My Project

Generated by Doxygen 1.8.17

Chapter 1

Source content

The 'src' folder contains all of the source code for the project. It is mostly divided into folders for specific modules:

1.1 Folders

1.1.1 tiles

Contains the implementations of different tiles that the game field consists of. Enemies and towers are placed in their representive tiles.

1.1.2 towers

Contains the implementations of different towers that the player can place in order to shoot enemies and stop their advance.

1.1.3 enemies

Contains the implementations of different enemies, that move through the path tiles in the map and are shot at by the towers.

1.1.4 GUI

Contains the graphical interface elements that the main game uses to visualize the game for the player.

1.1.5 maps

Contains the maps used in the game in a custom text format.

1.2 Separate files

The rest of the files in 'src' can be roughly categorized into map building tools, exceptions and the main Game.cpp file which is where the game actually runs.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 TDImageFiles Namespace Reference

Variables

- const std::map< const std::string, const std::string > Textures

6.1.1 Variable Documentation

6.1.1.1 Textures

```
const std::map<const std::string, const std::string> TDImageFiles::Textures
```

Initial value:

```
{
    {"pawn", "../assets/towers/chess-pawn.png"},
    {"king", "../assets/towers/chess-king.png"},
    {"queen", "../assets/towers/chess-queen.png"},
    {"rook", "../assets/towers/chess-rook.png"},
    {"smallEnemy", "../assets/enemies/token.png"},
    {"bigEnemy", "../assets/enemies/meeple.png"},
    {"splitEnemy", "../assets/enemies/two-coins.png"},
    {"bossEnemy", "../assets/enemies/abbot-meeple.png"},
    {"finalBoss", "../assets/enemies/meeple-king.png"},
}
```

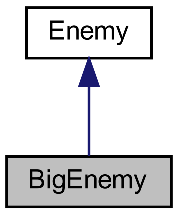
Chapter 7

Class Documentation

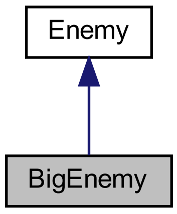
7.1 BigEnemy Class Reference

```
#include <BigEnemy.hpp>
```

Inheritance diagram for BigEnemy:



Collaboration diagram for BigEnemy:



Public Member Functions

- BigEnemy ()

  BigEnemy class constructor. BigEnemy inherits the Enemy class.
- ∼BigEnemy ()

  BigEnemy class destructor.

7.1.1 Constructor & Destructor Documentation

7.1.1.1 BigEnemy()

```
BigEnemy::BigEnemy ( )
```

BigEnemy class constructor. BigEnemy inherits the Enemy class.

7.1.1.2 ∼BigEnemy()

```
BigEnemy::∼BigEnemy ( )
```

BigEnemy class destructor.

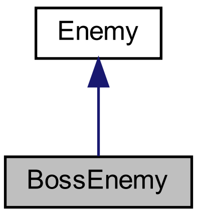
The documentation for this class was generated from the following files:

- src/enemies/BigEnemy.hpp
- src/enemies/BigEnemy.cpp

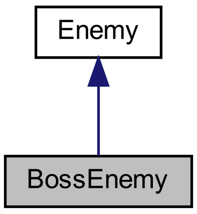
7.2 BossEnemy Class Reference

```
#include <BossEnemy.hpp>
```

Inheritance diagram for BossEnemy:



Collaboration diagram for BossEnemy:



Public Member Functions

- BossEnemy ()

    BossEnemy class constructor. BossEnemy inherits the Enemy class.
- ∼BossEnemy ()

    BossEnemy class destructor.

7.2.1 Constructor & Destructor Documentation

7.2.1.1 BossEnemy()

```
BossEnemy::BossEnemy ( )
```

BossEnemy class constructor. BossEnemy inherits the Enemy class.

7.2.1.2 ∼BossEnemy()

```
BossEnemy::∼BossEnemy ( )
```

BossEnemy class destructor.

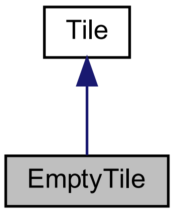
The documentation for this class was generated from the following files:

- src/enemies/BossEnemy.hpp
- src/enemies/BossEnemy.cpp

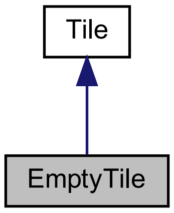
7.3 EmptyTile Class Reference

```
#include <EmptyTile.hpp>
```

Inheritance diagram for EmptyTile:



Collaboration diagram for EmptyTile:



Public Member Functions

- EmptyTile ()

  EmptyTile class constructor. EmptyTile inherits the Tile class. Represents the spots where there is no tower yet on the map.
- ∼EmptyTile ()

  EmptyTile destructor.

Additional Inherited Members

7.3.1 Constructor & Destructor Documentation

7.3.1.1 EmptyTile()

```
EmptyTile::EmptyTile ( )
```

EmptyTile class constructor. EmptyTile inherits the Tile class. Represents the spots where there is no tower yet on the map.

7.3.1.2 ∼EmptyTile()

```
EmptyTile::∼EmptyTile ( )
```

EmptyTile destructor.

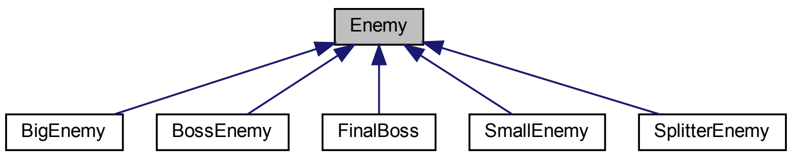
The documentation for this class was generated from the following files:

- src/tiles/EmptyTile.hpp
- src/tiles/EmptyTile.cpp

7.4 Enemy Class Reference

```
#include <Enemy.hpp>
```

Inheritance diagram for Enemy:



Public Member Functions

- Enemy (const std::string &type, int health, size_t reward)

  Enemy class constructor. Enemy is abstract base class for different enemy types.
- virtual ∼Enemy ()=0

  Enemy class destructor.
- const std::string GetType () const

  Get the enemy type.
- int GetHealth () const

  Get the enemy's current health points.
- size_t GetReward () const

  Get the enemy's reward.
- void GetHit (size_t amount)

  Reduce the enemy's health.

Friends

- std::ostream & operator<< (std::ostream &out, const Enemy &e)

7.4.1 Constructor & Destructor Documentation

7.4.1.1 Enemy()

```
Enemy::Enemy (
             const std::string & type,
             int health,
             size_t reward )
```

Enemy class constructor. Enemy is abstract base class for different enemy types.

Parameters

type	"Small", "Big" or "Splitter"
health	How much health enemy should have
reward	The reward amount when enemy dies

7.4.1.2 ∼Enemy()

```
Enemy::∼Enemy ( )  [pure virtual]
```

Enemy class destructor.

7.4.2 Member Function Documentation

7.4.2.1 GetHealth()

```
int Enemy::GetHealth ( ) const
```

Get the enemy's current health points.

Returns

int

7.4.2.2 GetHit()

```
void Enemy::GetHit (
             size_t amount )
```

Reduce the enemy's health.

Parameters

amount	How much damage was dealt to the enemy

7.4.2.3 GetReward()

```
size_t Enemy::GetReward ( ) const
```

Get the enemy's reward.

Returns

size_t

7.4.2.4 GetType()

```
const std::string Enemy::GetType ( ) const
```

Get the enemy type.

Returns

"Small", "Big" or "Splitter"

7.4.3 Friends And Related Function Documentation

7.4.3.1 operator<<

```
std::ostream& operator<< (
             std::ostream & out,
             const Enemy & e )  [friend]
```

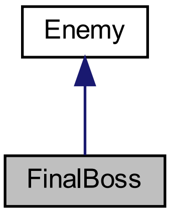
The documentation for this class was generated from the following files:

- src/enemies/Enemy.hpp
- src/enemies/Enemy.cpp

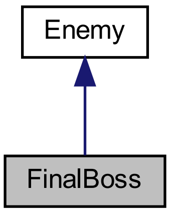
7.5 FinalBoss Class Reference

```
#include <FinalBossEnemy.hpp>
```

Inheritance diagram for FinalBoss:



Collaboration diagram for FinalBoss:



Public Member Functions

- FinalBoss ()

  FinalBoss class constructor. FinalBoss inherits the Enemy class.
- ∼FinalBoss ()

  FinalBoss class destructor.

7.5.1 Constructor & Destructor Documentation

7.5.1.1 FinalBoss()

```
FinalBoss::FinalBoss ( )
```

FinalBoss class constructor. FinalBoss inherits the Enemy class.

7.5.1.2 ∼FinalBoss()

```
FinalBoss::∼FinalBoss ( )
```

FinalBoss class destructor.

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

- src/enemies/FinalBossEnemy.hpp
- src/enemies/FinalBossEnemy.cpp

7.6 GUI Class Reference

```
#include <GUI.hpp>
```

Public Member Functions

- GUI (int width, int height, std::vector< std::vector< Tile ∗ >> ∗tiles)

  Construct a new GUI::GUI object that represents the base of the graphical user interface. It consists of a tilegrid as well as a menu.
- ∼GUI ()
- bool isRunning () const

  Determinates whether the window is open.
- void loadTextures ()

  Preloads the textures in TextureAssets.hpp to textures_ for more efficient use.
- void updateTiles ()

  Updates the latest Tile information into the frame. It in practise refreshes the graphical elements to represent the latest state in the game. Since tiles also hold information about the enemies and towers it also updates their position etc.
- bool createEnemy (PathTile ∗tile, int row, int column)

  Create an graphical enemy based on the information in the tile.
- bool createTower (TowerTile ∗tile, int row, int column)

  Create a graphical Tower based on the information in the tile.
- void render ()

  Renders the new frame and displays it in the window. This makes the changes of updateTiles visible to the user. The inner state of the game affects what e.g. menu elements are drawn, as some of them are mutually exclusive.
- void resize (int width, int height)

  Rezises the window to new values.
- void resetSize ()

  Resizes the window back to original values. Used to deny user resizing of the window as the SFML standard implementation didn't behave properly.

- bool pollEvent (sf::Event &e)

  Requests an event from the window and hands it down. The class by itself doesn't handle events but offers a multitude of methods to help with it.
- void close ()

  Closes the window.
- void initializeMenu ()

  Initializes all of the menu elements (positions, colours etc.). Needs to be called once after initialization and before the first render.
- void pause (bool isPaused)

  Sets the game status to paused/unpaused. Pausing affects e.g. what menu elements are drawn to the user.
- bool checkPauseButtonBounds (sf::Vector2f pos)

  Checks whether a given position is within the pause button.
- bool checkTowerButtonBounds (sf::Vector2f pos)

  Checks whether a given position is within any of the tower buttons. If it is it sets selectedTower_ to reflect which one.
- bool checkUpgradeButtonBounds (sf::Vector2f pos)

  Checks whether a given position is within the upgrade button AND a tower is currently selected within the game.
- void loadUpgradeInfo ()

  Refreshes the upgrade info text to represent the relevant upgrade.
- bool upgradeTower (int ∗score)

  Attempts to upgrade the currently selected tower. Fails if the amount of points available is insufficient.
- void createAmmunition (std::pair< int, int > from, Tile ∗to)

  Create a graphical shell that flies from the tower to the enemy when shooting. The amount of movement steps taken in the flying animation is set by SHELLSTEPS. The idea is that all of the shells complete their flight before the next game step. Therefore the length of the steps is affected by the distance travelled, so that each shell completes their flight simultaneously.
- bool moveAmmunition ()

  Moves all of the ammunition based on the movement vector stored with them. When their path is finished it destroys the shells.
- sf::Vector2f getMousePos ()

  returns mouse position in local coordinates
- bool posAsGrid (sf::Vector2f pos, sf::Vector2i ∗grid)

  Translates a position into a place in the grid if possible.
- void selectTile ()

  Selects the tile that is currently under the mouse. The effects of the selection are determined by the state of the game, e.g. whether a tower is currently being held. Trying to select a tile while the mouse is off the tile grid does nothing.
- Tile ∗ getTile (sf::Vector2i pos) const

  Fetches the tile according to the vector given.
- sf::Time getAmmoTime () const

  Returns the time needed for a single ammo flight step.
- bool release (Tower ∗∗spot, int ∗score)

  Handles the actions connected to releasing the mouse. Places a tower if one is currently being held in a suitable position.
- void highlight (sf::Vector2i gridpos, sf::Color color)

  Highlights the given tile with an outline of given color. Calling with an illegal position will cause undefined behaviour. Should only be used when the position is already confirmed correct, e.g. with the posAsGrid function.
- void highlight (std::vector< Tile ∗ > tiles, sf::Color color)

  Highlights the graphical tiles that represent the tiles in the given list with the given color. Calling it with tiles not in the game will cause undefined behaviour.
- void visualizeRange (sf::Vector2i gridpos)

  Visualizes the currently selected towers range if it was placed in the given position.
- void setScore (int value)

  Sets the score counter to the given value.
- void setWave (int value)

  Sets the wave counter to the given value.

7.6.1 Constructor & Destructor Documentation

7.6.1.1 GUI()

```
GUI::GUI (
            int width,
            int height,
            std::vector< std::vector< Tile * >> * tiles )
```

Construct a new GUI::GUI object that represents the base of the graphical user interface. It consists of a tilegrid as well as a menu.

Parameters

width	the width of the window (in tiles)
heigth	the height of the window (in tiles)
tiles	the tile grid used in game (should match the size parameters given)

7.6.1.2 ∼GUI()

```
GUI::∼GUI ( )  [inline]
```

7.6.2 Member Function Documentation

7.6.2.1 checkPauseButtonBounds()

```
bool GUI::checkPauseButtonBounds (
            sf::Vector2f pos )  [inline]
```

Checks whether a given position is within the pause button.

Parameters

pos	position to be checked

Returns

whether it was within the button

7.6.2.2 checkTowerButtonBounds()

```
bool GUI::checkTowerButtonBounds (
            sf::Vector2f pos )
```

Checks whether a given position is within any of the tower buttons. If it is it sets selectedTower_ to reflect which one.

Parameters

pos	position to be checked

Returns

whether it was within any of the buttons

7.6.2.3 checkUpgradeButtonBounds()

```
bool GUI::checkUpgradeButtonBounds (
            sf::Vector2f pos )  [inline]
```

Checks whether a given position is within the upgrade button AND a tower is currently selected within the game.

Parameters

pos	position to be checked

Returns

whether it was within the button

7.6.2.4 close()

```
void GUI::close ( )  [inline]
```

Closes the window.

7.6.2.5 createAmmunition()

```
void GUI::createAmmunition (
            std::pair< int, int > from,
            Tile * to )
```

Create a graphical shell that flies from the tower to the enemy when shooting. The amount of movement steps taken in the flying animation is set by SHELLSTEPS. The idea is that all of the shells complete their flight before the next game step. Therefore the length of the steps is affected by the distance travelled, so that each shell completes their flight simultaneously.

Parameters

from	The towers coords (x, y)
to	The tile where the enemy is

7.6.2.6 createEnemy()

```
bool GUI::createEnemy (
             PathTile * tile,
             int row,
             int column )
```

Create an graphical enemy based on the information in the tile.

