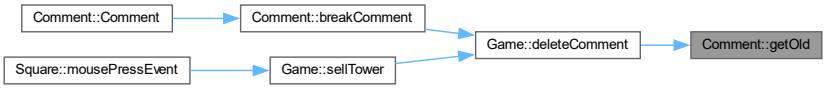
```
Path * Comment::getOld ( )
```

Comment::getOld (p. 11) Returns the old path which was replaced when the **Comment** (p. 9) tower was built.

Returns

The path the **Comment** (p. 9) tower previously replaced.

Here is the caller graph for this function:



4.2.3.2 towerCenter()

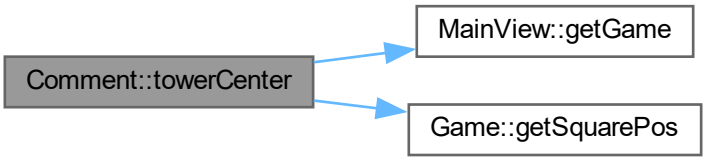
```
QPointF Comment::towerCenter ( )
```

Comment::towerCenter (p. 12) Gets the absolute position of the **Comment** (p. 9) tower.

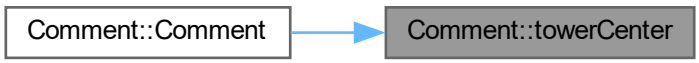
Returns

The absolute position of the **Comment** (p. 9) tower.

Here is the call graph for this function:



Here is the caller graph for this function:



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

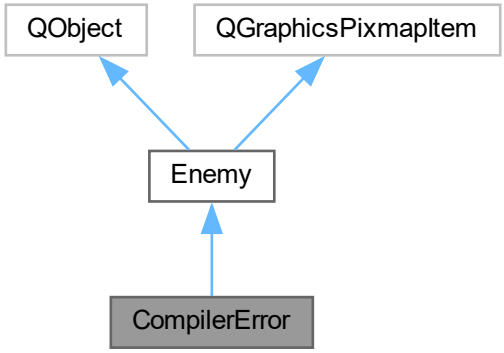
- **comment.h**
- **comment.cpp**

4.3 CompilerError Class Reference

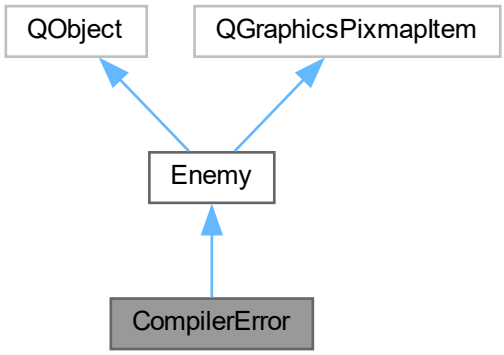
Class for **CompilerError** (p. 13) type enemies Inherits from class **Enemy** (p. 18).

`#include <compilererror.h>`

Inheritance diagram for CompilerError:



Collaboration diagram for CompilerError:



Public Member Functions

- **CompilerError** (CompilerErrorType subType, QList< QPointF > path, QList< QPoint > matrixPath)

  Construct a new Compiler Error:: Compiler Error object Checks the subType of the enemy and sets the member values accordingly.
- void **die** ()

  *After being killed calls the **explodeException()** (p. 14) function.*
- void **explodeException** ()

  Spawns the amount of SyntaxError enemies on the location of the enemy and sets different speeds to each of them.

Additional Inherited Members

4.3.1 Detailed Description

Class for **CompilerError** (p. 13) type enemies Inherits from class **Enemy** (p. 18).

4.3.2 Constructor & Destructor Documentation

4.3.2.1 CompilerError()

```
CompilerError::CompilerError (
            CompilerErrorType subType,
            QList< QPointF > path,
            QList< QPoint > matrixPath )
```

Construct a new Compiler Error:: Compiler Error object Checks the subType of the enemy and sets the member values accordingly.

Parameters

subType	
path	
matrixPath	

4.3.3 Member Function Documentation

4.3.3.1 die()

```
void CompilerError::die ( )  [virtual]
```

After being killed calls the **explodeException()** (p. 14) function.

Reimplemented from **Enemy** (p. 21).

Here is the call graph for this function:

4.3.3.2 explodeException()

```
void CompilerError::explodeException ( )
```

Spawns the amount of SyntaxError enemies on the location of the enemy and sets different speeds to each of them.

Here is the call graph for this function:

Here is the caller graph for this function:

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

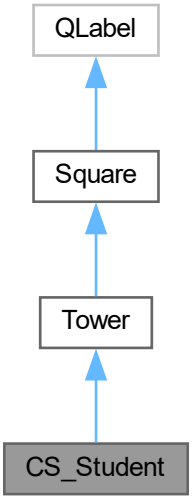
- **compilererror.h**
- **compilererror.cpp**

4.4 CS_Student Class Reference

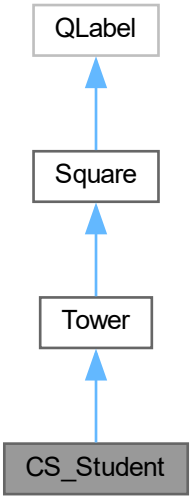
A CS Student tower type. Can attack enemies in range. Inherited from **Tower** (p. 101) class.

```
#include <cs_student.h>
```

Inheritance diagram for CS_Student:



Collaboration diagram for CS_Student:



Public Member Functions

- **CS_Student** (int row, int column, QWidget ∗parent=nullptr)

  CS_Student *(*p. *15) constructor, create a CS Student tower at the given posion on the grid map.*
- virtual bool **upgrade** ()

  Upgrade this tower if possible.

Additional Inherited Members

4.4.1 Detailed Description

A CS Student tower type. Can attack enemies in range. Inherited from **Tower** (p. 101) class.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 CS_Student()

```
CS_Student::CS_Student (
            int row,
            int column,
            QWidget * parent = nullptr )
```

CS_Student (p. 15) constructor, create a CS Student tower at the given posion on the grid map.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

4.4.3 Member Function Documentation

4.4.3.1 upgrade()

```
bool CS_Student::upgrade ( )  [virtual]
```

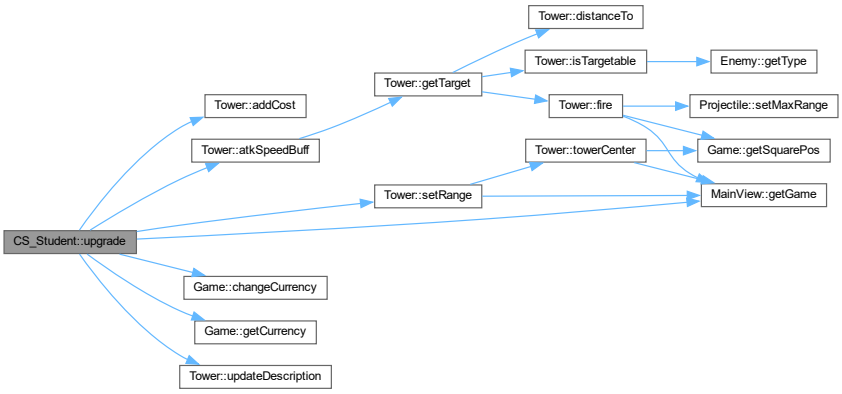
Upgrade this tower if possible.

Returns

true if the upgrade was successful, false otherwise.

Implements **Tower** (p. 101).

Here is the call graph for this function:



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

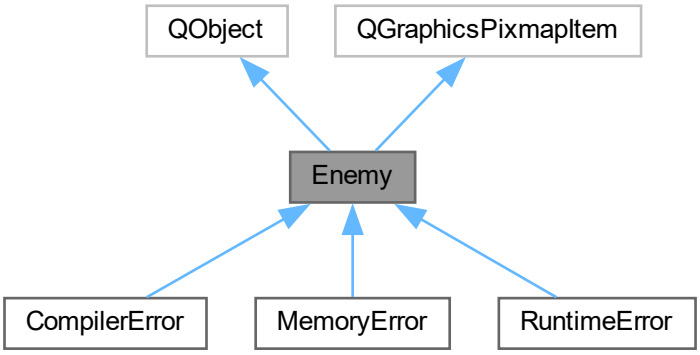
- **cs_student.h**
- cs_student.cpp

4.5 Enemy Class Reference

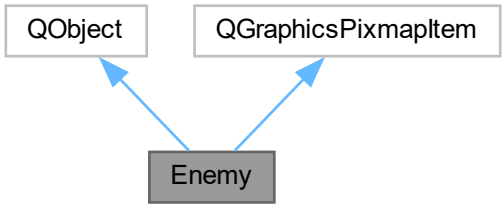
Virtual class of the enemies. All enemies in the game are inherited from this class.

`#include <enemy.h>`

Inheritance diagram for Enemy:



Collaboration diagram for Enemy:



Public Slots

- void **move** ()

  *Moves the enemy along on the path by STEP_SIZE. Emits a signal dealsDamage(int) when reaches the final destination which is handles in **game.cpp** (p. 135).*

Signals

- void **enemyDies** (int)
- void **dealsDamage** (int)
- void **addedEnemy** (**Enemy** ∗)

Public Member Functions

- **Enemy** (EnemyType type, QList< QPointF > path, QList< QPoint > matrixPath, int health=0, int damage=0, int speed=0, QGraphicsItem ∗parent=0)

  *Construct a new **Enemy** (p. 18):: **Enemy** (p. 18) object, the necessary members are initialized.*
- virtual void **attack** ()
- virtual void **die** ()

  Is called when the enemy dies. Emits a signal to notify that the enemy has died and calls deleteLater().
- virtual void **takeDamage** (int damage)

  *Is called when the enemy takes damage from towers. Either decreases the enemy's health or calls **die()** (p. 21).*
- void **setSpeed** (int speed)

  Sets the speed of the enemy.
- void **startMove** ()

  *Starts the timer that calls **move()** (p. 23) after the specified interval.*
- void **setPath** (QList< QPoint > matrixPath, QList< QPointF > path)

  Sets the path of the enemy.
- QPoint **getMatrixLocation** () const

  Returns the grid location of the enemy.
- QTimer ∗ **getTimer** ()

  Returns the timer of the enemy.
- QPoint **getNextLocation** () const

  Returns the next location of the enemy.
- EnemyType **getType** () const

  Returns the type of the enemy.

Protected Attributes

- int **health_**

  The health of the enemy.
- int **damage_**

  The damage what the enemy will deal when it reaches to the end of the path.
- int **speed_**

  The speed at which the enemies move along the path. Should be set between the values 0-70.
- int **pointValue_**

  The value of the enemy that is turned into points and currency once the enemy is killed.
- QList< QPointF > **path_**

  The QPointF(pixels) list of the path where the enemy moves along.
- QList< QPoint > **matrixPath_**

  The QPoint list of the path. (Grid locations)
- QPointF **dest_**

  The next destination of the enemy.
- int **point_index_**

  An index to keep track of the enemy movement along the path.
- EnemyType **type_**

  The type of the enemy specified by the enum EnemyType.
- QTimer * **timer_**

  The timer which is used to move the enemy.

4.5.1 Detailed Description

Virtual class of the enemies. All enemies in the game are inherited from this class.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Enemy()

```
Enemy::Enemy (
            EnemyType type,
            QList< QPointF > path,
            QList< QPoint > matrixPath,
            int health = 0,
            int damage = 0,
            int speed = 0,
            QGraphicsItem * parent = 0 )
```

Construct a new **Enemy** (p. 18):: **Enemy** (p. 18) object, the necessary members are initialized.

Parameters

type	
path	
matrixPath	
health	
damage	
speed	
parent	

4.5.3 Member Function Documentation

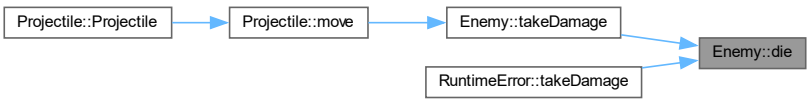
4.5.3.1 die()

```
void Enemy::die ( )  [virtual]
```

Is called when the enemy dies. Emits a signal to notify that the enemy has died and calls deleteLater().

Reimplemented in **CompilerError** (p. 14).

Here is the caller graph for this function:



4.5.3.2 getMatrixLocation()

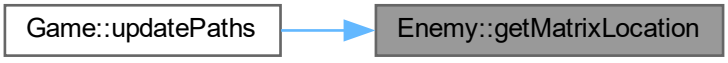
```
QPoint Enemy::getMatrixLocation ( ) const
```

Returns the grid location of the enemy.

Returns

QPoint

Here is the caller graph for this function:



4.5.3.3 getNextLocation()

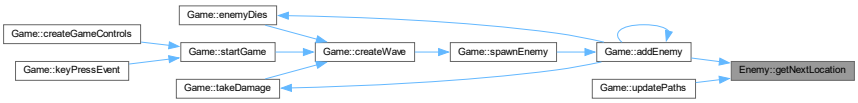
```
QPoint Enemy::getNextLocation ( ) const
```

Returns the next location of the enemy.

Returns

QPoint

Here is the caller graph for this function:



4.5.3.4 getTimer()

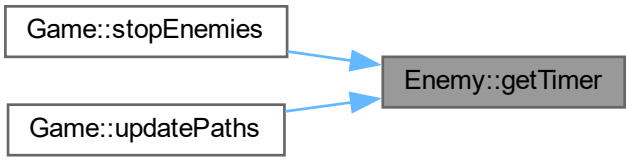
```
QTimer * Enemy::getTimer ( )
```

Returns the timer of the enemy.

Returns

QTimer*

Here is the caller graph for this function:



4.5.3.5 getType()

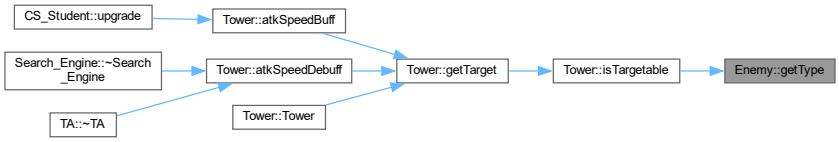
```
EnemyType Enemy::getType ( ) const
```

Returns the type of the enemy.

Returns

EnemyType

Here is the caller graph for this function:



4.5.3.6 move

```
void Enemy::move ( )  [slot]
```

Moves the enemy along on the path by STEP_SIZE. Emits a signal dealsDamage(int) when reaches the final destination which is handles in **game.cpp** (p. 135).

Here is the caller graph for this function:

4.5.3.7 setPath()

```
void Enemy::setPath (
            QList< QPoint > matrixPath,
            QList< QPointF > path )
```

Sets the path of the enemy.

Parameters

matrixPath	The grid representation of the path.
path	The QPointF(Scene pixels) representation of the path.

Here is the caller graph for this function:

4.5.3.8 setSpeed()

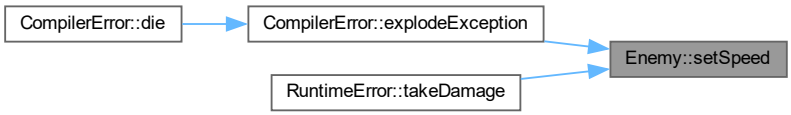
```
void Enemy::setSpeed (
            int speed )
```

Sets the speed of the enemy.

Parameters

speed	The new speed of the enemy.

Here is the caller graph for this function:



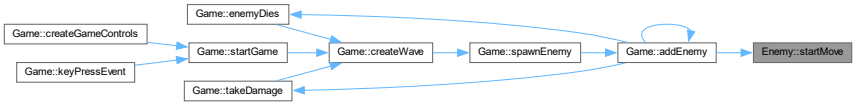
4.5.3.9 startMove()

```
void Enemy::startMove ( )
```

Starts the timer that calls **move()** (p. 23) after the specified interval.

Here is the call graph for this function:

Here is the caller graph for this function:



4.5.3.10 takeDamage()

```
void Enemy::takeDamage (
            int damage )  [virtual]
```

Is called when the enemy takes damage from towers. Either decreases the enemy's health or calls **die()** (p. 21).

Parameters

damage	The damage of the projectile, fired by a tower.

Reimplemented in **RuntimeError** (p. 88).

Here is the call graph for this function:

Here is the caller graph for this function:

4.5.4 Member Data Documentation

4.5.4.1 damage_

```
int Enemy::damage_  [protected]
```

The damage what the enemy will deal when it reaches to the end of the path.

4.5.4.2 dest_

```
QPointF Enemy::dest_  [protected]
```

The next destination of the enemy.

4.5.4.3 health_

```
int Enemy::health_  [protected]
```

The health of the enemy.

4.5.4.4 matrixPath_

```
QList<QPoint> Enemy::matrixPath_  [protected]
```

The QPoint list of the path. (Grid locations)

4.5.4.5 path_

```
QList<QPointF> Enemy::path_  [protected]
```

The QPointF(pixels) list of the path where the enemy moves along.

4.5.4.6 point_index_

```
int Enemy::point_index_  [protected]
```

An index to keep track of the enemy movement along the path.

4.5.4.7 pointValue_

```
int Enemy::pointValue_  [protected]
```

The value of the enemy that is turned into points and currency once the enemy is killed.

4.5.4.8 speed_

```
int Enemy::speed_  [protected]
```

The speed at which the enemies move along the path. Should be set between the values 0-70.

4.5.4.9 timer_

```
QTimer* Enemy::timer_  [protected]
```

The timer which is used to move the enemy.

4.5.4.10 type_

```
EnemyType Enemy::type_  [protected]
```

The type of the enemy specified by the enum EnemyType.

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

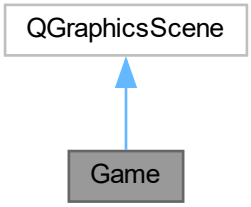
- **enemy.h**
- **enemy.cpp**

4.6 Game Class Reference

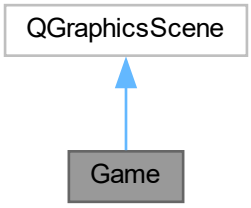
The class holding everything related to the actual tower defence gameplay.

`#include <game.h>`

Inheritance diagram for Game:



Collaboration diagram for Game:



Public Slots

- void **showMenu** ()

  Exits the game into the main menu.
- void **enterUpgradeMode** ()

  Set the game to upgrade mode for upgrading tower.
- void **enterBuildCS** ()

  Set the game to build mode, with the tower type CS Student.
- void **enterBuildTA** ()

  *Set the game to build mode, with the tower type **TA** (p. 97).*
- void **enterBuildSE** ()

  Set the game to build mode, with the tower type Search Engine.
- void **enterBuildLS** ()

  Set the game to build mode, with the tower type Language Server.

- void **enterBuildVal** ()

  *Set the game to build mode, with the tower type **Valgrind** (p. 118).*
- void **enterSellMode** ()

  Set the game to sell mode.
- void **enemyDies** (int value)

  Is called when an enemy dies. Updates the activeEnemies_ list and performs level and game ending checks.
- void **spawnEnemy** (int type)

  *Is called to spawn an enemy of the specified type. (Enum is located in **enemy.h** (p. 132)) Calls the slot addEnemy which will create the enemy and adds it to the scene.*
- void **updateLeaderboard** ()

  Is called when the game is lost or won. Reads the data from the leaderboard file (or creates it), compares the current score to the leaderboard and asks the name of the player if it is in the top10. Updates the leaderboard accordingly. Uses QInputDialog for communicating with the player.
- void **showError** (QString message)

  Displays an error message in a QMessageBox.
- void **addEnemy** (**Enemy** ∗)

  Adds an enemy to the game. Connects all signals and slots relevant to the enemy.
- void **takeDamage** (int dHealth)

  Is called when the player takes damage from an enemy.
- void **enterBuildCom** ()
- void **updatePaths** ()

  *Updates the paths of all the enemies. **Path** (p. 79) is calculated using breath-first search and returned by **getShortestPath(QPoint)** (p. 44). Is connected to the wallAction() signal.*
- void **updateEnemyCount** ()

  Updates the currently active enemy count.
- void **stopEnemies** ()

  Stops the timers of all the enemies in the game.
- void **createWave** ()

  Creates a wave of enemies for one level according to the description in wave.txt. The intervals between enemies and types of enemies is implemented using QTimer timers.
- void **startGame** ()

  Starts the game.

Signals

- void **gameWon** ()
- void **gameLost** ()
- void **waveWon** ()
- void **error** (QString)
- void **wallAction** ()

Public Member Functions

- **Game** (QObject ∗parent, int gamemode=0)
- QPointF **getSquarePos** (int row, int column)

  Returns the scene location of a square in the grid.
- QWidget ∗ **getWidgetAt** (int row, int column)

  Get a widget at the given position on the game's grid map.
- bool **buildTower** (int row, int column)
- bool **buildTower** (int row, int column, TowerTypes::TYPES type)

  Build a tower of a certain type at the given coordinates on the grid map.

- bool **sellTower** (int row, int column)

  Sell the tower at the given position on the grid map, which destroy it and give back some money to the player.
- bool **upgradeTower** (int row, int column)

  Upgrade the tower at the given coordinates on the grid map.
- bool **isLost** () const

  Performs a check if the game is lost.
- bool **isWon** () const

  Performs a check if the game is won.
- bool **isWaveWon** ()

  Performs a check if the level is won.
- void **createMap** ()

  Reads the map file and uses the information to construct the map's tile and path objects at the appropriate locations as denoted by the file data.
- void **createGameControls** ()

  Creates the game controls and hooks them up to their actions. Places all the controls into the gameLayout.
- void **readWaveFile** ()

  Reads the wave.txt file which specifies the levels of the games. README.txt can be found in /src/files. Adds waves to the QList<QStringList> waves_ member which will be used for waves' creation and the finalLevel_ member.
- void **playHitsound** ()

  Plays a random keyboard noise (hitsound) from the four available hitsounds.
- void **playClicksound** ()

  Plays a mouseclick sound.
- QList< QPointF > **convertCoordinates** (QList< QPoint > path)

  Converts grid matrix coordinates to scene coordinates for the enemy path.
- QList< QPoint > **getShortestPath** (QPoint start)

  Gets the shortest list of path coordinates which the enemies can take to get from an arbitrary path coordinate to the ending point. Uses the following priority:
- QList< QPoint > **BFS** (QPoint start, bool blocked)

  Breadth-first search algorithm, assuming one can only move either vertically or horizontally.
- void **breakComment** (int row, int column)

  The enemy's method of destroying a comment tower. Breaks the comment tower after its duration value passes.
- void **deleteComment** (int row, int column)

  Deletes a comment tower at a given location.
- int **getGamemode** () const

  Get the Gamemode: 0 = sandbox, 1 = easy, 2 = hard.
- int **getHealth** () const

  Returns the health of the player.
- int **getScore** () const

  Returns the current score.
- int **getLevel** () const

  Returns the currently active level.
- int **getCurrency** () const

  Returns the amount of currency the player has.
- int **getEnemyCount** () const

  Returns the amount of currently active enemies.
- Modes::MODES **getMode** () const

  Get the mode.
- TowerTypes::TYPES **getBuildType** () const

  Get the tower type.
- void **changeScore** (int points)

  Changes the score.

- void **changeCurrency** (int dCurrency)

  Changes the currency amount.
- void **setMode** (Modes::MODES m)

  Set the game's mode to the given mode.
- void **advanceLevel** ()

  Advances the level by 1.
- void **addSpawnedEnemies** (int)

  Adds amount of enemies that have been spawned during this level up to this point.
- void **keyPressEvent** (QKeyEvent ∗keyEvent)

  Initiates the correct functions related to the key pressed.
- void **resetButtonHighlights** ()

  Resets the button highlighting back to default.
- void **hideAllAttackAreasExcept** (QPointF exclude=QPointF(-1, -1))

  Hides attack areas of all the towers except the tower on the specified coordinate.
- bool **isTower** (int row, int column)

  *Check if the item at the given position in the grid map is a **Tower*** (p. *101) or not.*
- bool **isPath** (int row, int column)

  Checks if the square at the given location is a path tile.
- bool **isPathEnd** (int row, int column)

  Checks if the square at the given location is an end of the enemy's path.
- bool **isComment** (int row, int column)

  Checks if the square at the given location contains a comment tower.
- bool **isEnemy** (int row, int column)

  Checks if the location is populated by enemies.

Public Attributes

- QGraphicsGridLayout ∗ **mapLayout**
- QGraphicsLinearLayout ∗ **gameLayout**
- QGraphicsGridLayout ∗ **controlsLayout**

4.6.1 Detailed Description

The class holding everything related to the actual tower defence gameplay.

