

Railroad Tycoon Prototyp

1.0.0

Erzeugt von Doxygen 1.8.17

Inhaltsverzeichnis

Kapitel 1

SWT Praktikum

Hier ist eine kleine Anleitung wie man das Projekt auf seinem eigenen Rechner synchronisiert:

1. git installieren
2. `>> git clone https://github.com/davidtraum/swt/`
3. `>> cd swt`

Wenn man was geändert hat:

(0. Ins Basisverzeichnis vom Projekt gehen)

1. `>> git add *`
1. `>> git commit -m "Kurze Nachricht was man gemacht hat"`
2. `>> git push origin master` (Oder eigenen Branch angeben)

1.1 Changelog

| Datum | Funktion |
|--------|--|
| 28.10. | Start Changelog |
| 28.10. | Animation beim Klick auf Städte |
| 28.10. | Übersichtskarte mit Taste O |
| 29.10. | Statuspanel hinzugefügt |
| 04.11. | Menübar hinzugefügt |
| 05.11. | Tooltip-Widget hinzugefügt |
| 22.11. | Toolbar und Statusanzeige hinzugefügt |
| 24.11. | Minimap und Verbindungsanzeige hinzugefügt |

Kapitel 2

Verzeichnis der Namensbereiche

2.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

Ui ??

Kapitel 3

Hierarchie-Verzeichnis

3.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

| | |
|-----------------------------|----|
| AnimationEntity | ?? |
| AnimationMovement | ?? |
| City | ?? |
| GraphicsManager | ?? |
| MainMenu | ?? |
| MapTile | ?? |
| Player | ?? |
| Point | ?? |
| QDockWidget | |
| RouteInterface | ?? |
| QGraphicsRectItem | |
| Highlighter | ?? |
| QGraphicsScene | |
| Scene | ?? |
| QGraphicsView | |
| View | ?? |
| QMainWindow | |
| MainWindow | ?? |
| QObject | |
| DataModel | ?? |
| QThread | |
| Client | ?? |
| GameLoop | ?? |
| QToolBar | |
| MenuBar | ?? |
| QWidget | |
| GoodSelector | ?? |
| MapRenderer | ?? |
| Minimap | ?? |
| SidePanel | ?? |
| ToolTipMenu | ?? |
| TrainRenderer | ?? |
| River | ?? |

Kapitel 4

Klassen-Verzeichnis

4.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

| | |
|-------------------|----|
| AnimationEntity | ?? |
| AnimationMovement | ?? |
| City | ?? |
| Client | ?? |
| DataModel | ?? |
| GameLoop | ?? |
| GoodSelector | ?? |
| GraphicsManager | ?? |
| Highlighter | ?? |
| MainMenu | ?? |
| MainWindow | ?? |
| MapRenderer | ?? |
| MapTile | ?? |
| MenuBar | ?? |
| Minimap | ?? |
| Player | ?? |
| Point | ?? |
| River | ?? |
| RouteInterface | ?? |
| Scene | ?? |
| SidePanel | ?? |
| ToolTipMenu | ?? |
| TrainRenderer | ?? |
| View | ?? |

Kapitel 5

Datei-Verzeichnis

5.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

| | |
|--|----|
| src/application_server/animationentity.cpp | ?? |
| src/application_server/animationentity.h | ?? |
| src/application_server/animationmovement.cpp | ?? |
| src/application_server/animationmovement.h | ?? |
| src/application_server/city.cpp | ?? |
| src/application_server/city.h | ?? |
| src/application_server/client.cpp | ?? |
| src/application_server/client.h | ?? |
| src/application_server/datamodel.cpp | ?? |
| src/application_server/datamodel.h | ?? |
| src/application_server/gameloop.cpp | ?? |
| src/application_server/gameloop.h | ?? |
| src/application_server/goodselector.cpp | ?? |
| src/application_server/goodselector.h | ?? |
| src/application_server/graphicsmanager.cpp | ?? |
| src/application_server/graphicsmanager.h | ?? |
| src/application_server/highlighter.cpp | ?? |
| src/application_server/highlighter.h | ?? |
| src/application_server/main.cpp | ?? |
| src/application_server/main.h | ?? |
| src/application_server/mainmenu.cpp | ?? |
| src/application_server/mainmenu.h | ?? |
| src/application_server/mainwindow.cpp | ?? |
| src/application_server/mainwindow.h | ?? |
| src/application_server/maprender.cpp | ?? |
| src/application_server/maprender.h | ?? |
| src/application_server/maptile.cpp | ?? |
| src/application_server/maptile.h | ?? |
| src/application_server/menubar.cpp | ?? |
| src/application_server/menubar.h | ?? |
| src/application_server/minimap.cpp | ?? |
| src/application_server/minimap.h | ?? |
| src/application_server/player.cpp | ?? |
| src/application_server/player.h | ?? |
| src/application_server/point.cpp | ?? |

| | |
|---|----|
| src/application_server/point.h | ?? |
| src/application_server/river.cpp | ?? |
| src/application_server/river.h | ?? |
| src/application_server/routeinterface.cpp | ?? |
| src/application_server/routeinterface.h | ?? |
| src/application_server/scene.cpp | ?? |
| src/application_server/scene.h | ?? |
| src/application_server/sidepanel.cpp | ?? |
| src/application_server/sidepanel.h | ?? |
| src/application_server/tooltipmenu.cpp | ?? |
| src/application_server/tooltipmenu.h | ?? |
| src/application_server/trainrenderer.cpp | ?? |
| src/application_server/trainrenderer.h | ?? |
| src/application_server/view.cpp | ?? |
| src/application_server/view.h | ?? |

Kapitel 6

Dokumentation der Namensbereiche

6.1 Ui-Namensbereichsreferenz

Kapitel 7

Klassen-Dokumentation

7.1 AnimationEntity Klassenreferenz

```
#include <animationentity.h>
```

Zusammengehörigkeiten von AnimationEntity:

| AnimationEntity |
|--|
| + autoRotate + rotation - image - x - y |
| + AnimationEntity() + AnimationEntity() + getPosition() + move() + getX() + getY() + setPosition() + getImage() |

Öffentliche Methoden

- [AnimationEntity](#) (QImage *, int, int)
[AnimationEntity::AnimationEntity](#) Erzeugt ein neues Animation-Entity.
- [AnimationEntity](#) (QImage *)
- [Point](#) [getPosition](#) ()
[AnimationEntity::getPosition](#) Liefert die Position.
- void [move](#) (double, double)

- [`AnimationEntity::move`](#) Verschiebt das Element um einen Vektor.
- double [`getX`](#) ()
- double [`getY`](#) ()
- void [`setPosition`](#) (int, int)
- [`AnimationEntity::setPosition`](#) Setzt die Position des Elements.
- QImage * [`getImage`](#) ()
- [`AnimationEntity::getImage`](#) Liefert die Grafik des Animationselements.

Öffentliche Attribute

- bool [`autoRotate`](#) {true}
- int [`rotation`](#) {0}

Private Attribute

- QImage * [`image`](#)
- double [`x`](#) {0}
- double [`y`](#) {0}

7.1.1 Beschreibung der Konstruktoren und Destruktoren

7.1.1.1 AnimationEntity() [1/2]

```
AnimationEntity::AnimationEntity (
    QImage * pImage,
    int pX,
    int pY )
```

[`AnimationEntity::AnimationEntity`](#) Erzeugt ein neues Animation-Entity.

Parameter

| | |
|-------------------------------------|----------------------|
| <code>pImage</code> | Das Bild. |
| <code>pX</code> | Die X-Startposition. |
| <code>pY</code> | Die Y-Startposition. |

7.1.1.2 AnimationEntity() [2/2]

```
AnimationEntity::AnimationEntity (
    QImage * pImage )
```

7.1.2 Dokumentation der Elementfunktionen

7.1.2.1 getImage()

```
QImage * AnimationEntity::getImage ( )
```

[AnimationEntity::getImage](#) Liefert die Grafik des Animationselements.

Rückgabe

Ein QImage.

7.1.2.2 getPosition()

```
Point AnimationEntity::getPosition ( )
```

[AnimationEntity::getPosition](#) Liefert die Position.

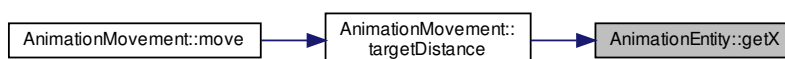
Rückgabe

Die Position als [Point](#).

7.1.2.3 getX()

```
double AnimationEntity::getX ( )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.1.2.4 getY()

```
double AnimationEntity::getY ( )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.1.2.5 move()

```
void AnimationEntity::move (
    double vx,
    double vy )
```

[AnimationEntity::move](#) Verschiebt das Element um einen Vektor.

Parameter

| | |
|-----------------|-------------------|
| <code>vx</code> | Die X-Komponente. |
| <code>vy</code> | Die Y-Komponente. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.1.2.6 setPosition()

```
void AnimationEntity::setPosition (
    int px,
    int py )
```

[AnimationEntity::setPosition](#) Setzt die Position des Elements.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.1.3 Dokumentation der Datenelemente

7.1.3.1 autoRotate

```
bool AnimationEntity::autoRotate {true}
```

7.1.3.2 image

```
QImage* AnimationEntity::image [private]
```

7.1.3.3 rotation

```
int AnimationEntity::rotation {0}
```

7.1.3.4 x

```
double AnimationEntity::x {0} [private]
```

7.1.3.5 y

```
double AnimationEntity::y {0} [private]
```

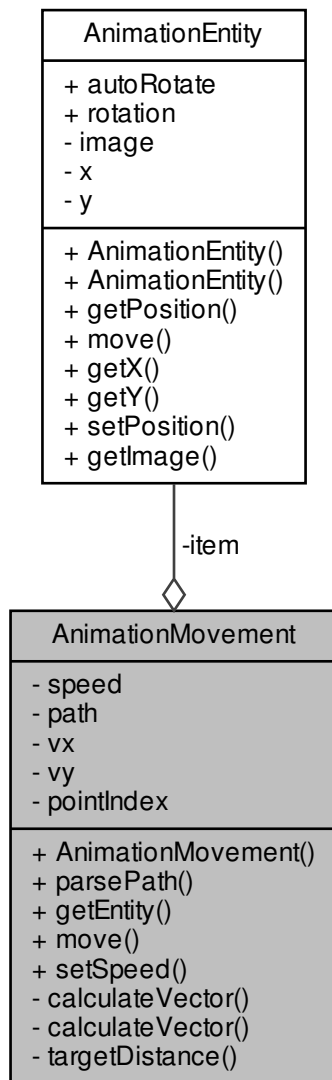
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- src/application_server/[animationentity.h](#)
- src/application_server/[animationentity.cpp](#)

7.2 AnimationMovement Klassenreferenz

```
#include <animationmovement.h>
```

Zusammengehörigkeiten von AnimationMovement:



Öffentliche Methoden

- [AnimationMovement](#) ([AnimationEntity](#) *, [QString](#))
[AnimationMovement::AnimationMovement](#) Erzeugt eine neue Bewegungsanimation.
- void [parsePath](#) ([QString](#) [path](#))
[AnimationMovement::parsePath](#) Liest einen Animationspfad aus einem String ein.
- [AnimationEntity](#) * [getEntity](#) ()

- bool `move` ()
AnimationMovement::move Führt einen Schritt der Animation durch.
- void `setSpeed` (double)
AnimationMovement::setSpeed Setzt die Geschwindigkeit der Animation.