Parameters

tile	The tile in question

Returns

Whether a new enemy was created

7.6.2.7 createTower()

```
bool GUI::createTower (
             TowerTile * tile,
             int row,
             int column )
```

Create a graphical Tower based on the information in the tile.

Parameters

tile	the tile in question.

Returns

Whether a new tower was created

7.6.2.8 getAmmoTime()

```
sf::Time GUI::getAmmoTime ( ) const  [inline]
```

Returns the time needed for a single ammo flight step.

Returns

time in milliseconds (constant)

7.6.2.9 getMousePos()

```
sf::Vector2f GUI::getMousePos ( )  [inline]
```

returns mouse position in local coordinates

Returns

sf::Vector2f

7.6.2.10 getTile()

```
Tile* GUI::getTile (
            sf::Vector2i pos ) const  [inline]
```

Fetches the tile according to the vector given.

7.6.2.11 highlight() [1/2]

```
void GUI::highlight (
            sf::Vector2i gridpos,
            sf::Color color )
```

Highlights the given tile with an outline of given color. Calling with an illegal position will cause undefined behaviour. Should only be used when the position is already confirmed correct, e.g. with the posAsGrid function.

Parameters

gridpos	the position of the tile in the grid

7.6.2.12 highlight() [2/2]

```
void GUI::highlight (
            std::vector< Tile * > tiles,
            sf::Color color )
```

Highlights the graphical tiles that represent the tiles in the given list with the given color. Calling it with tiles not in the game will cause undefined behaviour.

Parameters

tiles	vector of tiles to be highlighted

7.6.2.13 initializeMenu()

```
void GUI::initializeMenu ( )
```

Initializes all of the menu elements (positions, colours etc.). Needs to be called once after initialization and before the first render.

7.6.2.14 isRunning()

```
bool GUI::isRunning ( ) const  [inline]
```

Determinates whether the window is open.

7.6.2.15 loadTextures()

```
void GUI::loadTextures ( )
```

Preloads the textures in TextureAssets.hpp to textures_ for more efficient use.

7.6.2.16 loadUpgradeInfo()

```
void GUI::loadUpgradeInfo ( )
```

Refreshes the upgrade info text to represent the relevant upgrade.

7.6.2.17 moveAmmunition()

```
bool GUI::moveAmmunition ( )
```

Moves all of the ammunition based on the movement vector stored with them. When their path is finished it destroys the shells.

Returns

Whether the shells reached their target

7.6.2.18 pause()

```
void GUI::pause (
            bool isPaused )
```

Sets the game status to paused/unpaused. Pausing affects e.g. what menu elements are drawn to the user.

Parameters

isPaused	true if pausing the game, false when continuing

7.6.2.19 pollEvent()

```
bool GUI::pollEvent (
             sf::Event & e )
```

Requests an event from the window and hands it down. The class by itself doesn't handle events but offers a multitude of methods to help with it.

7.6.2.20 posAsGrid()

```
bool GUI::posAsGrid (
             sf::Vector2f pos,
             sf::Vector2i * grid )  [inline]
```

Translates a position into a place in the grid if possible.

Parameters

pos	position to be translated
grid	a pointer to a int vector for the result

Returns

true It was possible, the value is now in grid.

false It wasn't possible, nothing happens.

7.6.2.21 release()

```
bool GUI::release (
             Tower ** spot,
             int * score )
```

Handles the actions connected to releasing the mouse. Places a tower if one is currently being held in a suitable position.

Parameters

spot	a pointer to a spot for the tower pointer to be stored in
score	a pointer to the current available points

Returns

true a tower was placed, the pointer is stored in spot and the score is deducted.

false a tower was not placed, nothing happens

7.6.2.22 render()

```
void GUI::render ( )
```

Renders the new frame and displays it in the window. This makes the changes of updateTiles visible to the user. The inner state of the game affects what e.g. menu elements are drawn, as some of them are mutually exclusive.

7.6.2.23 resetSize()

```
void GUI::resetSize ( )
```

Resizes the window back to original values. Used to deny user resizing of the window as the SFML standard implementation didn't behave properly.

7.6.2.24 resize()

```
void GUI::resize (
             int width,
             int height )
```

Rezises the window to new values.

7.6.2.25 selectTile()

```
void GUI::selectTile ( )
```

Selects the tile that is currently under the mouse. The effects of the selection are determined by the state of the game, e.g. whether a tower is currently being held. Trying to select a tile while the mouse is off the tile grid does nothing.

7.6.2.26 setScore()

```
void GUI::setScore (
             int value )  [inline]
```

Sets the score counter to the given value.

7.6.2.27 setWave()

```
void GUI::setWave (
            int value )  [inline]
```

Sets the wave counter to the given value.

7.6.2.28 updateTiles()

```
void GUI::updateTiles ( )
```

Updates the latest Tile information into the frame. It in practise refreshes the graphical elements to represent the latest state in the game. Since tiles also hold information about the enemies and towers it also updates their position etc.

7.6.2.29 upgradeTower()

```
bool GUI::upgradeTower (
            int * score )
```

Attempts to upgrade the currently selected tower. Fails if the amount of points available is insufficient.

Parameters

score	Available points

Returns

Whether the upgrade was successful

7.6.2.30 visualizeRange()

```
void GUI::visualizeRange (
            sf::Vector2i gridpos )
```

Visualizes the currently selected towers range if it was placed in the given position.

Parameters

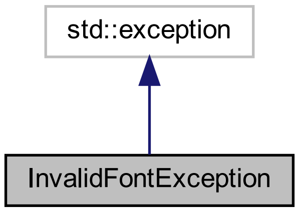
gridpos	position the tower is placed

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

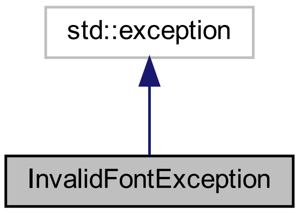
- src/GUI/GUI.hpp
- src/GUI/GUI.cpp

7.7 InvalidFontException Class Reference

Inheritance diagram for InvalidFontException:



Collaboration diagram for InvalidFontException:



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

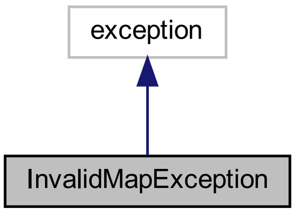
- src/GUI/GUI.cpp

7.8 InvalidMapException Class Reference

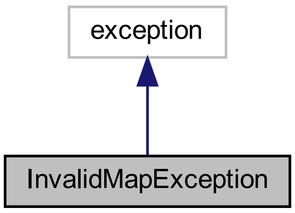
InvalidMapException class that derives std library exception class.

```
#include <InvalidMapException.hpp>
```

Inheritance diagram for InvalidMapException:



Collaboration diagram for InvalidMapException:



7.8.1 Detailed Description

InvalidMapException class that derives std library exception class.

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

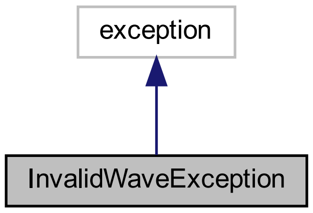
- src/InvalidMapException.hpp

7.9 InvalidWaveException Class Reference

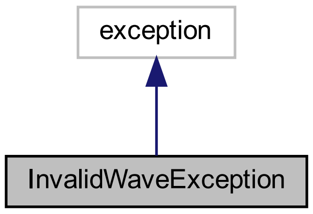
InvalidWaveException class that derives std library exception class.

```
#include <InvalidWaveException.hpp>
```

Inheritance diagram for InvalidWaveException:



Collaboration diagram for InvalidWaveException:



7.9.1 Detailed Description

InvalidWaveException class that derives std library exception class.

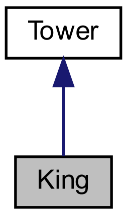
The documentation for this class was generated from the following file:

- src/InvalidWaveException.hpp

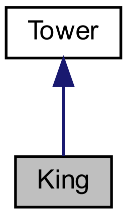
7.10 King Class Reference

```
#include <king.hpp>
```

Inheritance diagram for King:



Collaboration diagram for King:



Public Member Functions

- King (int x, int y, std::vector< std::vector< Tile ∗ >> ∗level)
- ∼King ()
- std::pair< int, Tile ∗ > Shoot ()

    Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.
- void GetTargetTiles (std::vector< std::vector< Tile ∗ >> ∗)

    Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.
- std::vector< PathTile ∗ > TileRecurring (int x, int y, std::vector< PathTile ∗ > tiles, int range, std::vector< std::vector< Tile ∗ >> ∗level)
- std::vector< Tile ∗ > GetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

    Gets all the tiles where the tower can shoot, and returns them.

Static Public Member Functions

- static int GetPrice ()
- static std::vector< Tile ∗ > StaticGetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level, int x, int y)
- static std::vector< Tile ∗ > AllTileRecurring (int x, int y, std::vector< Tile ∗ > tiles, int range, std::vector< std::vector< Tile ∗ >> ∗level)

Additional Inherited Members

7.10.1 Constructor & Destructor Documentation

7.10.1.1 King()

```
King::King (
            int x,
            int y,
            std::vector< std::vector< Tile * >> * level )
```

7.10.1.2 ∼King()

```
King::∼King ( )
```

7.10.2 Member Function Documentation

7.10.2.1 AllTileRecurring()

```
std::vector< Tile * > King::AllTileRecurring (
            int x,
            int y,
            std::vector< Tile * > tiles,
            int range,
            std::vector< std::vector< Tile * >> * level )  [static]
```

7.10.2.2 GetAllTargetTiles()

```
std::vector< Tile * > King::GetAllTargetTiles (
            std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the tiles where the tower can shoot, and returns them.

Parameters

level	The level where the tiles are

Returns

returns vector of Tile∗ where tower can see

Reimplemented from Tower.

7.10.2.3 GetPrice()

```
int King::GetPrice ( )  [static]
```

7.10.2.4 GetTargetTiles()

```
void King::GetTargetTiles (
             std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.

Parameters

level	The level where the tiles are

Reimplemented from Tower.

7.10.2.5 Shoot()

```
std::pair< int, Tile * > King::Shoot ( )  [virtual]
```

Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.

Reimplemented from Tower.

7.10.2.6 StaticGetAllTargetTiles()

```
std::vector< Tile * > King::StaticGetAllTargetTiles (
             std::vector< std::vector< Tile * >> * level,
             int x,
             int y )  [static]
```

7.10.2.7 TileRecurring()

```
std::vector< PathTile * > King::TileRecurring (
             int x,
             int y,
             std::vector< PathTile * > tiles,
             int range,
             std::vector< std::vector< Tile * >> * level )
```

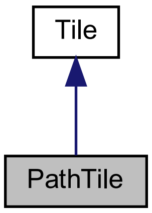
The documentation for this class was generated from the following files:

- src/towers/king.hpp
- src/towers/king.cpp

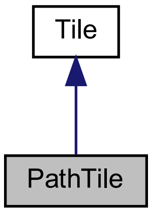
7.11 PathTile Class Reference

```
#include <PathTile.hpp>
```

Inheritance diagram for PathTile:



Collaboration diagram for PathTile:



Public Member Functions

- PathTile (size_t order, PathTile ∗nextPathTile=nullptr, PathTile ∗previousPathTile=nullptr)

  PathTile class constructor. PathTile inherits the Tile class. This class represents a path on the map.
- ∼PathTile ()

  PathTile destructor.
- size_t GetOrder () const

  Get the tile's order number.
- Enemy ∗ GetEnemy () const

  Get the current enemy, that is on this PathTile.
- PathTile ∗ GetNextPathTile () const

  Get the next PathTile.
- PathTile ∗ GetPreviousPathTile () const

  Get the previous PathTile.
- void SetEnemy (Enemy ∗enemy)

  Add enemy on this PathTile.
- void SetNextPathTile (PathTile ∗pathTile)

  Set next PathTile.
- void SetPrevoiusPathTile (PathTile ∗pathTile)

  Set previous PathTile.
- size_t AttackEnemy (size_t damage)

  Attack enemy. If its HP changes to 0 or less, kill the enemy.

Protected Member Functions

- virtual void Print (std::ostream &out) const

  Print PathTile info to terminal.

7.11.1 Constructor & Destructor Documentation

7.11.1.1 PathTile()

```
PathTile::PathTile (
             size_t order,
             PathTile ∗ nextPathTile = nullptr,
             PathTile ∗ previousPathTile = nullptr )
```

PathTile class constructor. PathTile inherits the Tile class. This class represents a path on the map.

Parameters

order	The order number of this tile
nextPathTile	Pointer to next PathTile
previousPathTilePointer	to previous PathTile

7.11.1.2 ∼PathTile()

```
PathTile::∼PathTile ( )
```

PathTile destructor.

7.11.2 Member Function Documentation

7.11.2.1 AttackEnemy()

```
size_t PathTile::AttackEnemy (
             size_t damage )
```

Attack enemy. If its HP changes to 0 or less, kill the enemy.

Parameters

damage	

Returns

return enemy reward if it dies, else 0

7.11.2.2 GetEnemy()

```
Enemy * PathTile::GetEnemy ( ) const
```

Get the current enemy, that is on this PathTile.

Returns

Pointer to enemy

7.11.2.3 GetNextPathTile()

```
PathTile * PathTile::GetNextPathTile ( ) const
```

Get the next PathTile.

Returns

Pointer to PathTile

7.11.2.4 GetOrder()

```
size_t PathTile::GetOrder ( ) const
```

Get the tile's order number.

Returns

size_t

7.11.2.5 GetPreviousPathTile()

```
PathTile * PathTile::GetPreviousPathTile ( ) const
```

Get the previous PathTile.

Returns

Pointer to PathTile

7.11.2.6 Print()

```
void PathTile::Print (
             std::ostream & out ) const  [protected], [virtual]
```

Print PathTile info to terminal.

Parameters

out	ostream

Reimplemented from Tile.

7.11.2.7 SetEnemy()

```
void PathTile::SetEnemy (
             Enemy * enemy )
```

Add enemy on this PathTile.

Parameters

enemy	Pointer to enemy

7.11.2.8 SetNextPathTile()

```
void PathTile::SetNextPathTile (
             PathTile * pathTile )
```

Set next PathTile.

Parameters

pathTile	Pointer to PathTile

7.11.2.9 SetPrevoiusPathTile()

```
void PathTile::SetPrevoiusPathTile (
             PathTile * pathTile )
```

Set previous PathTile.

Parameters

pathTile	Pointer to PathTile

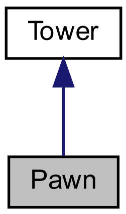
The documentation for this class was generated from the following files:

- src/tiles/PathTile.hpp
- src/tiles/PathTile.cpp

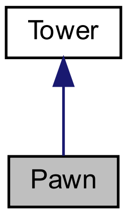
7.12 Pawn Class Reference

```
#include <pawn.hpp>
```

Inheritance diagram for Pawn:



Collaboration diagram for Pawn:



Public Member Functions

- Pawn (int x, int y, std::vector< std::vector< Tile ∗ >> ∗level)
- ∼Pawn ()
- std::pair< int, Tile ∗ > Shoot ()

    Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.
- void GetTargetTiles (std::vector< std::vector< Tile ∗ >> ∗)

    Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.
- std::vector< Tile ∗ > GetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

    Gets all the tiles where the tower can shoot, and returns them.