4.6.2 Member Function Documentation

4.6.2.1 addEnemy

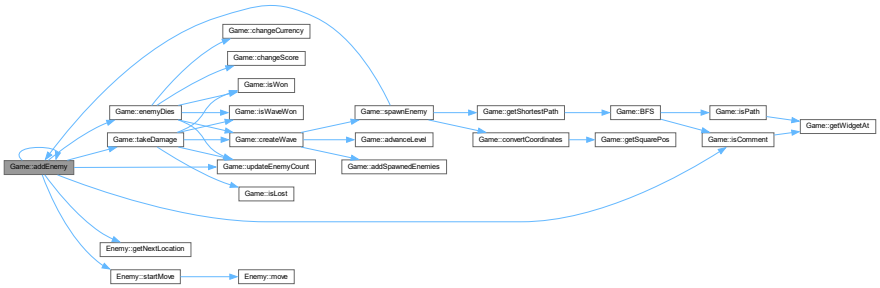
```
void Game::addEnemy (
            Enemy * enemy )  [slot]
```

Adds an enemy to the game. Connects all signals and slots relevant to the enemy.

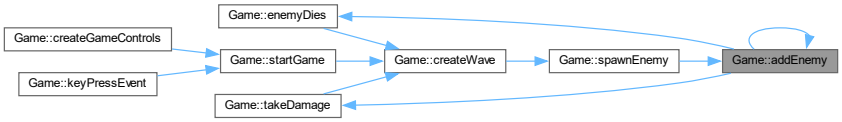
Parameters

enemy	The enemy to be added.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.2 addSpawnedEnemies()

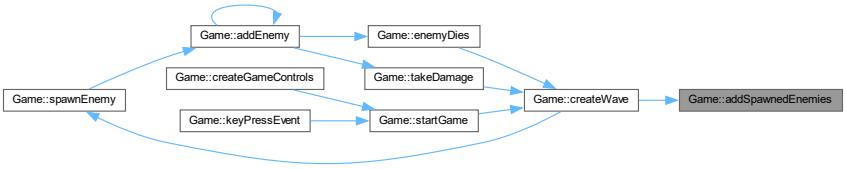
```
void Game::addSpawnedEnemies (
            int amount )
```

Adds amount of enemies that have been spawned during this level up to this point.

Parameters

amount	The amount of enemies added.

Here is the caller graph for this function:

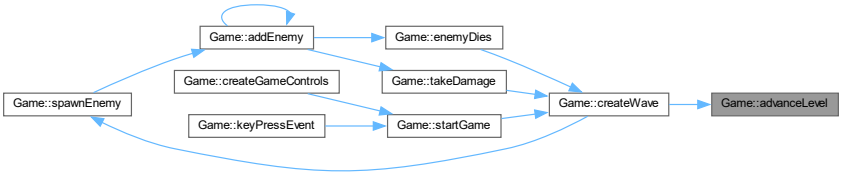


4.6.2.3 advanceLevel()

```
void Game::advanceLevel ( )
```

Advances the level by 1.

Here is the caller graph for this function:



4.6.2.4 BFS()

```
QList< QPoint > Game::BFS (
            QPoint start,
            bool blocked )
```

Breadth-first search algorithm, assuming one can only move either vertically or horizontally.

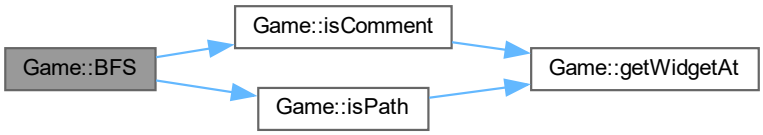
Parameters

start	The path coordinate to start from.
blocked	Whether to search in obstructed or unobstructed mode. That is: • Obstructed mode: All paths to the end are assumed to be obstructed, so search for the shortest route while assuming there are no obstructions. • Unobstructed mode: It is assumed that there is at least one path to the ending point. Find the shortest route.

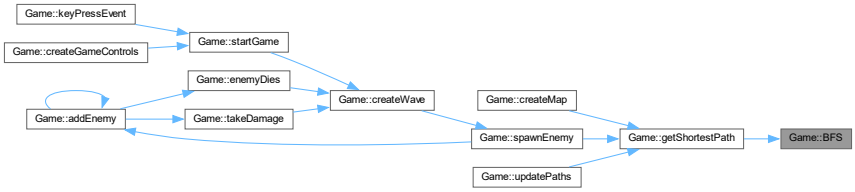
Returns

A list of path coordinates, representing the shortest path enemies can take to get to the ending point.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.5 breakComment()

```
void Game::breakComment (
            int row,
            int column )
```

The enemy's method of destroying a comment tower. Breaks the comment tower after its duration value passes.

Parameters

row	The row of the comment tower.
column	The column of the comment tower.

Here is the call graph for this function:

4.6.2.6 buildTower()

```
bool Game::buildTower (
            int row,
            int column,
            TowerTypes::TYPES type )
```

Build a tower of a certain type at the given coordinates on the grid map.

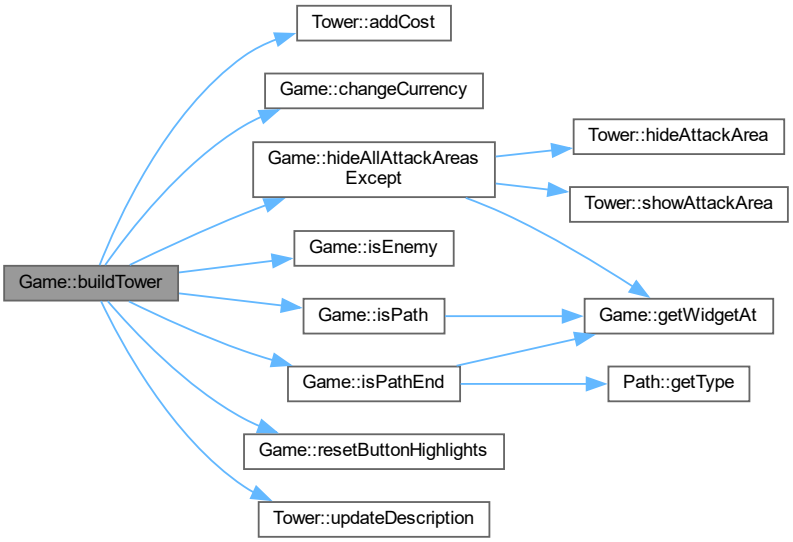
Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
type	an enum that represent certain tower types

Returns

true if the built was successful, false otherwise

Here is the call graph for this function:



4.6.2.7 changeCurrency()

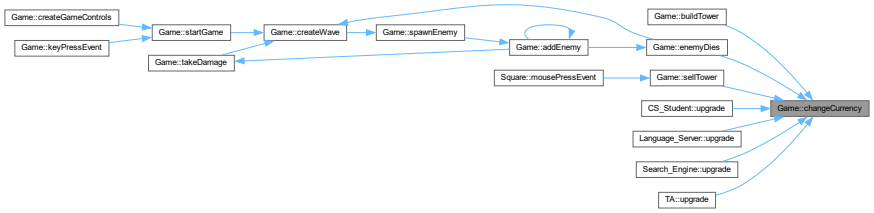
```
void Game::changeCurrency (
            int dCurrency )
```

Changes the currency amount.

Parameters

dCurrency	The amount to be added to currency.

Here is the caller graph for this function:



4.6.2.8 changeScore()

```
void Game::changeScore (
            int dPoints )
```

Changes the score.

Parameters

dPoints	The amount of points to be added.

Here is the caller graph for this function:

4.6.2.9 convertCoordinates()

```
QList< QPointF > Game::convertCoordinates (
            QList< QPoint > path )
```

Converts grid matrix coordinates to scene coordinates for the enemy path.

Parameters

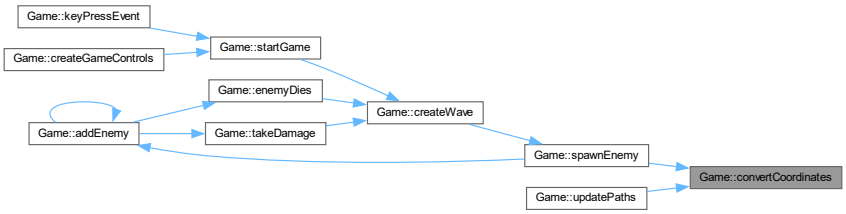
path	The grid representation of the enemy path.

Returns

QList<QPointF>

Here is the call graph for this function:

Here is the caller graph for this function:

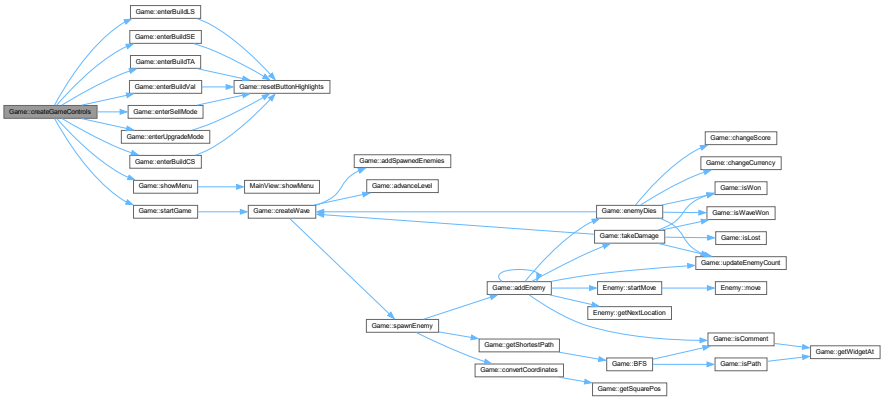


4.6.2.10 createGameControls()

```
void Game::createGameControls ( )
```

Creates the game controls and hooks them up to their actions. Places all the controls into the gameLayout.

Here is the call graph for this function:

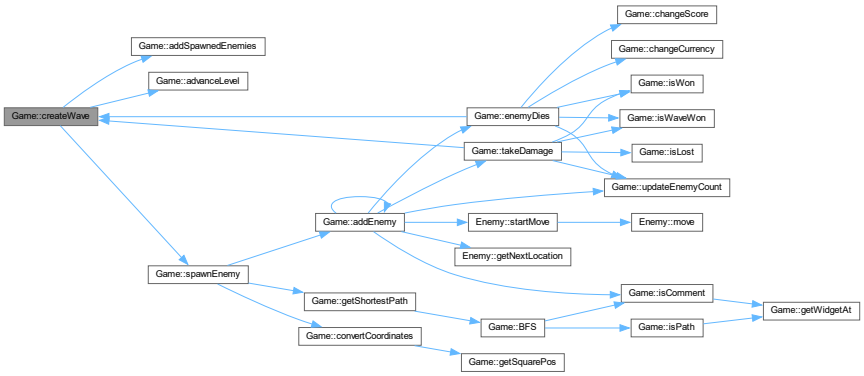


4.6.2.11 createWave

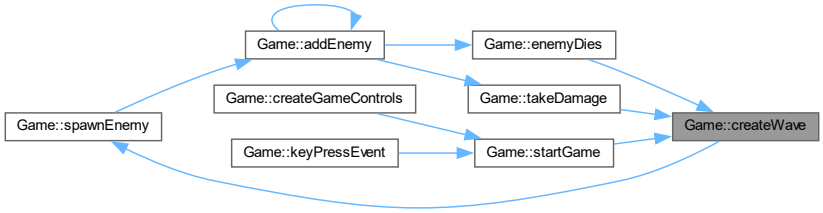
```
void Game::createWave ( )  [slot]
```

Creates a wave of enemies for one level according to the description in wave.txt. The intervals between enemies and types of enemies is implemented using QTimer timers.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.12 deleteComment()

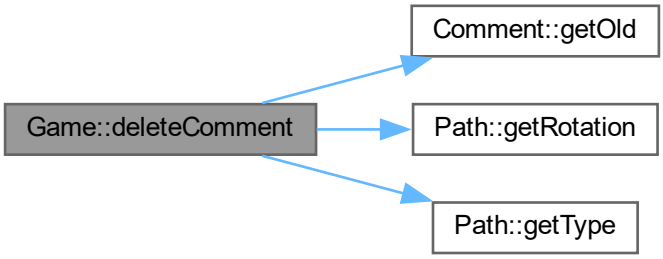
```
void Game::deleteComment (
            int row,
            int column )
```

Deletes a comment tower at a given location.

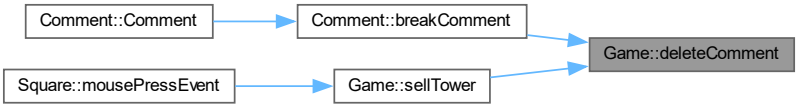
Parameters

row	The row of the comment tower.
column	The column of the comment tower.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.13 enemyDies

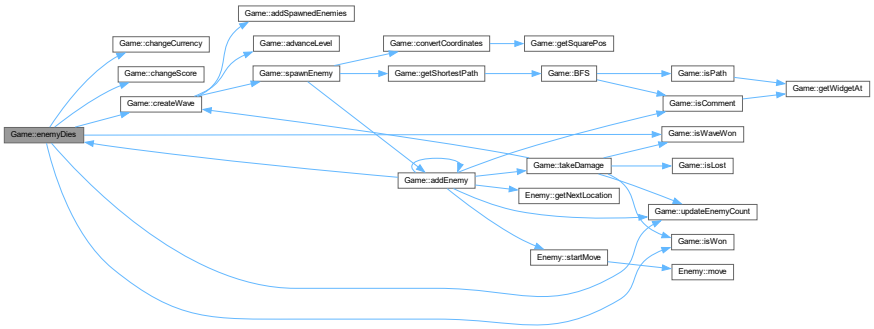
```
void Game::enemyDies (
            int value )  [slot]
```

Is called when an enemy dies. Updates the activeEnemies_ list and performs level and game ending checks.

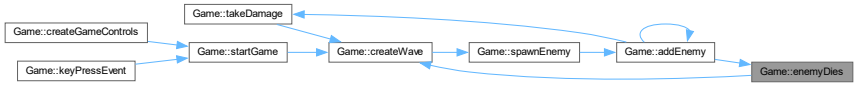
Parameters

value	The value that will be added the score and currency.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.14 getBuildType()

```
TowerTypes::TYPES Game::getBuildType ( ) const
```

Get the tower type.

Returns

TowerTypes::TYPES The type of tower selected

Here is the caller graph for this function:

4.6.2.15 getCurrency()

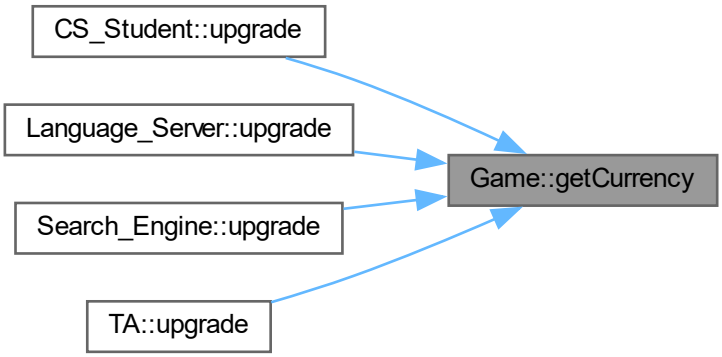
```
int Game::getCurrency ( ) const
```

Returns the amount of currency the player has.

Returns

int

Here is the caller graph for this function:



4.6.2.16 getEnemyCount()

```
int Game::getEnemyCount ( ) const
```

Returns the amount of currently active enemies.

Returns

int

4.6.2.17 getGamemode()

```
int Game::getGamemode ( ) const  [inline]
```

Get the Gamemode: 0 = sandbox, 1 = easy, 2 = hard.

Returns

int Corresponds to the gamemode of the game instance.

4.6.2.18 getHealth()

```
int Game::getHealth ( ) const
```

Returns the health of the player.

Returns

int

4.6.2.19 getLevel()

```
int Game::getLevel ( ) const
```

Returns the currently active level.

Returns

int

4.6.2.20 getMode()

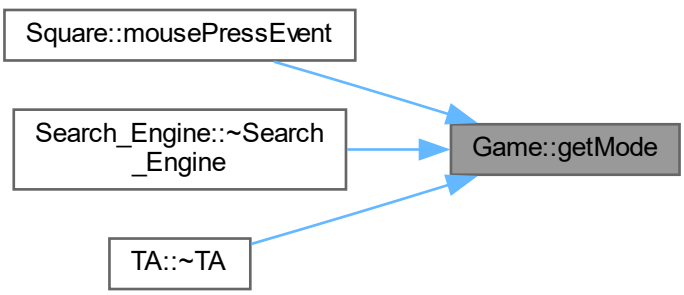
```
Modes::MODES Game::getMode ( ) const
```

Get the mode.

Returns

Modes::MODES Mode which is active

Here is the caller graph for this function:



4.6.2.21 getScore()

```
int Game::getScore ( ) const
```

Returns the current score.

Returns

int

4.6.2.22 getShortestPath()

```
QList< QPoint > Game::getShortestPath (
            QPoint start )
```

Gets the shortest list of path coordinates which the enemies can take to get from an arbitrary path coordinate to the ending point. Uses the following priority:

- If there is at least one unobstructed path between the given start point and the ending point, take the shortest path.
- If all paths are obstructed, take the shortest path to the ending point disregarding any obstructions.

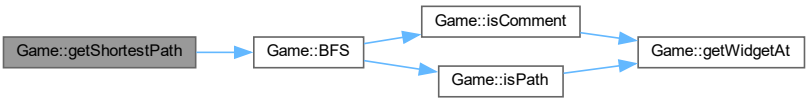
Parameters

start	The path coordinate to start from.

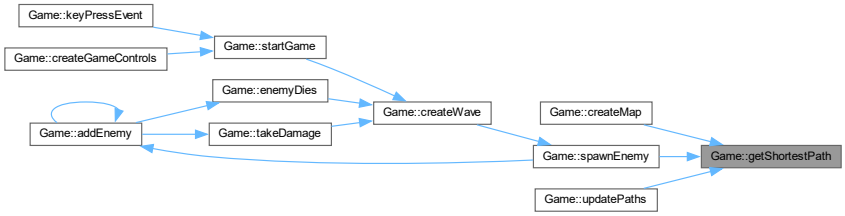
Returns

A list of path coordinates, representing the shortest path enemies can take to get to the ending point.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.23 getSquarePos()

```
QPointF Game::getSquarePos (
            int row,
            int column )
```

Returns the scene location of a square in the grid.

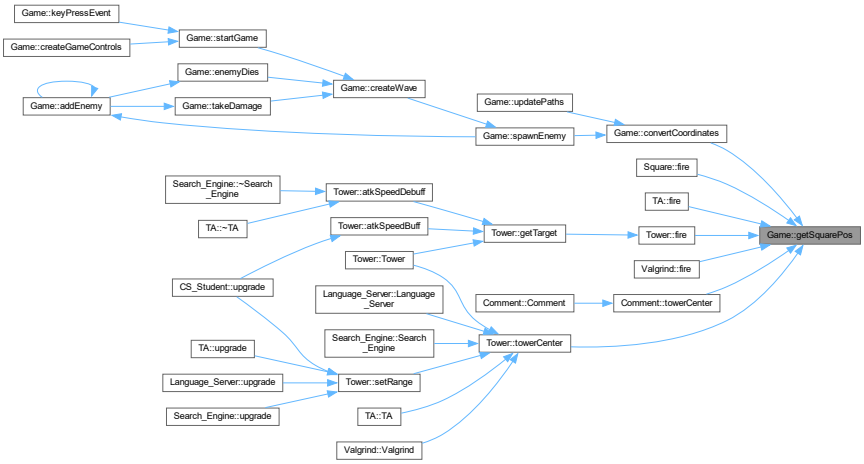
Parameters

row	The row in the grid.
column	The column in the grid.

Returns

QPointF

Here is the caller graph for this function:



4.6.2.24 getWidgetAt()

```
QWidget * Game::getWidgetAt (
            int row,
            int column )
```

Get a widget at the given position on the game's grid map.

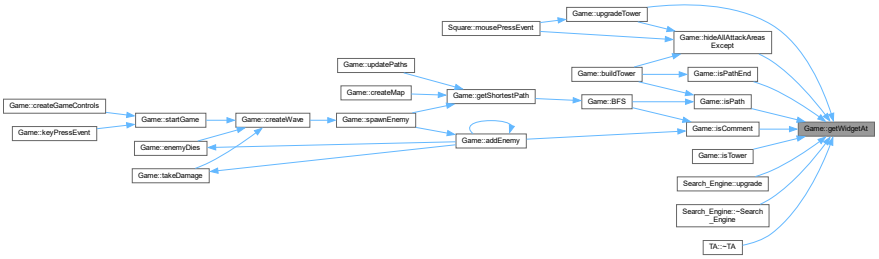
Parameters

row	a row in the game's grid map
column	a column in the game's grid map

Returns

a pointer to the widget at the given position in the grid map

Here is the caller graph for this function:



4.6.2.25 hideAllAttackAreasExcept()

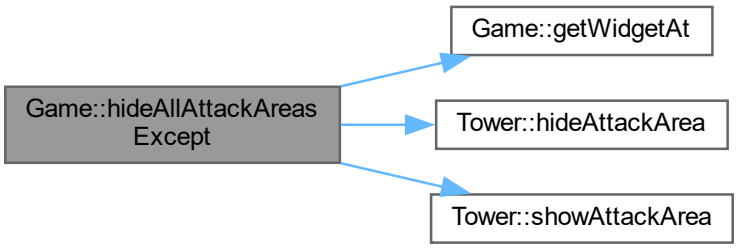
```
void Game::hideAllAttackAreasExcept (
            QPointF exclude = QPointF(-1, -1) )
```

Hides attack areas of all the towers except the tower on the specified coordinate.

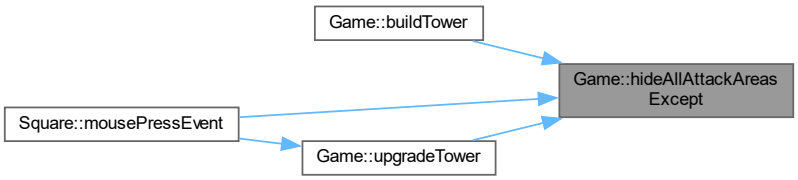
Parameters

exclude	Coordinates of tower of which attack area will not be hidden by this function.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.26 isComment()

```
bool Game::isComment (
            int row,
            int column )
```

Checks if the square at the given location contains a comment tower.

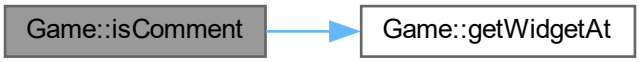
Parameters

row	The row of the location in question.
column	The column of the location in question.

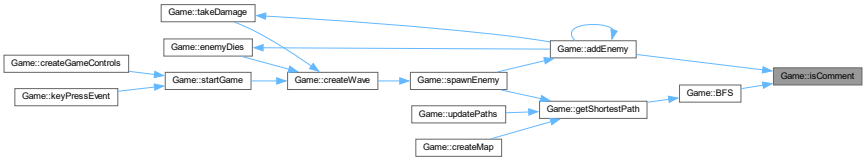
Returns

A boolean denoting if the square at the given location contains a comment tower or not.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.27 isEnemy()

```
bool Game::isEnemy (
            int row,
            int column )
```

Checks if the location is populated by enemies.

Parameters

row	The row in the grid.
column	The column in the grid.

Returns

true

false

Here is the caller graph for this function:

4.6.2.28 isLost()

```
bool Game::isLost ( ) const
```

Performs a check if the game is lost.

Returns

true

false

Here is the caller graph for this function:

4.6.2.29 isPath()

```
bool Game::isPath (
            int row,
            int column )
```

Checks if the square at the given location is a path tile.

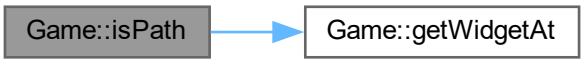
Parameters

row	The row of the square in question.
column	The column of the square in question.

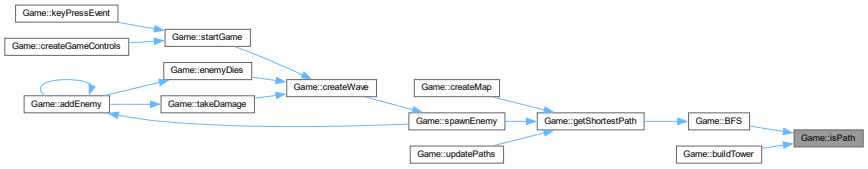
Returns

A boolean denoting if the square located at the given location is a path tile or not.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.30 isPathEnd()

```
bool Game::isPathEnd (
            int row,
            int column )
```

Checks if the square at the given location is an end of the enemy's path.

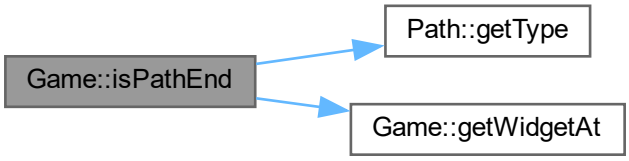
Parameters

row	The row of the square in question.
column	The column of the square in question.