Private Methoden

- void `calculateVector` (Point origin, Point target)
- void `calculateVector` (int origin, int target)
- double `targetDistance` ()

Private Attribute

- double `speed` {1}
- AnimationEntity * `item`
- QList< Point * > `path`
- double `vx` {0}
- double `vy` {0}
- int `pointIndex` {-1}

7.2.1 Beschreibung der Konstruktoren und Destruktoren

7.2.1.1 AnimationMovement()

```
AnimationMovement::AnimationMovement (
    AnimationEntity * pItem,
    QString pPath )
```

AnimationMovement::AnimationMovement Erzeugt eine neue Bewegungsanimation.

Parameter

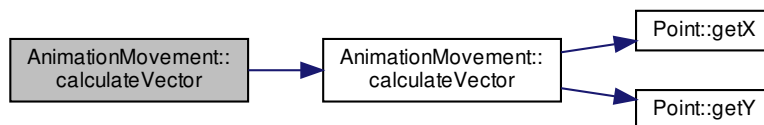
| | |
|--------------|------------------------------|
| <i>pItem</i> | Das zu animierende Grafikum. |
|--------------|------------------------------|

7.2.2 Dokumentation der Elementfunktionen

7.2.2.1 calculateVector() [1/2]

```
void AnimationMovement::calculateVector (
    int origin,
    int target ) [private]
```

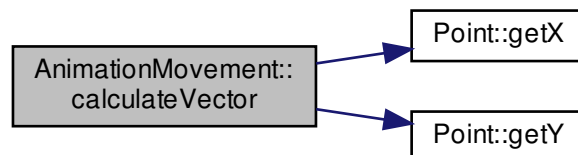
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



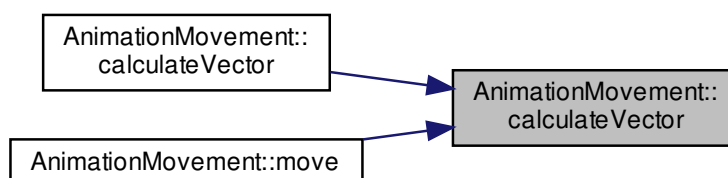
7.2.2.2 `calculateVector()` [2/2]

```
void AnimationMovement::calculateVector (  
    Point origin,  
    Point target ) [private]
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.2.3 getEntity()

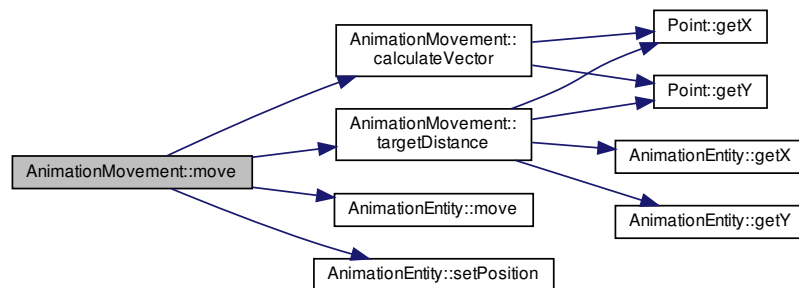
```
AnimationEntity * AnimationMovement::getEntity ( )
```

7.2.2.4 move()

```
bool AnimationMovement::move ( )
```

[AnimationMovement::move](#) Führt einen Schritt der Animation durch.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.2.2.5 parsePath()

```
void AnimationMovement::parsePath (
    QString data )
```

[AnimationMovement::parsePath](#) Liest einen Animationspfad aus einem String ein.

Parameter

| | |
|-------------|---|
| <i>path</i> | Ein String im Format X1:Y1;X2:Y2;X3:Y3... |
|-------------|---|

7.2.2.6 setSpeed()

```
void AnimationMovement::setSpeed (
    double pSpeed )
```

[AnimationMovement::setSpeed](#) Setzt die Geschwindigkeit der Animation.

Parameter

| | |
|---------------|--|
| <i>pSpeed</i> | Die Geschwindigkeit (1=normal, 2=doppelt...) |
|---------------|--|

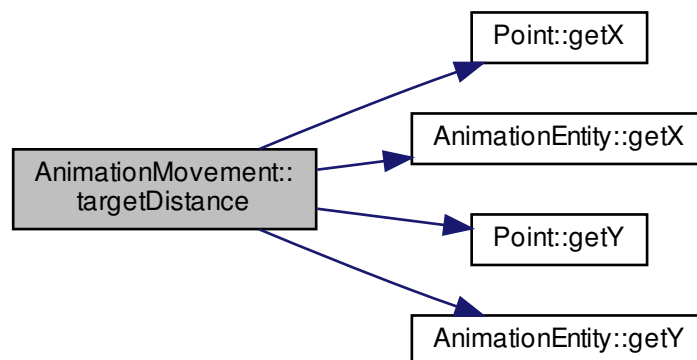
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



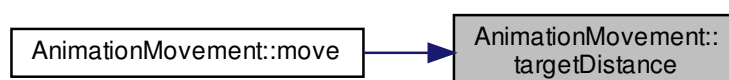
7.2.2.7 targetDistance()

```
double AnimationMovement::targetDistance ( ) [private]
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3 Dokumentation der Datenelemente

7.2.3.1 item

```
AnimationEntity* AnimationMovement::item [private]
```

7.2.3.2 path

```
QList<Point *> AnimationMovement::path [private]
```

7.2.3.3 pointIndex

```
int AnimationMovement::pointIndex {-1} [private]
```

7.2.3.4 speed

```
double AnimationMovement::speed {1} [private]
```

7.2.3.5 vx

```
double AnimationMovement::vx {0} [private]
```

7.2.3.6 vy

```
double AnimationMovement::vy {0} [private]
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- src/application_server/[animationmovement.h](#)
- src/application_server/[animationmovement.cpp](#)

7.3 City Klassenreferenz

```
#include <city.h>
```

Zusammengehörigkeiten von City:

| City |
|---|
| - size - centerX - centerY - name |
| + City() + City() + getSize() + getCenterX() + getCenterY() + getName() + setSize() + setCenter() + setName() |

Öffentliche Methoden

- [City](#) (int pX, int pY, int pSize)
[City::City](#) Erzeugt eine Stadt mit vorgegebenen Parametern.
- [City](#) ()
[City::City](#) Erzeugt eine leere Stadt.
- int [getSize](#) ()
[City::getSize](#) Gibt die Anzahl der Felder zurück die zur Stadt gehören.
- int [getCenterX](#) ()
[City::getCenterX](#) Gibt den X-Index des Mittelpunktes.
- int [getCenterY](#) ()
[City::getCenterX](#) Gibt den Y-Index des Mittelpunktes.
- std::string [getName](#) ()
[City::getName](#) Gibt den Namen der Stadt.
- void [setSize](#) (int pSize)
[City::setSize](#) Gibt die Größe der Stadt zurück (Anzahl der Gebäude)
- void [setCenter](#) (int pX, int pY)
[City::setCenter](#) Setzt den Mittelpunkt der Stadt.
- void [setName](#) (std::string pName)
[City::setName](#) Setzt den Namen der Stadt.

Private Attribute

- int `size`
- int `centerX`
- int `centerY`
- std::string `name`

7.3.1 Beschreibung der Konstruktoren und Destruktoren

7.3.1.1 City() [1/2]

```
City::City (  
    int pX,  
    int pY,  
    int pSize )
```

`City::City` Erzeugt eine Stadt mit vorgegebenen Parametern.

Parameter

| | |
|--------------|--------------------------------|
| <i>pX</i> | Der X-Index des Mittelpunktes. |
| <i>pY</i> | Der Y-Index des Mittelpunktes. |
| <i>pSize</i> | Die gröÙe der Stadt. |

7.3.1.2 City() [2/2]

```
City::City ( )
```

`City::City` Erzeugt eine leere Stadt.

7.3.2 Dokumentation der Elementfunktionen

7.3.2.1 getCenterX()

```
int City::getCenterX ( )
```

`City::getCenterX` Gibt den X-Index des Mittelpunktes.

Rückgabe

Der X-Index des Mittelpunktes der Stadt.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.3.2.2 getCenterY()**

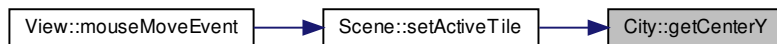
```
int City::getCenterY ( )
```

[City::getCenterX](#) Gibt den Y-Index des Mittelpunktes.

Rückgabe

Der Y-Index des Mittelpunktes der Stadt.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.3.2.3 getName()**

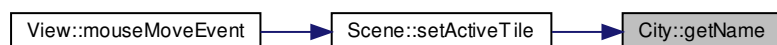
```
std::string City::getName ( )
```

[City::getName](#) Gibt den Namen der Stadt.

Rückgabe

Der Name der Stadt.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.3.2.4 getSize()

```
int City::getSize ( )
```

[City::getSize](#) Gibt die Anzahl der Felder zurück die zur Stadt gehören.

Rückgabe

Die Größe.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.3.2.5 setCenter()

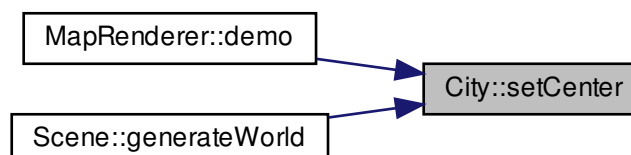
```
void City::setCenter (
    int pX,
    int pY )
```

[City::setCenter](#) Setzt den Mittelpunkt der Stadt.

Parameter

| | |
|-----------|--------------|
| <i>pX</i> | Der X-Index. |
| <i>pY</i> | Der Y-Index. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.3.2.6 setName()

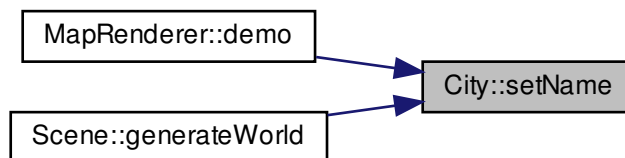
```
void City::setName (
    std::string pName )
```

[City::setName](#) Setzt den Namen der Stadt.

Parameter

| | |
|--------------|----------------|
| <i>pName</i> | Der neue Name. |
|--------------|----------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.3.2.7 setSize()

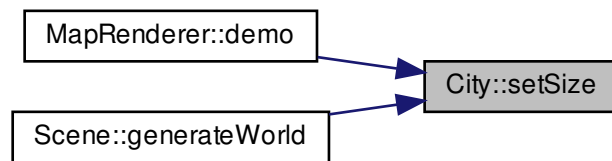
```
void City::setSize (
    int pSize )
```

[City::setSize](#) Gibt die Größe der Stadt zurück (Anzahl der Gebäude)

Parameter

| | |
|--------------|---------------------|
| <i>pSize</i> | Die Größe der Stadt |
|--------------|---------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.3.3 Dokumentation der Datenelemente

7.3.3.1 centerX

```
int City::centerX [private]
```

7.3.3.2 centerY

```
int City::centerY [private]
```

7.3.3.3 name

```
std::string City::name [private]
```

7.3.3.4 size

```
int City::size [private]
```

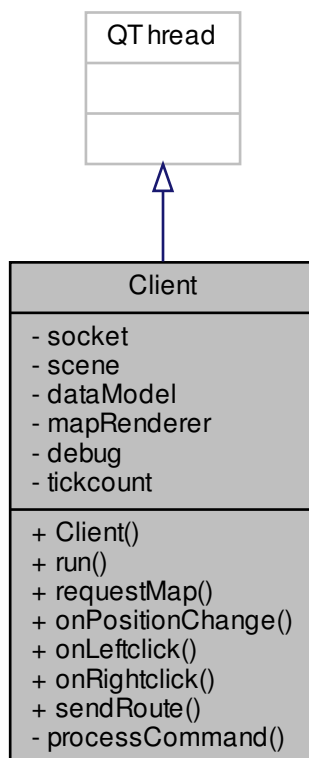
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/city.h](#)
- [src/application_server/city.cpp](#)

7.4 Client Klassenreferenz