Static Public Member Functions

- static int GetPrice ()
- static std::vector< Tile ∗ > StaticGetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level, int x, int y)

Additional Inherited Members

7.12.1 Constructor & Destructor Documentation

7.12.1.1 Pawn()

```
Pawn::Pawn (
             int x,
             int y,
             std::vector< std::vector< Tile ∗ >> ∗ level )
```

7.12.1.2 ∼Pawn()

```
Pawn::∼Pawn ( )
```

7.12.2 Member Function Documentation

7.12.2.1 GetAllTargetTiles()

```
std::vector< Tile * > Pawn::GetAllTargetTiles (
             std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the tiles where the tower can shoot, and returns them.

Parameters

level	The level where the tiles are

Returns

returns vector of Tile∗ where tower can see

Reimplemented from Tower.

7.12.2.2 GetPrice()

```
int Pawn::GetPrice ( )  [static]
```

7.12.2.3 GetTargetTiles()

```
void Pawn::GetTargetTiles (
             std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.

Parameters

level	The level where the tiles are

Reimplemented from Tower.

7.12.2.4 Shoot()

```
std::pair< int, Tile * > Pawn::Shoot ( )  [virtual]
```

Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.

Reimplemented from Tower.

7.12.2.5 StaticGetAllTargetTiles()

```
std::vector< Tile * > Pawn::StaticGetAllTargetTiles (
            std::vector< std::vector< Tile * >> * level,
            int x,
            int y )  [static]
```

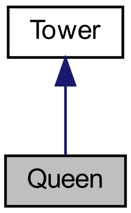
The documentation for this class was generated from the following files:

- src/towers/pawn.hpp
- src/towers/pawn.cpp

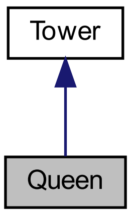
7.13 Queen Class Reference

```
#include <queen.hpp>
```

Inheritance diagram for Queen:



Collaboration diagram for Queen:



Public Member Functions

- Queen (int x, int y, std::vector< std::vector< Tile ∗ >> ∗level)
- ∼Queen ()
- std::pair< int, Tile ∗ > Shoot ()

  Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.
- void GetTargetTiles (std::vector< std::vector< Tile ∗ >> ∗)

  Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.
- std::vector< PathTile ∗ > TileRecurring (int x, int y, std::vector< PathTile ∗ > tiles, int range, std::vector< std::vector< Tile ∗ >> ∗level)
- std::vector< Tile ∗ > GetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

  Gets all the tiles where the tower can shoot, and returns them.

Static Public Member Functions

- static int GetPrice ()
- static std::vector< Tile ∗ > StaticGetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level, int x, int y)
- static std::vector< Tile ∗ > AllTileRecurring (int x, int y, std::vector< Tile ∗ > tiles, int range, std::vector< std::vector< Tile ∗ >> ∗level)

Additional Inherited Members

7.13.1 Constructor & Destructor Documentation

7.13.1.1 Queen()

```
Queen::Queen (
              int x,
              int y,
              std::vector< std::vector< Tile ∗ >> ∗ level )
```

7.13.1.2 ∼Queen()

```
Queen::∼Queen ( )
```

7.13.2 Member Function Documentation

7.13.2.1 AllTileRecurring()

```
std::vector< Tile ∗ > Queen::AllTileRecurring (
            int x,
            int y,
            std::vector< Tile ∗ > tiles,
            int range,
            std::vector< std::vector< Tile ∗ >> ∗ level ) [static]
```

7.13.2.2 GetAllTargetTiles()

```
std::vector< Tile ∗ > Queen::GetAllTargetTiles (
            std::vector< std::vector< Tile ∗ >> ∗ level ) [virtual]
```

Gets all the tiles where the tower can shoot, and returns them.

Parameters

level	The level where the tiles are

Returns

returns vector of Tile∗ where tower can see

Reimplemented from Tower.

7.13.2.3 GetPrice()

```
int Queen::GetPrice ( ) [static]
```

7.13.2.4 GetTargetTiles()

```
void Queen::GetTargetTiles (
            std::vector< std::vector< Tile ∗ >> ∗ level ) [virtual]
```

Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.

Parameters

level	The level where the tiles are

Reimplemented from Tower.

7.13.2.5 Shoot()

```
std::pair< int, Tile * > Queen::Shoot ( ) [virtual]
```

Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.

Reimplemented from Tower.

7.13.2.6 StaticGetAllTargetTiles()

```
std::vector< Tile * > Queen::StaticGetAllTargetTiles (
             std::vector< std::vector< Tile * >> * level,
             int x,
             int y ) [static]
```

7.13.2.7 TileRecurring()

```
std::vector< PathTile * > Queen::TileRecurring (
             int x,
             int y,
             std::vector< PathTile * > tiles,
             int range,
             std::vector< std::vector< Tile * >> * level )
```

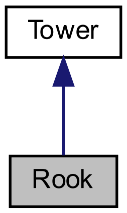
The documentation for this class was generated from the following files:

- src/towers/queen.hpp
- src/towers/queen.cpp

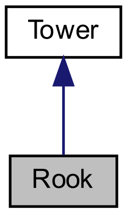
7.14 Rook Class Reference

`#include <rook.hpp>`

Inheritance diagram for Rook:



Collaboration diagram for Rook:



Public Types

- enum directions {
  all, up, down, left,
  right }

Public Member Functions

- Rook (int x, int y, std::vector< std::vector< Tile ∗ >> ∗level)
- ∼Rook ()
- std::pair< int, Tile ∗ > Shoot ()

  shoots at the enemy with most hp
- void GetTargetTiles (std::vector< std::vector< Tile ∗ >> ∗)

  Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.
- std::vector< PathTile ∗ > TileRecurring (int x, int y, std::vector< PathTile ∗ > tiles, int range, std::vector< std::vector< Tile ∗ >> ∗level, directions direction)

  Gets all path tiles in diagonal directions.
- std::vector< Tile ∗ > GetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

  Gets all the tiles where the tower can shoot, and returns them.

Static Public Member Functions

- static int GetPrice ()
- static std::vector< Tile * > StaticGetAllTargetTiles (std::vector< std::vector< Tile * >> *level, int x, int y)
- static std::vector< Tile * > AllTileRecurring (int x, int y, std::vector< Tile * > tiles, int range, std::vector< std::vector< Tile * >> *level, directions direction)

  Gets all tiles in diagonal directions.

Additional Inherited Members

7.14.1 Member Enumeration Documentation

7.14.1.1 directions

```
enum Rook::directions
```

Enumerator

all	
up	
down	
left	
right	

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Rook()

```
Rook::Rook (
            int x,
            int y,
            std::vector< std::vector< Tile * >> * level )
```

7.14.2.2 ∼Rook()

```
Rook::∼Rook ( )
```

7.14.3 Member Function Documentation

7.14.3.1 AllTileRecurring()

```
std::vector< Tile * > Rook::AllTileRecurring (
             int x,
             int y,
             std::vector< Tile * > tiles,
             int range,
             std::vector< std::vector< Tile * >> * level,
             directions direction ) [static]
```

Gets all tiles in diagonal directions.

Parameters

x	x of the tile searching
y	y of the tile searching
tiles	currently found tiles
range	range left for the search
level	level where we are searching
direction	the cardinal direction where we are searching, first time all

Returns

returns tiles.

7.14.3.2 GetAllTargetTiles()

```
std::vector< Tile * > Rook::GetAllTargetTiles (
             std::vector< std::vector< Tile * >> * level ) [virtual]
```

Gets all the tiles where the tower can shoot, and returns them.

Parameters

level	The level where the tiles are

Returns

returns vector of Tile∗ where tower can see

Reimplemented from Tower.

7.14.3.3 GetPrice()

```
int Rook::GetPrice ( ) [static]
```

7.14.3.4 GetTargetTiles()

```
void Rook::GetTargetTiles (
             std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.

Parameters

level	The level where the tiles are

Reimplemented from Tower.

7.14.3.5 Shoot()

```
std::pair< int, Tile * > Rook::Shoot ( )  [virtual]
```

shoots at the enemy with most hp

Reimplemented from Tower.

7.14.3.6 StaticGetAllTargetTiles()

```
std::vector< Tile * > Rook::StaticGetAllTargetTiles (
             std::vector< std::vector< Tile * >> * level,
             int x,
             int y )  [static]
```

7.14.3.7 TileRecurring()

```
std::vector< PathTile * > Rook::TileRecurring (
             int x,
             int y,
             std::vector< PathTile * > tiles,
             int range,
             std::vector< std::vector< Tile * >> * level,
             directions direction )
```

Gets all path tiles in diagonal directions.

Parameters

x	x of the tile searching
y	y of the tile searching
tiles	currently found tiles
range	range left for the search
level	level where we are searching
direction	the cardinal direction where we are searching, first time all

Returns

returns tiles.

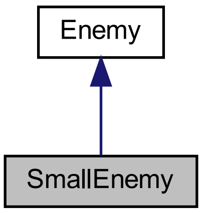
The documentation for this class was generated from the following files:

- src/towers/rook.hpp
- src/towers/rook.cpp

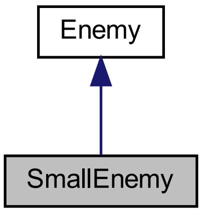
7.15 SmallEnemy Class Reference

```
#include <SmallEnemy.hpp>
```

Inheritance diagram for SmallEnemy:



Collaboration diagram for SmallEnemy:



Public Member Functions

- SmallEnemy ()

  SmallEnemy class constructor. SmallEnemy inherits the Enemy class.
- ∼SmallEnemy ()

  SmallEnemy class destructor.

7.15.1 Constructor & Destructor Documentation

7.15.1.1 SmallEnemy()

```
SmallEnemy::SmallEnemy ( )
```

SmallEnemy class constructor. SmallEnemy inherits the Enemy class.

7.15.1.2 ∼SmallEnemy()

```
SmallEnemy::∼SmallEnemy ( )
```

SmallEnemy class destructor.

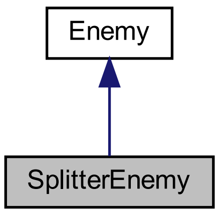
The documentation for this class was generated from the following files:

- src/enemies/SmallEnemy.hpp
- src/enemies/SmallEnemy.cpp

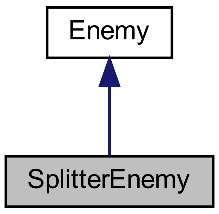
7.16 SplitterEnemy Class Reference

```
#include <SplitterEnemy.hpp>
```

Inheritance diagram for SplitterEnemy:



Collaboration diagram for SplitterEnemy:



Public Member Functions

- SplitterEnemy ()

  SplitterEnemy class constructor. SplitterEnemy inherits the Enemy class.
- ∼SplitterEnemy ()

  SplitterEnemy class destructor.
- void Split (PathTile ∗currentTile, PathTile ∗previousTile)

  if Splitter dies, it spawns two small enemies in the previous and own tile.

7.16.1 Constructor & Destructor Documentation

7.16.1.1 SplitterEnemy()

```
SplitterEnemy::SplitterEnemy ( )
```

SplitterEnemy class constructor. SplitterEnemy inherits the Enemy class.

7.16.1.2 ∼SplitterEnemy()

```
SplitterEnemy::∼SplitterEnemy ( )
```

SplitterEnemy class destructor.

7.16.2 Member Function Documentation

7.16.2.1 Split()

```
void SplitterEnemy::Split (
            PathTile ∗ currentTile,
            PathTile ∗ previousTile )
```

if Splitter dies, it spawns two small enemies in the previous and own tile.

Parameters

currentTile	PathTile∗
previousTile	PathTile∗

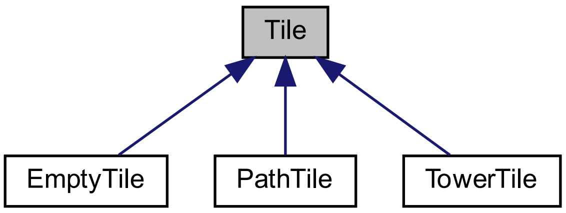
The documentation for this class was generated from the following files:

- src/enemies/SplitterEnemy.hpp
- src/enemies/SplitterEnemy.cpp

7.17 Tile Class Reference

`#include <Tile.hpp>`

Inheritance diagram for Tile:



Public Member Functions

- Tile (const std::string &type)

    Tile class constructor. Tile is abstract base class for different tile types.
- virtual ∼Tile ()=0

    Tile class destructor.
- const std::string GetType () const

    Get the tile type.

Protected Member Functions

- virtual void Print (std::ostream &out) const

    Print tile to terminal.

Friends

- std::ostream & operator<< (std::ostream &o, const Tile &t)

7.17.1 Constructor & Destructor Documentation

7.17.1.1 Tile()

```
Tile::Tile (
            const std::string & type )
```

Tile class constructor. Tile is abstract base class for different tile types.

Parameters

type	"path", "tower" or "empty"

7.17.1.2 ∼Tile()

```
Tile::∼Tile ( )  [pure virtual]
```

Tile class destructor.

7.17.2 Member Function Documentation

7.17.2.1 GetType()

```
const std::string Tile::GetType ( ) const
```

Get the tile type.

Returns

"path", "tower" or "empty"

7.17.2.2 Print()

```
void Tile::Print (
            std::ostream & out ) const  [protected], [virtual]
```

Print tile to terminal.

Parameters

out	ostream

Reimplemented in PathTile, and TowerTile.

7.17.3 Friends And Related Function Documentation

7.17.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & o,
            const Tile & t )  [friend]
```

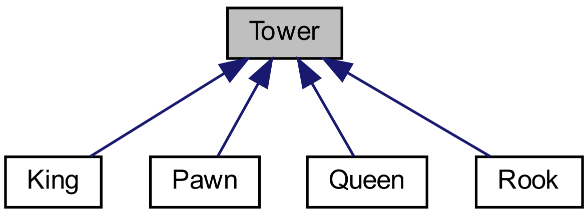
The documentation for this class was generated from the following files:

- src/tiles/Tile.hpp
- src/tiles/Tile.cpp

7.18 Tower Class Reference

```
#include <tower.hpp>
```

Inheritance diagram for Tower:



Public Member Functions

- Tower (std::string type, int speed, int damage, int range, int price, int x, int y, std::vector< std::tuple< int, upgradeType, int >> &upgradeTable)
- virtual ∼Tower ()
- std::string GetType () const
- int GetSpeed () const
- int GetDamage () const
- int GetRange () const
- int GetLevel () const
- std::pair< int, int > getPos ()
- std::tuple< int, upgradeType, int > GetNextUpgrade () const

  Returns the next upgrade level of the tower.
- bool Upgrade (int money, std::vector< std::vector< Tile ∗ >> ∗level)

  Upgrades the tower to the next level if possible.
- virtual std::pair< int, Tile ∗ > Shoot ()

  Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.
- virtual void GetTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

  Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.
- virtual std::vector< Tile ∗ > GetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level)

  Gets all the tiles where the tower can shoot, and returns them.

Static Public Member Functions

- static int GetPrice ()
- static std::vector< Tile ∗ > StaticGetAllTargetTiles (std::vector< std::vector< Tile ∗ >> ∗level, int x, int y)

  Statically gets all the tiles where the tower can shoot, and returns them.