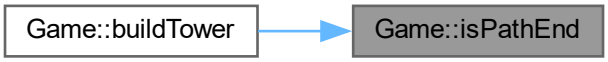
Returns

A boolean denoting if the square located at the given location is an end of the enemy's path.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.31 isTower()

```
bool Game::isTower (
            int row,
            int column )
```

Check if the item at the given position in the grid map is a **Tower** (p. 101) or not.

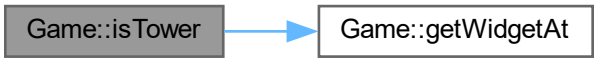
Parameters

row	a row in the game's grid map
column	a column in the game's grid map

Returns

true if the item is a **Tower** (p. 101), false otherwise

Here is the call graph for this function:



4.6.2.32 isWaveWon()

```
bool Game::isWaveWon ( )
```

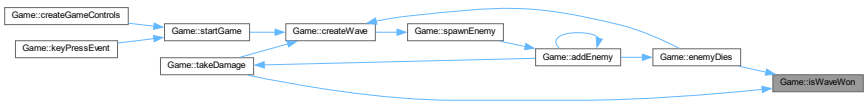
Performs a check if the level is won.

Returns

true

false

Here is the caller graph for this function:



4.6.2.33 isWon()

```
bool Game::isWon ( ) const
```

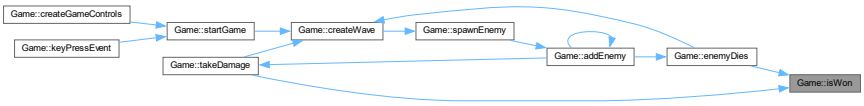
Performs a check if the game is won.

Returns

true

false

Here is the caller graph for this function:



4.6.2.34 keyPressEvent()

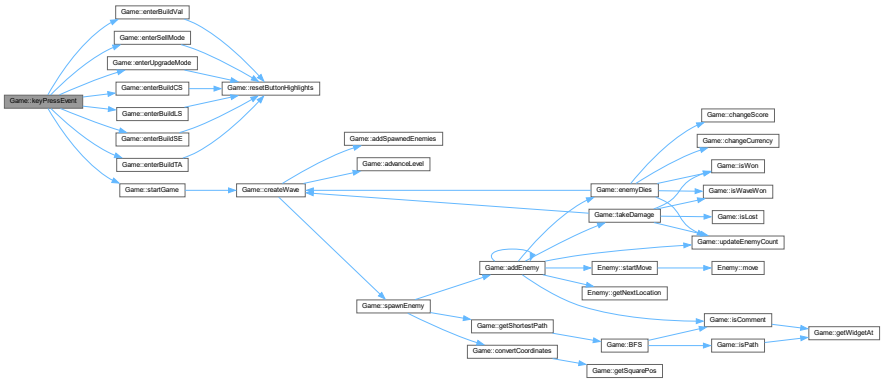
```
void Game::keyPressEvent (
            QKeyEvent * event )
```

Initiates the correct functions related to the key pressed.

Parameters

event	The event. Mainly it contains the character which corresponds to the key which was pressed.

Here is the call graph for this function:



4.6.2.35 readWaveFile()

```
void Game::readWaveFile ( )
```

Reads the wave.txt file which specifies the levels of the games. README.txt can be found in /src/files. Adds waves to the QList<QStringList> waves_ member which will be used for waves' creation and the finalLevel_ member.

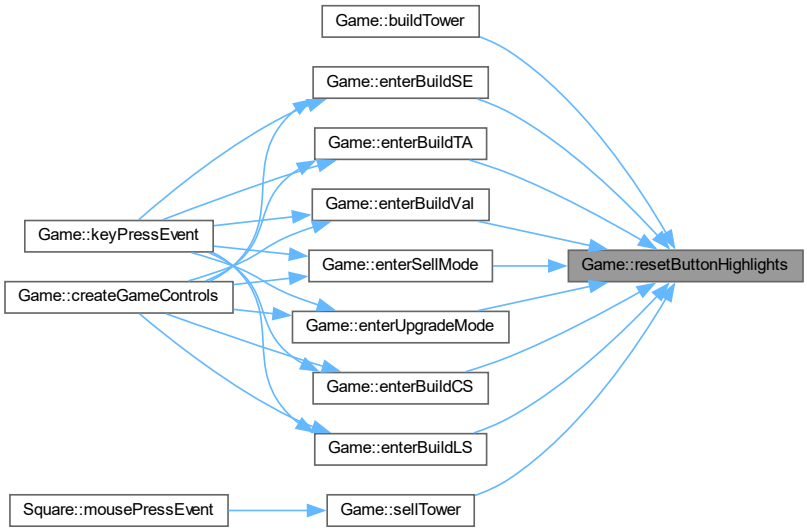
Here is the caller graph for this function:

4.6.2.36 resetButtonHighlights()

```
void Game::resetButtonHighlights ( )
```

Resets the button highlighting back to default.

Here is the caller graph for this function:



4.6.2.37 sellTower()

```
bool Game::sellTower (
            int row,
            int column )
```

Sell the tower at the given position on the grid map, which destroy it and give back some money to the player.

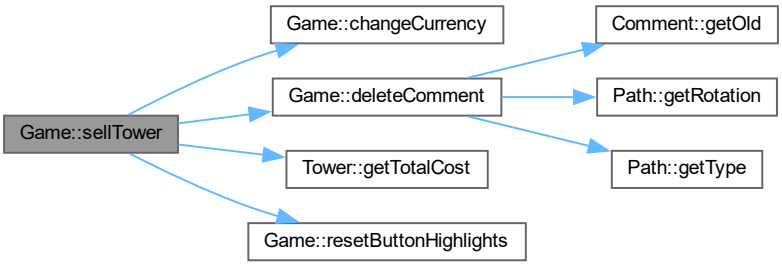
Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map

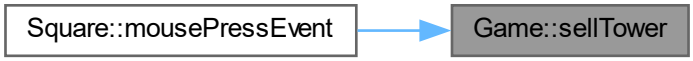
Returns

true if the sale was successful, false otherwise

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.38 setMode()

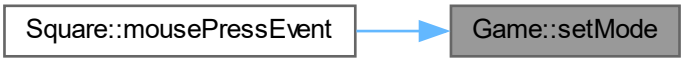
```
void Game::setMode (
            Modes::MODES m )
```

Set the game's mode to the given mode.

Parameters

m	an enum that represents a certain game mode

Here is the caller graph for this function:



4.6.2.39 showError

```
void Game::showError (
            QString message )  [slot]
```

Displays an error message in a QMessageBox.

Parameters

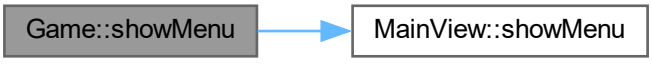
message	The message in QString

4.6.2.40 showMenu

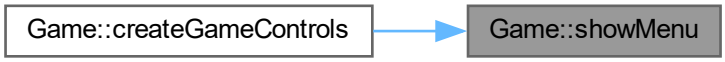
```
void Game::showMenu ( )  [slot]
```

Exits the game into the main menu.

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.41 spawnEnemy

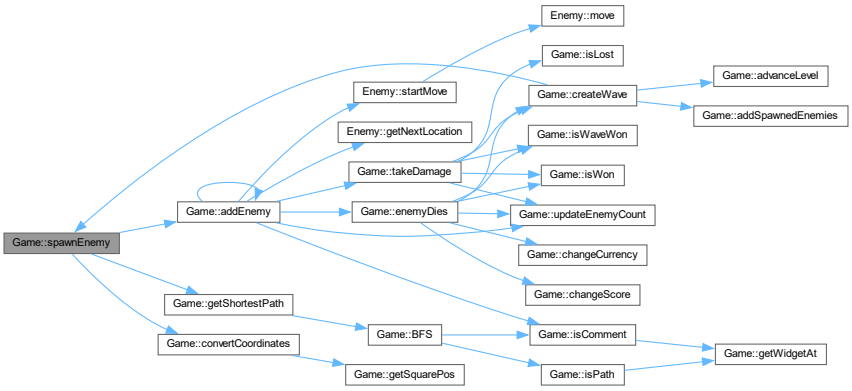
```
void Game::spawnEnemy (
            int type )  [slot]
```

Is called to spawn an enemy of the specified type. (Enum is located in **enemy.h** (p. 132)) Calls the slot addEnemy which will create the enemy and adds it to the scene.

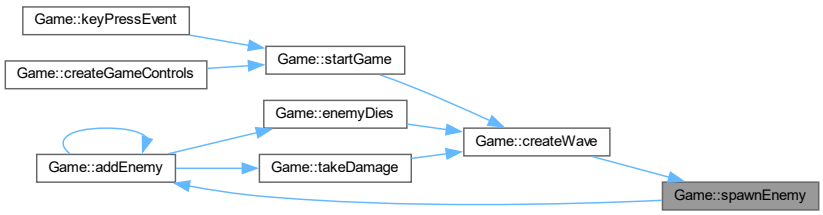
Parameters

type	One of the values of the enum EnemyType

Here is the call graph for this function:



Here is the caller graph for this function:



4.6.2.42 stopEnemies

```
void Game::stopEnemies ( )  [slot]
```

Stops the timers of all the enemies in the game.

Here is the call graph for this function:

4.6.2.43 takeDamage

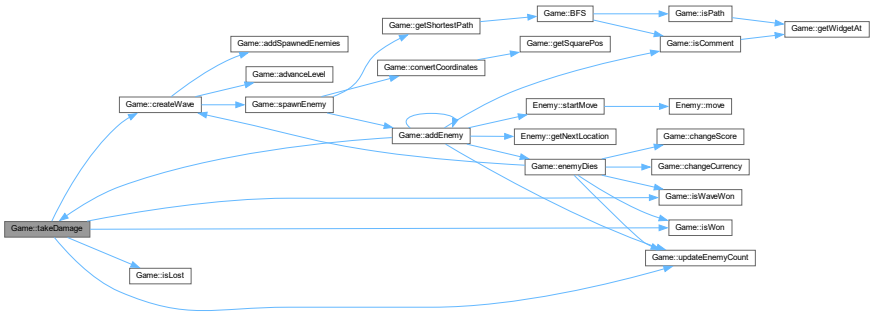
```
void Game::takeDamage (
            int dHealth )  [slot]
```

Is called when the player takes damage from an enemy.

Parameters

dHealth	The amount of damage dealt by an enemy.

Here is the call graph for this function:



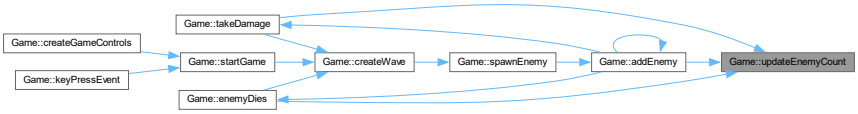
Here is the caller graph for this function:

4.6.2.44 updateEnemyCount

```
void Game::updateEnemyCount ( )  [slot]
```

Updates the currently active enemy count.

Here is the caller graph for this function:

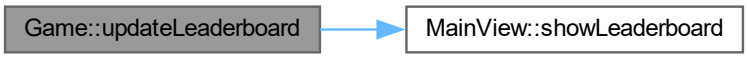


4.6.2.45 updateLeaderboard

```
void Game::updateLeaderboard ( )  [slot]
```

Is called when the game is lost or won. Reads the data from the leaderboard file (or creates it), compares the current score to the leaderboard and asks the name of the player if it is in the top10. Updates the leaderboard accordingly. Uses QInputDialog for communicating with the player.

Here is the call graph for this function:



4.6.2.46 upgradeTower()

```
bool Game::upgradeTower (
            int row,
            int column )
```

Upgrade the tower at the given coordinates on the grid map.

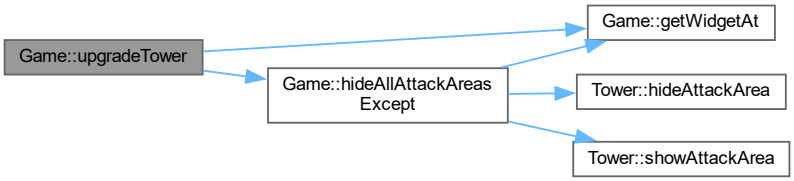
Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map

Returns

true if the upgrade was successful, false otherwise

Here is the call graph for this function:



Here is the caller graph for this function:

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

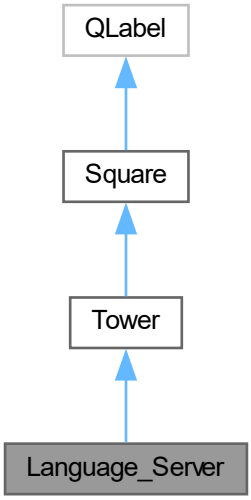
- **game.h**
- **game.cpp**

4.7 Language_Server Class Reference

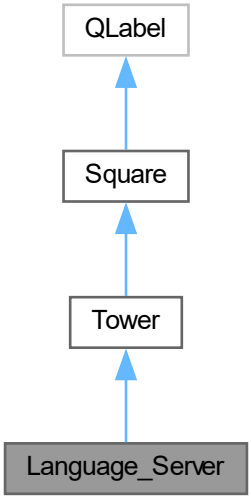
A Language Server tower type. Can attack all enemies in range at the same time. Inherited from **Tower** (p. 101) class.

`#include <language_server.h>`

Inheritance diagram for Language_Server:



Collaboration diagram for Language_Server:



Public Member Functions

- **Language_Server** (int row, int column, QWidget *parent=nullptr)

  Language_Server *(p. 61) constructor, create a Language Server tower at the given posion on the grid map.*
- void **fire** (QPointF targetPos)

  Send a pulse-like projectile that damage all enemies in range.
- virtual bool **upgrade** ()

  Upgrade this tower if possible.
- void **updateDescription** ()

  Update the tower's text description base on the tower's current stats.

Additional Inherited Members

4.7.1 Detailed Description

A Language Server tower type. Can attack all enemies in range at the same time. Inherited from **Tower** (p. 101) class.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Language_Server()

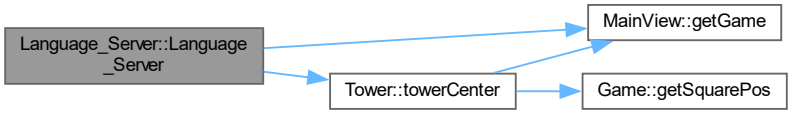
```
Language_Server::Language_Server (
            int row,
            int column,
            QWidget * parent = nullptr )
```

Language_Server (p. 61) constructor, create a Language Server tower at the given posion on the grid map.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

Here is the call graph for this function:



4.7.3 Member Function Documentation

4.7.3.1 fire()

```
void Language_Server::fire (
            QPointF targetPos )  [virtual]
```

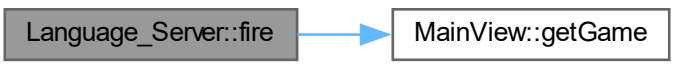
Send a pulse-like projectile that damage all enemies in range.

Parameters

QPointF	unused, only for matching the overloading signature of the base class

Reimplemented from **Tower** (p. 109).

Here is the call graph for this function:



4.7.3.2 updateDescription()

```
void Language_Server::updateDescription ( )  [virtual]
```

Update the tower's text description base on the tower's current stats.

Reimplemented from **Tower** (p. 117).

Here is the caller graph for this function:

4.7.3.3 upgrade()

```
bool Language_Server::upgrade ( )  [virtual]
```

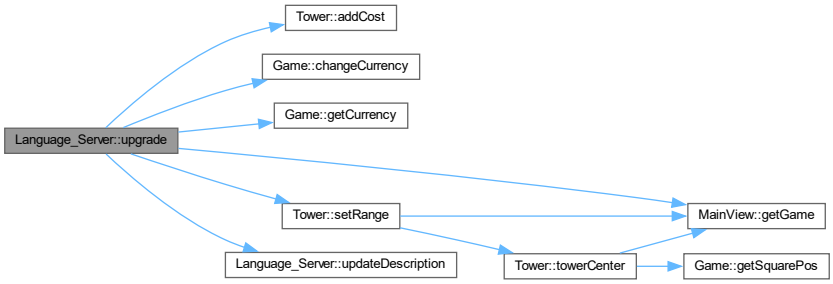
Upgrade this tower if possible.

Returns

true if the upgrade was successful, false otherwise.

Implements **Tower** (p. 101).

Here is the call graph for this function:



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

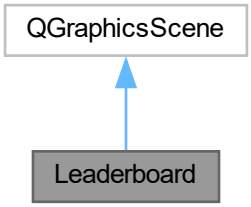
- **language_server.h**
- language_server.cpp

4.8 Leaderboard Class Reference

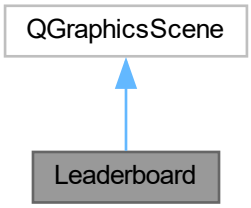
This is the leaderboard scene.

`#include <leaderboard.h>`

Inheritance diagram for Leaderboard:



Collaboration diagram for Leaderboard:



Public Slots

- void **showMenu** ()

  Shows menu.

Public Member Functions

- **Leaderboard** (QObject ∗parent=0)

  *Construct a new **Leaderboard** (p. 66):: **Leaderboard** (p. 66) object. Places the graphics into the leaderboard scene.*
- void **setLeaderBoard** (QList< QPair< QString, int > > leaderboard)

  Sets leaderboard.
- void **readFile** ()

  Reads the leaderboard data file from OS specific data location. On windows appdata%.

4.8.1 Detailed Description

This is the leaderboard scene.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Leaderboard()

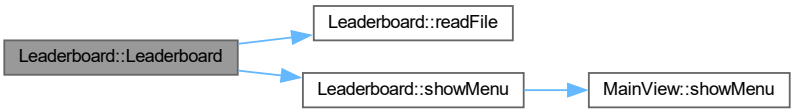
```
Leaderboard::Leaderboard (
            QObject * parent = 0 )
```

Construct a new **Leaderboard** (p. 66):: **Leaderboard** (p. 66) object. Places the graphics into the leaderboard scene.

Parameters

parent	The parent object of the leaderboard scene.

Here is the call graph for this function:



4.8.3 Member Function Documentation

4.8.3.1 readFile()

```
void Leaderboard::readFile ( )
```

Reads the leaderboard data file from OS specific data location. On windows appdata%.

Here is the caller graph for this function:

4.8.3.2 setLeaderBoard()

```
void Leaderboard::setLeaderBoard (
            QList< QPair< QString, int > > leaderboard )
```

Sets leaderboard.

Parameters

leaderboard	New leaderboard.

4.8.3.3 showMenu

```
void Leaderboard::showMenu ( )  [slot]
```

Shows menu.

Here is the call graph for this function:

Here is the caller graph for this function:

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

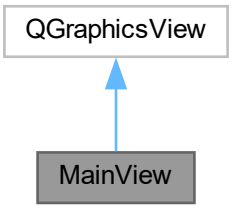
- **leaderboard.h**
- **leaderboard.cpp**

4.9 MainView Class Reference

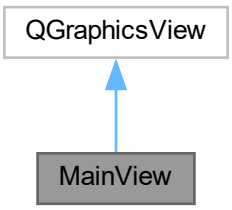
Class for representing the View that contains the **Menu** (p. 75), **Game** (p. 28) and **Leaderboard** (p. 66).

`#include <mainview.h>`

Inheritance diagram for MainView:



Collaboration diagram for MainView:



Public Slots

- void **showGame** (int gamemode)

  Switches to the game scene.
- void **showMenu** ()

  Switches to the menu scene.
- void **showLeaderboard** ()

  Switches to the leaderboard scene.

Public Member Functions

- **MainView** (QWidget *parent=nullptr)

  Construct a new Main View:: Main View object.
- **Game** * **getGame** ()

  Returns the game pointer.
- **Menu** * **getMenu** ()

  Returns the menu pointer.
- **Leaderboard** * **getLeaderboard** ()

  Returns the leaderboard pointer.

4.9.1 Detailed Description

Class for representing the View that contains the **Menu** (p. 75), **Game** (p. 28) and **Leaderboard** (p. 66).

4.9.2 Constructor & Destructor Documentation

4.9.2.1 MainView()

```
MainView::MainView (
            QWidget * parent = nullptr )
```

Construct a new Main View:: Main View object.

Parameters

parent	

4.9.3 Member Function Documentation

4.9.3.1 getGame()

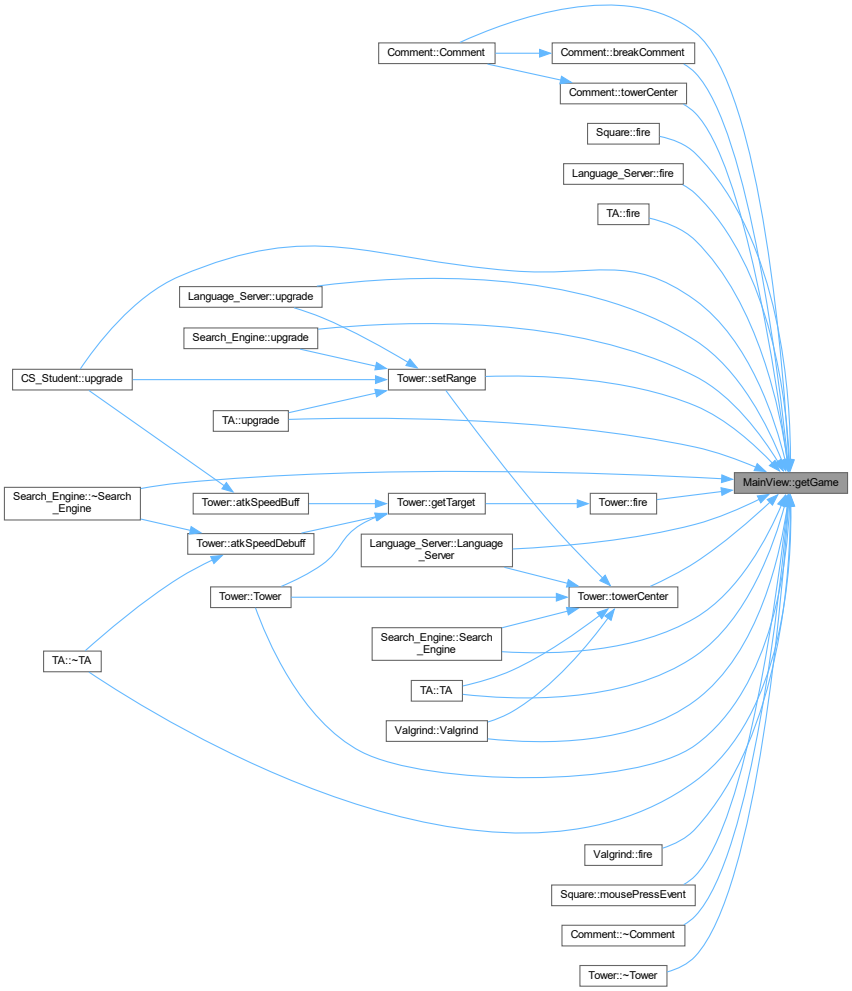
```
Game * MainView::getGame ( )
```

Returns the game pointer.

Returns

Game*

Here is the caller graph for this function:



4.9.3.2 getLeaderboard()

```
Leaderboard * MainView::getLeaderboard ( )
```

Returns the leaderboard pointer.

Returns

Leaderboard*

4.9.3.3 getMenu()

```
Menu * MainView::getMenu ( )
```

Returns the menu pointer.

Returns

Menu*

4.9.3.4 showGame

```
void MainView::showGame (
            int gamemode )  [slot]
```

Switches to the game scene.

Parameters

gamemode	The mode of the game to use.

Here is the call graph for this function:

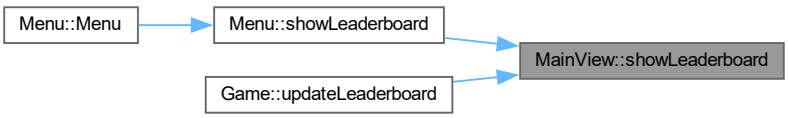
Here is the caller graph for this function:

4.9.3.5 showLeaderboard

```
void MainView::showLeaderboard ( )  [slot]
```

Switches to the leaderboard scene.

Here is the caller graph for this function:

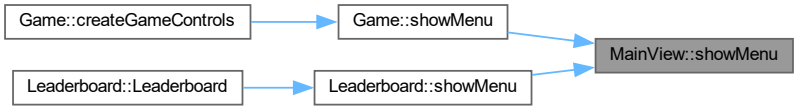


4.9.3.6 showMenu

```
void MainView::showMenu ( )  [slot]
```

Switches to the menu scene.