```
#include <client.h>
```

Klassendiagramm für Client:



Signale

- void `mapLoaded` ()
- void `tileChanged` (int, int, int, int)
- void `playerPositionChange` (int, int)
- void `onMapLoaded` (bool)

Öffentliche Methoden

- `Client` (QString *connectionInfo, `Scene` *pScene, `MapRenderer` *pRenderer, `View` *pView, `DataModel` *pDataModel)
`Client::Client` Erzeugt einen neuen `Client`.
- void `run` () override
`Client::run` Startet den Client-Thread.
- void `requestMap` ()

Private Methoden

- void `processCommand` (QString command)
`Client::processCommand` Führt einen empfangenen Befehl aus dem Serverprotokoll aus.

Private Attribute

- QTcpSocket * `socket`
- `Scene` * `scene`
- `DataModel` * `dataModel`
- `MapRenderer` * `mapRenderer`
- bool `debug`
- int `tickcount` {0}

7.4.1 Beschreibung der Konstruktoren und Destruktoren

7.4.1.1 Client()

```
Client::Client (
    QString * connectionInfo,
    Scene * pScene,
    MapRenderer * pRenderer,
    View * pView,
    DataModel * pDataModel )
```

`Client::Client` Erzeugt einen neuen `Client`.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.4.2 Dokumentation der Elementfunktionen

7.4.2.1 mapLoaded

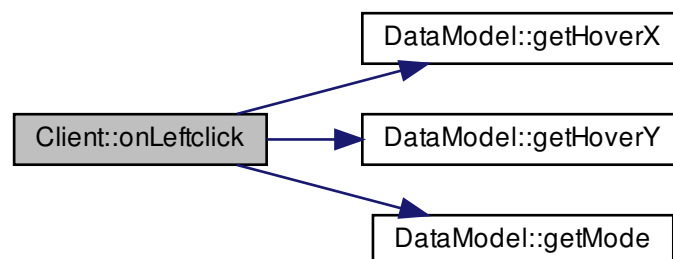
```
void Client::mapLoaded ( ) [signal]
```

7.4.2.2 onLeftclick

```
void Client::onLeftclick ( ) [slot]
```

[Client::onLeftclick](#) Führt einen Linksklick durch.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



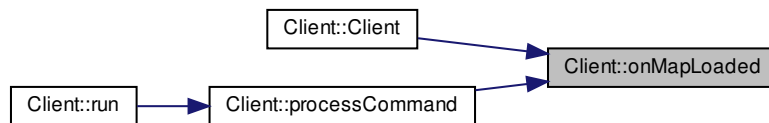
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.3 onMapLoaded

```
void Client::onMapLoaded (
    bool ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.4 onPositionChange

```
void Client::onPositionChange (
    int pX,
    int pY ) [slot]
```

[Client::onPositionChange](#) Slot für Ändern der Position.

Parameter

| | |
|-----------|--------------|
| <i>pX</i> | Der X-Index. |
| <i>pY</i> | Der Y-Index. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.5 onRightclick

```
void Client::onRightclick ( ) [slot]
```

[Client::onRightclick](#) Führt einen Rechtsklick durch.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.4.2.6 playerPositionChange

```
void Client::playerPositionChange (
    int ,
    int ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.7 processCommand()

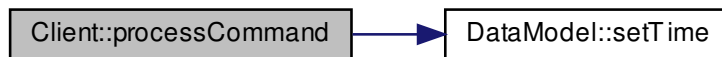
```
void Client::processCommand (
    QString cmd ) [private]
```

[Client::processCommand](#) Führt einen empfangenen Befehl aus dem Serverprotokoll aus.

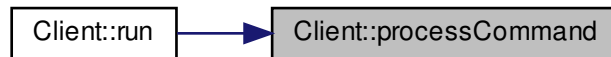
Parameter

| | |
|------------|------------------------|
| <i>cmd</i> | Der Befehl als String. |
|------------|------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



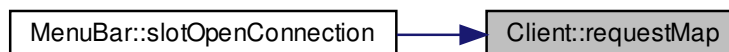
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.8 requestMap()

```
void Client::requestMap ( )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

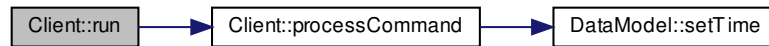


7.4.2.9 run()

```
void Client::run ( ) [override]
```

[Client::run](#) Startet den Client-Thread.

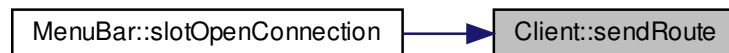
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.4.2.10 sendRoute

```
void Client::sendRoute (
    QString routeString ) [slot]
```

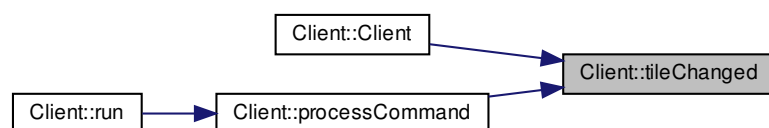
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.2.11 tileChanged

```
void Client::tileChanged (
    int ,
    int ,
    int ,
    int ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.4.3 Dokumentation der Datenelemente

7.4.3.1 dataModel

```
DataModel* Client::dataModel [private]
```

7.4.3.2 debug

```
bool Client::debug [private]
```

7.4.3.3 mapRenderer

```
MapRenderer* Client::mapRenderer [private]
```

7.4.3.4 scene

```
Scene* Client::scene [private]
```

7.4.3.5 socket

```
QTcpSocket* Client::socket [private]
```

7.4.3.6 tickcount

```
int Client::tickcount {0} [private]
```

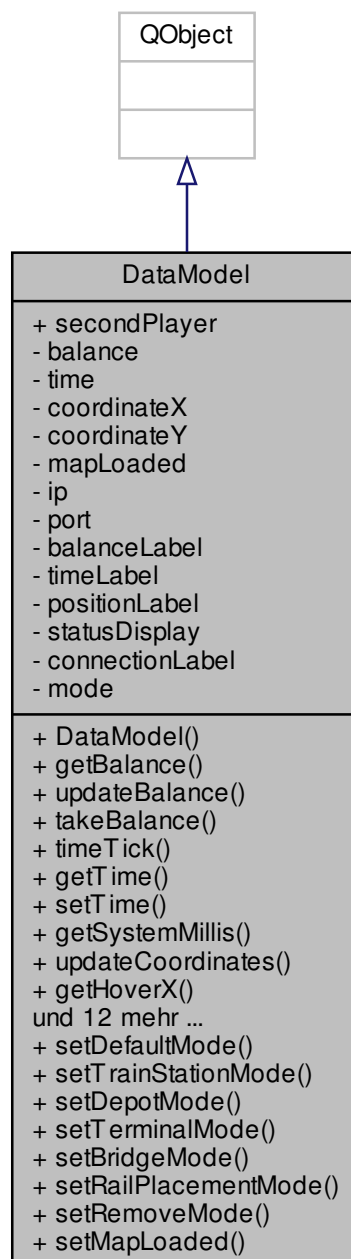
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/client.h](#)
- [src/application_server/client.cpp](#)

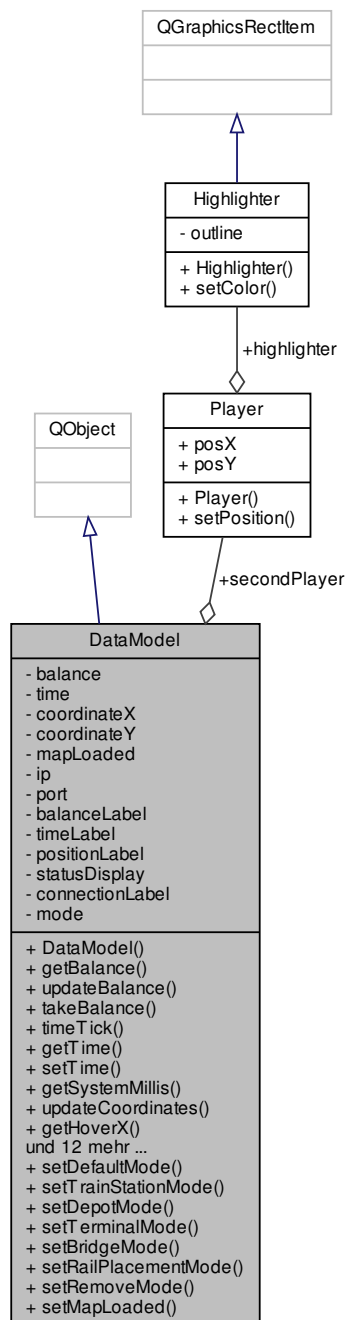
7.5 DataModel Klassenreferenz

```
#include <datamodel.h>
```

Klassendiagramm für DataModel:



Zusammengehörigkeiten von DataModel:



Öffentliche Typen

- enum `MODE` {
`DEFAULT`, `TRAIN_STATION`, `TRAIN_DEPOT`, `TRAIN_TERMINAL`,
`BRIDGE`, `RAIL_PLACEMENT`, `REMOVE` }

Öffentliche Slots

- void `setDefaultMode` ()
DataModel::setDefaultMode Signal um in den Standard Bearbeitungsmodus zu wechseln.
- void `setTrainStationMode` ()
DataModel::setRailPlacementMode Signal um in den Bahnhofseditor zu wechseln.
- void `setDepotMode` ()
DataModel::setRailPlacementMode Signal um in den Bahnhofseditor zu wechseln.
- void `setTerminalMode` ()
DataModel::setRailPlacementMode Signal um in den Bahnhofseditor zu wechseln.
- void `setBridgeMode` ()
DataModel::setRailPlacementMode Signal um in den Brückeneditor zu wechseln.
- void `setRailPlacementMode` ()
DataModel::setRailPlacementMode Signal um in den Gleiseditor zu wechseln.
- void `setRemoveMode` ()
DataModel::setRailPlacementMode Signal um in den Removeeditor zu wechseln.
- void `setMapLoaded` (bool)
DataModel::setMapLoaded Setzt das die Karte geladen wurde.

Signale

- void `positionChange` (int, int)
- void `viewChange` ()

Öffentliche Methoden

- `DataModel` ()
DataModel::DataModel Diese Klasse verwaltet alle globalen Daten rund um den Spielverlauf, z.B. den Kontostand.
- int `getBalance` ()
DataModel::getBalance Liefert den aktuellen Kontostand zurück.
- void `updateBalance` (int pBalance)
DataModel::updateBalance Aktualisiert den Kontostand. Auch in Anzeigen etc.
- bool `takeBalance` (int pAmount)
DataModel::takeBalance Zieht Geld ab falls noch genug da ist.
- void `timeTick` ()
DataModel::timeTick Wird aufgerufen wenn eine Zeiteinheit verstrichen ist. Erhöht den Timecode.
- long `getTime` ()
DataModel::getTime Liefert die aktuelle Zeit als Timecode. (Zahl die je nach Geschwindigkeit wächst)
- void `setTime` (long)
DataModel::setTime Setzt den aktuellen Zeitstempel.
- long `getSystemMillis` ()
DataModel::getSystemMillis Gibt die Zahl der Millisekunden seit 1970 zurück.
- void `updateCoordinates` (int pX, int pY)
DataModel::updateCoordinates Aktualisiert die Koordinaten des fokussierten Quadranten.
- int `getHoverX` ()
DataModel::getHoverX Gibt die aktuelle Koordinate der Kachel zurück über der die Maus ist.
- int `getHoverY` ()
DataModel::getHoverX Gibt die aktuelle Koordinate der Kachel zurück über der die Maus ist.
- std::string `formatTime` (long pTime)
DataModel::formatTime Formattiert einen Timecode als String.

- void `setConnectionInfo` (QString pString)
DataModel::setConnectionInfo Setzt die Verbindungsinformation als String.
- QString * `getIP` ()
DataModel::getIP Gibt die IP Adresse zur Verbindung zurück.
- quint16 `getPort` ()
DataModel::getPort Gibt den Port zur Verbindung zurück.
- void `setGuiBalanceLabel` (QLabel *label)
DataModel::setGuiBalanceLabel Setzt das Label in welchem der Kontostand dargestellt wird.
- void `setGuiTimeLabel` (QLabel *label)
DataModel::setGuiTimeLabel Setzt das Label in welchem die Zeit dargestellt wird.
- void `setGuiPositionLabel` (QLabel *label)
DataModel::setGuiTimeLabel Setzt das Label in welchem die Koordinate dargestellt wird.
- void `setStatusDisplayLabel` (QLabel *label)
DataModel::setStatusDisplayLabel Setzt das Label in dem der aktuelle Bearbeitungsmodus angezeigt wird.
- void `setConnectionLabel` (QLabel *label)
DataModel::setStatusDisplayLabel Setzt das Label in dem der aktuelle Bearbeitungsmodus angezeigt wird.
- void `setMode` (MODE)
DataModel::setMode Setzt den aktuellen Bearbeitungsmodus.
- MODE `getMode` ()
DataModel::getMode Gibt den aktuellen Bearbeitungsmodus.

Öffentliche Attribute

- Player * `secondPlayer`

Private Attribute

- int `balance`
- long `time`
- int `coordinateX`
- int `coordinateY`
- bool `mapLoaded` {false}
- QString `ip`
- quint16 `port`
- QLabel * `balanceLabel`
- QLabel * `timeLabel`
- QLabel * `positionLabel`
- QLabel * `statusDisplay`
- QLabel * `connectionLabel`
- MODE `mode` {MODE::DEFAULT}

7.5.1 Dokumentation der Aufzählungstypen

7.5.1.1 MODE