Protected Attributes

- std::string type_
- int speed_
- int damage_
- int range_
- int level_
- int coolDown_
- int price_
- int x_
- int y_
- std::vector< PathTile ∗ > canSee_
- std::vector< std::tuple< int, upgradeType, int > > & upgradeTable_

Friends

- std::ostream & operator<< (std::ostream &stream, Tower &tower)

7.18.1 Constructor & Destructor Documentation

7.18.1.1 Tower()

```
Tower::Tower (
            std::string type,
            int speed,
            int damage,
            int range,
            int price,
            int x,
            int y,
            std::vector< std::tuple< int, upgradeType, int >> & upgradeTable )
```

7.18.1.2 ∼Tower()

```
Tower::∼Tower ( )  [virtual]
```

7.18.2 Member Function Documentation

7.18.2.1 GetAllTargetTiles()

```
std::vector< Tile * > Tower::GetAllTargetTiles (
            std::vector< std::vector< Tile * >> * level )  [virtual]
```

Gets all the tiles where the tower can shoot, and returns them.

Parameters

level	The level where the tiles are

Returns

returns vector of Tile∗ where tower can see

Reimplemented in Rook, King, Queen, and Pawn.

7.18.2.2 GetDamage()

```
int Tower::GetDamage ( ) const
```

7.18.2.3 GetLevel()

```
int Tower::GetLevel ( ) const
```

7.18.2.4 GetNextUpgrade()

```
std::tuple< int, upgradeType, int > Tower::GetNextUpgrade ( ) const
```

Returns the next upgrade level of the tower.

Returns

Tuple including the price, upgrade type and increase, in this order. Price is -1 if tower is at max level.

7.18.2.5 getPos()

```
std::pair<int, int> Tower::getPos ( ) [inline]
```

7.18.2.6 GetPrice()

```
int Tower::GetPrice ( ) [static]
```

7.18.2.7 GetRange()

```
int Tower::GetRange ( ) const
```

7.18.2.8 GetSpeed()

```
int Tower::GetSpeed ( ) const
```

7.18.2.9 GetTargetTiles()

```
void Tower::GetTargetTiles (
             std::vector< std::vector< Tile * >> * level ) [virtual]
```

Gets all the path tiles where the tower can shoot, and adds them to canSee_, in descending order. Takes no input and returns nothing.

Parameters

level	The level where the tiles are

Reimplemented in King, Pawn, Queen, and Rook.

7.18.2.10 GetType()

```
std::string Tower::GetType ( ) const
```

7.18.2.11 Shoot()

```
std::pair< int, Tile * > Tower::Shoot ( ) [virtual]
```

Shoots at enemies. Done damage and cooldown after attack is done internally, so this method can be called anytime.

Reimplemented in King, Pawn, Queen, and Rook.

7.18.2.12 StaticGetAllTargetTiles()

```
std::vector< Tile * > Tower::StaticGetAllTargetTiles (
            std::vector< std::vector< Tile * >> * level,
            int x,
            int y ) [static]
```

Statically gets all the tiles where the tower can shoot, and returns them.

Parameters

level	Level where tiles are
x	x of the tower
y	y of the tower

Returns

returns vector of Tile∗ where tower can see

7.18.2.13 Upgrade()

```
bool Tower::Upgrade (
            int money,
            std::vector< std::vector< Tile * >> * level )
```

Upgrades the tower to the next level if possible.

Parameters

money	The money the player has. HOX! This method does not change the amount of money the player has, should be handled by gamestate!

Returns

Returns true if the upgrade was successful. Otherwise returns false

7.18.3 Friends And Related Function Documentation

7.18.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & stream,
            Tower & tower )  [friend]
```

7.18.4 Member Data Documentation

7.18.4.1 canSee_

```
std::vector<PathTile *> Tower::canSee_  [protected]
```

7.18.4.2 coolDown_

```
int Tower::coolDown_  [protected]
```

7.18.4.3 damage_

```
int Tower::damage_  [protected]
```

7.18.4.4 level_

```
int Tower::level_  [protected]
```

7.18.4.5 price_

```
int Tower::price_  [protected]
```

7.18.4.6 range_

```
int Tower::range_  [protected]
```

7.18.4.7 speed_

```
int Tower::speed_  [protected]
```

7.18.4.8 type_

```
std::string Tower::type_  [protected]
```

7.18.4.9 upgradeTable_

```
std::vector<std::tuple<int, upgradeType, int> >& Tower::upgradeTable_  [protected]
```

7.18.4.10 x_

```
int Tower::x_  [protected]
```

7.18.4.11 y_

```
int Tower::y_ [protected]
```

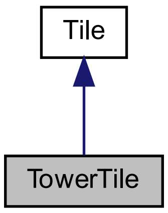
The documentation for this class was generated from the following files:

- src/towers/tower.hpp
- src/towers/tower.cpp

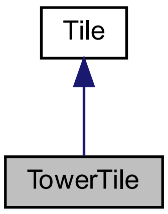
7.19 TowerTile Class Reference

```
#include <TowerTile.hpp>
```

Inheritance diagram for TowerTile:



Collaboration diagram for TowerTile:



Public Member Functions

- TowerTile ()

    TowerTile class constructor. TowerTile inherits the Tile class. Represents the towers on the map.
- ∼TowerTile ()

    TowerTile destructor. Deletes tower pointer.
- Tower ∗ GetTower ()
- void SetTower (Tower ∗tower)

Protected Member Functions

- virtual void Print (std::ostream &out) const

  Print tower info to terminal.

7.19.1 Constructor & Destructor Documentation

7.19.1.1 TowerTile()

```
TowerTile::TowerTile ( )
```

TowerTile class constructor. TowerTile inherits the Tile class. Represents the towers on the map.

7.19.1.2 ∼TowerTile()

```
TowerTile::∼TowerTile ( )
```

TowerTile destructor. Deletes tower pointer.

7.19.2 Member Function Documentation

7.19.2.1 GetTower()

```
Tower* TowerTile::GetTower ( )  [inline]
```

7.19.2.2 Print()

```
void TowerTile::Print (
             std::ostream & out ) const  [protected], [virtual]
```

Print tower info to terminal.

Parameters

out	ostream

Reimplemented from Tile.

7.19.2.3 SetTower()

```
void TowerTile::SetTower (
            Tower * tower )  [inline]
```

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

- src/tiles/TowerTile.hpp
- src/tiles/TowerTile.cpp

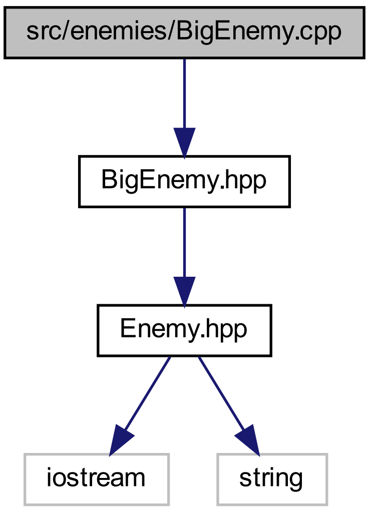
Chapter 8

File Documentation

8.1 src/enemies/BigEnemy.cpp File Reference

```
#include "BigEnemy.hpp"
```

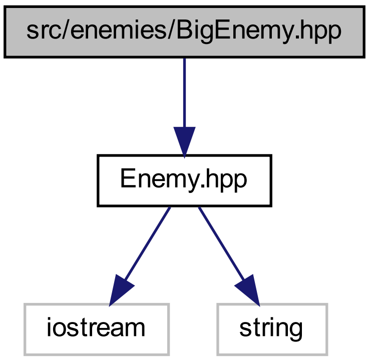
Include dependency graph for BigEnemy.cpp:



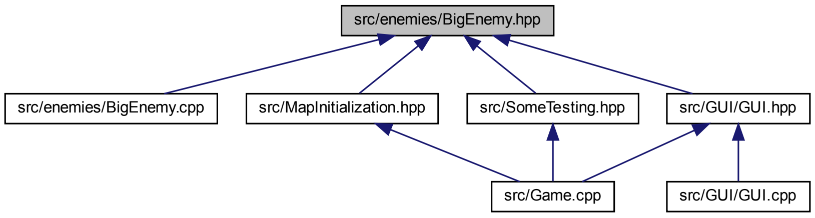
8.2 src/enemies/BigEnemy.hpp File Reference

```
#include "Enemy.hpp"
```

Include dependency graph for BigEnemy.hpp:



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



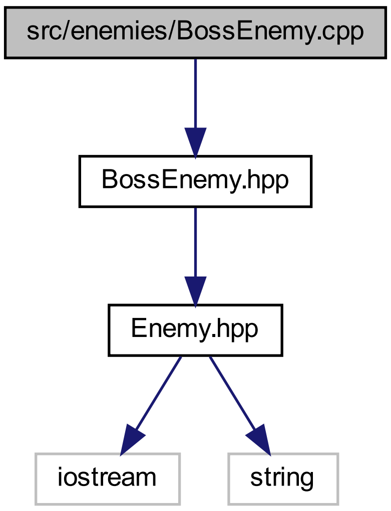
Classes

- class BigEnemy

8.3 src/enemies/BossEnemy.cpp File Reference

```
#include "BossEnemy.hpp"
```

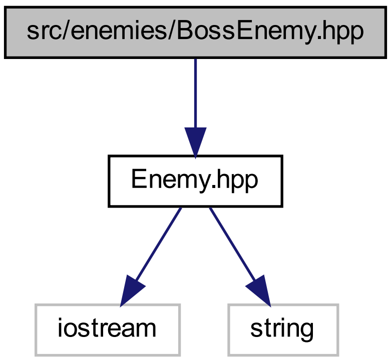
Include dependency graph for BossEnemy.cpp:



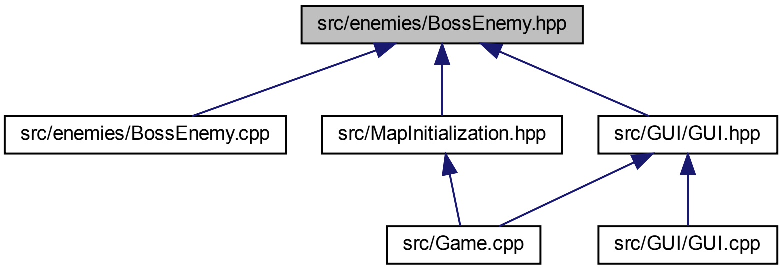
8.4 src/enemies/BossEnemy.hpp File Reference

```
#include "Enemy.hpp"
```

Include dependency graph for BossEnemy.hpp:



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


src/enemies/BossEnemy.hpp
src/enemies/BossEnemy.cpp
src/MapInitialization.hpp
src/GUI/GUI.hpp
src/Game.cpp
src/GUI/GUI.cpp

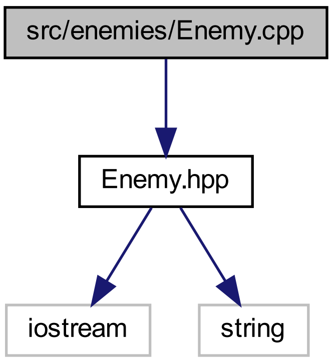
Classes

- class BossEnemy

8.5 src/enemies/Enemy.cpp File Reference

```
#include "Enemy.hpp"
```

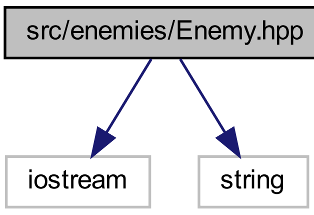
Include dependency graph for Enemy.cpp:



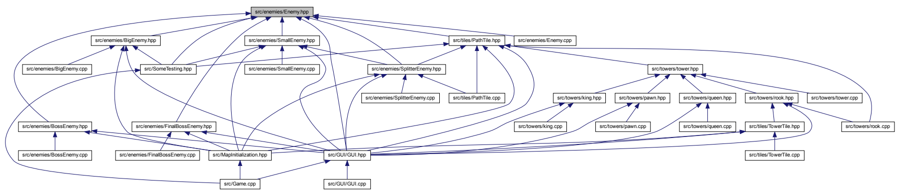
8.6 src/enemies/Enemy.hpp File Reference

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

Include dependency graph for Enemy.hpp:



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



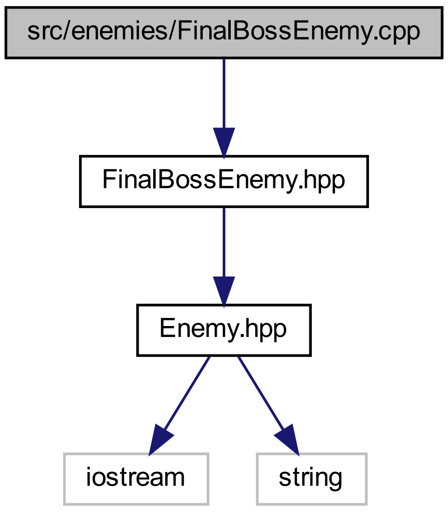
Classes

- class Enemy

8.7 src/enemies/FinalBossEnemy.cpp File Reference

```
#include "FinalBossEnemy.hpp"
```

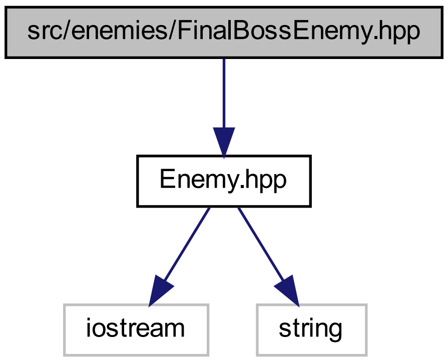
Include dependency graph for FinalBossEnemy.cpp:



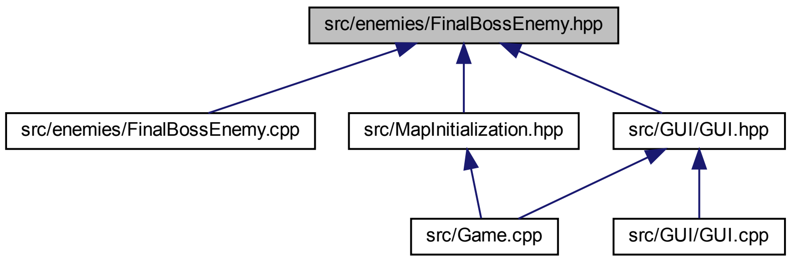
8.8 src/enemies/FinalBossEnemy.hpp File Reference

```
#include "Enemy.hpp"
```

Include dependency graph for FinalBossEnemy.hpp:



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


src/enemies/FinalBossEnemy.hpp
src/enemies/FinalBossEnemy.cpp
src/MapInitialization.hpp
src/GUI/GUI.hpp
src/Game.cpp
src/GUI/GUI.cpp

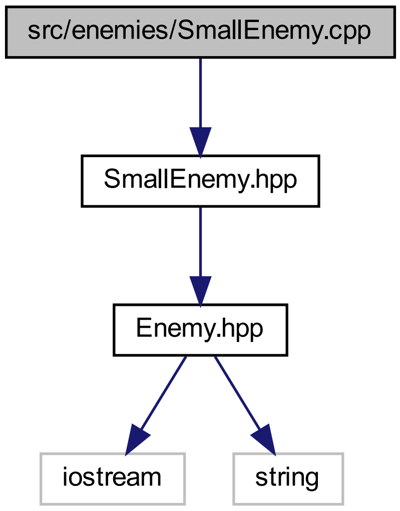
Classes

- class FinalBoss

8.9 src/enemies/SmallEnemy.cpp File Reference