Here is the caller graph for this function:



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

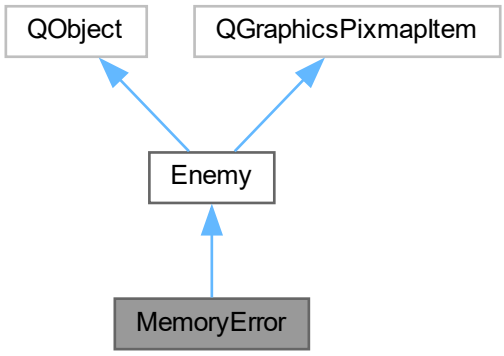
- **mainview.h**
- **mainview.cpp**

4.10 MemoryError Class Reference

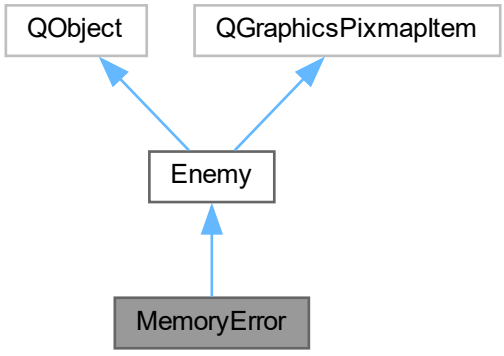
Class for the **MemoryError** (p. 74) type enemy.

`#include <memoryerror.h>`

Inheritance diagram for MemoryError:



Collaboration diagram for MemoryError:



Public Member Functions

- **MemoryError** (MemoryErrorType subType, QList< QPointF > path, QList< QPoint > matrixPath)

  Construct a new Memory Error:: Memory Error object Checks the subtype of the error and set the members accordingly.

Additional Inherited Members

4.10.1 Detailed Description

Class for the **MemoryError** (p. 74) type enemy.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 MemoryError()

```
MemoryError::MemoryError (
            MemoryErrorType subType,
            QList< QPointF > path,
            QList< QPoint > matrixPath )
```

Construct a new Memory Error:: Memory Error object Checks the subtype of the error and set the members accordingly.

Parameters

subType	
path	
matrixPath	

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

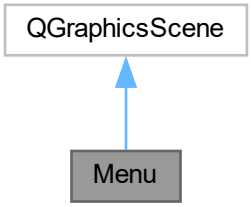
- **memoryerror.h**
- **memoryerror.cpp**

4.11 Menu Class Reference

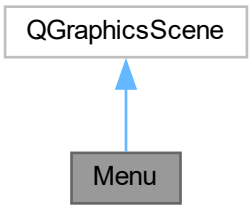
This is the menu scene.

```
#include <menu.h>
```

Inheritance diagram for Menu:



Collaboration diagram for Menu:



Public Slots

- void **showGame** (int gamemode)

  Shows game.
- void **showLeaderboard** ()

  Shows leaderboard.
- void **quit** ()

  Quits the whole program.

Public Member Functions

- **Menu** (QObject ∗parent=nullptr)

  *Construct a new **Menu** (p. 75):: **Menu** (p. 75) object. Places all the graphics in the menu scene.*

4.11.1 Detailed Description

This is the menu scene.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Menu()

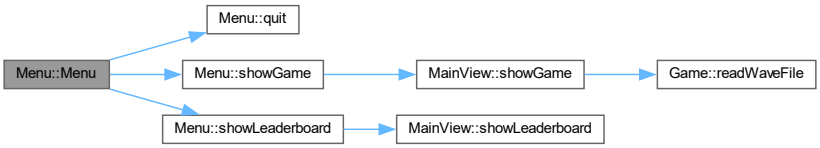
```
Menu::Menu (
            QObject * parent = nullptr )
```

Construct a new **Menu** (p. 75):: **Menu** (p. 75) object. Places all the graphics in the menu scene.

Parameters

parent	The parent object of the menu scene.

Here is the call graph for this function:



4.11.3 Member Function Documentation

4.11.3.1 quit

```
void Menu::quit ( )  [slot]
```

Quits the whole program.

Here is the caller graph for this function:

4.11.3.2 showGame

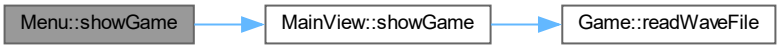
```
void Menu::showGame (
            int gamemode )  [slot]
```

Shows game.

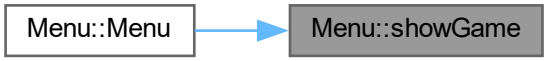
Parameters

gamemode	Specifies the gamemode of the game showed.

Here is the call graph for this function:



Here is the caller graph for this function:

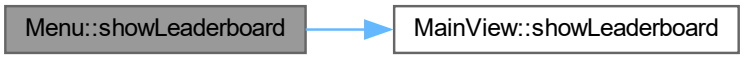


4.11.3.3 showLeaderboard

```
void Menu::showLeaderboard ( )  [slot]
```

Shows leaderboard.

Here is the call graph for this function:



Here is the caller graph for this function:

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

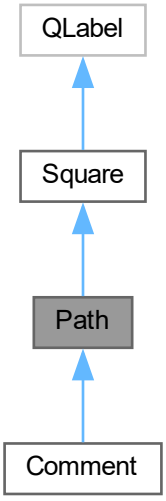
- **menu.h**
- **menu.cpp**

4.12 Path Class Reference

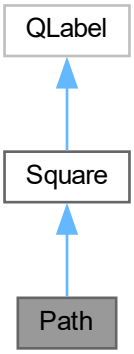
The **Path** (p. 79) class. Represents the path tiles on which enemies travel on.

`#include <path.h>`

Inheritance diagram for Path:



Collaboration diagram for Path:



Public Member Functions

- **Path** (int x, int y, PathType type, int rotation, **Path** *old=nullptr, QWidget *parent=nullptr)

  *The constructor class for the **Path** (p. 79).*
- PathType **getType** ()

  Gets the type of the path.
- int **getRotation** ()

  Gets the rotational orientation of the path.

Additional Inherited Members

4.12.1 Detailed Description

The **Path** (p. 79) class. Represents the path tiles on which enemies travel on.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 Path()

```
Path::Path (
            int x,
            int y,
            PathType type,
            int rotation,
             Path * old = nullptr,
            QWidget * parent = nullptr )
```

The constructor class for the **Path** (p. 79).

Parameters

x	The x-coordinate of the path.
y	The y-coordinate of the path.
type	The type of path.
rotation	The rotational orientation of the path, in degrees.
old	If the path contains a comment tower, this will hold the path tile the comment tower replaced when it was built.
parent	The parent widget of the path.

4.12.3 Member Function Documentation

4.12.3.1 getRotation()

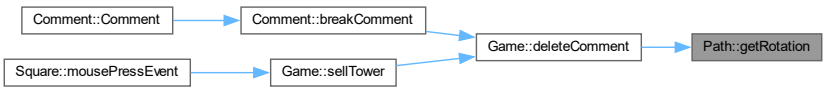
```
int Path::getRotation ( )
```

Gets the rotational orientation of the path.

Returns

The rotational orientation of the path, in degrees.

Here is the caller graph for this function:



4.12.3.2 getType()

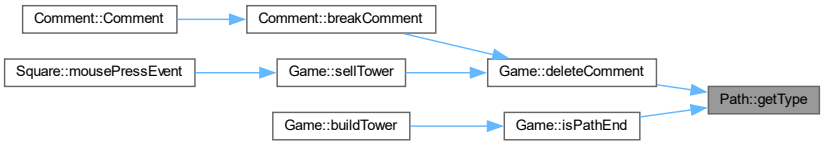
```
PathType Path::getType ( )
```

Gets the type of the path.

Returns

The type of the path.

Here is the caller graph for this function:



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

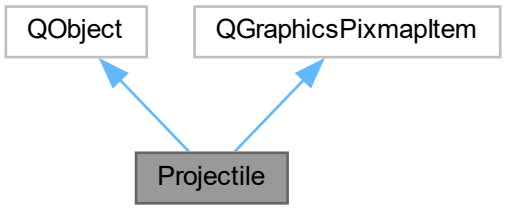
- **path.h**
- **path.cpp**

4.13 Projectile Class Reference

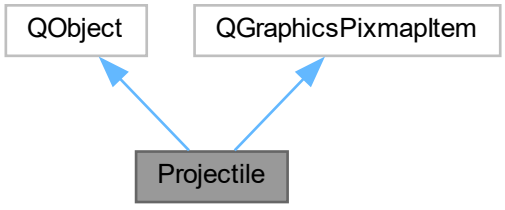
This is the projectile object.

`#include <projectile.h>`

Inheritance diagram for Projectile:



Collaboration diagram for Projectile:



Public Slots

- void **move** ()

  Moves the projectiles forward the stepSize amount. Deletes the projectile when it has traveled far enough. Damages colliding enemies.

Public Member Functions

- **Projectile** (int damage, QString imgPath=":/images/CStudent_projectile.png", int pierce=0, int stepSize=6, int maxLifeTime=100000, QGraphicsItem ∗parent=0)

  *Construct a new **Projectile** (p. 82):: **Projectile** (p. 82) object. Initializes timers to cause moving.*
- double **getMaxRange** ()

  Returns the maxRange of the projectile.
- double **getDistanceTravelled** ()

  Returns the distance which the projectile has traveled.
- void **setMaxRange** (double rng)

  Sets the maxRange_ variable.
- void **setDistanceTravelled** (double dist)

  Sets the variable distanceTravelled_.
- void **setMoveFrequency** (int newValue)

4.13.1 Detailed Description

This is the projectile object.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Projectile()

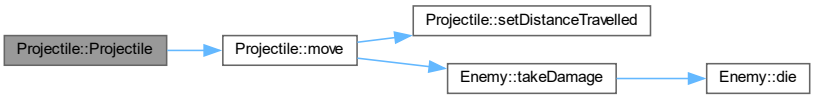
```
Projectile::Projectile (
            int damage,
            QString imgPath = ":/images/CStudent_projectile.png",
            int pierce = 0,
            int stepSize = 6,
            int maxLifeTime = 100000,
            QGraphicsItem * parent = 0 )
```

Construct a new **Projectile** (p. 82):: **Projectile** (p. 82) object. Initializes timers to cause moving.

Parameters

damage	Damage that the projectile causes on enemies it hits
imgPath	**Path** (p. 79) to the image file
pierce	Number of enemies the projectile can damage
stepSize	Size of the step the projectile will make when it moves
maxLifeTime	Maximum time the projectile can exist
parent	The parent of the projectile.

Here is the call graph for this function:



4.13.3 Member Function Documentation

4.13.3.1 getDistanceTravelled()

```
double Projectile::getDistanceTravelled ( )
```

Returns the distance which the projectile has traveled.

Returns

the distance which the projectile has traveled.

4.13.3.2 getMaxRange()

```
double Projectile::getMaxRange ( )
```

Returns the maxRange of the projectile.

Returns

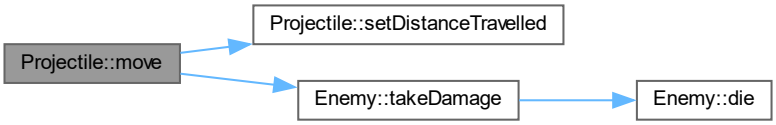
the maxRange of the projectile.

4.13.3.3 move

```
void Projectile::move ( )  [slot]
```

Moves the projectiles forward the stepSize amount. Deletes the projectile when it has traveled far enough. Damages colliding enemies.

Here is the call graph for this function:



Here is the caller graph for this function:

4.13.3.4 setDistanceTravelled()

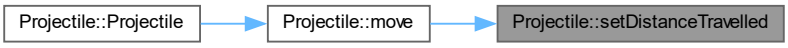
```
void Projectile::setDistanceTravelled (
            double distance )
```

Sets the variable distanceTravelled_.

Parameters

distance	New value for the distance.

Here is the caller graph for this function:



4.13.3.5 setMaxRange()

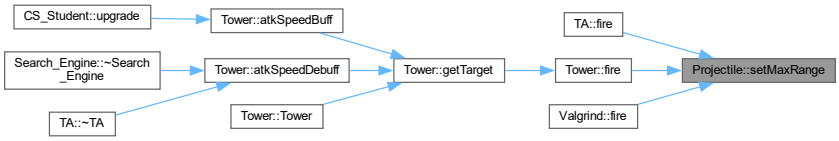
```
void Projectile::setMaxRange (
            double range )
```

Sets the maxRange_ variable.

Parameters

range	New value for the range.

Here is the caller graph for this function:



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

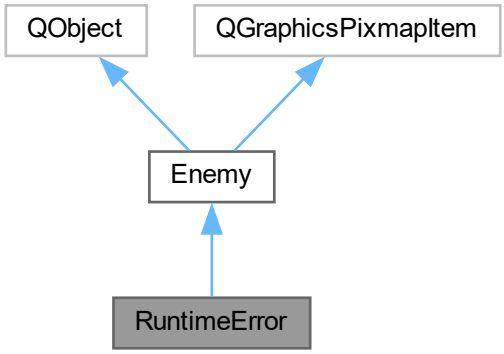
- **projectile.h**
- **projectile.cpp**

4.14 RuntimeError Class Reference

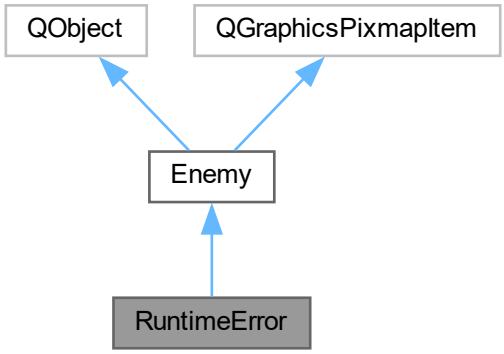
Class for the **RuntimeError** (p. 86) type enemies.

```
#include <runtimeerror.h>
```

Inheritance diagram for RuntimeError:



Collaboration diagram for RuntimeError:



Public Member Functions

- **RuntimeError** (RuntimeErrorType subType, QList< QPointF > path, QList< QPoint > matrixPath)

  Construct a new Runtime Error:: Runtime Error object Checks the subType of the enemy and sets the members accordingly.
- void **takeDamage** (int damage)

  Is called when the enemy takes damage from a projectile. Since this a boss enemy it releases faster minions and has a nr of stages. Minions are released after every stage.

Additional Inherited Members

4.14.1 Detailed Description

Class for the **RuntimeError** (p. 86) type enemies.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 RuntimeError()

```
RuntimeError::RuntimeError (
            RuntimeErrorType subType,
            QList< QPointF > path,
            QList< QPoint > matrixPath )
```

Construct a new Runtime Error:: Runtime Error object Checks the subType of the enemy and sets the members accordingly.

Parameters

subType	
path	
matrixPath	

4.14.3 Member Function Documentation

4.14.3.1 takeDamage()

```
void RuntimeError::takeDamage (
            int damage )  [virtual]
```

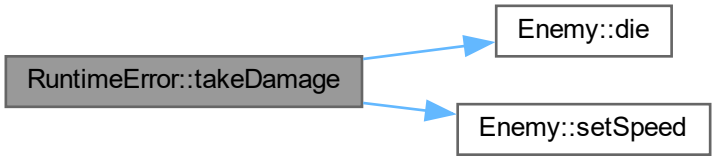
Is called when the enemy takes damage from a projectile. Since this a boss enemy it releases faster minions and has a nr of stages. Minions are released after every stage.

Parameters

damage	The amount of damage taken by the projectile.

Reimplemented from **Enemy** (p. 25).

Here is the call graph for this function:



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

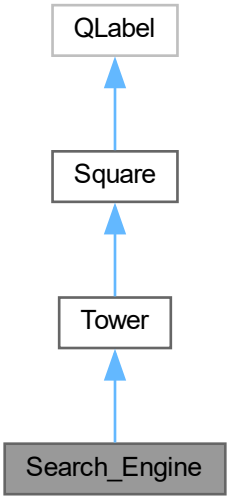
- **runtimeerror.h**
- **runtimeerror.cpp**

4.15 Search_Engine Class Reference

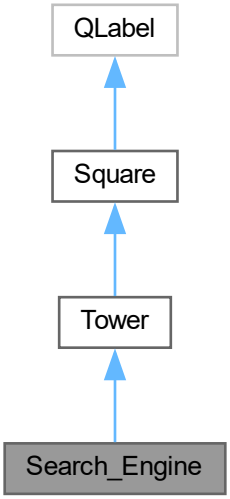
A Search Engine tower type. Can buff nearby tower, but does not shoot at enemies. Inherited from **Tower** (p. 101) class.

`#include <search_engine.h>`

Inheritance diagram for Search_Engine:



Collaboration diagram for Search_Engine:



Public Slots

- void **buffPulse** ()

  Send out a buff pulse that buff all towers in range that haven't been buffed yet.

Public Member Functions

- **Search_Engine** (int row, int column, QWidget ∗parent=nullptr)

  Search_Engine *(*p. *89) constructor, create a Search Engine tower at the given posion on the grid map.*
- ∼**Search_Engine** ()

  Search_Engine *(*p. *89) destructor, delete the Search Engine's members and remove the buff from all the towers buffed by this one.*
- bool **upgrade** ()

  Upgrade this tower if possible.
- void **updateDescription** ()

  Update the tower's text description base on the tower's current stats.

Additional Inherited Members

4.15.1 Detailed Description

A Search Engine tower type. Can buff nearby tower, but does not shoot at enemies. Inherited from **Tower** (p. 101) class.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 Search_Engine()

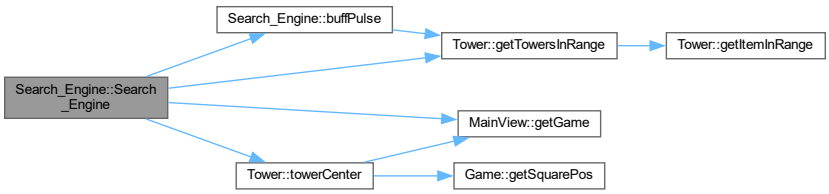
```
Search_Engine::Search_Engine (
            int row,
            int column,
            QWidget * parent = nullptr )
```

Search_Engine (p. 89) constructor, create a Search Engine tower at the given posion on the grid map.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

Here is the call graph for this function:



4.15.3 Member Function Documentation

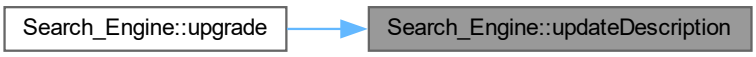
4.15.3.1 updateDescription()

```
void Search_Engine::updateDescription ( )  [virtual]
```

Update the tower's text description base on the tower's current stats.

Reimplemented from **Tower** (p. 117).

Here is the caller graph for this function:



4.15.3.2 upgrade()

```
bool Search_Engine::upgrade ( )  [virtual]
```

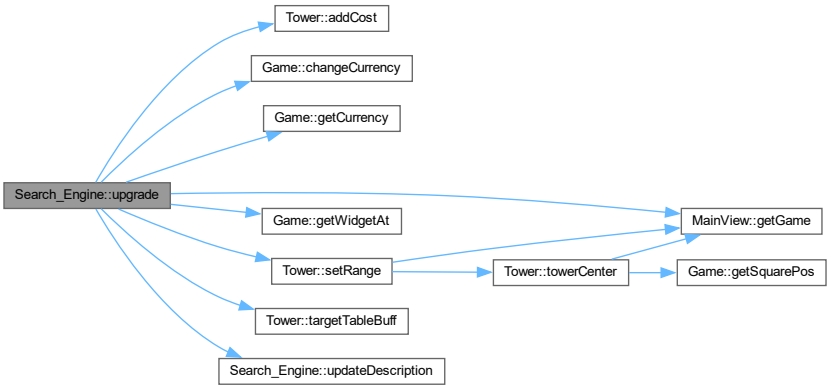
Upgrade this tower if possible.

Returns

true if the upgrade was successful, false otherwise.

Implements **Tower** (p. 101).

Here is the call graph for this function:



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

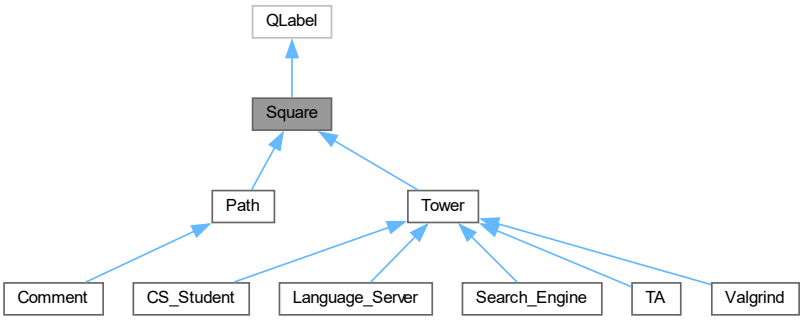
- **search_engine.h**
- search_engine.cpp

4.16 Square Class Reference

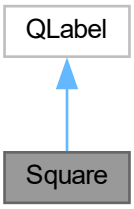
This is the square object. It represents a square which is a tower or an empty square where a tower can be placed or a path.

```
#include <square.h>
```

Inheritance diagram for Square:



Collaboration diagram for Square:



Public Slots

- virtual void **getTarget** ()

Public Member Functions

- **Square** (int x, int y, QWidget ∗parent=nullptr)

  *Construct a new **Square** (p. 92):: **Square** (p. 92) object.*
- void **mousePressEvent** (QMouseEvent ∗event)

  Based on the mode, builds a tower or upgrades the tower on the square. Changes mode back to normal mode.
- virtual bool **isTower** ()

  Checks if the square is a tower.
- virtual void **showHideAttackArea** ()
- QPointF **getCoords** ()
- virtual void **buffPulse** ()

Protected Member Functions

- void **fire** (QPointF target)

  Fires a projectile at the targetPos.

Protected Attributes

- int **x_**

  The X-coordinate of the square.
- int **y_**

  The Y-coordinate of the square.

4.16.1 Detailed Description

This is the square object. It represents a square which is a tower or an empty square where a tower can be placed or a path.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Square()

```
Square::Square (
            int x,
            int y,
            QWidget * parent = nullptr )
```

Construct a new **Square** (p. 92):: **Square** (p. 92) object.

Parameters

x	X-coordinate of the square.
y	Y-coordinate of the square.
parent	The parent widget of the square.

4.16.3 Member Function Documentation

4.16.3.1 fire()

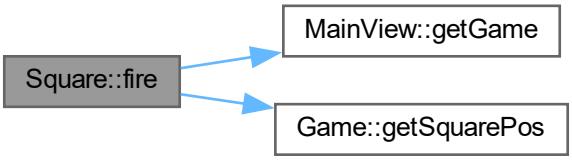
```
void Square::fire (
            QPointF targetPos )  [protected]
```

Fires a projectile at the targetPos.

Parameters

targetPos	Position the projectile is fired at.

Here is the call graph for this function:



4.16.3.2 isTower()

```
bool Square::isTower ( )  [virtual]
```

Checks if the square is a tower.

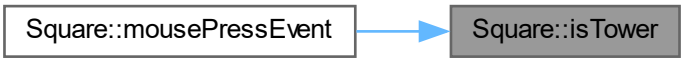
Returns

true **Square** (p. 92) is a tower.

false **Square** (p. 92) is not a tower.

Reimplemented in **Tower** (p. 113).

Here is the caller graph for this function:

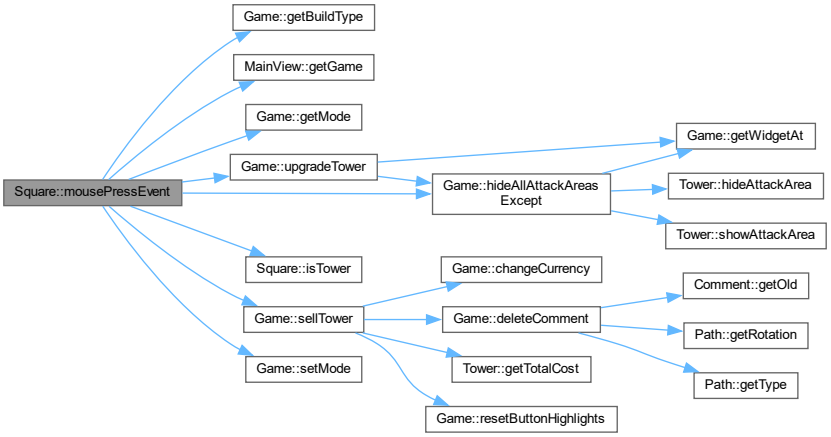


4.16.3.3 mousePressEvent()

```
void Square::mousePressEvent (
            QMouseEvent * event )
```

Based on the mode, builds a tower or upgrades the tower on the square. Changes mode back to normal mode.

Here is the call graph for this function:



4.16.3.4 showHideAttackArea()

```
virtual void Square::showHideAttackArea ( )  [inline], [virtual]
```

Reimplemented in **Tower** (p. 115).