```
enum DataModel::MODE
```

Aufzählungswerte

| | |
|----------------|--|
| DEFAULT | |
| TRAIN_STATION | |
| TRAIN_DEPOT | |
| TRAIN_TERMINAL | |
| BRIDGE | |
| RAIL_PLACEMENT | |
| REMOVE | |

7.5.2 Beschreibung der Konstruktoren und Destruktoren

7.5.2.1 DataModel()

```
DataModel::DataModel ( )
```

[DataModel::DataModel](#) Diese Klasse verwaltet alle globalen Daten rund um den Spielverlauf, z.B. den Kontostand.

7.5.3 Dokumentation der Elementfunktionen

7.5.3.1 formatTime()

```
std::string DataModel::formatTime (
    long pTime )
```

[DataModel::formatTime](#) Formattiert einen Timecode als String.

Parameter

| | |
|--------------|---------------|
| <i>pTime</i> | Der Timecode. |
|--------------|---------------|

Rückgabe

Der Text.

7.5.3.2 getBalance()

```
int DataModel::getBalance ( )
```

[DataModel::getBalance](#) Liefert den aktuellen Kontostand zurück.

Rückgabe

Der aktuelle Kontostand.

7.5.3.3 getHoverX()

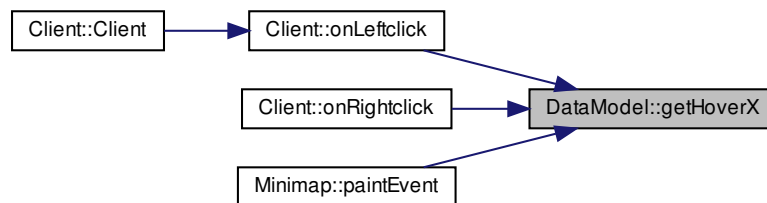
```
int DataModel::getHoverX ( )
```

[DataModel::getHoverX](#) Gibt die aktuelle Koordinate der Kachel zurück über der die Maus ist.

Rückgabe

Eine Kachel-Koordinate.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.5.3.4 getHoverY()**

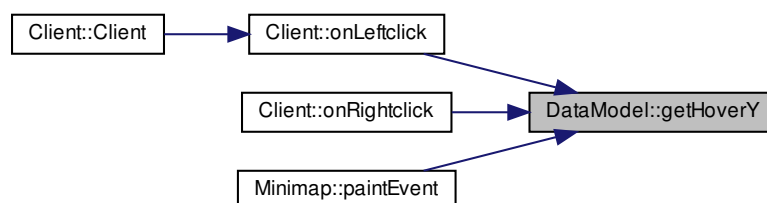
```
int DataModel::getHoverY ( )
```

[DataModel::getHoverX](#) Gibt die aktuelle Koordinate der Kachel zurück über der die Maus ist.

Rückgabe

Eine Kachel-Koordinate.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.5 getIP()

```
QString * DataModel::getIP ( )
```

[DataModel::getIP](#) Gibt die IP Adresse zur Verbindung zurück.

Rückgabe

Die IP Adresse als QString

7.5.3.6 getMode()

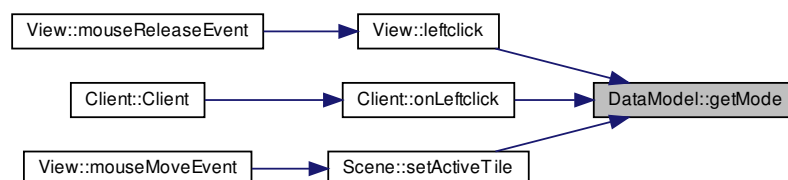
```
DataModel::MODE DataModel::getMode ( )
```

[DataModel::getMode](#) Gibt den aktuellen Bearbeitungsmodus.

Rückgabe

Der aktuelle Modus.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.7 getPort()

```
quint16 DataModel::getPort ( )
```

[DataModel::getPort](#) Gibt den Port zur Verbindung zurück.

Rückgabe

Der Port als int.

7.5.3.8 `getSystemMillis()`

```
long DataModel::getSystemMillis ( )
```

[DataModel::getSystemMillis](#) Gibt die Zahl der Millisekunden seit 1970 zurück.

Rückgabe

Die Zahl der Millisekunden.

7.5.3.9 `getTime()`

```
long DataModel::getTime ( )
```

[DataModel::getTime](#) Liefert die aktuelle Zeit als Timecode. (Zahl die je nach Geschwindigkeit wächst)

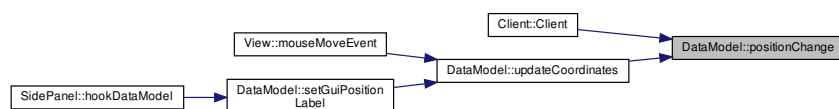
Rückgabe

Der Timecode.

7.5.3.10 `positionChange`

```
void DataModel::positionChange (
    int ,
    int ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

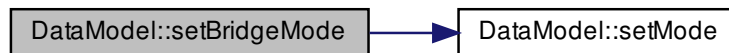


7.5.3.11 setBridgeMode

```
void DataModel::setBridgeMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Brückeneditor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.5.3.12 setConnectionInfo()

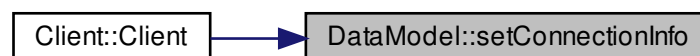
```
void DataModel::setConnectionInfo (
    QString pString )
```

[DataModel::setConnectionInfo](#) Setzt die Verbindungsinformation als String.

Parameter

| | |
|----------------------|---------------------------------------|
| <code>pString</code> | Die IP und der Port im Format IP:PORT |
|----------------------|---------------------------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.13 setConnectionLabel()

```
void DataModel::setConnectionLabel (
    QLabel * label )
```

[DataModel::setStatusDisplayLabel](#) Setzt das Label in dem der aktuelle Bearbeitungsmodus angezeigt wird.

Parameter

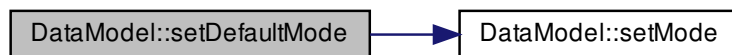
| | |
|--------------|------------------------------------|
| <i>label</i> | Ein Pointer auf ein QLabel Objekt. |
|--------------|------------------------------------|

7.5.3.14 setDefaultMode

```
void DataModel::setDefaultMode ( ) [slot]
```

[DataModel::setDefaultMode](#) Signal um in den Standard Bearbeitungsmodus zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**7.5.3.15 setDepotMode**

```
void DataModel::setDepotMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Bahnhofseditor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**7.5.3.16 setGuiBalanceLabel()**

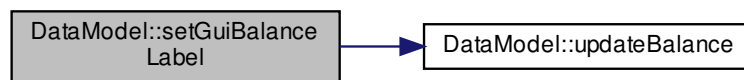
```
void DataModel::setGuiBalanceLabel (
    QLabel * label )
```

[DataModel::setGuiBalanceLabel](#) Setzt das Label in welchem der Kontostand dargestellt wird.

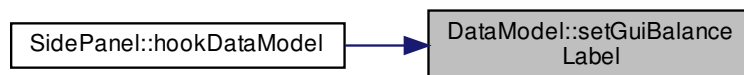
Parameter

| | |
|--------------|------------------------------------|
| <i>label</i> | Ein Pointer auf ein QLabel Objekt. |
|--------------|------------------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.17 setGuiPositionLabel()

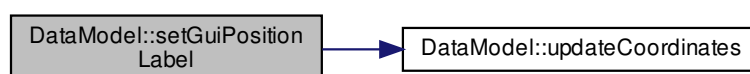
```
void DataModel::setGuiPositionLabel (
    QLabel * label )
```

[DataModel::setGuiTimeLabel](#) Setzt das Label in welchem die Koordinate dargestellt wird.

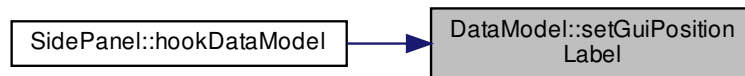
Parameter

| | |
|--------------|------------------------------------|
| <i>label</i> | Ein Pointer auf ein QLabel Objekt. |
|--------------|------------------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.18 setGuiTimeLabel()

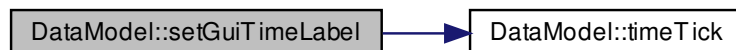
```
void DataModel::setGuiTimeLabel (
    QLabel * label )
```

[DataModel::setGuiTimeLabel](#) Setzt das Label in welchem die Zeit dargestellt wird.

Parameter

| | |
|--------------|------------------------------------|
| <i>label</i> | Ein Pointer auf ein QLabel Objekt. |
|--------------|------------------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.19 setMapLoaded

```
void DataModel::setMapLoaded (
    bool status ) [slot]
```

[DataModel::setMapLoaded](#) Setzt das die Karte geladen wurde.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.20 setMode()

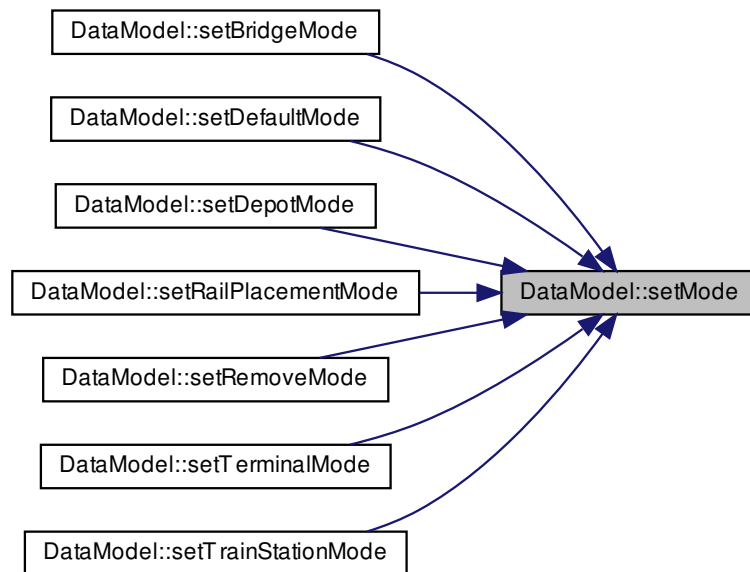
```
void DataModel::setMode (
    DataModel::MODE pMode )
```

[DataModel::setMode](#) Setzt den aktuellen Bearbeitungsmodus.