```
#include "SmallEnemy.hpp"
```

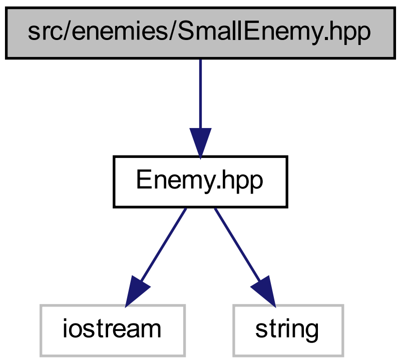
Include dependency graph for SmallEnemy.cpp:



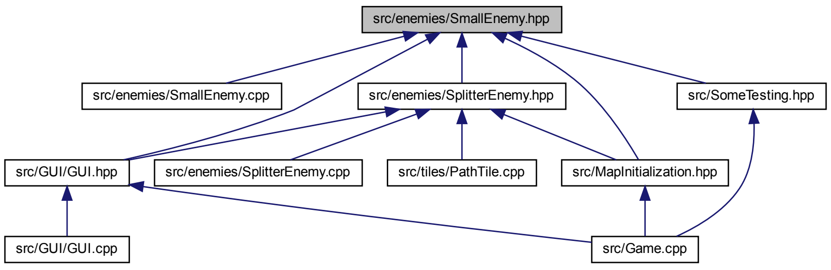
8.10 src/enemies/SmallEnemy.hpp File Reference

```
#include "Enemy.hpp"
```

Include dependency graph for SmallEnemy.hpp:



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


src/enemies/SmallEnemy.hpp
src/enemies/SmallEnemy.cpp
src/enemies/SplitterEnemy.hpp
src/SomeTesting.hpp
src/GUI/GUI.hpp
src/enemies/SplitterEnemy.cpp
src/tiles/PathTile.cpp
src/MapInitialization.hpp
src/GUI/GUI.cpp
src/Game.cpp

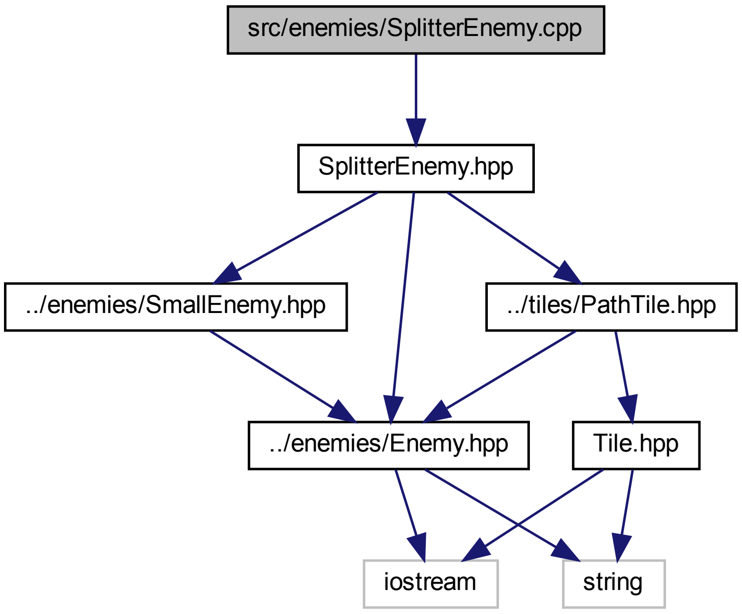
Classes

- class SmallEnemy

8.11 src/enemies/SplitterEnemy.cpp File Reference

```
#include "SplitterEnemy.hpp"
```

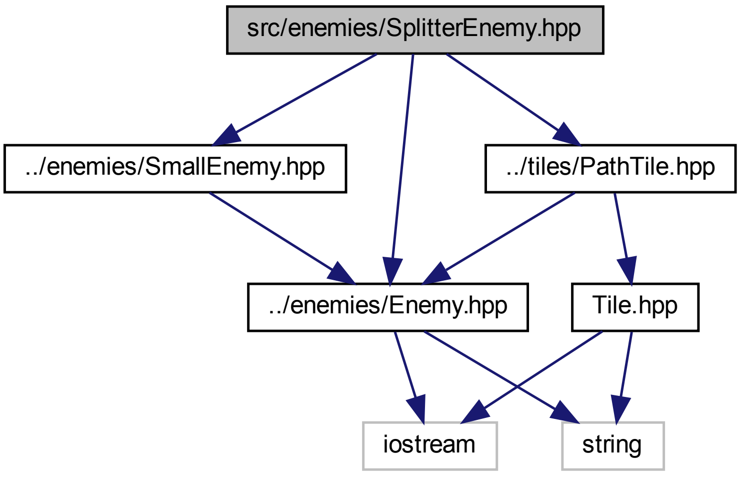
Include dependency graph for SplitterEnemy.cpp:



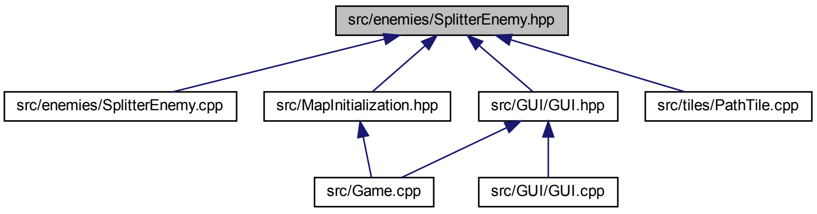
8.12 src/enemies/SplitterEnemy.hpp File Reference

```
#include "../tiles/PathTile.hpp"
#include "Enemy.hpp"
#include "../enemies/SmallEnemy.hpp"
```

Include dependency graph for SplitterEnemy.hpp:



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


src/enemies/SplitterEnemy.hpp
src/enemies/SplitterEnemy.cpp
src/MapInitialization.hpp
src/GUI/GUI.hpp
src/tiles/PathTile.cpp
src/Game.cpp
src/GUI/GUI.cpp

Classes

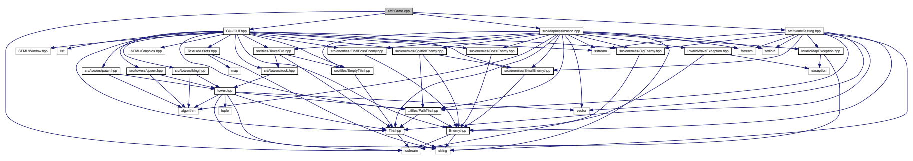
- class SplitterEnemy

8.13 src/Game.cpp File Reference

```
#include <iostream>
#include "GUI/GUI.hpp"
#include "src/MapInitialization.hpp"
```

`#include "src/SomeTesting.hpp"`

Include dependency graph for Game.cpp:



Functions

- std::string MainLoop (std::vector< std::vector< Tile ∗ >> tiles, std::vector< std::vector< Enemy ∗ >> waves)

  Game main loop.
- int main ()

8.13.1 Function Documentation

8.13.1.1 main()

```
int main ( )
```

8.13.1.2 MainLoop()

```
std::string MainLoop (
            std::vector< std::vector< Tile ∗ >> tiles,
            std::vector< std::vector< Enemy ∗ >> waves )
```

Game main loop.

Parameters

tiles	
waves	

Returns

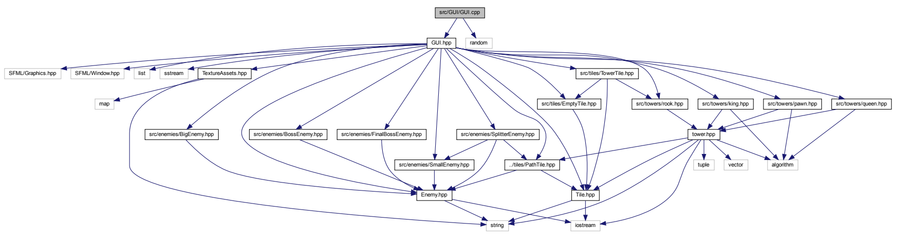
string that tells the status of game level. "win", "gameover" or "quit". Win: the current level/map is cleared, Gameover: enemy got into the last pathTile, Quit: user quitted the game

8.14 src/GUI/GUI.cpp File Reference

`#include "GUI.hpp"`

```
#include <random>
```

Include dependency graph for GUI.cpp:



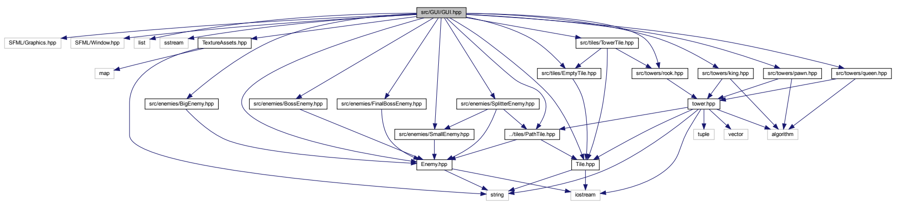
Classes

- class InvalidFontException

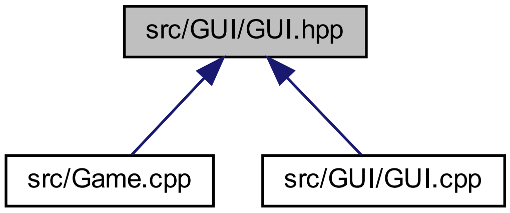
8.15 src/GUI/GUI.hpp File Reference

```
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include <list>
#include <sstream>
#include "TextureAssets.hpp"
#include "src/enemies/BigEnemy.hpp"
#include "src/enemies/BossEnemy.hpp"
#include "src/enemies/FinalBossEnemy.hpp"
#include "src/enemies/Enemy.hpp"
#include "src/enemies/SmallEnemy.hpp"
#include "src/enemies/SplitterEnemy.hpp"
#include "src/tiles/EmptyTile.hpp"
#include "src/tiles/PathTile.hpp"
#include "src/tiles/Tile.hpp"
#include "src/tiles/TowerTile.hpp"
#include "src/towers/king.hpp"
#include "src/towers/pawn.hpp"
#include "src/towers/queen.hpp"
#include "src/towers/rook.hpp"
```

Include dependency graph for GUI.hpp:



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



Classes

- class GUI

Macros

- #define TILESIZE 50
- #define ENEMYSIZE 40
- #define MENUWIDTH 200
- #define SHELLSTEPS 15
- #define FLIGHTTIME 250

Enumerations

- enum SelectMode { none, empty, tower }

  Depicts different tile selection modes:
- enum TowerTypes {
  NoTower, pawn, king, rook,
  queen }

  Different tower types to tell which one is currently being placed.

8.15.1 Macro Definition Documentation

8.15.1.1 ENEMYSIZE

```
#define ENEMYSIZE 40
```

8.15.1.2 FLIGHTTIME

```
#define FLIGHTTIME 250
```

8.15.1.3 MENUWIDTH

```
#define MENUWIDTH 200
```

8.15.1.4 SHELLSTEPS

```
#define SHELLSTEPS 15
```

8.15.1.5 TILESIZE

```
#define TILESIZE 50
```

8.15.2 Enumeration Type Documentation

8.15.2.1 SelectMode

```
enum SelectMode
```

Depicts different tile selection modes:

Parameters

empty	A tower is being held over a tile viable for placement.
tower	A tower tile with a tower is selected.
none	No tile is selected

Enumerator

none	
empty	
tower	

8.15.2.2 TowerTypes

```
enum TowerTypes
```

Different tower types to tell which one is currently being placed.

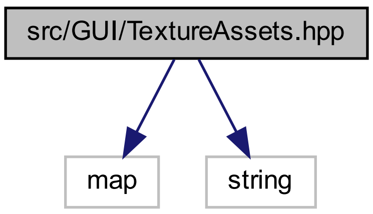
Enumerator

NoTower	
pawn	
king	
rook	
queen	

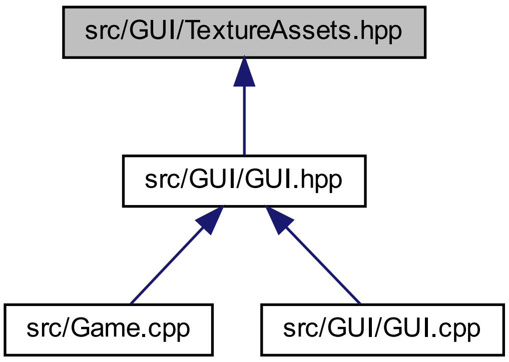
8.16 src/GUI/TextureAssets.hpp File Reference

```
#include <map>
#include <string>
```

Include dependency graph for TextureAssets.hpp:



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



Namespaces

- TDImageFiles

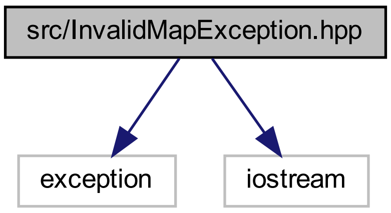
Variables

- const std::map< const std::string, const std::string > TDImageFiles::Textures

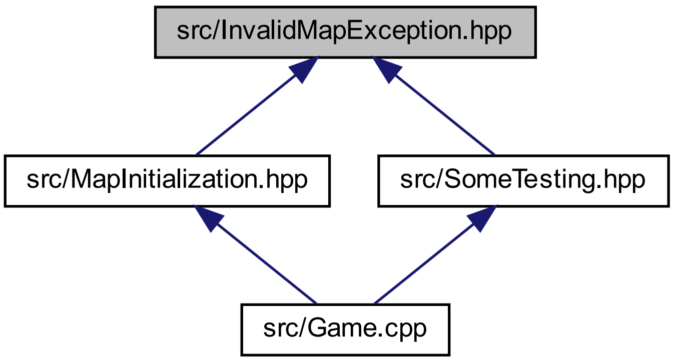
8.17 src/InvalidMapException.hpp File Reference

```
#include <exception>
#include <iostream>
```

Include dependency graph for InvalidMapException.hpp:



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



Classes

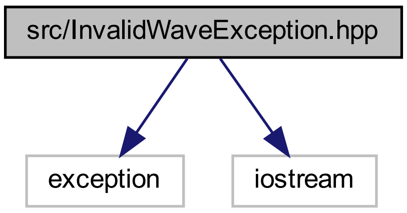
- class InvalidMapException

  InvalidMapException class that derives std library exception class.

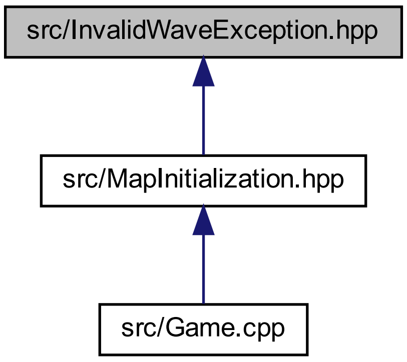
8.18 src/InvalidWaveException.hpp File Reference