4.16.4 Member Data Documentation

4.16.4.1 x_

```
int Square::x_  [protected]
```

The X-coordinate of the square.

4.16.4.2 y_

```
int Square::y_  [protected]
```

The Y-coordinate of the square.

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

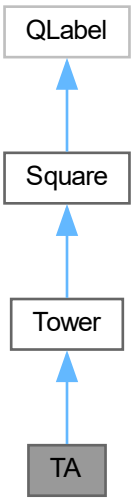
- **square.h**
- **square.cpp**

4.17 TA Class Reference

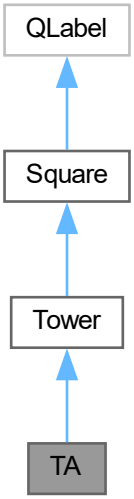
A **TA** (p. 97) tower type. Can attack enemies in range and buff nearby towers. Inherited from **Tower** (p. 101) class.

```
#include <ta.h>
```

Inheritance diagram for TA:



Collaboration diagram for TA:



Public Slots

- void **buffPulse** ()

  Send out a buff pulse that buff all towers in range that haven't been buffed yet.

Public Member Functions

- **TA** (int row, int column, QWidget ∗parent=nullptr)

  TA *(p. 97) constructor, create a* ***TA*** *(p. 97) tower at the given posion on the grid map.*
- ∼**TA** ()

  TA *(p. 97) destructor, delete the* ***TA*** *(p. 97)'s members and remove the buff from all the towers buffed by this one.*
- void **fire** (QPointF targetPos)

  Fire a projectile at the given target position.
- bool **upgrade** ()

  Upgrade this tower if possible.
- void **updateDescription** ()

  Update the tower's text description base on the tower's current stats.

Additional Inherited Members

4.17.1 Detailed Description

A **TA** (p. 97) tower type. Can attack enemies in range and buff nearby towers. Inherited from **Tower** (p. 101) class.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 TA()

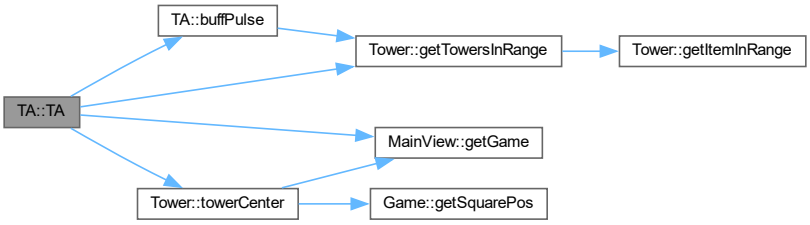
```
TA::TA (
            int row,
            int column,
            QWidget * parent = nullptr )
```

TA (p. 97) constructor, create a **TA** (p. 97) tower at the given posion on the grid map.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

Here is the call graph for this function:



4.17.3 Member Function Documentation

4.17.3.1 fire()

```
void TA::fire (
            QPointF targetPos )  [virtual]
```

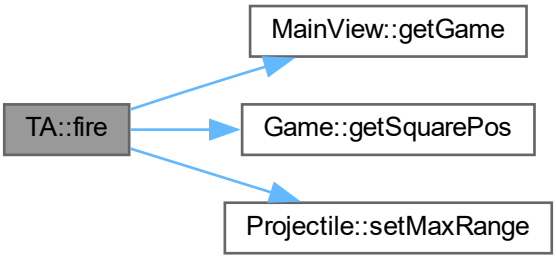
Fire a projectile at the given target position.

Parameters

targetPos	the position of the target to fire at

Reimplemented from **Tower** (p. 109).

Here is the call graph for this function:



4.17.3.2 updateDescription()

```
void TA::updateDescription ( )  [virtual]
```

Update the tower's text description base on the tower's current stats.

Reimplemented from **Tower** (p. 117).

Here is the caller graph for this function:

4.17.3.3 upgrade()

```
bool TA::upgrade ( )  [virtual]
```

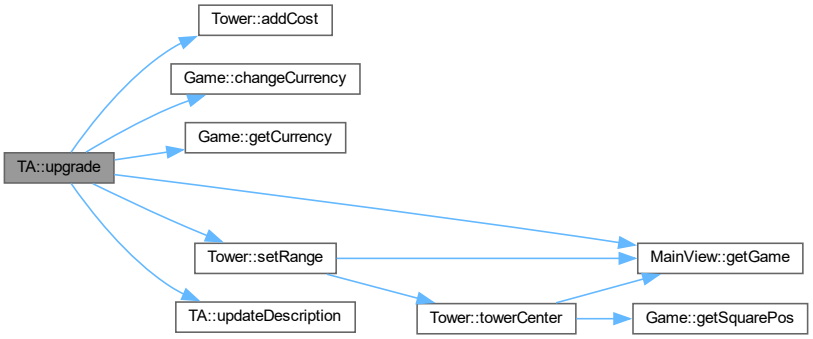
Upgrade this tower if possible.

Returns

true if the upgrade was successful, false otherwise.

Implements **Tower** (p. 101).

Here is the call graph for this function:



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

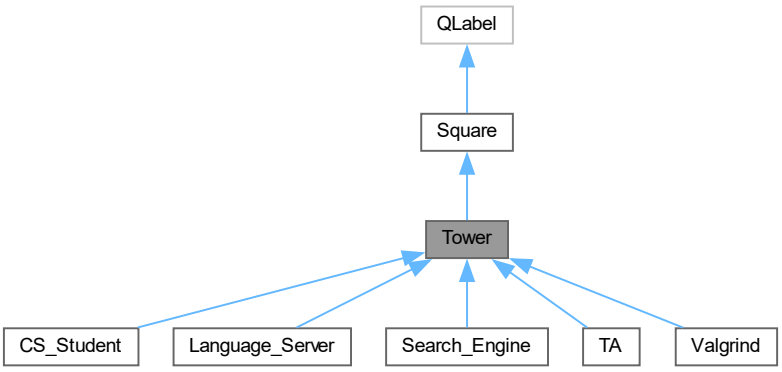
- **ta.h**
- ta.cpp

4.18 Tower Class Reference

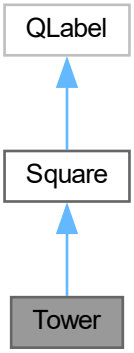
Base class for all towers in the game, inherited from **Square** (p. 92).

`#include <tower.h>`

Inheritance diagram for Tower:



Collaboration diagram for Tower:



Public Slots

- void **getTarget** ()

  Get a target for the tower to shoot at.

Public Member Functions

- **Tower** (QWidget ∗parent=nullptr)
- **Tower** (int row, int column, QWidget ∗parent=nullptr)

  Tower (p. *101) basic constructor, build a default tower (CS Student) at the given position.*
- **Tower** (int row, int column, int range, int damage, int attackSpeed, QWidget ∗parent=nullptr)

  Tower (p. *101) specific constructor, initialize the member of the tower with given parameter.*
- ∼**Tower** ()

  Tower (p. *101)'s destructor.*
- QPointF **towerCenter** ()

  Return the position of the tower's center on the screen.
- double **distanceTo** (QGraphicsItem ∗item)

  Calculate the distance from this tower to the given item on the screen.
- virtual void **fire** (QPointF targetPos)

  Fire a projectile at the given target position.
- bool **isTargetable** (**Enemy** ∗enemy)

  Check if the tower can target the given enemy.
- void **damageBuff** (double buffFactor)

  Increase the tower's damage by the given buff factor.
- void **atkSpeedBuff** (double buffFactor)

  Increase the tower's attack speed by the given buff factor.
- void **targetTableBuff** (EnemyTypes::TYPES type)

  Allow the tower to target the given enemy type.
- void **atkSpeedDebuff** (double debuffFactor)

  Decrease the tower's attack speed by the given buff factor.

- void **targetTableDebuff** (EnemyTypes::TYPES type)

  Remove an existing targetable buff from the tower. This will not affect the tower's own targetable ability. In other words if the tower was able to target an enemy type without any buff, it will still be able to do so.
- bool **hasAtkSpdBuff** ()

  Check if the tower has any attack speed buff or not.
- QGraphicsEllipseItem * **getAttackArea** ()

  Returns the tower's attack area.
- QList< QGraphicsItem * > **getItemInRange** ()

  Get a list of items that are currently in the tower's range.
- QList< **Tower** * > **getTowersInRange** ()

  Get a list of towers that are currently in this tower's range.
- int **getTotalCost** ()

  Calculate the current total value of the tower.
- void **setRange** (int range)

  Set a new range of the tower, which changes the attack area accordingly.
- void **addCost** (int cost)

  Add to the tower's total cost.
- virtual bool **upgrade** ()=0
- virtual bool **isTower** ()

  Check if this is a tower or not.
- void **showAttackArea** ()

  Show the tower's attack area on the scene.
- void **hideAttackArea** ()

  Hide the tower's attack area on the scene.
- virtual void **showHideAttackArea** ()

  Show the tower's attack area if it is currently hidden, and hide the attack area if it is currently displayed.
- virtual void **updateDescription** ()

  Update the tower's text description base on the tower's current stats.

Public Attributes

- int **projectileStepSize_** = 6

Protected Attributes

- QString **type_**

  the type of the tower
- int **range_**

  the attack range of the tower
- double **damage_**

  the damage of the tower
- double **damageMultiplier_**

  the damage multiplier of the tower, can be use for buff mechanic
- double **attackInterval_**

  the interval between the tower's shot, in mili-seconds
- QTimer * **attackTimer_**

  timer for timing the tower's attack
- int **pierce_**

  how many enemies (after the first one) can the projectile of this tower pierce through
- int **maxLevel_**

the max upgrade level of the tower

- int **upgradeLevel_**
- int **totalCost_**

  the tower current total value
- int **rotationAngle_**

  the rotation angle of the tower
- QGraphicsPixmapItem * **towerImg**

  the tower's image in use
- QString **ogImagePath_**

  the original image path in use
- QString **projectileImagePath_**

  the tower's projectile image
- QGraphicsEllipseItem * **attack_area_**

  the physical attack area of the tower, represent by an ellipse item
- QString **description_**
- bool **canFire_**

  true if the tower can fire, false otherwise (ex: support towers)
- bool **targetAble_** [3]

  array for checking if an enemy is targetable, an enemy is targetable if the coresponding position for that enemy is true
- bool **targetAbleBuff_** [3]

  array for checking if this tower has been allowed to target an enemy by a support tower, logic to targetAble but is used for buffs

Additional Inherited Members

4.18.1 Detailed Description

Base class for all towers in the game, inherited from **Square** (p. 92).

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Tower() [1/2]

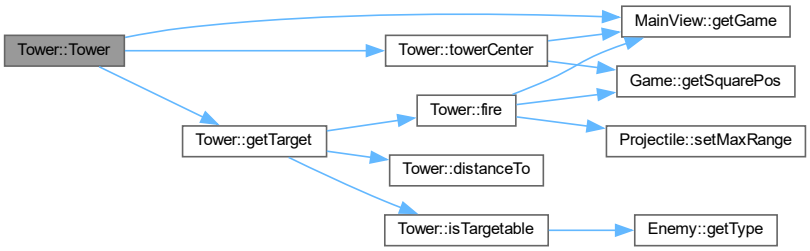
```
Tower::Tower (
            int row,
            int column,
            QWidget * parent = nullptr )
```

Tower (p. 101) basic constructor, build a default tower (CS Student) at the given position.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

Here is the call graph for this function:



4.18.2.2 Tower() [2/2]

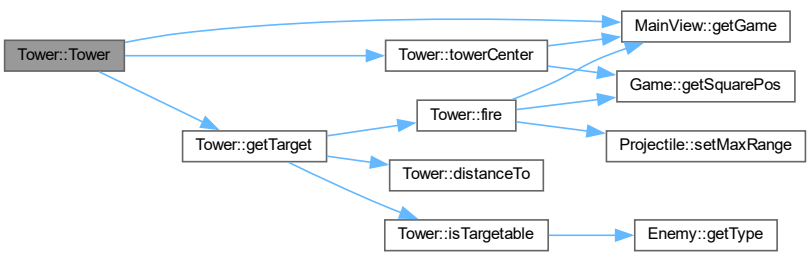
```
Tower::Tower (
            int row,
            int column,
            int range,
            int damage,
            int attackInterval,
            QWidget * parent = nullptr )
```

Tower (p. 101) specific constructor, initialize the member of the tower with given parameter.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
range	the attack range of the tower
damage	the damage of the tower
attackInterval	the attack interval of the tower, in mili-seconds
parent	the parent of this item

Here is the call graph for this function:



4.18.3 Member Function Documentation

4.18.3.1 addCost()

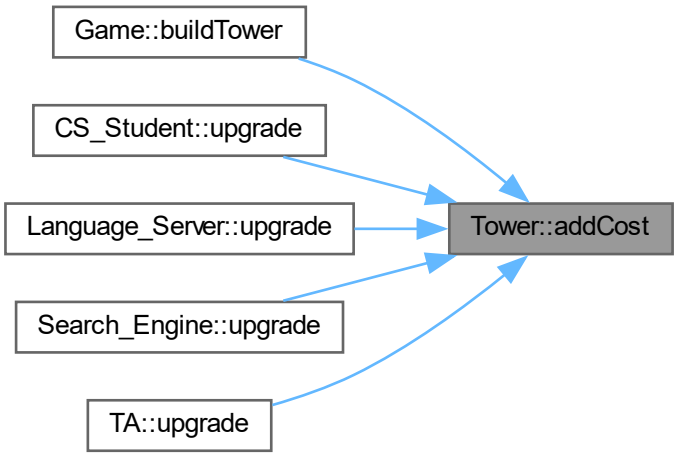
```
void Tower::addCost (
            int cost )
```

Add to the tower's total cost.

Parameters

cost	the cost to be added.

Here is the caller graph for this function:



4.18.3.2 atkSpeedBuff()

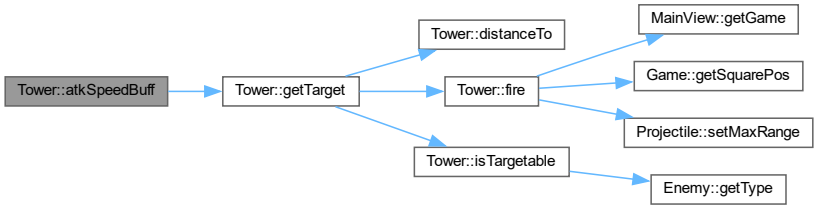
```
void Tower::atkSpeedBuff (
            double buffFactor )
```

Increase the tower's attack speed by the given buff factor.

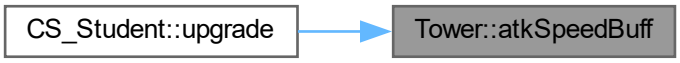
Parameters

buffFactor	the buff factor. A buff factor of 0.2 will coresponds to an increase of 20%.

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.3 atkSpeedDebuff()

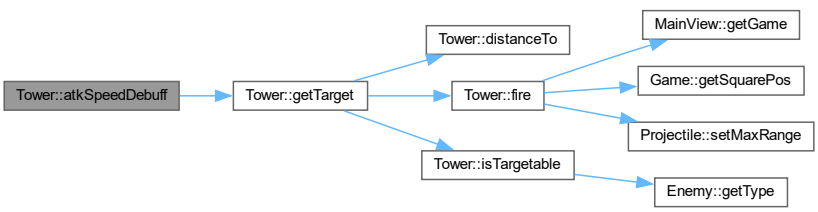
```
void Tower::atkSpeedDebuff (
            double debuffFactor )
```

Decrease the tower's attack speed by the given buff factor.

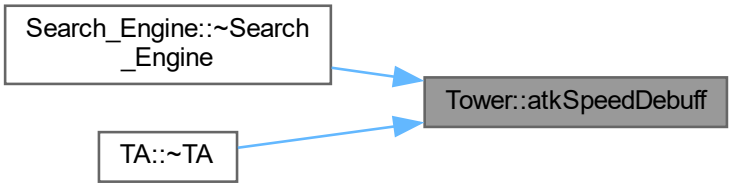
Parameters

buffFactor	the buff factor. A buff factor of 0.2 will coresponds to a decrease of 20%.

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.4 damageBuff()

```
void Tower::damageBuff (
            double buffFactor )
```

Increase the tower's damage by the given buff factor.

Parameters

buffFactor	the buff factor. A buff factor of 0.2 will coresponds to an increase of 20%.

4.18.3.5 distanceTo()

```
double Tower::distanceTo (
            QGraphicsItem * item )
```

Calculate the distance from this tower to the given item on the screen.

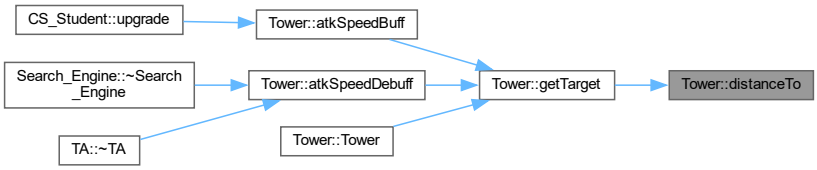
Parameters

item	a QGraphicsItem on the screen

Returns

the distance from this tower to the given item as a double

Here is the caller graph for this function:



4.18.3.6 fire()

```
void Tower::fire (
            QPointF targetPos )  [virtual]
```

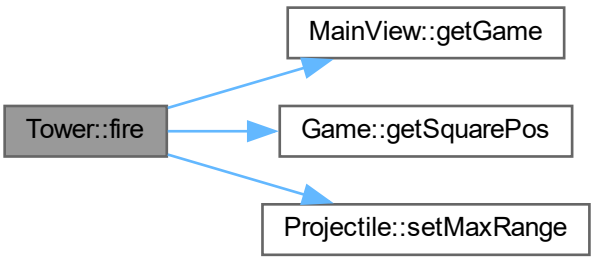
Fire a projectile at the given target position.

Parameters

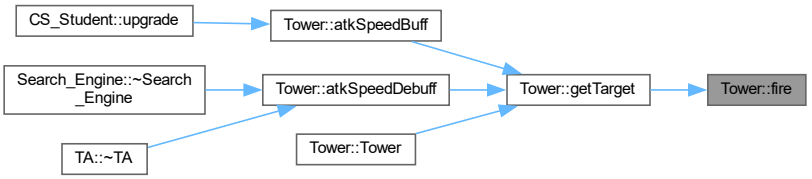
targetPos	the position of the target to fire at

Reimplemented in **Language_Server** (p. 64), **TA** (p. 99), and **Valgrind** (p. 120).

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.7 getAttackArea()

```
QGraphicsEllipseItem * Tower::getAttackArea ( )
```

Returns the tower's attack area.

Returns

a QGraphicsEllipseItem that represents the tower's current attack area

4.18.3.8 getItemInRange()

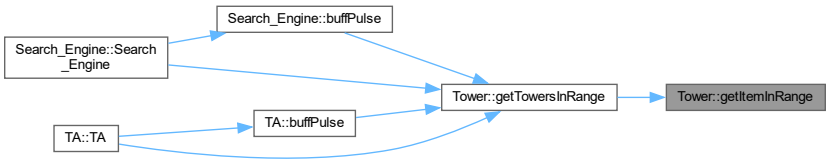
```
QList< QGraphicsItem * > Tower::getItemInRange ( )
```

Get a list of items that are currently in the tower's range.

Returns

a QList of QGraphicsItem* that contains every item that is in the tower's range

Here is the caller graph for this function:

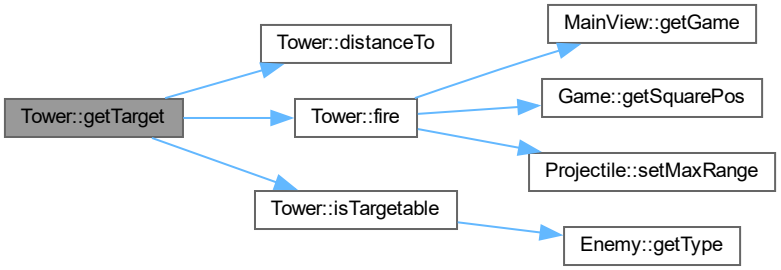


4.18.3.9 getTarget

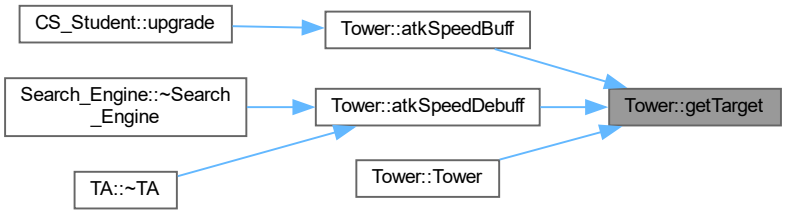
```
void Tower::getTarget ( )  [slot]
```

Get a target for the tower to shoot at.

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.10 getTotalCost()

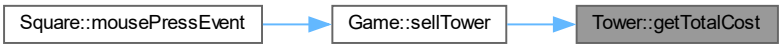
```
int Tower::getTotalCost ( )
```

Calculate the current total value of the tower.

Returns

the total value of the tower as an int

Here is the caller graph for this function:



4.18.3.11 getTowersInRange()

```
QList< Tower * > Tower::getTowersInRange ( )
```

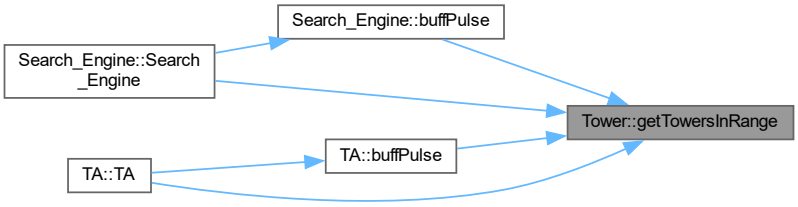
Get a list of towers that are currently in this tower's range.

Returns

a QList of Towers* that contains every tower that is in this tower's range

Here is the call graph for this function:

Here is the caller graph for this function:



4.18.3.12 hasAtkSpdBuff()

```
bool Tower::hasAtkSpdBuff ( )
```

Check if the tower has any attack speed buff or not.

Returns

true if the tower has attack speed buff, false otherwise

4.18.3.13 isTargetable()

```
bool Tower::isTargetable (
            Enemy * enemy )
```

Check if the tower can target the given enemy.

Parameters

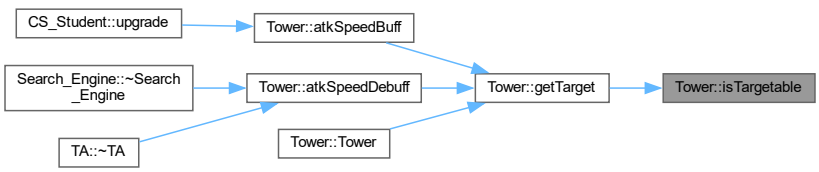
enemy	pointer to an **Enemy** (p. 18) class object

Returns

true if the enemy can be target by this tower, false otherwise

Here is the call graph for this function:

Here is the caller graph for this function:



4.18.3.14 isTower()

```
bool Tower::isTower ( )  [virtual]
```

Check if this is a tower or not.

Returns

true

Reimplemented from **Square** (p. 95).

4.18.3.15 setRange()

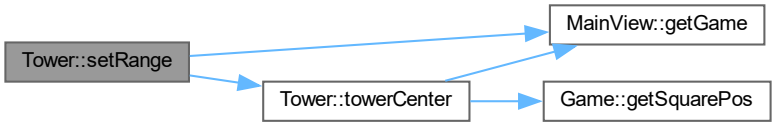
```
void Tower::setRange (
            int range )
```

Set a new range of the tower, which changes the attack area accordingly.

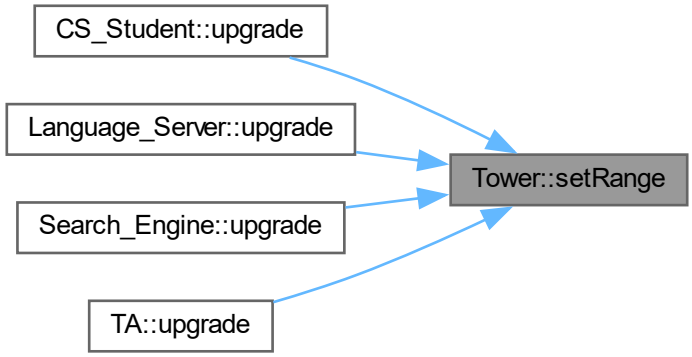
Parameters

range	the new range of the tower

Here is the call graph for this function:



Here is the caller graph for this function:



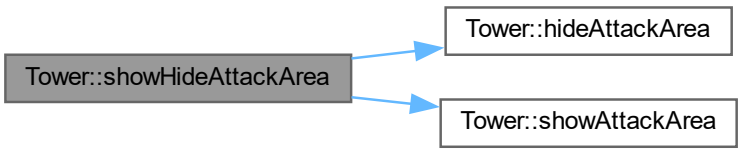
4.18.3.16 showHideAttackArea()

```
void Tower::showHideAttackArea ( )  [virtual]
```

Show the tower's attack area if it is currently hidden, and hide the attack area if it is currently displayed.