Parameter

| | |
|--------------|--------------------|
| <i>pMode</i> | Bearbeitungsmodus. |
|--------------|--------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

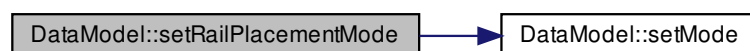


7.5.3.21 setRailPlacementMode

```
void DataModel::setRailPlacementMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Gleiseditor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.5.3.22 setRemoveMode

```
void DataModel::setRemoveMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Removeeditor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.5.3.23 setStatusDisplayLabel()

```
void DataModel::setStatusDisplayLabel (
    QLabel * label )
```

[DataModel::setStatusDisplayLabel](#) Setzt das Label in dem der aktuelle Bearbeitungsmodus angezeigt wird.

Parameter

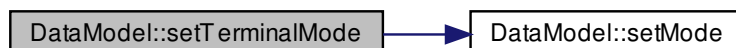
| | |
|--------------|------------------------------------|
| <i>label</i> | Ein Pointer auf ein QLabel Objekt. |
|--------------|------------------------------------|

7.5.3.24 setTerminalMode

```
void DataModel::setTerminalMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Bahnhofseditor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

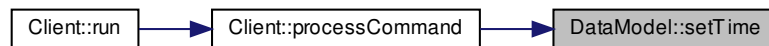


7.5.3.25 setTime()

```
void DataModel::setTime (
    long pTime )
```

[DataModel::setTime](#) Setzt den aktuellen Zeitstempel.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

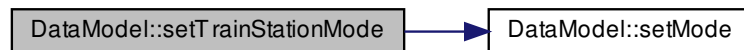


7.5.3.26 setTrainStationMode

```
void DataModel::setTrainStationMode ( ) [slot]
```

[DataModel::setRailPlacementMode](#) Signal um in den Bahnhofseitor zu wechseln.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.5.3.27 takeBalance()

```
bool DataModel::takeBalance (
    int pAmount )
```

[DataModel::takeBalance](#) Zieht Geld ab falls noch genug da ist.

Parameter

| | |
|-----------------------------|-----------------------------|
| <code><i>pAmount</i></code> | Die Geldzahl zum Entfernen. |
|-----------------------------|-----------------------------|

Rückgabe

true wenn genug Geld da war und entfernt wurde. false wenn nicht genug Geld da ist.

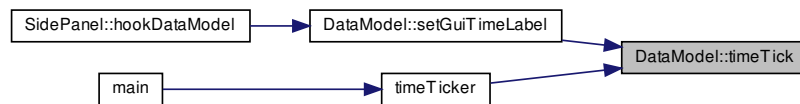
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

**7.5.3.28 timeTick()**

```
void DataModel::timeTick ( )
```

[DataModel::timeTick](#) Wird aufgerufen wenn eine Zeiteinheit verstrichen ist. Erhöht den Timecode.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.5.3.29 updateBalance()**

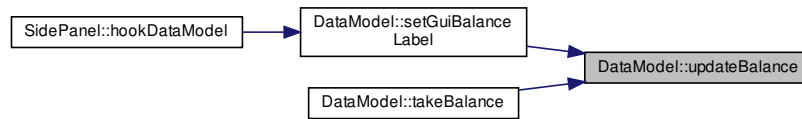
```
void DataModel::updateBalance (
    int pBalance )
```

[DataModel::updateBalance](#) Aktualisiert den Kontostand. Auch in Anzeigen etc.

Parameter

| | |
|-----------------------|----------------------|
| <code>pBalance</code> | Der neue Kontostand. |
|-----------------------|----------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.30 updateCoordinates()

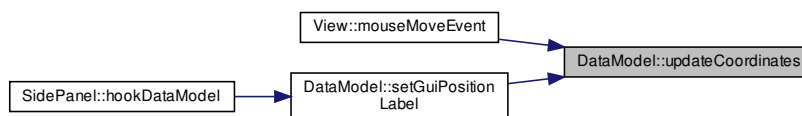
```
void DataModel::updateCoordinates (
    int pX,
    int pY )
```

[DataModel::updateCoordinates](#) Aktualisiert die Koordinaten des fokussierten Quadranten.

Parameter

| | |
|-----------------|-------------------|
| <code>pX</code> | Die X Koordinate. |
| <code>pY</code> | Die Y Koordinate. |

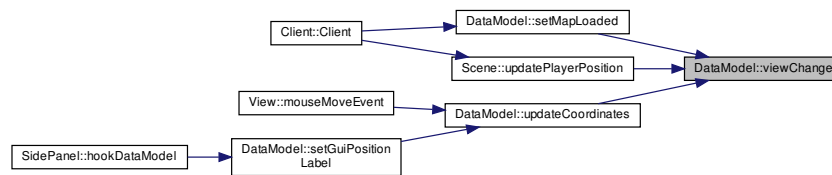
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.3.31 viewChange

```
void DataModel::viewChange ( ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.5.4 Dokumentation der Datenelemente

7.5.4.1 balance

```
int DataModel::balance [private]
```

7.5.4.2 balanceLabel

```
QLabel* DataModel::balanceLabel [private]
```

7.5.4.3 connectionLabel

```
QLabel* DataModel::connectionLabel [private]
```

7.5.4.4 coordinateX

```
int DataModel::coordinateX [private]
```

7.5.4.5 coordinateY

```
int DataModel::coordinateY [private]
```

7.5.4.6 ip

```
QString DataModel::ip [private]
```

7.5.4.7 mapLoaded

```
bool DataModel::mapLoaded {false} [private]
```

7.5.4.8 mode

```
MODE DataModel::mode {MODE::DEFAULT} [private]
```

7.5.4.9 port

```
quint16 DataModel::port [private]
```

7.5.4.10 positionLabel

```
QLabel* DataModel::positionLabel [private]
```

7.5.4.11 secondPlayer

```
Player* DataModel::secondPlayer
```

7.5.4.12 statusDisplay

```
QLabel* DataModel::statusDisplay [private]
```

7.5.4.13 time

```
long DataModel::time [private]
```

7.5.4.14 timeLabel

```
QLabel* DataModel::timeLabel [private]
```

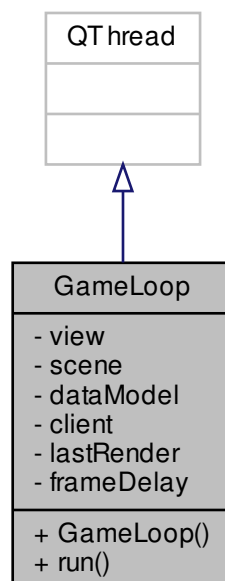
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/datamodel.h](#)
- [src/application_server/datamodel.cpp](#)

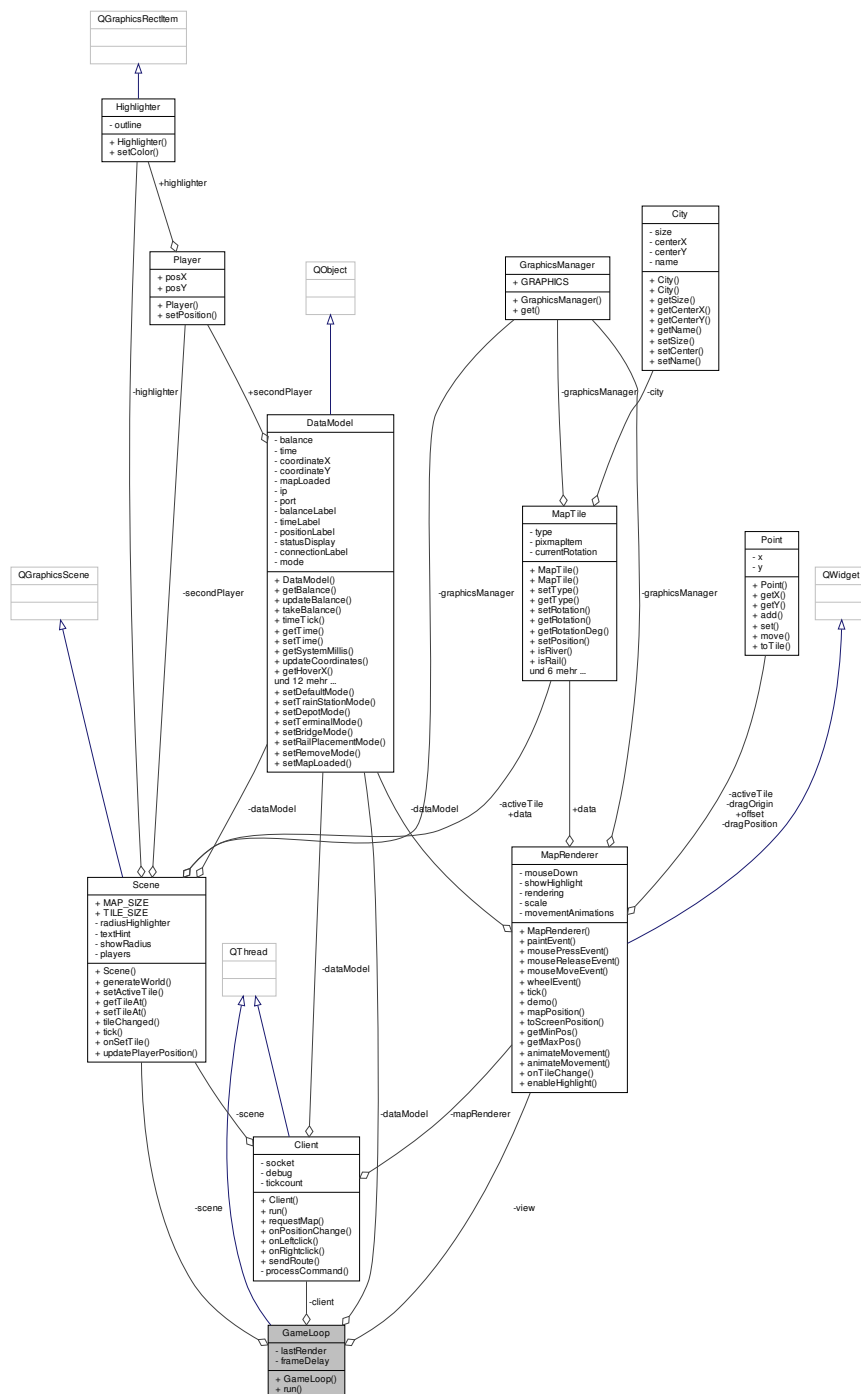
7.6 GameLoop Klassenreferenz

```
#include <gameloop.h>
```

Klassendiagramm für GameLoop:



Zusammengehörigkeiten von GameLoop:



Öffentliche Methoden

- `GameLoop (MapRenderer *, Scene *, DataModel *, Client *)`
- `void run ()` override

GameLoop::run Die Gameloop.