```
#include <exception>
#include <iostream>
```

Include dependency graph for InvalidWaveException.hpp:



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



Classes

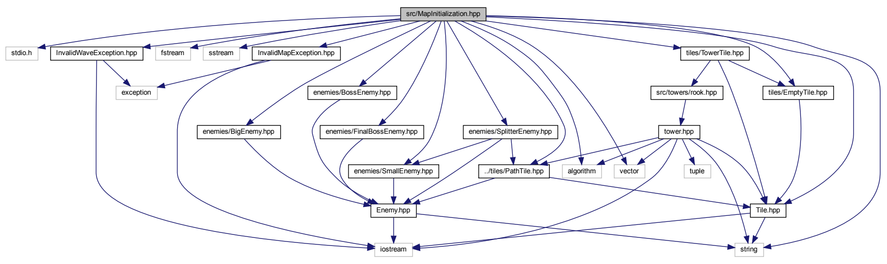
- class InvalidWaveException

    InvalidWaveException class that derives std library exception class.

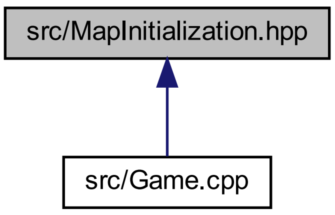
8.19 src/MapInitialization.hpp File Reference

```
#include <stdio.h>
#include <algorithm>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include "InvalidMapException.hpp"
#include "InvalidWaveException.hpp"
#include "enemies/BigEnemy.hpp"
#include "enemies/BossEnemy.hpp"
#include "enemies/FinalBossEnemy.hpp"
#include "enemies/SmallEnemy.hpp"
#include "enemies/SplitterEnemy.hpp"
#include "tiles/EmptyTile.hpp"
#include "tiles/PathTile.hpp"
#include "tiles/Tile.hpp"
#include "tiles/TowerTile.hpp"
```

Include dependency graph for MapInitialization.hpp:



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



Functions

- vector< string > SplitString (string s, char c)

  Splits a string into vector using a character as the split delimiter.
- void PrintVector (vector< string > strings)

  Prints vector of strings. Useful for debugging.
- void PrintVector (vector< Tile ∗ > ∗tiles)

  Prints vector of tiles. Useful for debugging.
- void PrintVector (vector< PathTile ∗ > ∗tiles)

  Prints vector of path tiles (not in order). Useful for debugging.
- void PrintVector (vector< Enemy ∗ > ∗enemies)

  Prints vector of enemies. Useful for debugging.
- Tile ∗ GetTilePtr (string type)

  Create new pointer to correct Tile class.
- Enemy ∗ GetEnemyPtr (string type)

  Create new pointer to correct Enemy class.
- bool CreateConnectionsInPathTiles (vector< PathTile ∗ > ∗pathTiles)

  Creates the "doubly linked list" between pathTiles.
- bool MapIsValid (vector< vector< Tile ∗ >> ∗map)

  Checks that map is valid and PathTiles are next to each other in order and map's rows are all same size. NOTICE!! that CreateConnectionsInPathTiles() -method must be called before for this to work correctly.
- pair< vector< vector< Tile ∗ > >, vector< vector< Enemy ∗ > > > GenerateMapAndWaves (string file)

  Generates game map as 2D matrix and enemy waves. Wave enemies are reversed in vector (first enemy is the last element of vector).

8.19.1 Function Documentation

8.19.1.1 CreateConnectionsInPathTiles()

```
bool CreateConnectionsInPathTiles (
            vector< PathTile * > * pathTiles )  [inline]
```

Creates the "doubly linked list" between pathTiles.

Parameters

pathTiles	vector<PathTiles*>*

Returns

true if success, otherwise false

8.19.1.2 GenerateMapAndWaves()

```
pair<vector<vector<Tile*> >, vector<vector<Enemy*> > > GenerateMapAndWaves (
            string file )  [inline]
```

Generates game map as 2D matrix and enemy waves. Wave enemies are reversed in vector (first enemy is the last element of vector).

Parameters

file	path to file

Returns

pair<vector<vector<Tile*>>, vector<vector<Enemy*>>>

8.19.1.3 GetEnemyPtr()

```
Enemy* GetEnemyPtr (
            string type )  [inline]
```

Create new pointer to correct Enemy class.

Parameters

type	abbreviation of enemy types ("s", "b" or 'sp)

Returns

Enemy∗

8.19.1.4 GetTilePtr()

```
Tile∗ GetTilePtr (
             string type )  [inline]
```

Create new pointer to correct Tile class.

Parameters

type	abbreviation of tile types ("e", "t" or digit as path)

Returns

Tile∗

8.19.1.5 MapIsValid()

```
bool MapIsValid (
             vector< vector< Tile ∗ >> ∗ map )  [inline]
```

Checks that map is valid and PathTiles are next to each other in order and map's rows are all same size. NOTICE!! that CreateConnectionsInPathTiles() -method must be called before for this to work correctly.

Parameters

map	2D tile vector

Returns

true if path is valid

8.19.1.6 PrintVector() [1/4]

```
void PrintVector (
             vector< Enemy ∗ > ∗ enemies )  [inline]
```

Prints vector of enemies. Useful for debugging.

Parameters

enemies	tile pointers in vector

8.19.1.7 PrintVector() [2/4]

```
void PrintVector (
            vector< PathTile * > * tiles )  [inline]
```

Prints vector of path tiles (not in order). Useful for debugging.

Parameters

tiles	tile pointers in vector

8.19.1.8 PrintVector() [3/4]

```
void PrintVector (
            vector< string > strings )  [inline]
```

Prints vector of strings. Useful for debugging.

Parameters

strings	

8.19.1.9 PrintVector() [4/4]

```
void PrintVector (
            vector< Tile * > * tiles )  [inline]
```

Prints vector of tiles. Useful for debugging.

Parameters

tiles	tile pointers in vector

8.19.1.10 SplitString()

```
vector<string> SplitString (
            string s,
            char c )  [inline]
```

Splits a string into vector using a character as the split delimiter.

Parameters

s	String to split
c	The delimeter char

Returns

vector of strings

8.20 src/maps/map1.txt File Reference

Variables

- Map __pad0__
- Map e
- Map e e e e e e e e e e e e e e e e e e e e Waves
- Map e e e e e e e e e e e e e e e e e e e e s
- Map e e e e e e e e e e e e e e e e e e e e b
- Map e e e e e e e e e e e e e e e e e e e e sp

8.20.1 Variable Documentation

8.20.1.1 __pad0__

```
Map __pad0__
```

8.20.1.2 b

```
Map t t t b
```

8.20.1.3 e

```
Map e
```

8.20.1.4 s

```
Map t t t s
```

8.20.1.5 sp

```
Map t t t sp
```

8.20.1.6 Waves

```
Map e e e e e e e e e e e e e e e e e e e e Waves
```

8.21 src/maps/map2.txt File Reference

Variables

- Map __pad1__
- Map e
- Map e e e e e e e e e e e e e e e e e e e e Waves
- Map e e e e e e e e e e e e e e e e e e e e b
- Map e e e e e e e e e e e e e e e e e e e e s
- Map e e e e e e e e e e e e e e e e e e e e sp

8.21.1 Variable Documentation

8.21.1.1 __pad1__

```
Map __pad1__
```

8.21.1.2 b

```
Map e e e e e e e e e e e e e e e e e e e e b
```

8.21.1.3 e

```
Map e e e e e e e e e e e e e e e e e e e e
```

8.21.1.4 s

```
Map e e e e e e e e e e e e e e e e e e e e s
```

8.21.1.5 sp

```
Map e e e e e e e e e e e e e e e e e e e e sp
```

8.21.1.6 Waves

```
Map e e e e e e e e e e e e e e e e e e e e Waves
```

8.22 src/maps/map3.txt File Reference

Variables

- Map __pad2__
- Map e
- Map e e e e e e e e e e e e e e e e e e Waves
- Map e e e e e e e e e e e e e e e e e e b
- Map e e e e e e e e e e e e e e e e e e s
- Map e e e e e e e e e e e e e e e e e e sp

8.22.1 Variable Documentation

8.22.1.1 __pad2__

```
Map __pad2__
```

8.22.1.2 b

```
Map e e e e e e e e e e e e e e e e e e b
```

8.22.1.3 e

```
Map e e e e e e e e e e e e e e e e e e
```

8.22.1.4 s

```
Map e e e e e e e e e e e e e e e e e e s
```

8.22.1.5 sp

```
Map e e e e e e e e e e e e e e e e e e sp
```

8.22.1.6 Waves

```
Map e e e e e e e e e e e e e e e e e e Waves
```

8.23 src/maps/map4.txt File Reference

Variables

- Map __pad3__
- Map e
- Map e e e e e e e e e e e e e e e e e e e e Waves
- Map e e e e e e e e e e e e e e e e e e e e s
- Map e e e e e e e e e e e e e e e e e e e e sp
- Map e e e e e e e e e e e e e e e e e e e e b
- Map e e e e e e e e e e e e e e e e e e e e m

8.23.1 Variable Documentation

8.23.1.1 __pad3__

```
Map __pad3__
```

8.23.1.2 b

```
Map e e e e e e e e e e e e e e e e e e e e b
```

8.23.1.3 e

```
Map e e e e e e e e e e e e e e e e e e e e
```

8.23.1.4 m

```
Map e e e e e e e e e e e e e e e e e e e e m
```

8.23.1.5 s

```
Map e e e e e e e e e e e e e e e e e e e e s
```

8.23.1.6 sp

```
Map e e e e e e e e e e e e e e e e e e e e sp
```

8.23.1.7 Waves

```
Map e e e e e e e e e e e e e e e e e e e e Waves
```

8.24 src/maps/map5.txt File Reference

Variables

- Map __pad4__
- Map e
- Map e e e e e e e e e e e e e e e e e e e Waves
- Map e e e e e e e e e e e e e e e e e e e s
- Map e e e e e e e e e e e e e e e e e e e b
- Map e e e e e e e e e e e e e e e e e e e sp

8.24.1 Variable Documentation

8.24.1.1 __pad4__

```
Map __pad4__
```

8.24.1.2 b

```
Map e e e e e e e e e e e e e e e e e e e b
```

8.24.1.3 e

```
Map e e e e e e e e e e e e e e e e e e e
```

8.24.1.4 s

```
Map e e e e e e e e e e e e e e e e e e e s
```

8.24.1.5 sp

```
Map e e e e e e e e e e e e e e e e e e e sp
```

8.24.1.6 Waves

```
Map e e e e e e e e e e e e e e e e e e e Waves
```

8.25 src/maps/test_map.txt File Reference

Variables

- Map __pad5__
- Map t
- Map e
- Map t t t Waves
- Map t t t s
- Map t t t b
- Map t t t sp

8.25.1 Variable Documentation

8.25.1.1 __pad5__

```
Map __pad5__
```

8.25.1.2 b

```
Map t t t b
```

8.25.1.3 e

```
Map e
```

8.25.1.4 s

```
Map t t t s
```

8.25.1.5 sp

```
Map t t t sp
```

8.25.1.6 t

```
Map t t t
```

8.25.1.7 Waves

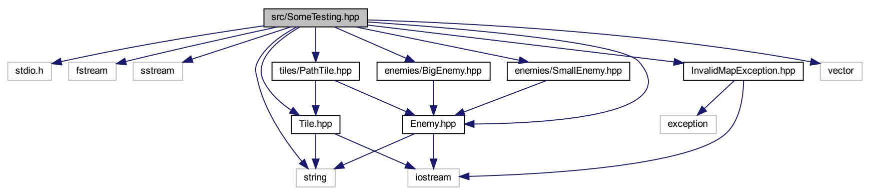
```
Map t t t Waves
```

8.26 src/readme.md File Reference

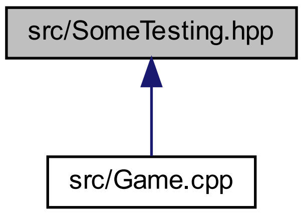
8.27 src/SomeTesting.hpp File Reference

```
#include <stdio.h>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include "InvalidMapException.hpp"
#include "enemies/BigEnemy.hpp"
#include "enemies/Enemy.hpp"
#include "enemies/SmallEnemy.hpp"
#include "tiles/PathTile.hpp"
#include "tiles/Tile.hpp"
```

Include dependency graph for SomeTesting.hpp:



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



Functions

- PathTile ∗ GetFirstPathTile (vector< vector< Tile ∗ >> ∗map)

  Find first PathTile from map.
- bool MoveEnemiesAndCheckGameover (PathTile ∗tile, Enemy ∗enemy=nullptr)

  Move all enemies forward one tile, and check if gameover.
- bool CheckIfAllEnemiesDied (PathTile ∗firstPathTile)

  Checks if all enemies are died.

8.27.1 Function Documentation

8.27.1.1 CheckIfAllEnemiesDied()

```
bool CheckIfAllEnemiesDied (
            PathTile * firstPathTile )  [inline]
```

Checks if all enemies are died.

Parameters

firstPathTile	

Returns

boolean

8.27.1.2 GetFirstPathTile()

```
PathTile* GetFirstPathTile (
            vector< vector< Tile * >> * map )  [inline]
```

Find first PathTile from map.

Parameters

map	2D tile vector

Returns

first PathTile

8.27.1.3 MoveEnemiesAndCheckGameover()

```
bool MoveEnemiesAndCheckGameover (
            PathTile * tile,
            Enemy * enemy = nullptr )  [inline]
```

Move all enemies forward one tile, and check if gameover.

Parameters

tile	first PathTile in map
enemy	New enemy that is added to map, if recursion is called first time.

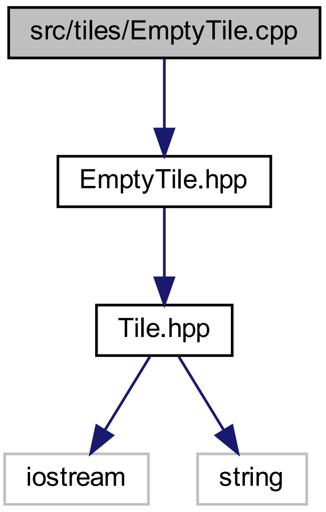
Returns

true if gameover

8.28 src/tiles/EmptyTile.cpp File Reference

```
#include "EmptyTile.hpp"
```

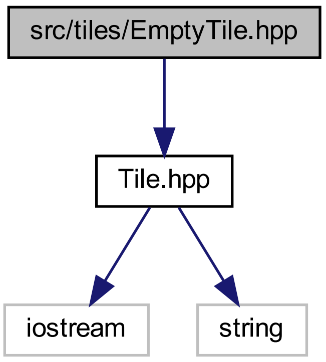
Include dependency graph for EmptyTile.cpp:



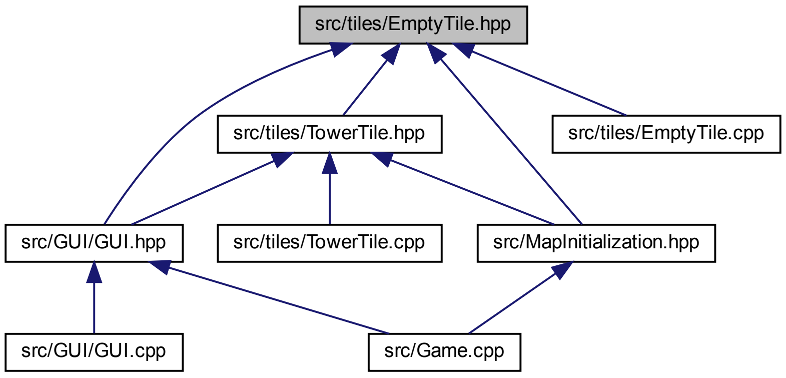
8.29 src/tiles/EmptyTile.hpp File Reference