Reimplemented from **Square** (p. 92).

Here is the call graph for this function:



4.18.3.17 targetTableBuff()

```
void Tower::targetTableBuff (
            EnemyTypes::TYPES type )
```

Allow the tower to target the given enemy type.

Parameters

type	an enemy type

Here is the caller graph for this function:

4.18.3.18 targetTableDebuff()

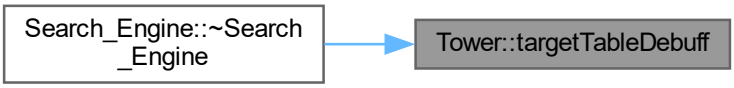
```
void Tower::targetTableDebuff (
            EnemyTypes::TYPES type )
```

Remove an existing targetable buff from the tower. This will not affect the tower's own targetable ability. In other words if the tower was able to target an enemy type without any buff, it will still be able to do so.

Parameters

type	an enemy type

Here is the caller graph for this function:



4.18.3.19 towerCenter()

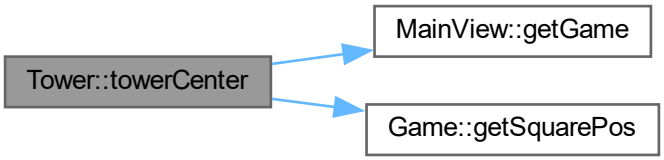
```
QPointF Tower::towerCenter ( )
```

Return the position of the tower's center on the screen.

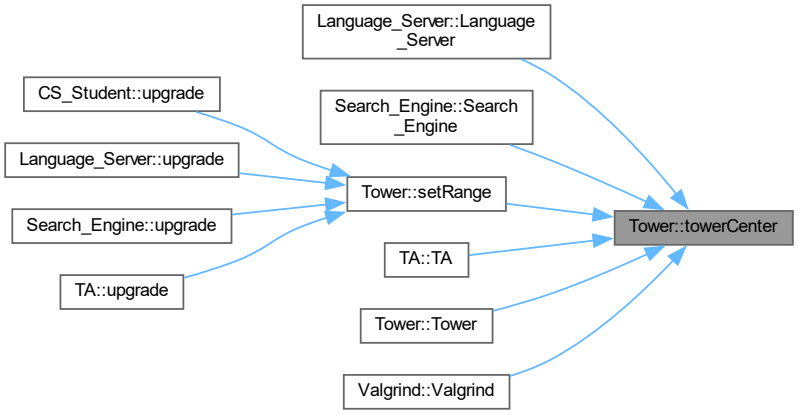
Returns

the tower's center as a QPointF item

Here is the call graph for this function:



Here is the caller graph for this function:



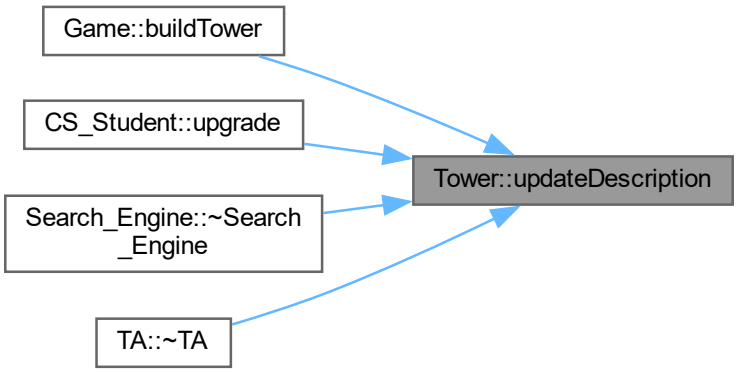
4.18.3.20 updateDescription()

```
void Tower::updateDescription ( )  [virtual]
```

Update the tower's text description base on the tower's current stats.

Reimplemented in **Language_Server** (p. 64), **Search_Engine** (p. 91), **TA** (p. 100), and **Valgrind** (p. 121).

Here is the caller graph for this function:



4.18.3.21 upgrade()

```
virtual bool Tower::upgrade ( )  [pure virtual]
```

Implemented in **CS_Student** (p. 17), **Language_Server** (p. 65), **Search_Engine** (p. 92), **TA** (p. 100), and **Valgrind** (p. 121).

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

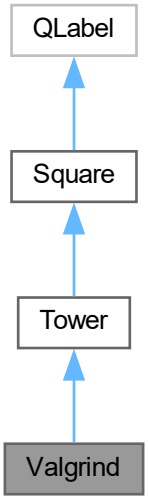
- **tower.h**
- tower.cpp

4.19 Valgrind Class Reference

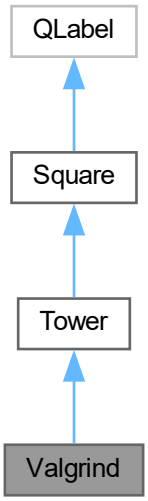
A **Valgrind** (p. 118) tower type. Can attack all enemies on the map. Inherited from **Tower** (p. 101) class.

```
#include <valgrind.h>
```

Inheritance diagram for Valgrind:



Collaboration diagram for Valgrind:



Public Member Functions

- **Valgrind** (int row, int column, QWidget ∗parent=nullptr)

  Valgrind *(*p. *118) constructor, create a* ***Valgrind*** *(*p. *118) tower at the given posion on the grid map.*

- virtual bool **upgrade** ()
- void **updateDescription** ()

  Update the tower's text description base on the tower's current stats.
- void **fire** (QPointF targetPos)

  Fire a projectile at the given target position.

Additional Inherited Members

4.19.1 Detailed Description

A **Valgrind** (p. 118) tower type. Can attack all enemies on the map. Inherited from **Tower** (p. 101) class.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 Valgrind()

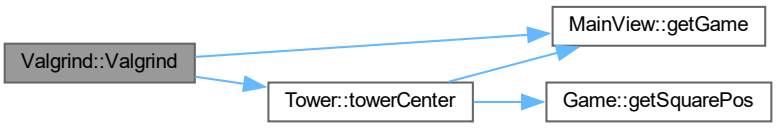
```
Valgrind::Valgrind (
            int row,
            int column,
            QWidget * parent = nullptr )
```

Valgrind (p. 118) constructor, create a **Valgrind** (p. 118) tower at the given posion on the grid map.

Parameters

row	the row of the tower in the game's grid map
column	the column of the tower in the game's grid map
parent	the parent of this item

Here is the call graph for this function:



4.19.3 Member Function Documentation

4.19.3.1 fire()

```
void Valgrind::fire (
            QPointF targetPos )  [virtual]
```

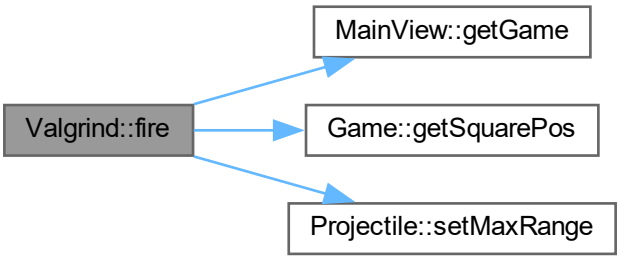
Fire a projectile at the given target position.

Parameters

targetPos	the position of the target to fire at

Reimplemented from **Tower** (p. 109).

Here is the call graph for this function:



4.19.3.2 updateDescription()

```
void Valgrind::updateDescription ( )  [virtual]
```

Update the tower's text description base on the tower's current stats.

Reimplemented from **Tower** (p. 117).

4.19.3.3 upgrade()

```
bool Valgrind::upgrade ( )  [virtual]
```

Valgrind (p. 118) does not have further level designed yet at the moment

Implements **Tower** (p. 101).

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

- **valgrind.h**
- valgrind.cpp

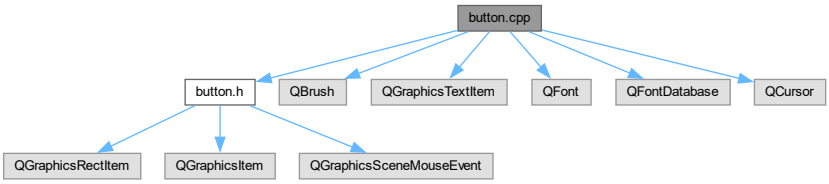
Chapter 5

File Documentation

5.1 button.cpp File Reference

```
#include "button.h"
#include <QBrush>
#include <QGraphicsTextItem>
#include <QFont>
#include <QFontDatabase>
#include <QCursor>
```

Include dependency graph for button.cpp:



5.1.1 Detailed Description

Author

Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

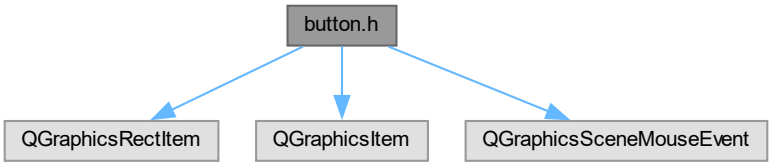
Copyright

Copyright (c) 2022

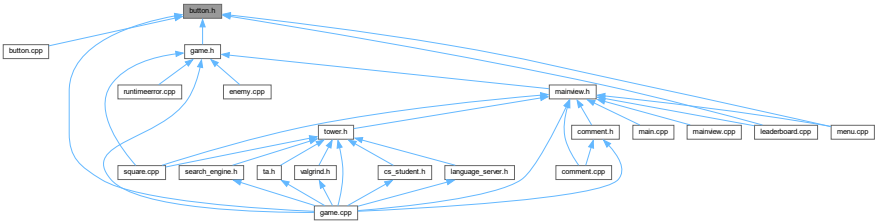
5.2 button.h File Reference

```
#include <QGraphicsRectItem>
#include <QGraphicsItem>
#include <QGraphicsSceneMouseEvent>
```

Include dependency graph for button.h:



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



Classes

- class **Button**

  This is the button object.

5.2.1 Detailed Description

Author

Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.3 button.h

Go to the documentation of this file.

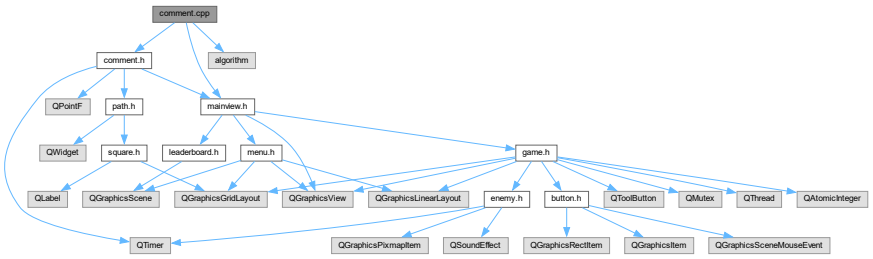
```
1
11 #ifndef BUTTON_H
12 #define BUTTON_H
13
14 #include <QGraphicsRectItem>
15 #include <QGraphicsItem>
16 #include <QGraphicsSceneMouseEvent>
17
22 class Button:public QObject, public QGraphicsRectItem{
23     Q_OBJECT
24 public:
25     Button(QString name, int w, int h, QColor backgroundColor = Qt::green
26             , QGraphicsItem* parent =NULL);
27
28     void mousePressEvent(QGraphicsSceneMouseEvent * event);
29     void hoverEnterEvent(QGraphicsSceneHoverEvent * event);
30     void hoverLeaveEvent(QGraphicsSceneHoverEvent * event);
31
32 signals:
33     void clicked();
34
35 private:
40     QGraphicsTextItem * text;
41
46     QColor backgroundColor_;
47 };
48
49 #endif // BUTTON_H
```

5.4 comment.cpp File Reference

File containing the comment tower class methods.

```
#include "comment.h"
#include "mainview.h"
#include <algorithm>
```

Include dependency graph for comment.cpp:



Variables

- **MainView * view**

5.4.1 Detailed Description

File containing the comment tower class methods.

Authors

Harvey Lim (harvey.lim@aalto.fi)

Version

0.1

Date

2022-12-11

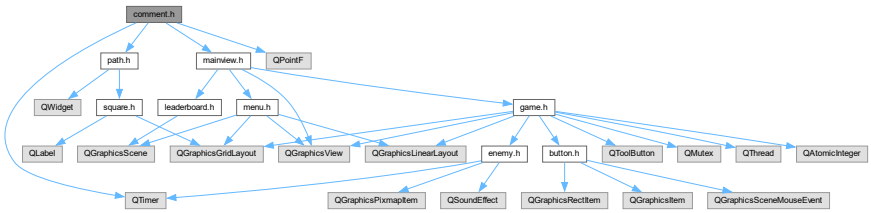
Copyright

5.5 comment.h File Reference

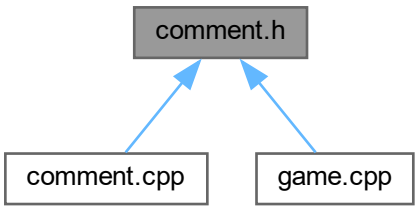
File containing the comment tower class and its members.

```
#include "path.h"
#include <QPointF>
#include <QTimer>
#include "mainview.h"
```

Include dependency graph for comment.h:



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



Classes

- class **Comment**

  *The **Comment** (p. 9) class. Represents the comment tower.*

Variables

- **MainView ∗ view**

5.5.1 Detailed Description

File containing the comment tower class and its members.

Authors

Harvey Lim (harvey.lim@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.6 comment.h

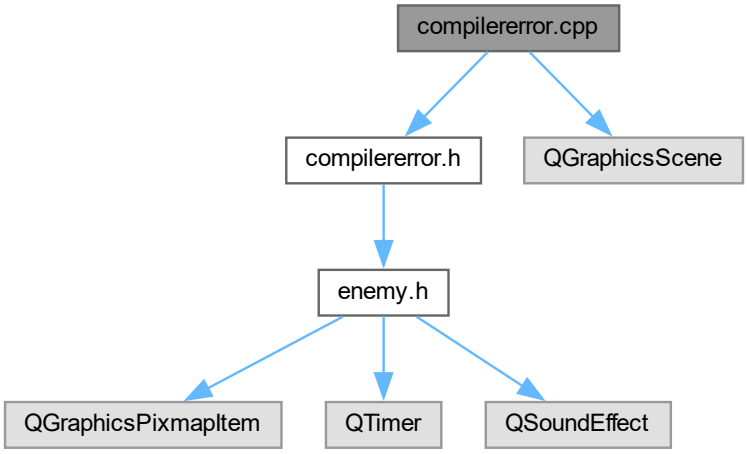
Go to the documentation of this file.

```
1
12 #ifndef COMMENT_H
13 #define COMMENT_H
14
15 #include "path.h"
16 #include <QPointF>
17 #include <QTimer>
18 #include "mainview.h"
19
20 extern MainView* view;
24 class Comment :  public Path {
25 public:
26     Comment(int x, int y, int duration, Path* old, QWidget* parent);
27     QPointF towerCenter();
28     Path* getOld();
29     ~Comment();
30     bool isTimerActive();
31     void startTimer();
32 public slots:
33     void breakComment();
34 private:
38     int duration_;
42     QGraphicsPixmapItem * towerImg;
46     Path* old_;
50     QTimer* breakdownTimer_;
54     int x_;
58     int y_;
59 };
60
61 #endif // COMMENT_H
```

5.7 compilererror.cpp File Reference

```
#include "compilererror.h"
#include <QGraphicsScene>
```

Include dependency graph for compilererror.cpp:



5.7.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

2022-12-11

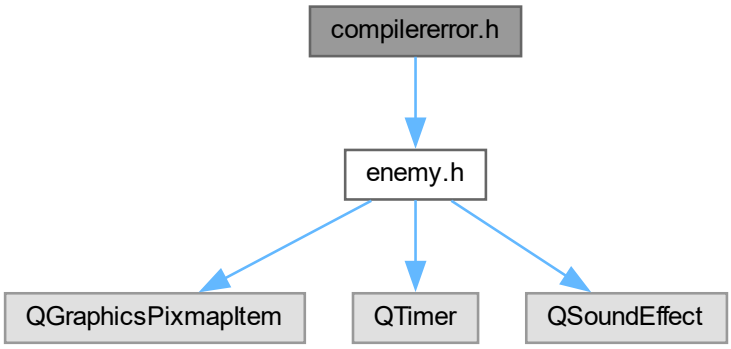
Copyright

Copyright (c) 2022

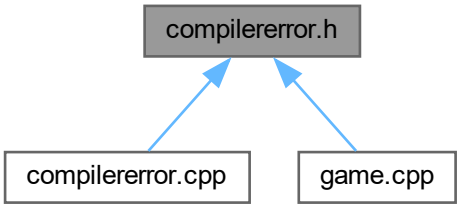
5.8 compilererror.h File Reference

```
#include "enemy.h"
```

Include dependency graph for compilererror.h:



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



Classes

- class **CompilerError**

  *Class for **CompilerError** (p. 13) type enemies Inherits from class **Enemy** (p. 18).*

5.8.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.9 compilererror.h

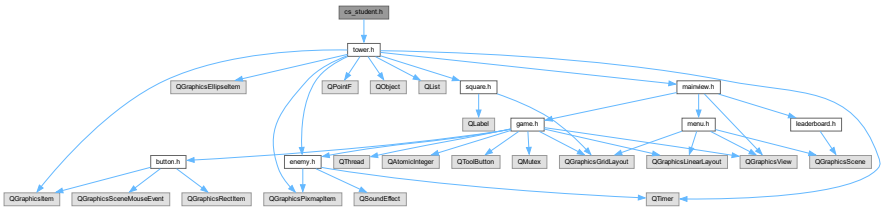
Go to the documentation of this file.

```
1
11 #ifndef COMPILERERROR_H
12 #define COMPILERERROR_H
13
14 #include "enemy.h"
20 class CompilerError:  public Enemy
21 {
22     Q_OBJECT
23 public:
24     CompilerError(CompilerErrorType subType, QList<QPointF> path,QList<QPoint> matrixPath);
25     void die();
26     void explodeException();
27
28 private:
33     CompilerErrorType name_;
34 };
35
36 #endif // COMPILERERROR_H
```

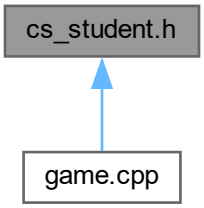
5.10 cs_student.h File Reference

`#include "tower.h"`

Include dependency graph for cs_student.h:



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



Classes

- class **CS_Student**

  *A CS Student tower type. Can attack enemies in range. Inherited from **Tower** (p. 101) class.*

5.10.1 Detailed Description

Authors

Hung Vu (hung.h.vu@aalto.fi), Saku (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.11 cs_student.h

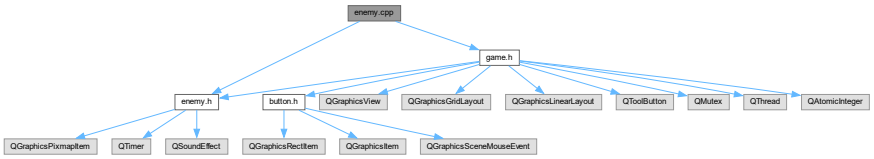
Go to the documentation of this file.

```
1
11 #ifndef CS_STUDENT_H
12 #define CS_STUDENT_H
13
14 #include "tower.h"
15
19 class CS_Student :  public Tower {
20 public:
21     CS_Student();
22     CS_Student(int row, int column, QWidget *parent=nullptr);
23     virtual bool upgrade();
24
25 };
26
27 #endif // CS_STUDENT_H
```

5.12 enemy.cpp File Reference

```
#include "enemy.h"
#include "game.h"
```

Include dependency graph for enemy.cpp:



5.12.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

2022-12-11

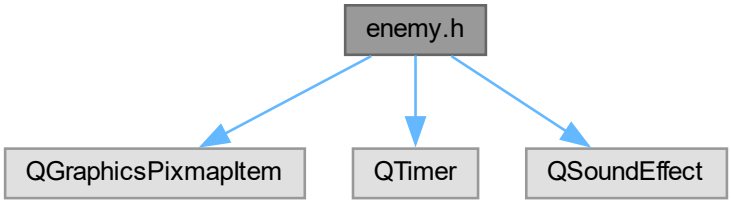
Copyright

Copyright (c) 2022

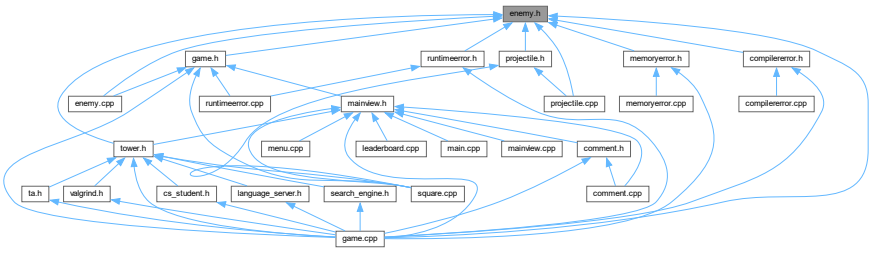
5.13 enemy.h File Reference

```
#include <QGraphicsPixmapItem>
#include <QTimer>
#include <QSoundEffect>
```

Include dependency graph for enemy.h:



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



Classes

- class **Enemy**

  Virtual class of the enemies. All enemies in the game are inherited from this class.