Private Attribute

- `MapRenderer * view`
- `Scene * scene`
- `DataModel * dataModel`
- `Client * client`
- `long lastRender = 0`
- `int frameDelay = 500`

7.6.1 Beschreibung der Konstruktoren und Destruktoren

7.6.1.1 GameLoop()

```
GameLoop::GameLoop (
    MapRenderer * pView,
    Scene * pScene,
    DataModel * pModel,
    Client * pClient )
```

7.6.2 Dokumentation der Elementfunktionen

7.6.2.1 run()

```
void GameLoop::run ( ) [override]
```

`GameLoop::run` Die Gameloop.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.6.3 Dokumentation der Datenelemente

7.6.3.1 client

```
Client* GameLoop::client [private]
```

7.6.3.2 dataModel

```
DataModel* GameLoop::dataModel [private]
```

7.6.3.3 frameDelay

```
int GameLoop::frameDelay = 500 [private]
```

7.6.3.4 lastRender

```
long GameLoop::lastRender = 0 [private]
```

7.6.3.5 scene

```
Scene* GameLoop::scene [private]
```

7.6.3.6 view

```
MapRenderer* GameLoop::view [private]
```

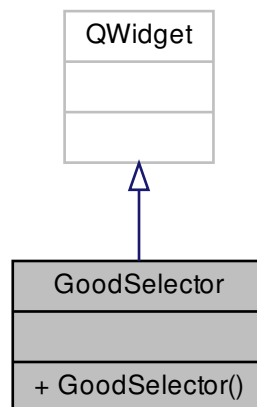
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/gameloop.h](#)
- [src/application_server/gameloop.cpp](#)

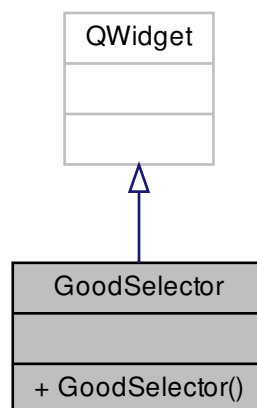
7.7 GoodSelector Klassenreferenz

```
#include <goodselector.h>
```

Klassendiagramm für GoodSelector:



Zusammengehörigkeiten von GoodSelector:



Signale

- void [addWagonClicked](#) (QString *)

Öffentliche Methoden

- [GoodSelector\(\)](#)

7.7.1 Beschreibung der Konstruktoren und Destruktoren

7.7.1.1 GoodSelector()

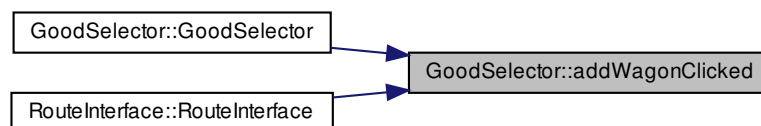
```
GoodSelector::GoodSelector ( )
```

7.7.2 Dokumentation der Elementfunktionen

7.7.2.1 addWagonClicked

```
void GoodSelector::addWagonClicked (
    QString * ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/goodselector.h](#)
- [src/application_server/goodselector.cpp](#)

7.8 GraphicsManager Klassenreferenz

```
#include <graphicsmanager.h>
```

Zusammengehörigkeiten von GraphicsManager:

| |
|--------------------------------|
| GraphicsManager |
| + GRAPHICS |
| + GraphicsManager() + get() |

Öffentliche Methoden

- [GraphicsManager \(\)](#)
[GraphicsManager::GraphicsManager](#) Lädt alle Grafiken. Neue bitte im selben Stil ergänzen.
- QPixmap [get](#) (std::string key)
[GraphicsManager::get](#) Liefert eine Grafik mit einem bestimmten Namen.

Öffentliche Attribute

- std::map< std::string, QPixmap > [GRAPHICS](#)

7.8.1 Beschreibung der Konstruktoren und Destruktoren

7.8.1.1 GraphicsManager()

```
GraphicsManager::GraphicsManager ( )
```

[GraphicsManager::GraphicsManager](#) Lädt alle Grafiken. Neue bitte im selben Stil ergänzen.

7.8.2 Dokumentation der Elementfunktionen

7.8.2.1 get()

```
QPixmap GraphicsManager::get (
    std::string key )
```

[GraphicsManager::get](#) Liefert eine Grafik mit einem bestimmten Namen.

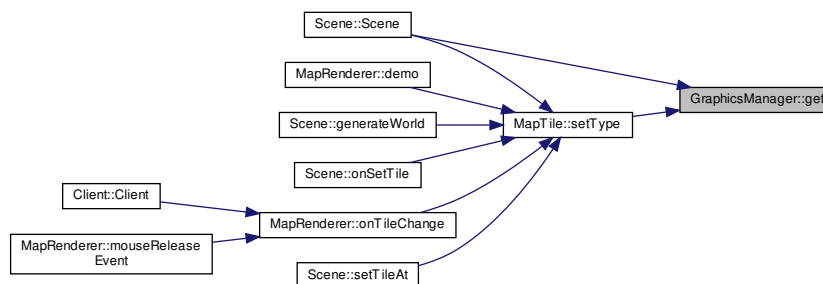
Parameter

| | |
|------------|------------------|
| <i>key</i> | Name der Grafik. |
|------------|------------------|

Rückgabe

Die Grafik.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.8.3 Dokumentation der Datenelemente****7.8.3.1 GRAPHICS**

```
std::map<std::string, QPixmap> GraphicsManager::GRAPHICS
```

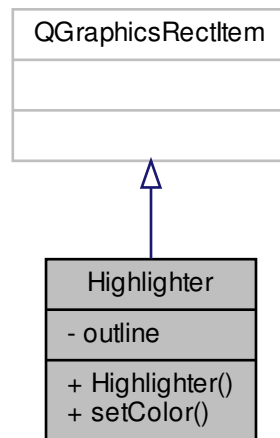
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/graphicsmanager.h](#)
- [src/application_server/graphicsmanager.cpp](#)

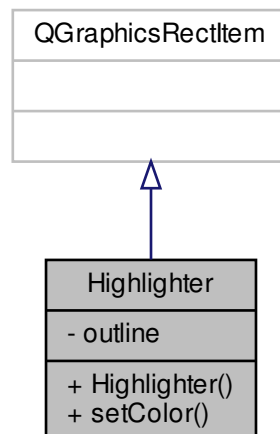
7.9 Highlighter Klassenreferenz

```
#include <highlighter.h>
```

Klassendiagramm für Highlighter:



Zusammengehörigkeiten von Highlighter:



Öffentliche Methoden

- [Highlighter](#) ()
- void [setColor](#) (QColor pColor)

Private Attribute

- QPen * [outline](#)

7.9.1 Beschreibung der Konstruktoren und Destruktoren

7.9.1.1 Highlighter()

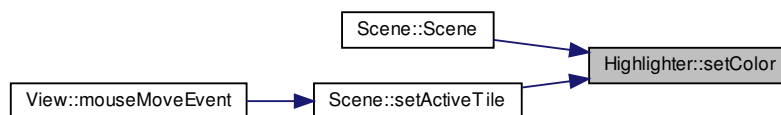
```
Highlighter::Highlighter ( )
```

7.9.2 Dokumentation der Elementfunktionen

7.9.2.1 setColor()

```
void Highlighter::setColor (
    QColor pColor )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.9.3 Dokumentation der Datenelemente

7.9.3.1 outline

```
QPen* Highlighter::outline [private]
```

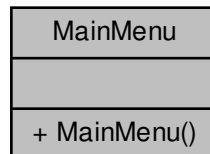
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/highlighter.h](#)
- [src/application_server/highlighter.cpp](#)

7.10 MainMenu Klassenreferenz

```
#include <mainmenu.h>
```

Zusammengehörigkeiten von MainMenu:



Öffentliche Methoden

- [MainMenu \(\)](#)

7.10.1 Beschreibung der Konstruktoren und Destruktoren

7.10.1.1 MainMenu()

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

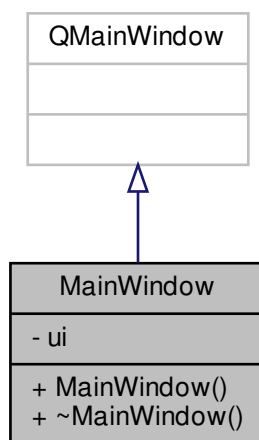
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/mainmenu.h](#)
- [src/application_server/mainmenu.cpp](#)

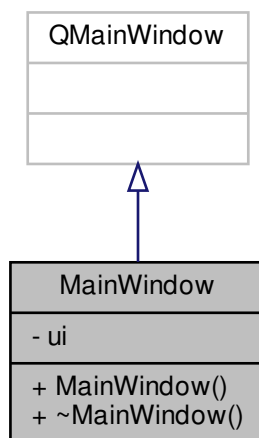
7.11 MainWindow Klassenreferenz

```
#include <mainwindow.h>
```

Klassendiagramm für MainWindow:



Zusammengehörigkeiten von MainWindow:



Öffentliche Methoden

- `MainWindow` (`QWidget *parent=nullptr`)
MainWindow::MainWindow.
- `~MainWindow` ()
MainWindow::~~MainWindow.

Private Attribute

- `Ui::MainWindow * ui`

7.11.1 Beschreibung der Konstruktoren und Destruktoren

7.11.1.1 MainWindow()

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

[MainWindow::MainWindow.](#)

Parameter

| | |
|---------------------|--|
| <code>parent</code> | |
|---------------------|--|

7.11.1.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

[MainWindow::~MainWindow.](#)

7.11.2 Dokumentation der Datenelemente

7.11.2.1 ui

```
Ui::MainWindow* MainWindow::ui [private]
```

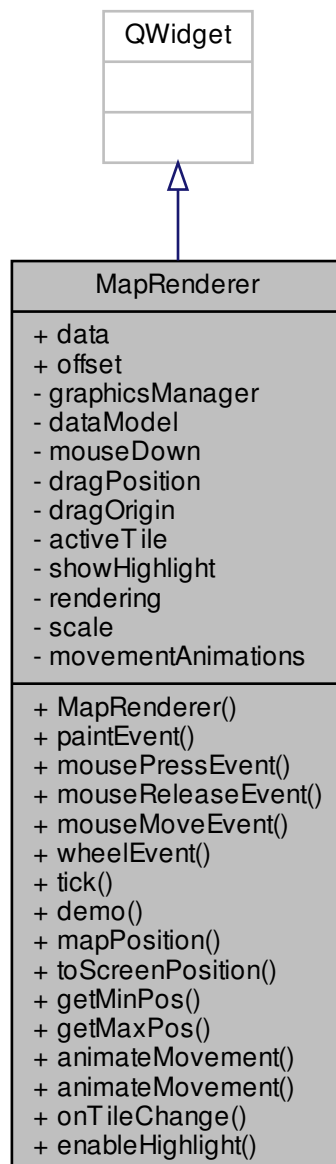
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/mainwindow.h](#)
- [src/application_server/mainwindow.cpp](#)