```
#include "Tile.hpp"
```

Include dependency graph for EmptyTile.hpp:



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



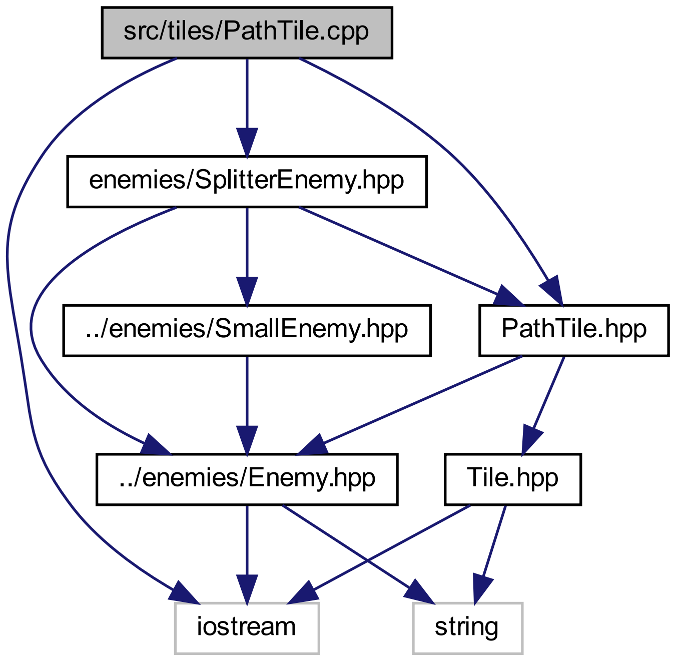
Classes

- class EmptyTile

8.30 src/tiles/PathTile.cpp File Reference

```
#include "PathTile.hpp"
#include <iostream>
#include "enemies/SplitterEnemy.hpp"
```

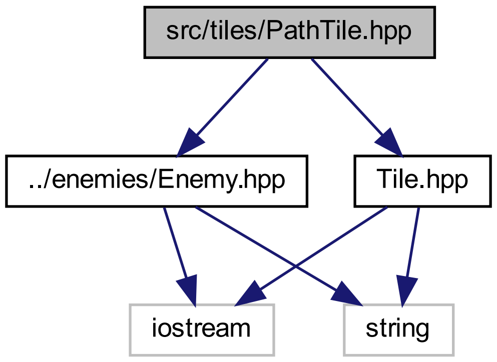
Include dependency graph for PathTile.cpp:



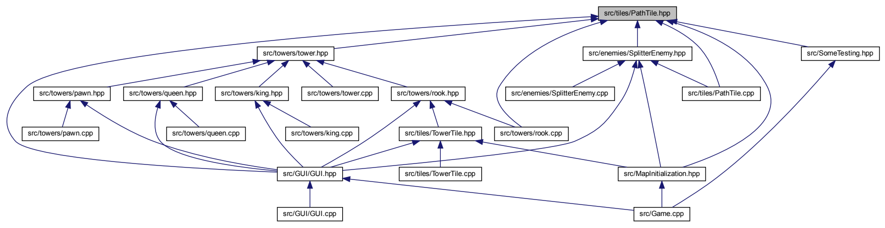
8.31 src/tiles/PathTile.hpp File Reference

```
#include "../enemies/Enemy.hpp"
#include "Tile.hpp"
```

Include dependency graph for PathTile.hpp:



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



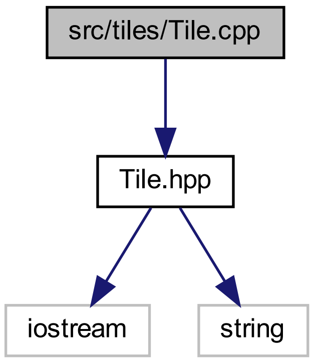
Classes

- class PathTile

8.32 src/tiles/Tile.cpp File Reference

```
#include "Tile.hpp"
```

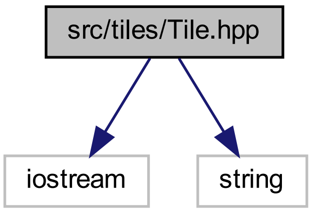
Include dependency graph for Tile.cpp:



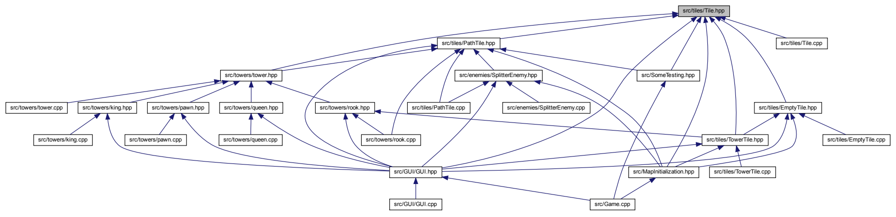
8.33 src/tiles/Tile.hpp File Reference

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

Include dependency graph for Tile.hpp:



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



Classes

- class Tile

Functions

- std::ostream & operator<< (std::ostream &out, const Tile &t)

8.33.1 Function Documentation

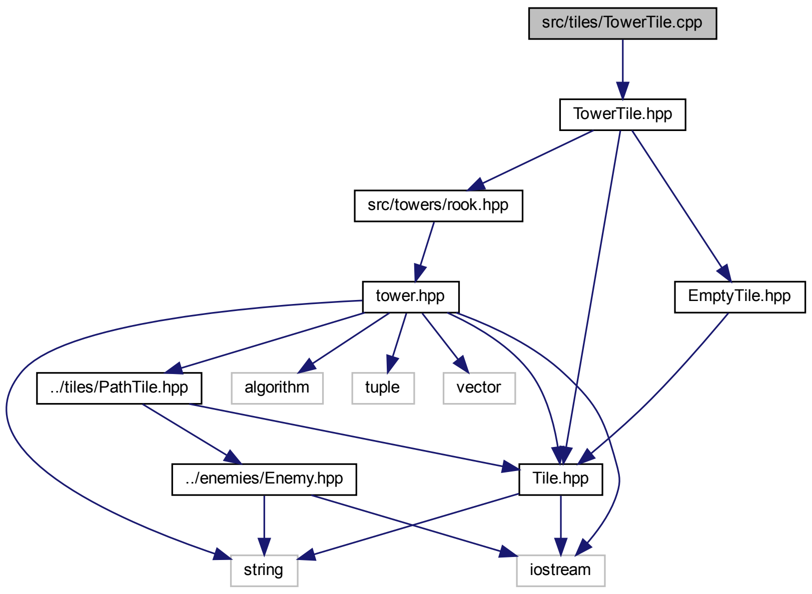
8.33.1.1 operator<<()

```
std::ostream& operator<< (
            std::ostream & out,
            const Tile & t )  [inline]
```

8.34 src/tiles/TowerTile.cpp File Reference

```
#include "TowerTile.hpp"
```

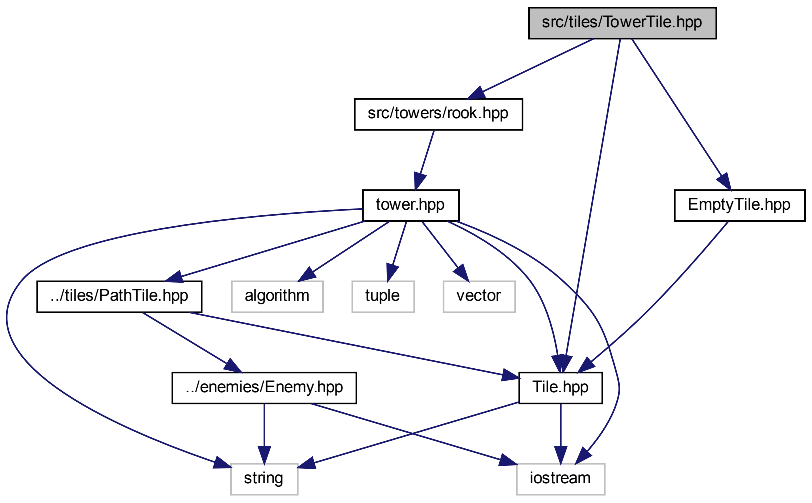
Include dependency graph for TowerTile.cpp:



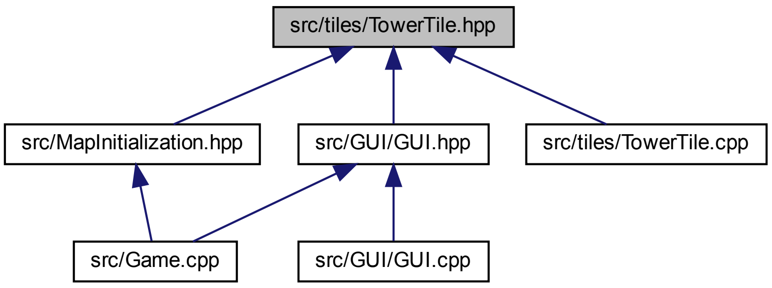
8.35 src/tiles/TowerTile.hpp File Reference

```
#include "EmptyTile.hpp"
#include "Tile.hpp"
#include "src/towers/rook.hpp"
```

Include dependency graph for TowerTile.hpp:



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



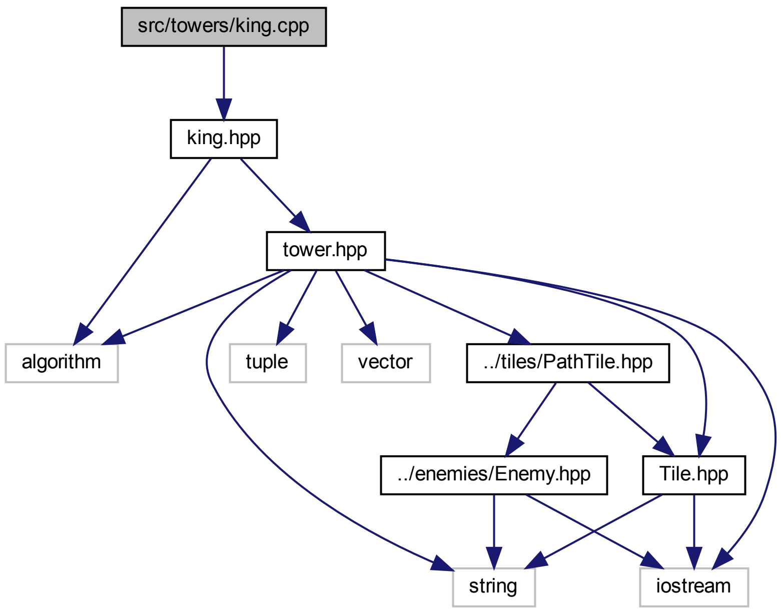
Classes

- class TowerTile

8.36 src/towers/king.cpp File Reference

```
#include "king.hpp"
```

Include dependency graph for king.cpp:



Variables

- std::vector< std::tuple< int, upgradeType, int > > upgradeTableKing

8.36.1 Variable Documentation

8.36.1.1 upgradeTableKing

```
std::vector<std::tuple<int, upgradeType, int> > upgradeTableKing
```

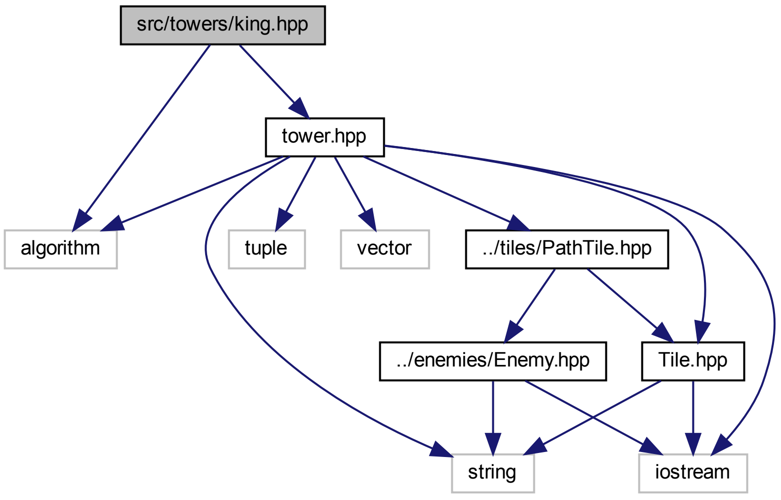
Initial value:

```
{
    {100, range, 1}, {200, damage, 1}, {300, speed, 3}}
```

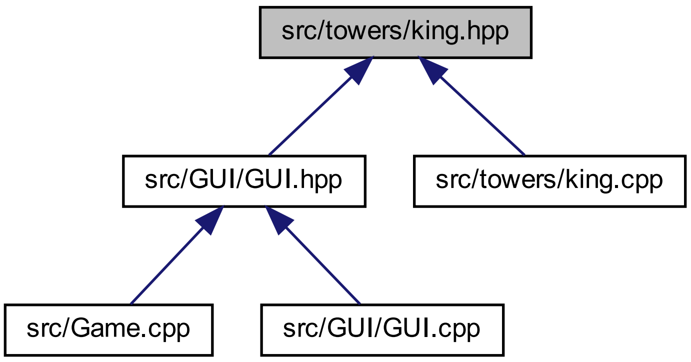
8.37 src/towers/king.hpp File Reference

```
#include <algorithm>
#include "tower.hpp"
```

Include dependency graph for king.hpp:



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



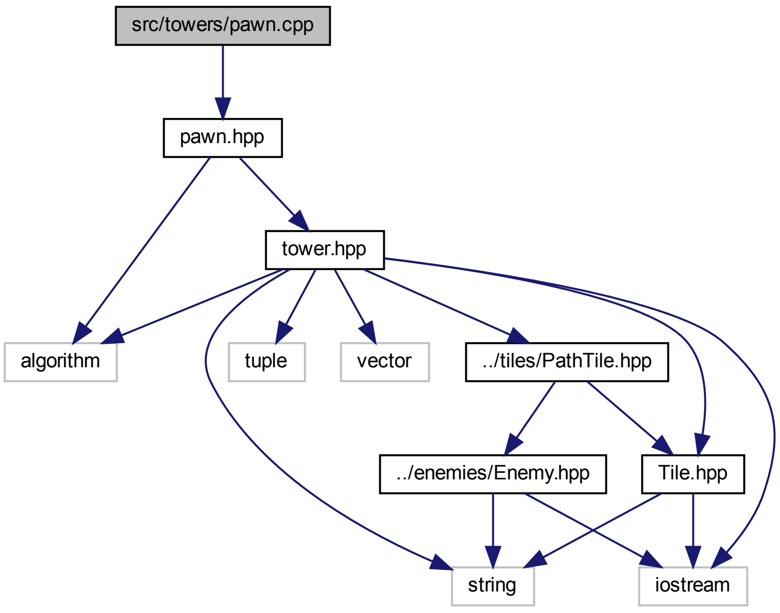
Classes

- class King

8.38 src/towers/pawn.cpp File Reference