Enumerations

- enum class **EnemyType** { **CompilerError** =1 , **MemoryError** =2 , **RuntimeError** =3 }
- enum **CompilerErrorType** { **SyntaxError** =1 , **Exception** =2 }
- enum **MemoryErrorType** { **InvalidRead** =3 , **InvalidWrite** =4 , **XBytesAreLost** =5 , **MismatchedDeleteFree** =6 }
- enum **RuntimeErrorType** { **MemoryStackMinion** =8 , **StackOverflow** =7 }

5.13.1 Detailed Description

Author

Siim Kasela (`siim.kasela@aalto.fi`)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.14 enemy.h

Go to the documentation of this file.

```
1
11 #ifndef ENEMY_H
12 #define ENEMY_H
13
14
15 #include <QGraphicsPixmapItem>
16 #include <QTimer>
17 #include <QSoundEffect>
18
19 //enums of main and subtypes, could help with scaling
20
21 //main 3 types of enemies
22 //is the type_ variable
23 enum class EnemyType
24 {
25     CompilerError=1,
26     MemoryError=2,
27     RuntimeError=3
28 };
29 //subtypes of enemies
30 //is the name_ variable of the enemies
31 enum CompilerErrorType
32 {
33     SyntaxError=1,
34     Exception=2
35 };
36 enum MemoryErrorType
37 {
38     InvalidRead=3,
39     InvalidWrite=4,
40     XBytesAreLost=5,
41     MismatchedDeleteFree=6
42 };
43 enum RuntimeErrorType
44 {
45     MemoryStackMinion=8,
46     StackOverflow=7
47 };
48
53 class Enemy:  public QObject, public QGraphicsPixmapItem
54 {
55     Q_OBJECT
56 public:
57     Enemy(EnemyType type, QList<QPointF> path,QList<QPoint> matrixPath, int health = 0, int damage = 0,
      int speed = 0, QGraphicsItem * parent=0);
58
59     virtual void attack(){}
60     virtual void die();
61     virtual void takeDamage(int damage);
62     void setSpeed(int speed);
63     void startMove();
64     void setPath(QList<QPoint> matrixPath, QList<QPointF> path);
65     QPoint getMatrixLocation() const;
66     QTimer *getTimer ();
67     QPoint getNextLocation() const;
68
69     EnemyType getType() const;
70
71 public slots:
72
73     void move();
74
75 signals:
76
77     void enemyDies(int);
78     void dealsDamage(int);
79     void addedEnemy(Enemy*);
80
81 protected:
86     int health_;
91     int damage_;
96     int speed_;
101     int pointValue_;
106     QList<QPointF> path_;
111     QList<QPoint> matrixPath_;
116     QPointF dest_;
121     int point_index_;
126     EnemyType type_;
131     QTimer* timer_;
132
133 };
134
135 #endif // ENEMY_H
```

5.15 game.cpp File Reference

```
#include "game.h"
#include "square.h"
#include "mainview.h"
#include "button.h"
#include <QGraphicsScene>
#include <QGraphicsGridLayout>
#include <QGraphicsPixmapItem>
#include <QLabel>
#include <QGraphicsProxyWidget>
#include <QGraphicsWidget>
#include <QGraphicsTextItem>
#include <QToolButton>
#include "enemy.h"
#include "tower.h"
#include "compilererror.h"
#include "memoryerror.h"
#include "runtimeerror.h"
#include "cs_student.h"
#include "search_engine.h"
#include "language_server.h"
#include "ta.h"
#include "valgrind.h"
#include "path.h"
#include "comment.h"
#include <QQueue>
#include <QSet>
#include <QPoint>
#include <QDataStream>
#include <QFile>
#include <QIcon>
#include <QScrollBar>
#include <QTextStream>
#include <QRegularExpression>
#include <QTimer>
#include <QMessageBox>
#include <QInputDialog>
#include <QDir>
#include <QStandardPaths>
#include <QRandomGenerator>
```

Include dependency graph for game.cpp:

Macros

- #define **BUILD_BUTTON_SIZE** 95
- #define **CS_COST** 50
- #define **TA_COST** 100
- #define **SE_COST** 250
- #define **COM_COST** 20

- #define **LS_COST** 120
- #define **VAL_COST** 200
- #define **SELL_PENALTY** 0.3

Functions

- QString **CS_DESCRIPTION** (" CS student:
" "A student on his quest to complete his CS degree. In his first year he had a hard time identifying memory errors." " He has somewhat mediocre ability at first, but as he advances through his degree," " he soon become a valuable member of the debugging force.
" "Shortcut: C ")
- QString **TA_DESCRIPTION** (" Teaching Assistant:
" "A Teaching Assistant who is rather experienced at his field." " With his hints, other members within his reach will have higher commit rate." " After having enough work experience, he can be promoted to a Teacher, with higher patches per commit" " and better reach. Teacher also have the ability to spot a major error type that **TA** cannot.
" "Shortcut: T ")
- QString **SE_DESCRIPTION** (" Search Engine:
" "A crucial tool for any programmer. It was recently found in a study that Search Engine can increase" " a programmer's commit rate up to 10%. Not all Search Engine are equal however," " some are just worse than others.
" "By using a good Search Engine (that start with a G), even an amatuer programmer could solve memory errors.
" "Shortcut: E or B ")
- QString **COM_DESCRIPTION** (" Comment:
" "\"Out of sight, out of mind\", a programmer can choose to comment a block of error away and deal with it later." " But everything has a deadline and the errors need to be dealt with enventually, though." " The errors block will be uncommented after a while and continue its way to wreak havoc on the project.
" "Shortcut: O or \\ ")
- QString **LS_DESCRIPTION** (" Language Server:
" "\"The Language Server Protocol (LSP) is an open, JSON-RPC-based protocol for...\". Yeah whatever, it can catch" " some bugs for you, very cool. Language Server can be more effective than humans at handling minor bugs," " but some special errors can only be manually dealt with. Unless you have a" " really good Language Server, of course.
" "Shortcut: L ")
- QString **VAL_DESCRIPTION** (" Valgrind:
" "A programmer's favorite tool for dealing with memory problems. **Valgrind** can search far and wide for bugs," " some might even say that nothing can escape its scope. It does take a bit long for **Valgrind** to detect" " something, but when it does, that bug is almost as good as dead.
" "Shortcut: V ")

5.15.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi), Harvey Lim (harvey.lim@aalto.fi), Hung Vu (hung.h.vu@aalto.fi)

Version

0.1

Date

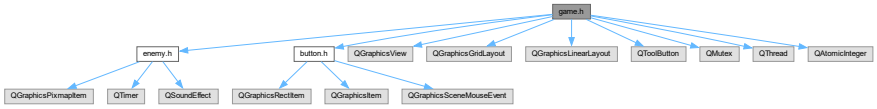
2022-12-11

Copyright

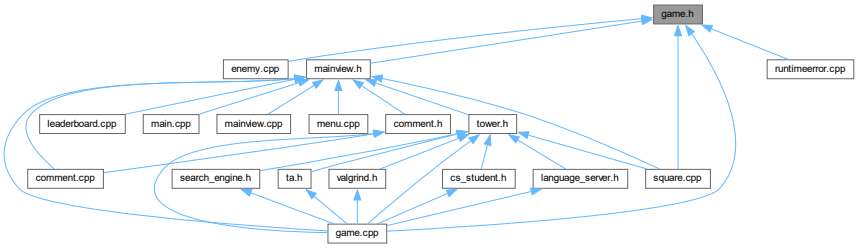
5.16 game.h File Reference

```
#include "enemy.h"
#include "button.h"
#include <QGraphicsView>
#include <QGraphicsGridLayout>
#include <QGraphicsLinearLayout>
#include <QToolButton>
#include <QMutex>
#include <QThread>
#include <QAtomicInteger>
```

Include dependency graph for game.h:



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



Classes

- class **Game**

  The class holding everything related to the actual tower defence gameplay.

Enumerations

- enum **MODES** {
  normal , **build** , **upgrade** , **sell** ,
  exit }
- enum **TYPES** {
  CS_Student , **TA** , **SearchEngine** , **LanguageServer** ,
  Valgrind , **Comment** }

5.16.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi), Harvey Lim (harvey.lim@aalto.fi), Hung Vu (hung.h.vu@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.17 game.h

Go to the documentation of this file.

```
1
11 #ifndef GAME_H
12 #define GAME_H
13
14 #include "enemy.h"
15 #include "button.h"
16 #include <QGraphicsView>
17 #include <QGraphicsGridLayout>
18 #include <QGraphicsLinearLayout>
19 #include <QToolButton>
20 #include <QMutex>
21 #include <QThread>
22 #include <QAtomicInteger>
23
24 // enumeration to keep track of the game's mode
25 namespace Modes{
26     enum MODES{normal, build, upgrade, sell, exit};
27 }
28
29 // enumeration to keep track of tower types for build
30 namespace TowerTypes{
31     enum TYPES{CS_Student, TA, SearchEngine, LanguageServer, Valgrind, Comment};
32 }
33
37 class Game:  public QGraphicsScene
38 {
39     Q_OBJECT
40     QThread thread;
41 public:
42
43     Game(QObject* parent, int gamemode = 0);
44     QPointF getSquarePos(int row, int column);
45     QWidget* getWidgetAt(int row, int column);
46     bool buildTower(int row, int column);
47     bool buildTower(int row, int column, TowerTypes::TYPES type);
48     bool sellTower(int row, int column);
49     bool upgradeTower(int row, int column);
50
51     bool isLost() const;
52     bool isWon() const;
53     bool isWaveWon();
54     void createMap();
55     void createGameControls();
56     void readWaveFile();
57
58     void playHitsound();
59     void playClicksound();
60
```

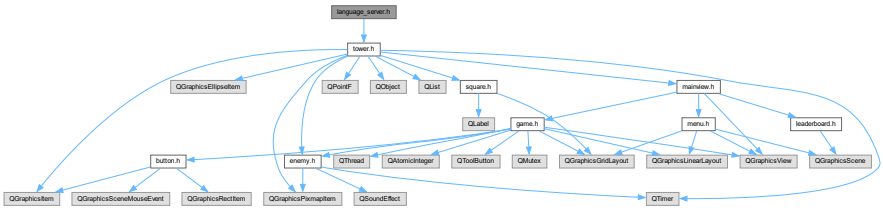
```
61      QList<QPointF> convertCoordinates(QList<QPoint> path);
62      QList<QPoint> getShortestPath(QPoint start);
63      QList<QPoint> BFS(QPoint start, bool blocked);
64
65      void breakComment(int row, int column);
66      void deleteComment(int row, int column);
67
76      int getGamemode()const {return gamemode_;};
77
78      int getHealth() const;
79      int getScore() const;
80      int getLevel() const;
81      int getCurrency() const;
82      int getEnemyCount() const;
83      Modes::MODES getMode() const;
84      TowerTypes::TYPES getBuildType() const;
85
86
87      void changeScore(int points);
88      void changeCurrency(int dCurrency);
89      void setMode(Modes::MODES m);
90      void advanceLevel();
91      void addSpawnedEnemies(int);
92
93      void keyPressEvent(QKeyEvent *keyEvent);
94      void resetButtonHighlights();
95
96      void hideAllAttackAreasExcept(QPointF exclude = QPointF(-1, -1));
97
98      QGraphicsGridLayout* mapLayout; //map area where the action is
99      QGraphicsLinearLayout* gameLayout; //the whole are of the game, including the controls
100      QGraphicsGridLayout* controlsLayout;//change this to your liking
101
102      bool isTower(int row, int column);
103      bool isPath(int row, int column);
104      bool isPathEnd(int row, int column);
105      bool isComment(int row, int column);
106      bool isEnemy(int row, int column);
107  public slots:
108      void showMenu();
109      void enterUpgradeMode();
110      void enterBuildCS();
111      void enterBuildTA();
112      void enterBuildSE();
113      void enterBuildLS();
114      void enterBuildVal();
115      void enterSellMode();
116      void enemyDies(int value);
117      void spawnEnemy(int type);
118      void updateLeaderboard();
119      void showError(QString message);
120      void addEnemy(Enemy*);
121      void takeDamage(int dHealth);
122      void enterBuildCom();
123      void updatePaths();
124      void updateEnemyCount();
125      void stopEnemies();
126
127      void createWave();
128      void startGame();
129
130  signals:
131      void gameWon();
132      void gameLost();
133      void waveWon();
134      void error(QString);
135      void wallAction();
136
137
138  private:
139      QPoint start_;
140      QPoint end_;
144      QList<QList<QString>> map_;
148      bool isBlocked_;
149
154      int gamemode_; // 0 = sandbox, 1 = easy, 2 = hard
155
156      int health_;
157      int currency_;
158      //int time_;
159      int wavesEnemyCount_;
160      QAtomicInteger<int> spawnedThisWave_;
161      int enemyCount_;
162      int level_;
163      int finalLevel_;
164      int score_;
169      double incomeMultiplier_ = 1.0;
```

```
174     double initialIncomeMultiplier_ = 2.0;
175     QList<QPoint> path_;
176     QList<QStringList> waves_;
177     QList<Enemy*> activeEnemies_;
178
182     QSoundEffect hitsound1_;
186     QSoundEffect hitsound2_;
190     QSoundEffect hitsound3_;
194     QSoundEffect hitsound4_;
198     QSoundEffect clicksound_;
199
200     Modes::MODES mode_;
201     TowerTypes::TYPES buildType_;
205     QList<QPoint> shortest_path_;
206
211     QGraphicsTextItem * roundDisplay;
216     QGraphicsTextItem * scoreDisplay;
221     QGraphicsTextItem * healthDisplay;
226     QGraphicsTextItem * wealthDisplay;
231     Button * upgradeButton;
236     Button * sellButton;
241     Button * nextRoundButton;
246     QToolButton* build_CSstudent;
251     QToolButton* build_TA;
256     QToolButton* build_SE;
261     QToolButton* build_LS;
266     QToolButton* build_Valgrind;
271     QToolButton* build_Comment;
276     QString buildButtonStylesheet;
281     QList<QPointF> coordsOfTowers;
282
283     void addLabelTo(QGraphicsLinearLayout* layout, QString towerName, QString stylesheet);
284 };
285
286 #endif // GAME_H
```

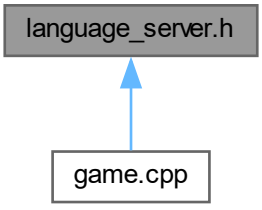
5.18 language_server.h File Reference

`#include "tower.h"`

Include dependency graph for language_server.h:



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



Classes

- class **Language_Server**

  A Language Server tower type. Can attack all enemies in range at the same time. Inherited from ***Tower*** *(p. 101) class.*

5.18.1 Detailed Description

Authors

Hung Vu (hung.h.vu@aalto.fi), Saku (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.19 language_server.h

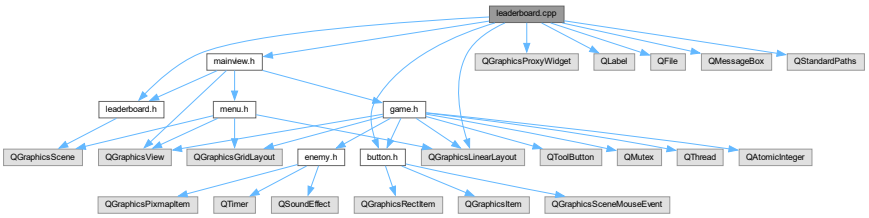
Go to the documentation of this file.

```
1
11 #ifndef LANGUAGE_SERVER_H
12 #define LANGUAGE_SERVER_H
13
14 #include "tower.h"
15
19 class Language_Server :  public Tower {
20 public:
21     Language_Server();
22     Language_Server(int row, int column, QWidget *parent=nullptr);
23
24     void fire(QPointF targetPos);
25     virtual bool upgrade();
26     void updateDescription();
27 };
28
29 #endif // LANGUAGE_SERVER_H
```

5.20 leaderboard.cpp File Reference

```
#include "leaderboard.h"
#include "mainview.h"
#include "button.h"
#include <QGraphicsLinearLayout>
#include <QGraphicsProxyWidget>
#include <QLabel>
#include <QFile>
#include <QMessageBox>
```

```
#include <QStandardPaths>
```
Include dependency graph for leaderboard.cpp:



5.20.1 Detailed Description

Authors

Siim (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

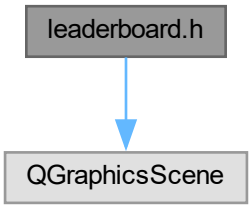
Date

2022-12-11

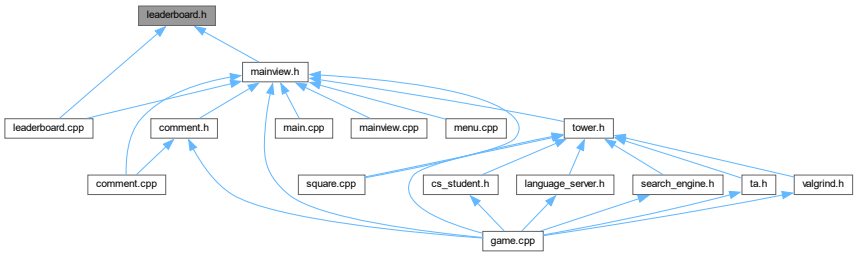
Copyright

5.21 leaderboard.h File Reference

```
#include <QGraphicsScene>
```
Include dependency graph for leaderboard.h:



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



Classes

- class **Leaderboard**

  This is the leaderboard scene.

5.21.1 Detailed Description

Authors

Siim (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.22 leaderboard.h

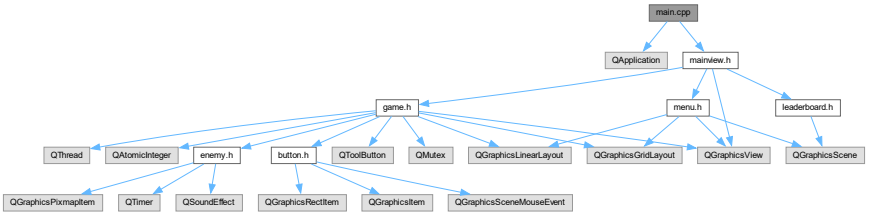
Go to the documentation of this file.

```
1
11 #ifndef LEADERBOARD_H
12 #define LEADERBOARD_H
13
14 #include <QGraphicsScene>
15
20 class Leaderboard:  public QGraphicsScene
21 {
22     Q_OBJECT
23 public:
24     Leaderboard(QObject* parent  = 0);
25     void setLeaderBoard(QList<QPair<QString,int>> leaderboard);
26     void readFile();
27
28 public slots:
29     void showMenu();
30
31 private:
36     QList<QPair<QString,int>> leaderboard_;
37
38 };
39
40 #endif // LEADERBOARD_H
```

5.23 main.cpp File Reference

```
#include <QApplication>
#include "mainview.h"
```

Include dependency graph for main.cpp:



Functions

- int **main** (int argc, char ∗argv[ ])

Variables

- **MainView** ∗ **view**

5.23.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi), Harvey Lim (harvey.lim@aalto.fi), Hung Vu (hung.h.vu@aalto.fi)

Version

0.1

Date

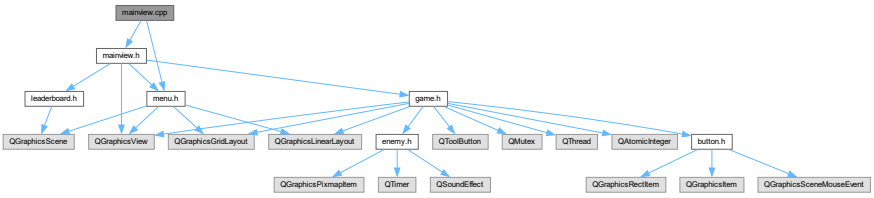
2022-12-11

Copyright

5.24 mainview.cpp File Reference

```
#include "mainview.h"
#include "menu.h"
```

Include dependency graph for mainview.cpp:



5.24.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

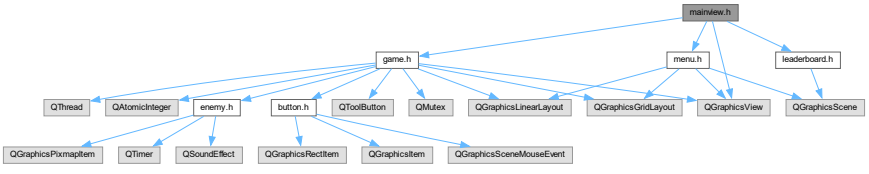
Copyright

Copyright (c) 2022

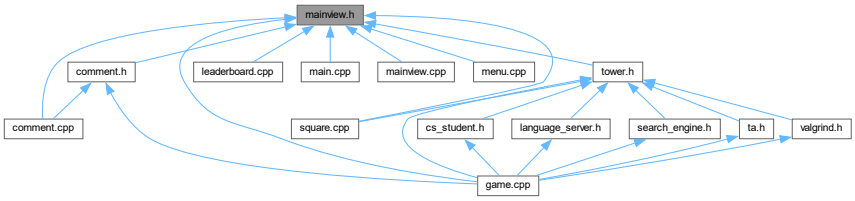
5.25 mainview.h File Reference

```
#include "game.h"
#include "menu.h"
#include "leaderboard.h"
#include <QGraphicsView>
```

Include dependency graph for mainview.h:



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



Classes

- class **MainView**

  *Class for representing the View that contains the **Menu** (p. 75), **Game** (p. 28) and **Leaderboard** (p. 66).*

5.25.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.26 mainview.h

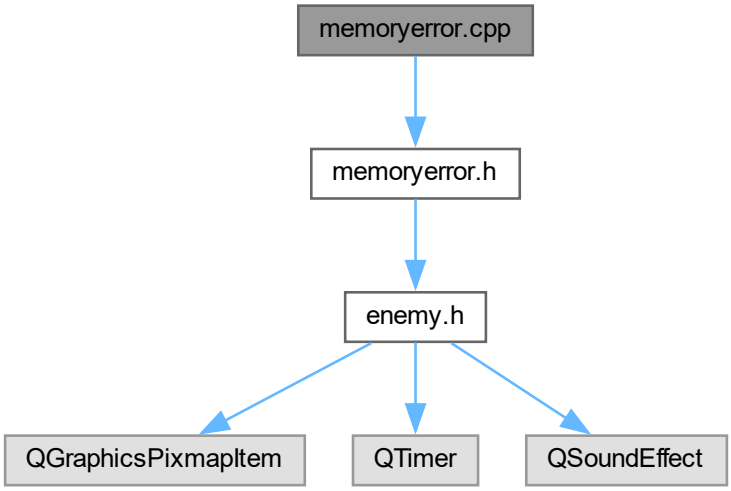
Go to the documentation of this file.

```
1
11 #ifndef MAINVIEW_H
12 #define MAINVIEW_H
13
14 #include "game.h"
15 #include "menu.h"
16 #include "leaderboard.h"
17
18 #include <QGraphicsView>
19
24 class MainView :    public QGraphicsView
25 {
26     Q_OBJECT
27
28 public:
29     MainView(QWidget *parent = nullptr);
30
31     Game* getGame();
```

```
32     Menu* getMenu();
33     Leaderboard* getLeaderboard();
34 public slots:
35     void showGame(int gamemode); // 0 = sandbox, 1 = easy, 2 = hard
36     void showMenu();
37     void showLeaderboard();
38
39
40 private:
45     Game* game_ = nullptr;
50     Menu* menu_;
55     Leaderboard* leaderboard_;
56 };
57
58 #endif // MAINVIEW_H
```

5.27 memoryerror.cpp File Reference

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



5.27.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

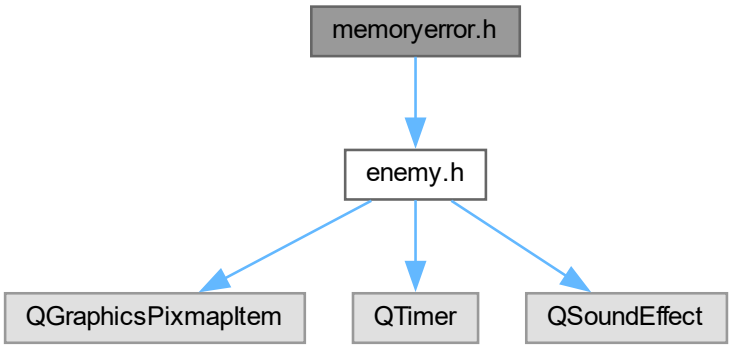
2022-12-11

Copyright

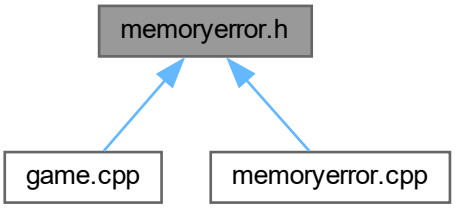
Copyright (c) 2022

5.28 memoryerror.h File Reference

`#include "enemy.h"`
Include dependency graph for memoryerror.h:



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



Classes

- class **MemoryError**

  *Class for the **MemoryError** (p. 74) type enemy.*

5.28.1 Detailed Description

Author

Siim Kasela (`siim.kasela@aalto.fi`)

Version

0.1

Date

2022-12-11

Copyright

5.29 memoryerror.h

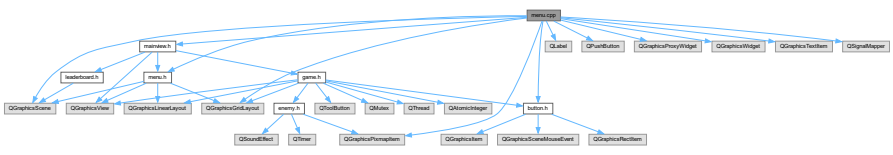
Go to the documentation of this file.

```
1
11 #ifndef MEMORYERROR_H
12 #define MEMORYERROR_H
13
14 #include "enemy.h"
19 class MemoryError :public Enemy
20 {
21     Q_OBJECT
22 public:
23
24     MemoryError(MemoryErrorType subType, QList<QPointF> path, QList<QPoint> matrixPath);
25
26 private:
31     MemoryErrorType name_;
32 };
33
34 #endif // MEMORYERROR_H
```

5.30 menu.cpp File Reference

```
#include "menu.h"
#include "mainview.h"
#include "button.h"
#include <QGraphicsScene>
#include <QGraphicsGridLayout>
#include <QGraphicsPixmapItem>
#include <QLabel>
#include <QPushButton>
#include <QGraphicsProxyWidget>
#include <QGraphicsWidget>
#include <QGraphicsTextItem>
#include <QSignalMapper>
```

Include dependency graph for menu.cpp:



5.30.1 Detailed Description

Authors

Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

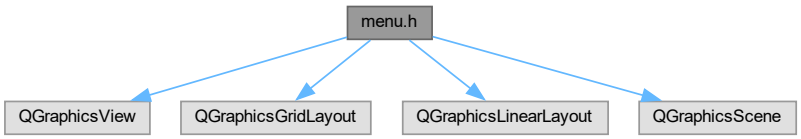
2022-12-11

Copyright

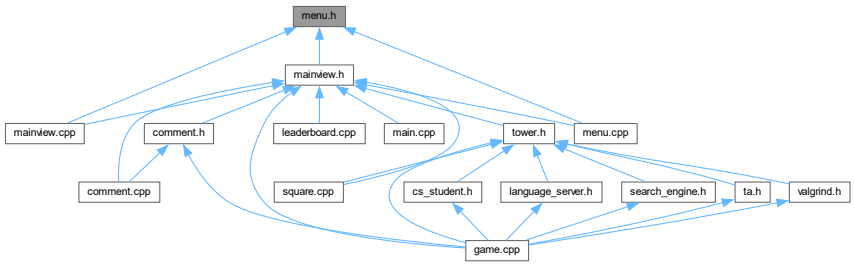
5.31 menu.h File Reference

```
#include <QGraphicsView>
#include <QGraphicsGridLayout>
#include <QGraphicsLinearLayout>
#include <QGraphicsScene>
```

Include dependency graph for menu.h:



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



Classes

- class **Menu**

  This is the menu scene.