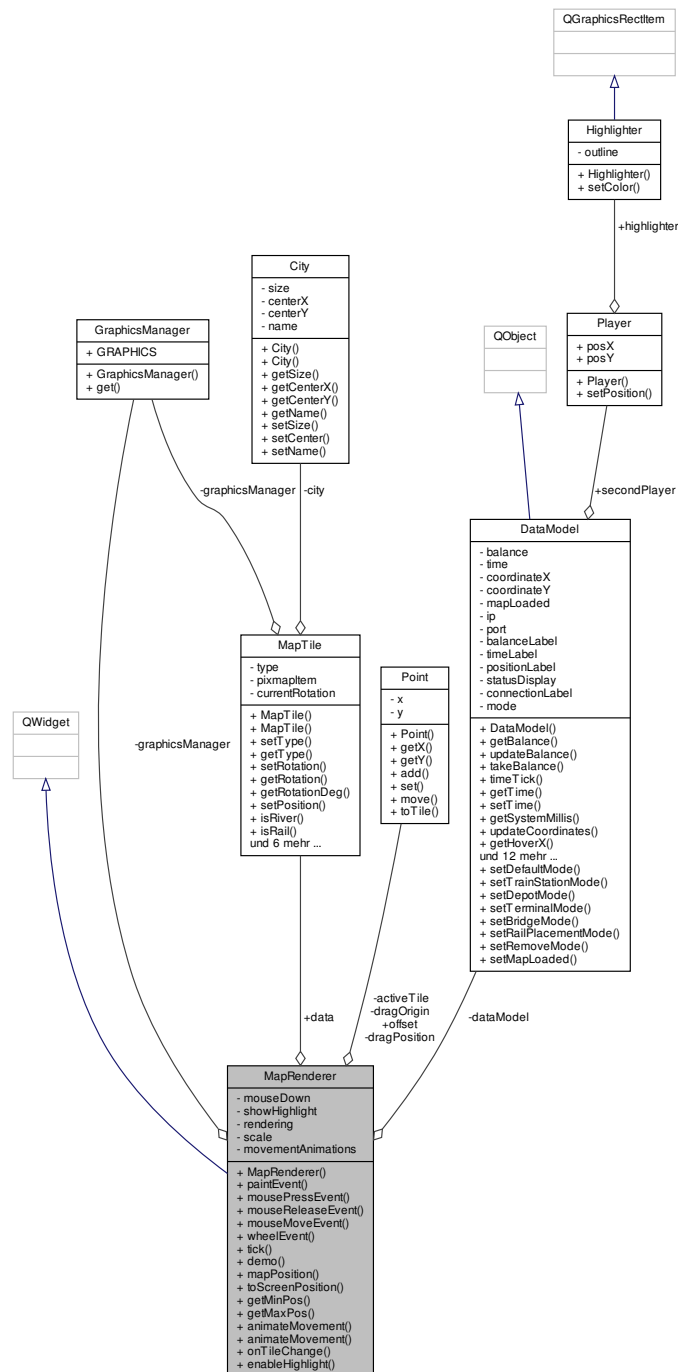
7.12 MapRenderer Klassenreferenz

```
#include <maprenderer.h>
```

Klassendiagramm für MapRenderer:



Zusammengehörigkeiten von MapRenderer:



Öffentliche Slots

- void **onTileChange** (int, int, int)
MapRenderer::onTileChange Wird aufgerufen wenn sich eine Kachel ändert.
- void **enableHighlight** (bool)
MapRenderer::enableHighlight Wird aufgerufen wenn der **Highlighter** aktiviert wird.

Signale

- void [tileClick](#) (int, int, int)
- void [leftclick](#) ()

Öffentliche Methoden

- [MapRenderer](#) ([GraphicsManager](#) *, [DataModel](#) *)
[MapRenderer::MapRenderer](#) Erzeugt einen neuen [MapRenderer](#).
- void [paintEvent](#) ([QPaintEvent](#) *event) override
[MapRenderer::paintEvent](#) Rendert die Spielansicht.
- void [mousePressEvent](#) ([QMouseEvent](#) *event) override
- void [mouseReleaseEvent](#) ([QMouseEvent](#) *event) override
- void [mouseMoveEvent](#) ([QMouseEvent](#) *event) override
- void [wheelEvent](#) ([QWheelEvent](#) *event) override
- void [tick](#) ()
- void [demo](#) ()
[MapRenderer::demo](#) Zeigt eine Demo-Visualisierung.
- [Point](#) [mapPosition](#) (int, int)
[MapRenderer::mapPosition](#) Wandelt eine Position im Fenster in eine Kachelkoordinate um.
- [Point](#) [toScreenPosition](#) (int, int)
[MapRenderer::toScreenPosition](#) Wandelt einen Kachelindex in eine Koordinate auf dem Bildschirm um.
- [Point](#) [getMinPos](#) ()
[MapRenderer::getMinPos](#) Gibt die minimale Position des Viewports zurück.
- [Point](#) [getMaxPos](#) ()
[MapRenderer::getMinPos](#) Gibt die maximale Position des Viewports zurück.
- void [animateMovement](#) ([QImage](#), [QString](#))
- void [animateMovement](#) ([QImage](#), [QString](#), double)

Öffentliche Attribute

- [MapTile](#) [data](#) [300][300]
- [Point](#) [offset](#) {0,0}

Private Attribute

- [GraphicsManager](#) * [graphicsManager](#)
- [DataModel](#) * [dataModel](#)
- bool [mouseDown](#) {false}
- [Point](#) [dragPosition](#) {0,0}
- [Point](#) [dragOrigin](#) {0,0}
- [Point](#) [activeTile](#) {0,0}
- bool [showHighlight](#) {false}
- bool [rendering](#) {false}
- double [scale](#) {1}
- [QList](#)< [AnimationMovement](#) * > [movementAnimations](#)

7.12.1 Beschreibung der Konstruktoren und Destruktoren

7.12.1.1 MapRenderer()

```
MapRenderer::MapRenderer (
    GraphicsManager * pGraphicsManager,
    DataModel * pDataModel )
```

[MapRenderer::MapRenderer](#) Erzeugt einen neuen [MapRenderer](#).

Parameter

| | |
|-------------------------|-------------------------------------|
| <i>pGraphicsManager</i> | Der GraphicsManager |
| <i>pDataModel</i> | Das Datenmodell. |

7.12.2 Dokumentation der Elementfunktionen

7.12.2.1 animateMovement() [1/2]

```
void MapRenderer::animateMovement (
    QImage img,
    QString path )
```

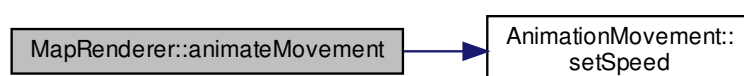
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.2 animateMovement() [2/2]

```
void MapRenderer::animateMovement (
    QImage img,
    QString path,
    double speed )
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

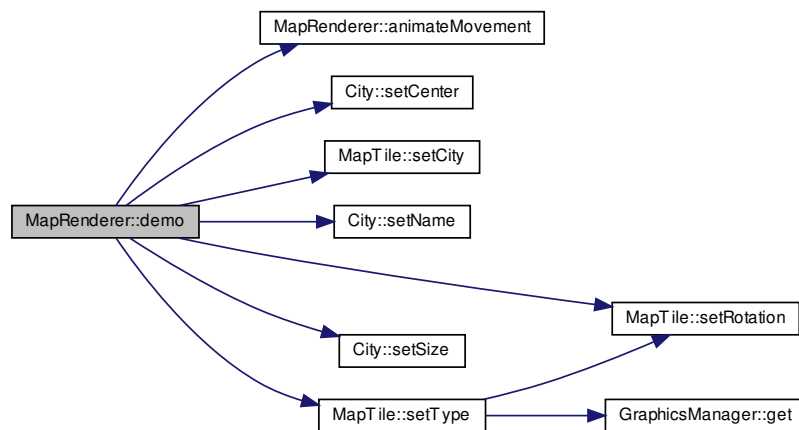


7.12.2.3 demo()

```
void MapRenderer::demo ( )
```

[MapRenderer::demo](#) Zeigt eine Demo-Visualisierung.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.12.2.4 enableHighlight

```
void MapRenderer::enableHighlight (
    bool status ) [slot]
```

[MapRenderer::enableHighlight](#) Wird aufgerufen wenn der [Highlighter](#) aktiviert wird.

Parameter

| | |
|---------------------|---------------------|
| <code>status</code> | Der Status (an/aus) |
|---------------------|---------------------|

7.12.2.5 getMaxPos()

```
Point MapRenderer::getMaxPos ( )
```

[MapRenderer::getMinPos](#) Gibt die maximale Position des Viewports zurück.

Rückgabe

Ein [Point](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.12.2.6 getMinPos()**

```
Point MapRenderer::getMinPos ( )
```

[MapRenderer::getMinPos](#) Gibt die minimale Position des Viewports zurück.

Rückgabe

Ein [Point](#).

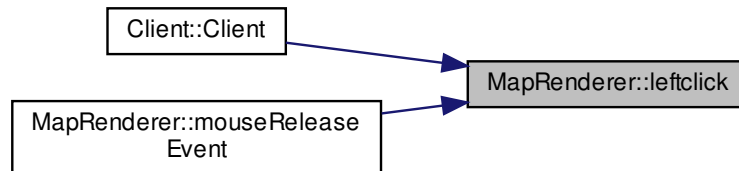
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.7 leftclick

```
void MapRenderer::leftclick ( ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.8 mapPosition()

```
Point MapRenderer::mapPosition (
    int px,
    int py )
```

[MapRenderer::mapPosition](#) Wandelt eine Position im Fenster in eine Kachelkoordinate um.

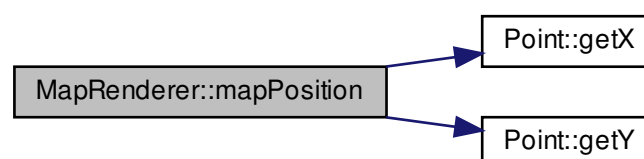
Parameter

| | |
|-----------|------------------------------|
| <i>px</i> | Die X-Koordinate im Fenster. |
| <i>py</i> | Die Y-Koordinate im Fenster. |

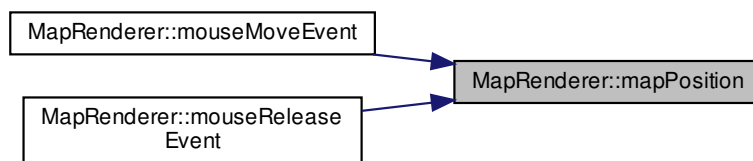
Rückgabe

Der Punkt.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



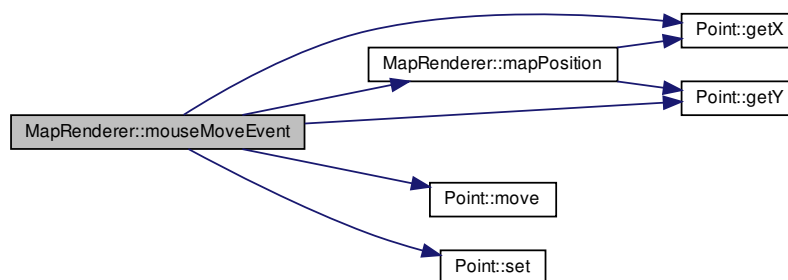
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.9 mouseMoveEvent()

```
void MapRenderer::mouseMoveEvent (
    QMouseEvent * event ) [override]
```

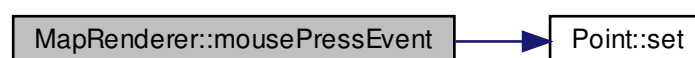
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.12.2.10 mousePressEvent()