```
#include "pawn.hpp"
```

Include dependency graph for pawn.cpp:



Variables

- std::vector< std::tuple< int, upgradeType, int > > upgradeTablePawn

8.38.1 Variable Documentation

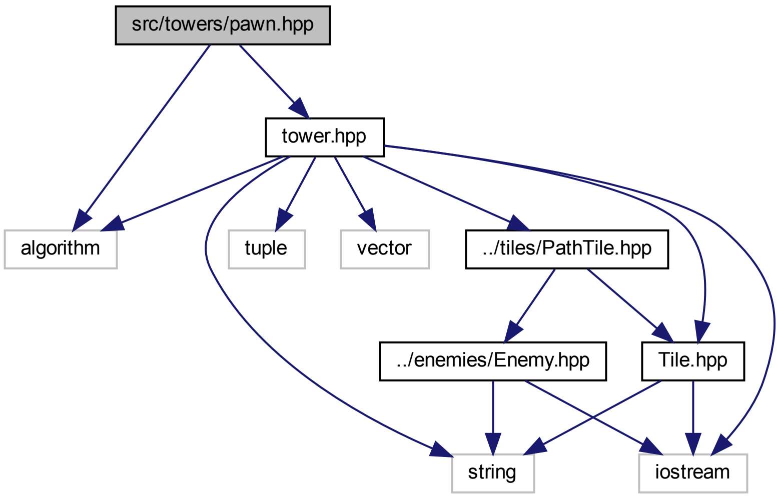
8.38.1.1 upgradeTablePawn

```
std::vector<std::tuple<int, upgradeType, int> > upgradeTablePawn
```

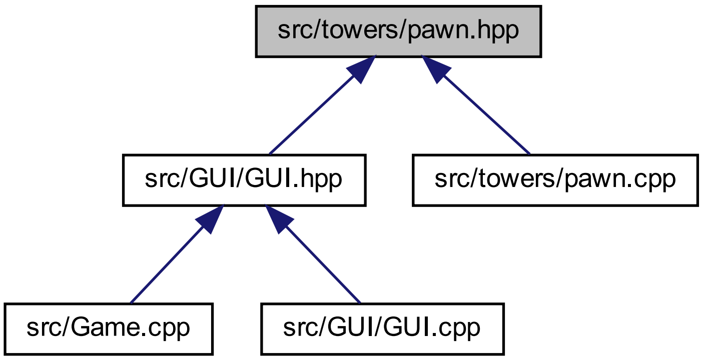
Initial value:
```
{
    {50, speed, 1}, {100, damage, 1}, {200, speed, 2}}
```

8.39 src/towers/pawn.hpp File Reference

```
#include <algorithm>
#include "tower.hpp"
```
Include dependency graph for pawn.hpp:



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



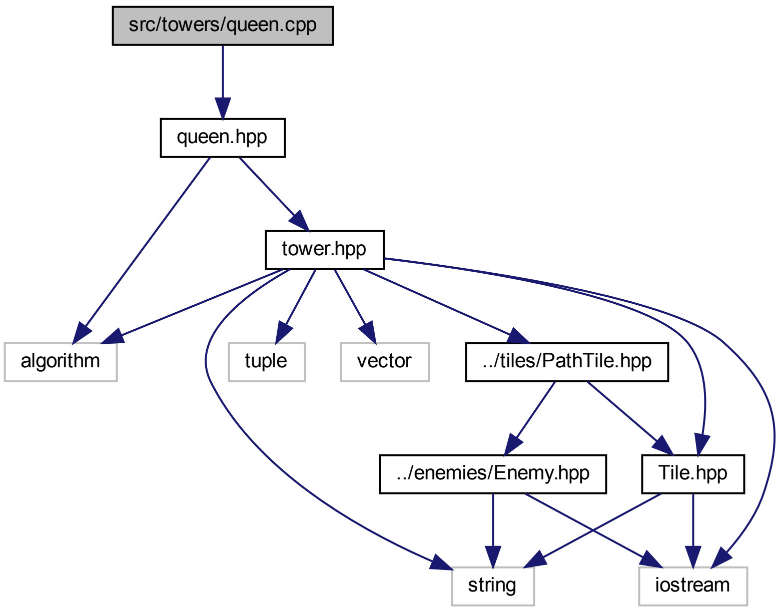
Classes

- class Pawn

8.40 src/towers/queen.cpp File Reference

```
#include "queen.hpp"
```

Include dependency graph for queen.cpp:



Variables

- std::vector< std::tuple< int, upgradeType, int > > upgradeTableQueen

8.40.1 Variable Documentation

8.40.1.1 upgradeTableQueen

```
std::vector<std::tuple<int, upgradeType, int> > upgradeTableQueen
```

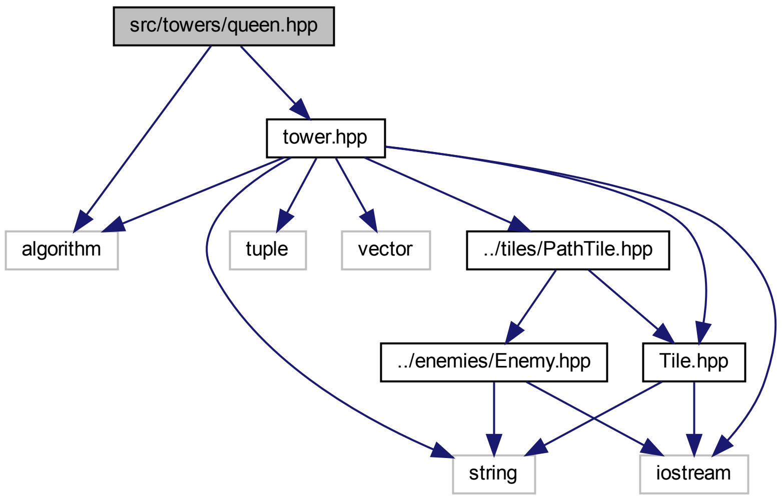
Initial value:

```
{
    {250, range, 2}, {500, damage, 2}, {300, speed, 5}}
```

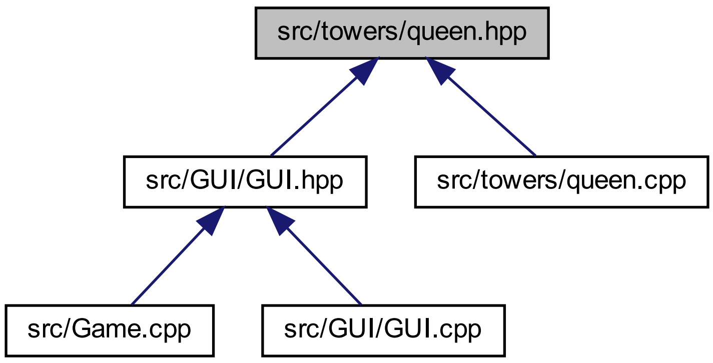
8.41 src/towers/queen.hpp File Reference

```
#include <algorithm>
#include "tower.hpp"
```

Include dependency graph for queen.hpp:



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



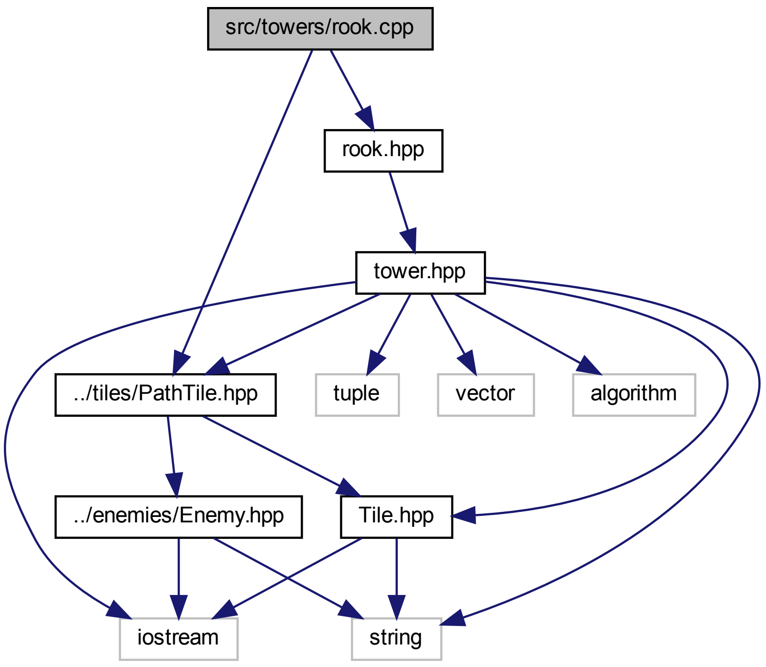
Classes

- class Queen

8.42 src/towers/rook.cpp File Reference

```
#include "rook.hpp"
#include "../tiles/PathTile.hpp"
```

Include dependency graph for rook.cpp:



Variables

- std::vector< std::tuple< int, upgradeType, int > > upgradeTableRook

8.42.1 Variable Documentation

8.42.1.1 upgradeTableRook

```
std::vector<std::tuple<int, upgradeType, int> > upgradeTableRook
```

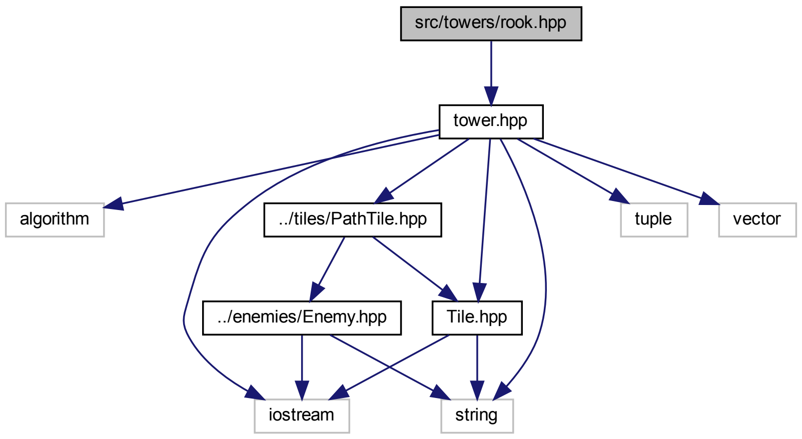
Initial value:

```
{
    {50, range, 1}, {150, damage, 5}, {400, speed, 3}}
```

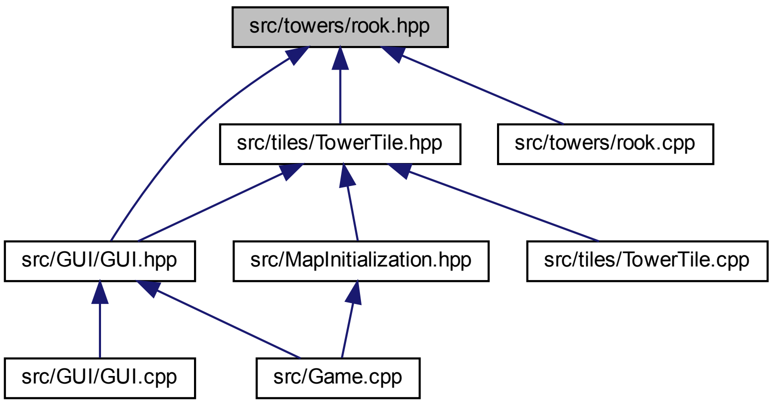
8.43 src/towers/rook.hpp File Reference

```
#include "tower.hpp"
```

Include dependency graph for rook.hpp:



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



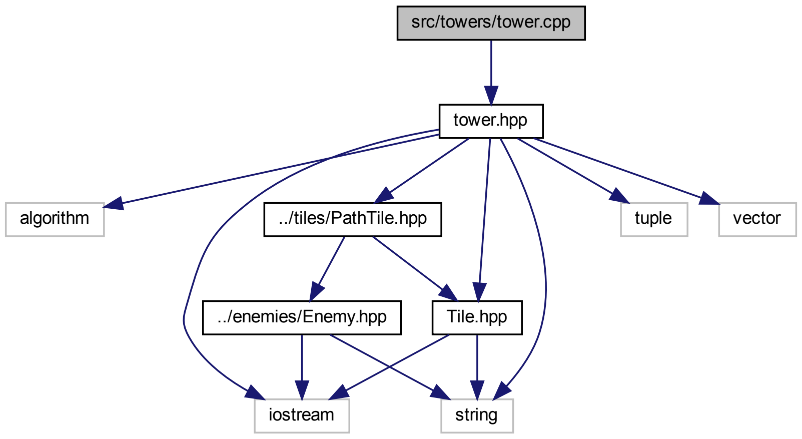
Classes

- class Rook

8.44 src/towers/tower.cpp File Reference

```
#include "tower.hpp"
```

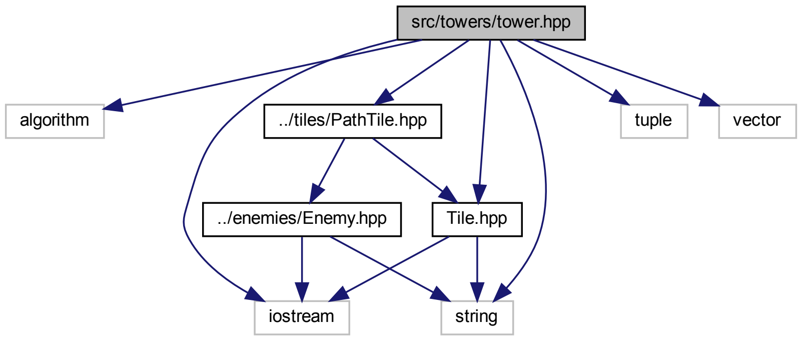
Include dependency graph for tower.cpp:



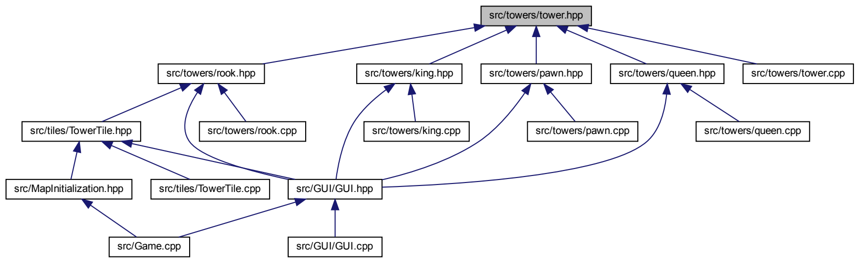
8.45 src/towers/tower.hpp File Reference

```
#include <algorithm>
#include <iostream>
#include <string>
#include <tuple>
#include <vector>
#include "../tiles/PathTile.hpp"
#include "../tiles/Tile.hpp"
```

Include dependency graph for tower.hpp:



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



Classes

- class Tower

Enumerations

- enum upgradeType { speed, damage, range }
  All the possible types of upgrades.

8.45.1 Enumeration Type Documentation

8.45.1.1 upgradeType

```
enum upgradeType
```

All the possible types of upgrades.

Enumerator

speed	
damage	
range	

Index