5.31.1 Detailed Description

Authors

Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.32 menu.h

Go to the documentation of this file.

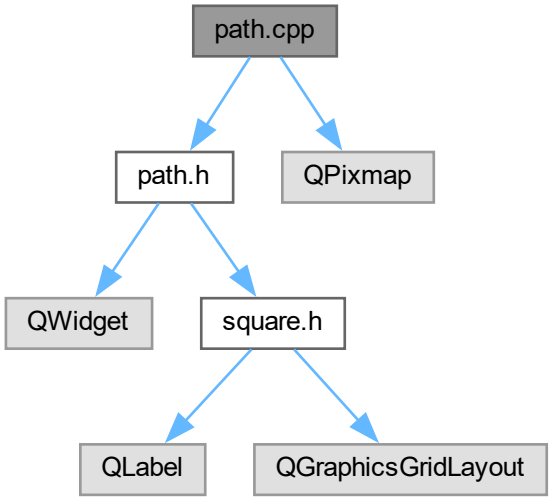
```
1
11 #ifndef MENU_H
12 #define MENU_H
13 #include <QGraphicsView>
14 #include <QGraphicsGridLayout>
15 #include <QGraphicsLinearLayout>
16 #include <QGraphicsScene>
17
22 class Menu :  public QGraphicsScene
23 {
24     Q_OBJECT
25 public:
26     Menu(QObject *parent = nullptr);
27 public slots:
28     void showGame(int gamemode); // 0 = sandbox, 1 = easy, 2 = hard
29     void showLeaderboard();
30     void quit();
31 private:
32 };
33
34 #endif // MENU_H
```

5.33 path.cpp File Reference

File containing the path tile methods.

```
#include "path.h"
#include <QPixmap>
```

Include dependency graph for path.cpp:



5.33.1 Detailed Description

File containing the path tile methods.

Authors

Harvey Lim (harvey.lim@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

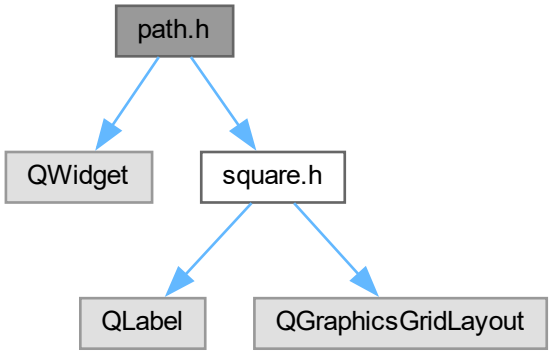
Copyright (c) 2022

5.34 path.h File Reference

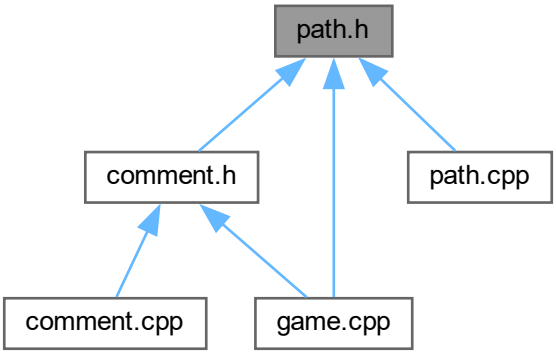
File containing the path tile class definition and its members.

```
#include <QWidget>
#include "square.h"
```

Include dependency graph for path.h:



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



Classes

- class **Path**

  *The **Path** (p. 79) class. Represents the path tiles on which enemies travel on.*

Enumerations

- enum **PathType** {
  Straight , **Turn** , **TSplit** , **XSplit** ,
  Start , **End** , **CommentType** }

5.34.1 Detailed Description

File containing the path tile class definition and its members.

Authors

Harvey Lim (harvey.lim@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.35 path.h

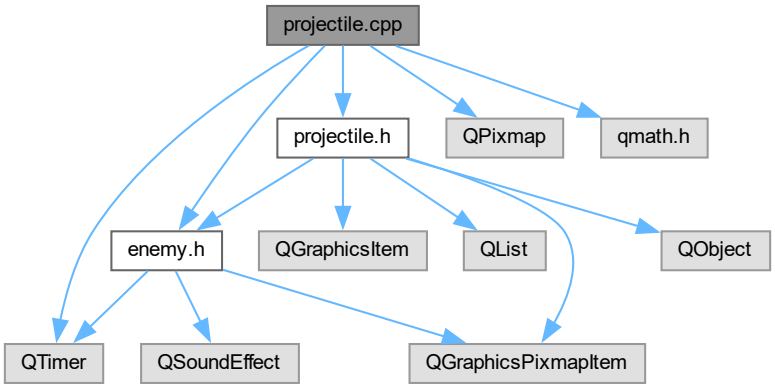
Go to the documentation of this file.

```
1
12 #ifndef PATH_H
13 #define PATH_H
14
15 #include <QWidget>
16 #include "square.h"
17
18 enum PathType {
19     Straight, Turn, TSplit, XSplit, Start, End, CommentType
20 };
21
25 class Path :  public Square {
26 public:
27     Path(int x, int y, PathType type, int rotation, Path* old = nullptr, QWidget* parent = nullptr);
28     PathType getType();
29     int getRotation();
30 private:
34     bool isComment_;
38     Path* old_;
42     PathType type_;
46     int rotation_;
47 };
48
49 #endif // PATH_H
```

5.36 projectile.cpp File Reference

```
#include "projectile.h"
#include "enemy.h"
#include <QPixmap>
#include <QTimer>
#include <qmath.h>
```

Include dependency graph for projectile.cpp:



5.36.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

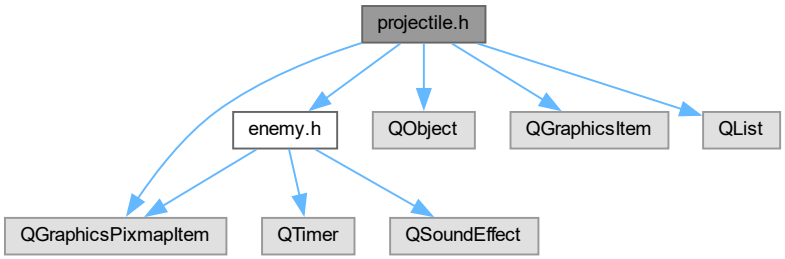
Copyright

Copyright (c) 2022

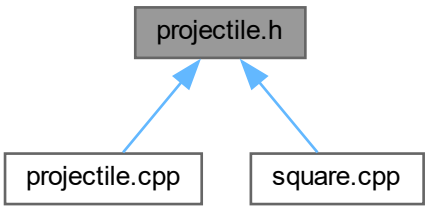
5.37 projectile.h File Reference

```
#include "enemy.h"
#include <QObject>
#include <QGraphicsPixmapItem>
#include <QGraphicsItem>
#include <QList>
```

Include dependency graph for projectile.h:



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



Classes

- class **Projectile**

  This is the projectile object.

5.37.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.38 projectile.h

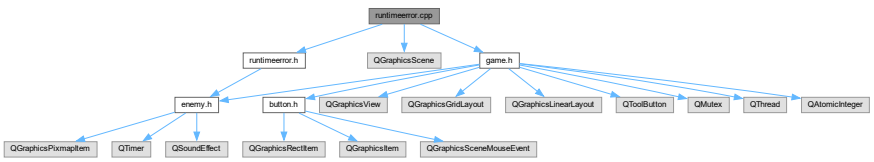
Go to the documentation of this file.

```
1
11 #ifndef PROJECTILE_H
12 #define PROJECTILE_H
13
14 #include "enemy.h"
15 #include <QObject>
16 #include <QGraphicsPixmapItem>
17 #include <QGraphicsItem>
18 #include <QList>
19
24 class Projectile :  public QObject, public QGraphicsPixmapItem
25 {
26     Q_OBJECT
27 public:
28     Projectile(int damage, QString imgPath = ":/images/CStudent_projectile.png",
29                int pierce = 0, int stepSize = 6, int maxLifeTime = 100000, QGraphicsItem * parent=0);
30     double getMaxRange();
31     double getDistanceTravelled();
32     void setMaxRange(double rng);
33     void setDistanceTravelled(double dist);
34     void setMoveFrequency(int newValue) {moveInterval_=newValue;};
35
36 public slots:
37     void move();
38
39 private:
44     double maxRange_;
49     double distanceTravelled_;
54     int damage_;
59     int pierce_;
64     int pierceCount_;
69     QList<Enemy*> enemiesHit_;
74     int stepSize_;
79     int maxLifetime_;
84     QTimer* lifeTimer_;
89     int moveInterval_ = 10;
90 };
91
92 #endif // PROJECTILE_H
```

5.39 runtimeerror.cpp File Reference

```
#include "runtimeerror.h"
#include <QGraphicsScene>
#include "game.h"
```

Include dependency graph for runtimeerror.cpp:



5.39.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

2022-12-11

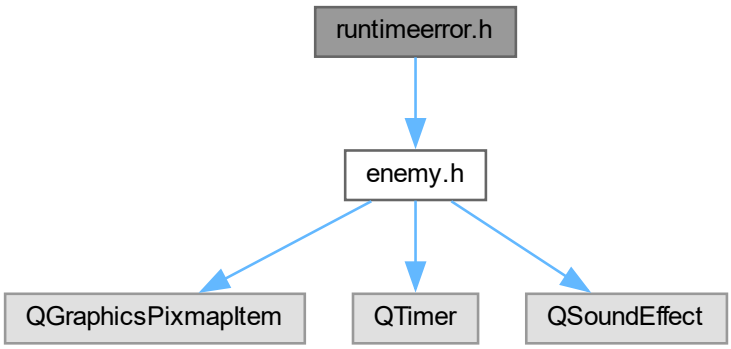
Copyright

Copyright (c) 2022

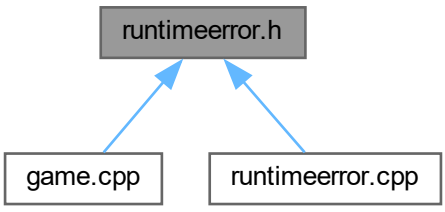
5.40 runtimeerror.h File Reference

```
#include "enemy.h"
```

Include dependency graph for runtimeerror.h:



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



Classes

- class **RuntimeError**

  *Class for the **RuntimeError** (p. 86) type enemies.*

5.40.1 Detailed Description

Author

Siim Kasela (siim.kasela@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.41 runtimeerror.h

Go to the documentation of this file.

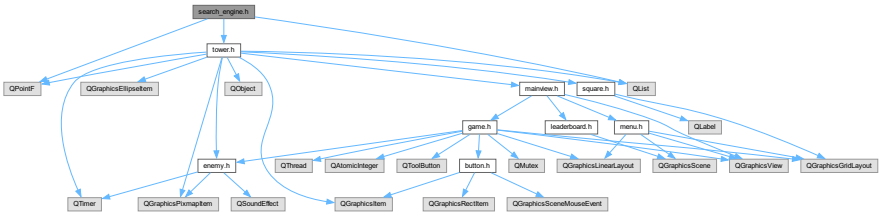
```
1
11 #ifndef RUNTIMEERROR_H
12 #define RUNTIMEERROR_H
13
14 #include "enemy.h"
19 class RuntimeError:  public Enemy
20 {
21     Q_OBJECT
22 public:
23
24     RuntimeError(RuntimeErrorType subType, QList<QPointF> path,QList<QPoint> matrixPath);
25     void takeDamage(int damage);
26
27 private:
32     RuntimeErrorType name_;
33 };
34
35 #endif // RUNTIMEERROR_H
```

5.42 search_engine.h File Reference

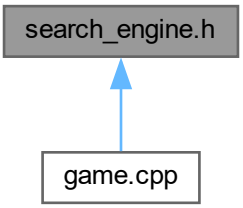
```
#include "tower.h"
#include <QList>
```

```
#include <QPointF>
```

Include dependency graph for search_engine.h:



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



Classes

- class **Search_Engine**

  A Search Engine tower type. Can buff nearby tower, but does not shoot at enemies. Inherited from ***Tower*** *(p. 101) class.*

5.42.1 Detailed Description

Authors

Hung Vu (hung.h.vu@aalto.fi), Saku (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.43 search_engine.h

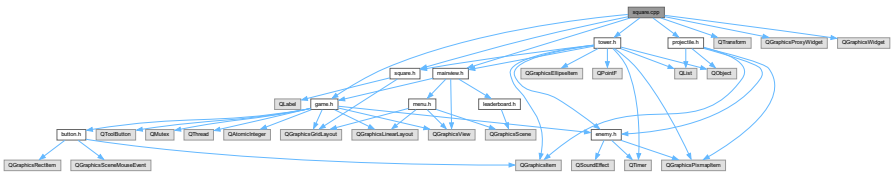
Go to the documentation of this file.

```
1
11 #ifndef SEARCH_ENGINE_H
12 #define SEARCH_ENGINE_H
13 #include "tower.h"
14 #include <QList>
15 #include <QPointF>
16
20 class Search_Engine :  public Tower {
21 public:
22     Search_Engine();
23     Search_Engine(int row, int column, QWidget *parent=nullptr);
24
25     ~Search_Engine();
26
27     bool upgrade();
28     void updateDescription();
29 public slots:
30     void buffPulse();
31 private:
32     double damageBuffFactor_;
33     double atkSpeedBuffFactor_;
34     int buffPulseInterval_;
35     QTimer* buffTimer_;
36
37     QList<QPointF> buffedTowers;
38 };
39
40 #endif // SEARCH_ENGINE_H
```

5.44 square.cpp File Reference

```
#include "square.h"
#include "projectile.h"
#include "tower.h"
#include "game.h"
#include "mainview.h"
#include <QTransform>
#include <QGraphicsProxyWidget>
#include <QGraphicsWidget>
```

Include dependency graph for square.cpp:



Variables

- **MainView ∗ view**

5.44.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi), Harvey Lim (harvey.lim@aalto.fi), Hung Vu (hung.h.vu@aalto.fi)

Version

0.1

Date

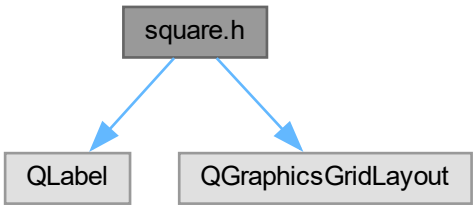
2022-12-11

Copyright

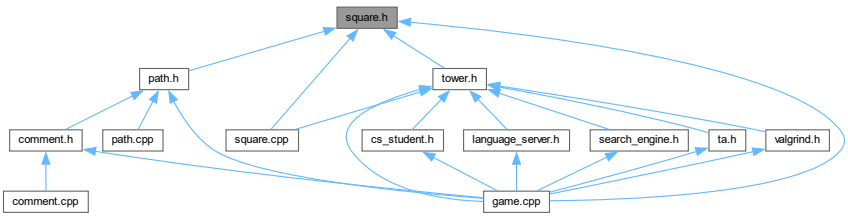
5.45 square.h File Reference

```
#include <QLabel>
#include <QGraphicsGridLayout>
```

Include dependency graph for square.h:



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



Classes

- class **Square**

  This is the square object. It represents a square which is a tower or an empty square where a tower can be placed or a path.

5.45.1 Detailed Description

Authors

Siim Kasela (siim.kasela@aalto.fi), Saku Kovanen (saku.kovanen@aalto.fi), Harvey Lim (harvey.lim@aalto.fi), Hung Vu (hung.h.vu@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

5.46 square.h

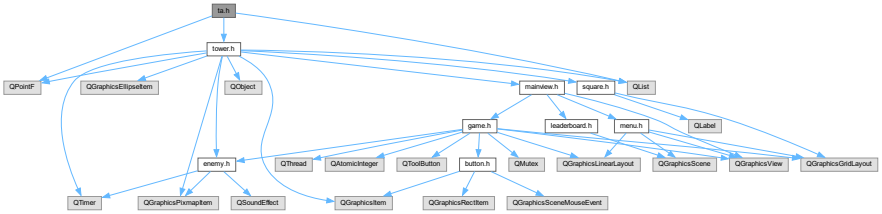
Go to the documentation of this file.

```
1
11 #ifndef SQUARE_H
12 #define SQUARE_H
13 #include <QLabel>
14 #include <QGraphicsGridLayout>
15
20 class Square :  public QLabel
21 {
22     Q_OBJECT
23 public:
24      Square(int x, int y, QWidget *parent = nullptr);
25      void mousePressEvent(QMouseEvent* event);
26      virtual bool isTower();
27      virtual void showHideAttackArea(){};
28      QPointF getCoords() {return QPointF(x_,y_);}
29 public slots:
30      virtual void getTarget() {};// virtual function for Tower class, needed for connecting the timer
31      virtual void buffPulse() {};// virtual function for support towers, needed for connecting the timer
32 protected:
37      int x_;
42      int y_;
43      void fire(QPointF target);
44
45 private:
46      QPixmap rotate(int angle, QPixmap pixmap);
47 };
48
49 #endif // SQUARE_H
```

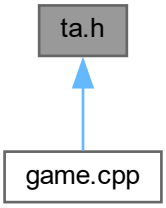
5.47 ta.h File Reference

```
#include "tower.h"
#include <QList>
#include <QPointF>
```

Include dependency graph for ta.h:



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



Classes

- class **TA**

  *A **TA** (p. 97) tower type. Can attack enemies in range and buff nearby towers. Inherited from **Tower** (p. 101) class.*

5.47.1 Detailed Description

Authors

Hung Vu (hung.h.vu@aalto.fi), Saku (saku.kovanen@aalto.fi)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.48 ta.h

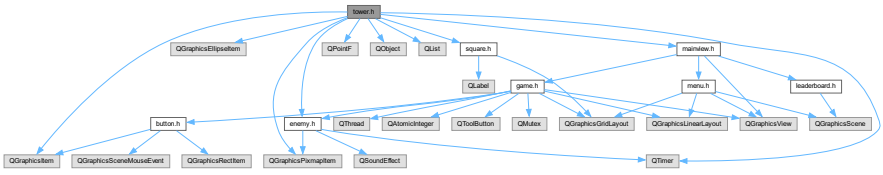
Go to the documentation of this file.

```
1
11 #ifndef TA_H
12 #define TA_H
13
14 #include "tower.h"
15 #include <QList>
16 #include <QPointF>
17
21 class TA : public Tower {
22 public:
23     TA();
24     TA(int row, int column, QWidget *parent=nullptr);
25
26     ~TA();
27
28     void fire(QPointF targetPos);
29
30     bool upgrade();
31     void updateDescription();
32 public slots:
33     void buffPulse();
34 private:
35     double damageBuffFactor_;
36     double atkSpeedBuffFactor_;
37     int buffPulseInterval_;
38     QTimer* buffTimer_;
39
40     int maxRangeMultiplier_ = 1;
41
42     QList<QPointF> buffedTowers;
43 };
44
45
46 #endif // TA_H
```

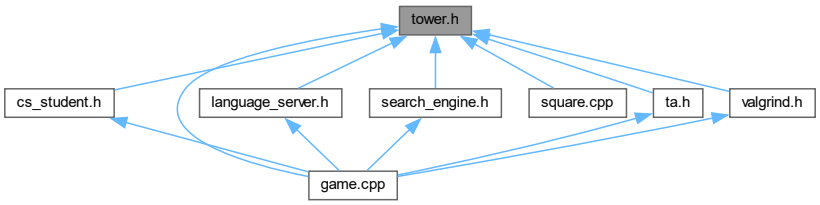
5.49 tower.h File Reference

```
#include "square.h"
#include <QGraphicsEllipseItem>
#include <QGraphicsPixmapItem>
#include <QGraphicsItem>
#include <QPointF>
#include <QObject>
#include <QList>
#include <QTimer>
#include "mainview.h"
#include "enemy.h"
```

Include dependency graph for tower.h:



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



Classes

- class **Tower**

  *Base class for all towers in the game, inherited from **Square** (p. 92).*

Enumerations

- enum **TYPES** { **CompilerError** =0 , **MemoryError** =1 , **RuntimeError** =2 }

Variables

- **MainView** ∗ **view**

5.49.1 Detailed Description

Authors

Hung Vu (`hung.h.vu@aalto.fi`), Saku (`saku.kovanen@aalto.fi`)

Version

0.1

Date

2022-12-11

Copyright

Copyright (c) 2022

5.50 tower.h

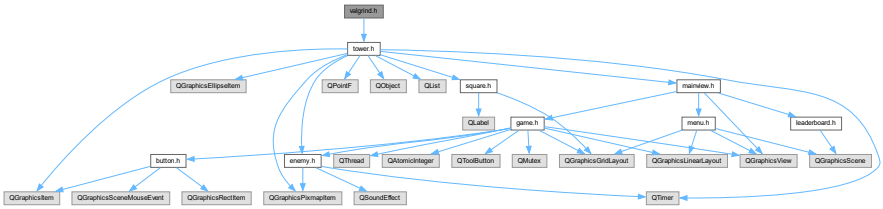
Go to the documentation of this file.

```
1
11 #ifndef TOWER_H
12 #define TOWER_H
13
14 #include "square.h"
15 #include <QGraphicsEllipseItem>
16 #include <QGraphicsPixmapItem>
17 #include <QGraphicsItem>
18 #include <QPointF>
19 #include <QObject>
20 #include <QList>
21 #include <QTimer>
22 #include "mainview.h"
23 #include "enemy.h"
24
25 // enum to act as shorthand that denotes enemy types
26 namespace EnemyTypes{
27     enum TYPES{CompilerError=0, MemoryError=1, RuntimeError=2};
28 }
29
30 extern MainView* view;
31
35 class Tower :  public Square {
36 public:
37     Tower(QWidget *parent = nullptr);
38     Tower(int row, int column, QWidget *parent=nullptr);
39     Tower(int row, int column, int range, int damage, int attackSpeed, QWidget *parent=nullptr);
40
41     ~Tower();
42
43     QPointF towerCenter();
44     double distanceTo(QGraphicsItem * item);
45     virtual void fire(QPointF targetPos);
46     bool isTargetable(Enemy* enemy);
47     void damageBuff(double buffFactor);
48     void atkSpeedBuff(double buffFactor);
49     void targetTableBuff(EnemyTypes::TYPES type);
50
51     void atkSpeedDebuff(double debuffFactor);
52     void targetTableDebuff(EnemyTypes::TYPES type);
53     bool hasAtkSpdBuff();
54
55     QGraphicsEllipseItem* getAttackArea();
56     QList<QGraphicsItem*> getItemInRange();
57     QList<Tower*> getTowersInRange();
58     int getTotalCost();
59     void setRange(int range);
60     void addCost(int cost);
61
62     virtual bool upgrade() = 0;
63     virtual bool isTower();
64     void showAttackArea();
65     void hideAttackArea();
66     virtual void showHideAttackArea();
67     virtual void updateDescription();
68
69     int projectileStepSize_ = 6;// how fast the tower's projectile move
70 public slots:
71     void getTarget();
72 protected:
73     QString type_;
74     int range_;
75     double damage_;
76     double damageMultiplier_;
77     double attackInterval_;
78     QTimer* attackTimer_;
79     int pierce_;
80     int maxLevel_;
81     int upgradeLevel_;// the tower's current upgrade level
82     int totalCost_;
83     int rotationAngle_;
84
85     QGraphicsPixmapItem * towerImg;
86     QString ogImagePath_;
87     QString projectileImagePath_;
88     QGraphicsEllipseItem* attack_area_;
89     QString description_;
90
91     bool canFire_;
92     bool targetAble_[3];
93     bool targetAbleBuff_[3];
94 private:
```

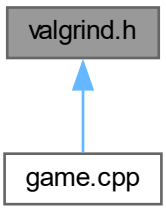
```
95     bool upgradeable_;
96
97     bool has_target_;
98     bool hasAtkSpdBuff_;
99
100     QPointF target_pos_;
101 };
102
103
104
105 #endif // TOWER_H
```

5.51 valgrind.h File Reference

`#include "tower.h"`
Include dependency graph for valgrind.h:



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



Classes

- class **Valgrind**

  *A **Valgrind** (p. 118) tower type. Can attack all enemies on the map. Inherited from **Tower** (p. 101) class.*

5.51.1 Detailed Description

Authors

Hung Vu (`hung.h.vu@aalto.fi`), Saku (`saku.kovanen@aalto.fi`)

Version

0.1

Date

2022-12-11

Copyright

5.52 valgrind.h

Go to the documentation of this file.

```
1
11 #ifndef VALGRIND_H
12 #define VALGRIND_H
13
14 #include "tower.h"
15
19 class Valgrind :  public Tower {
20 public:
21     Valgrind(int row, int column, QWidget *parent=nullptr);
22     virtual bool upgrade();
23     void updateDescription();
24     void fire(QPointF targetPos);
25 };
26
27 #endif // VALGRIND_H
```

Index