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

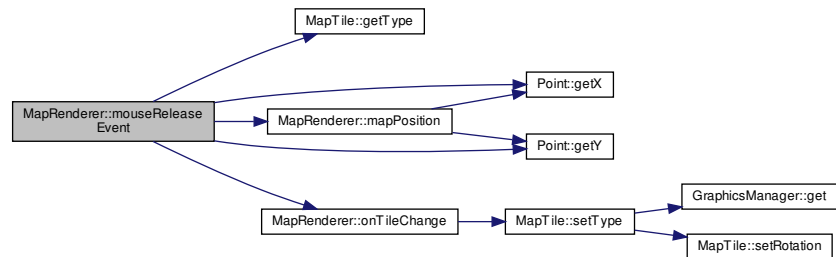
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.12.2.11 mouseReleaseEvent()

```
void MapRenderer::mouseReleaseEvent (
    QMouseEvent * event ) [override]
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.12.2.12 onTileChange

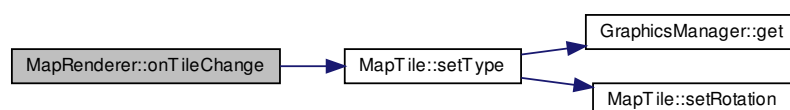
```
void MapRenderer::onTileChange (
    int px,
    int py,
    int type ) [slot]
```

[MapRenderer::onTileChange](#) Wird aufgerufen wenn sich eine Kachel ändert.

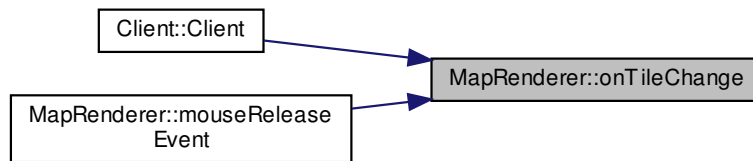
Parameter

| | |
|-------------|---|
| <i>px</i> | Der X-Index |
| <i>py</i> | Der Y-Index |
| <i>type</i> | Der neue Typ als Zahl (Index in MapTile::TYPE) |

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.13 paintEvent()

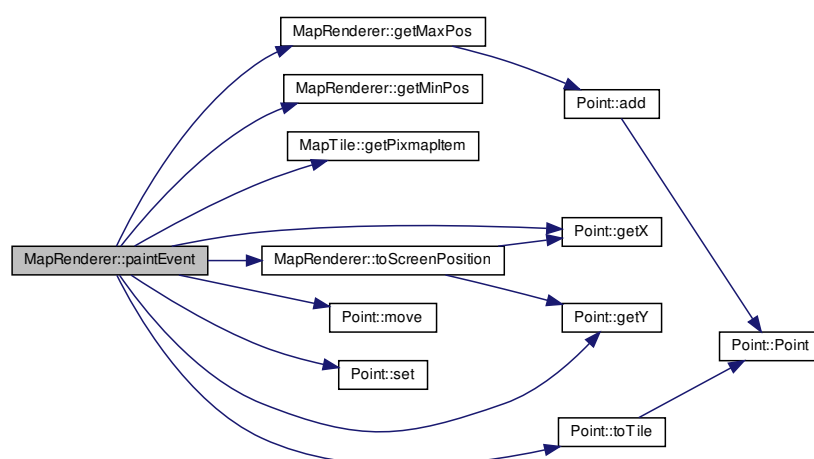
```
void MapRenderer::paintEvent (
    QPaintEvent * event ) [override]
```

[MapRenderer::paintEvent](#) Rendert die Spielansicht.

Parameter

| | |
|--------------|------------------|
| <i>event</i> | Ein QPaintEvent. |
|--------------|------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.12.2.14 tick()

```
void MapRenderer::tick ( )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.15 tileClick

```
void MapRenderer::tileClick (
    int ,
    int ,
    int ) [signal]
```

7.12.2.16 toScreenPosition()

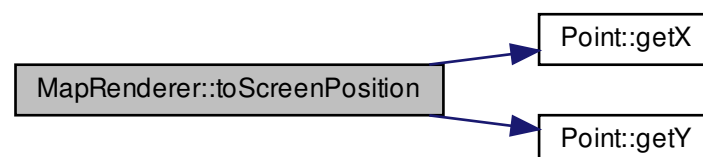
```
Point MapRenderer::toScreenPosition (
    int px,
    int py )
```

[MapRenderer::toScreenPosition](#) Wandelt einen Kachelindex in eine Koordinate auf dem Bildschirm um.

Rückgabe

Der Punkt.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.12.2.17 wheelEvent()

```
void MapRenderer::wheelEvent (
    QWheelEvent * event ) [override]

if(event->delta()>0){ scale-=0.1; }else{ scale+=0.1; }
```

7.12.3 Dokumentation der Datenelemente

7.12.3.1 activeTile

```
Point MapRenderer::activeTile {0,0} [private]
```

7.12.3.2 data

```
MapTile MapRenderer::data[300][300]
```

7.12.3.3 dataModel

```
DataModel* MapRenderer::dataModel [private]
```

7.12.3.4 dragOrigin

```
Point MapRenderer::dragOrigin {0,0} [private]
```

7.12.3.5 dragPosition

```
Point MapRenderer::dragPosition {0,0} [private]
```

7.12.3.6 graphicsManager

```
GraphicsManager* MapRenderer::graphicsManager [private]
```

7.12.3.7 mouseDown

```
bool MapRenderer::mouseDown {false} [private]
```

7.12.3.8 movementAnimations

```
QList<AnimationMovement *> MapRenderer::movementAnimations [private]
```

7.12.3.9 offset

```
Point MapRenderer::offset {0,0}
```

7.12.3.10 rendering

```
bool MapRenderer::rendering {false} [private]
```

7.12.3.11 scale

```
double MapRenderer::scale {1} [private]
```


7.12.3.12 showHighlight

```
bool MapRenderer::showHighlight {false} [private]
```

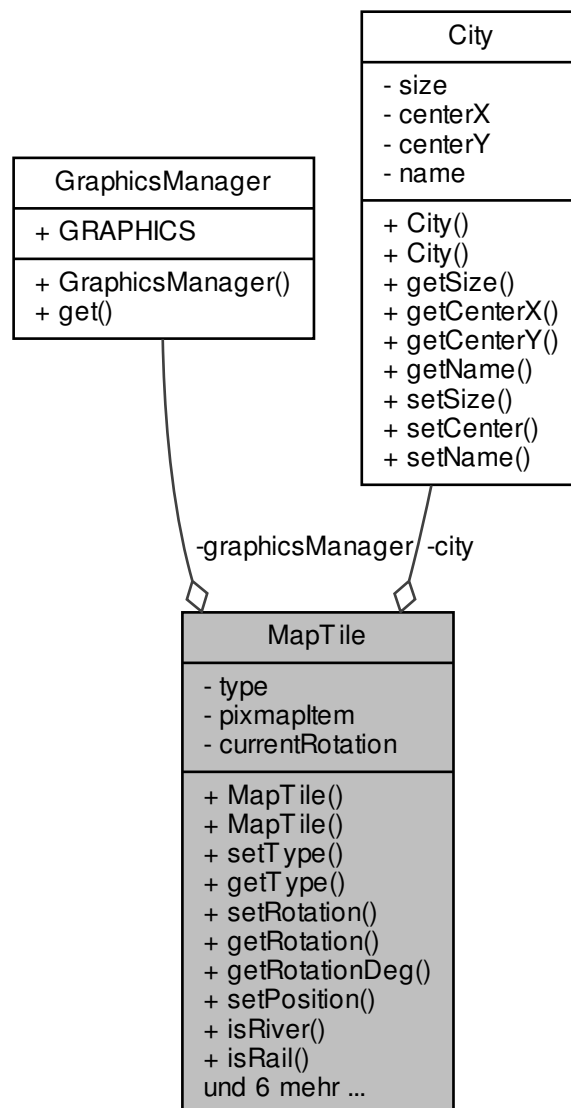
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/maprenderer.h](#)
- [src/application_server/maprenderer.cpp](#)

7.13 MapTile Klassenreferenz

```
#include <maptile.h>
```

Zusammengehörigkeiten von MapTile:



Öffentliche Typen

- enum [TYPE](#) {
[GRASS](#), [FORREST](#), [CITY](#), [RIVER_H](#),
[RIVER_V](#), [RIVER_LB](#), [RIVER_LT](#), [RIVER_RT](#),
[RIVER_RB](#), [RAIL_H](#), [RAIL_V](#), [RAIL_LB](#),
[RAIL_LT](#), [RAIL_RT](#), [RAIL_RB](#), [WATER](#),
[DEPOT_H](#), [DEPOT_V](#), [STATION_H](#), [STATION_V](#),
[TERMINAL_H](#), [TERMINAL_V](#), [BRIDGE_H](#), [BRIDGE_V](#) }

Öffentliche Methoden

- [MapTile](#) ([GraphicsManager](#) *pGraphicsManager)
[MapTile::MapTile](#) Konstruktor.
- [MapTile](#) ()
[MapTile::MapTile](#) Konstruktor.
- void [setType](#) ([TYPE](#) pType)
[MapTile::setType](#) Setzt den Typ der Kachel.
- [TYPE](#) [getType](#) ()
[MapTile::getType](#) Liefert den Typ des Quadranten.
- void [setRotation](#) (int pRotation)
[MapTile::setRotation](#) Hilfsfunktion zur Rotation im Quadrat.
- int [getRotation](#) ()
[MapTile::getRotation](#) Liefert die aktuelle Rotation. (Himmelsrichtung)
- int [getRotationDeg](#) ()
- void [setPosition](#) (int posX, int posY)
[MapTile::setPosition](#) Setzt die Position der Kachel. (In Pixeln)
- bool [isRiver](#) ()
[MapTile::isRiver](#) Checkt ob die Kachel ein Fluss ist.
- bool [isRail](#) ()
[MapTile::isRail](#) Checkt ob die Kachel eine Schiene ist.
- [QGraphicsPixmapItem](#) * [getPixmapItem](#) ()
[MapTile::getPixmapItem](#) Liefert das QPixmap Item.
- [City](#) * [getCity](#) ()
[MapTile::getCity](#) Die Informationen. Falls keine Stadt: null.
- void [setCity](#) ([City](#) *pCity)
[MapTile::setCity](#).
- int [getX](#) ()
[MapTile::getX](#).
- int [getY](#) ()
[MapTile::getY](#).
- void [attachGraphicsManager](#) ([GraphicsManager](#) *pGraphicsManager)
[MapTile::attachGraphicsManager](#) Setzte den [GraphicsManager](#).

Private Attribute

- [TYPE](#) type
- [QGraphicsPixmapItem](#) * [pixmapItem](#)
- int [currentRotation](#) {0}
- [City](#) * city
- [GraphicsManager](#) * [graphicsManager](#)

7.13.1 Dokumentation der Aufzählungstypen

7.13.1.1 TYPE

enum `MapTile::TYPE`

Aufzählungswerte

| | |
|------------|--|
| GRASS | |
| FORREST | |
| CITY | |
| RIVER_H | |
| RIVER_V | |
| RIVER_LB | |
| RIVER_LT | |
| RIVER_RT | |
| RIVER_RB | |
| RAIL_H | |
| RAIL_V | |
| RAIL_LB | |
| RAIL_LT | |
| RAIL_RT | |
| RAIL_RB | |
| WATER | |
| DEPOT_H | |
| DEPOT_V | |
| STATION_H | |
| STATION_V | |
| TERMINAL_H | |
| TERMINAL_V | |
| BRIDGE_H | |
| BRIDGE_V | |

7.13.2 Beschreibung der Konstruktoren und Destruktoren

7.13.2.1 MapTile() [1/2]

```
MapTile::MapTile (
    GraphicsManager * pGraphicsManager )
```

`MapTile::MapTile` Konstruktor.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.13.2.2 MapTile() [2/2]

```
MapTile::MapTile ( )
```

[MapTile::MapTile](#) Konstruktor.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3 Dokumentation der Elementfunktionen

7.13.3.1 attachGraphicsManager()

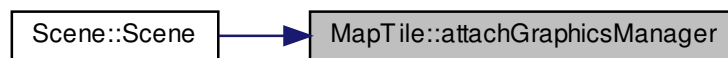
```
void MapTile::attachGraphicsManager (
    GraphicsManager * pGraphicsManager )
```

[MapTile::attachGraphicsManager](#) Setzte den [GraphicsManager](#).

Parameter

| | |
|-------------------------|---------------------------------------|
| <i>pGraphicsManager</i> | Ein GraphicsManager . |
|-------------------------|---------------------------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.2 `getCity()`

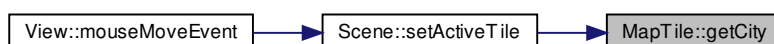
```
City * MapTile::getCity ( )
```

`MapTile::getCity` Die Informationen. Falls keine Stadt: null.

Rückgabe

Liefert die Informationen über eine Stadt auf der Kachel.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.3 `getPixmapItem()`

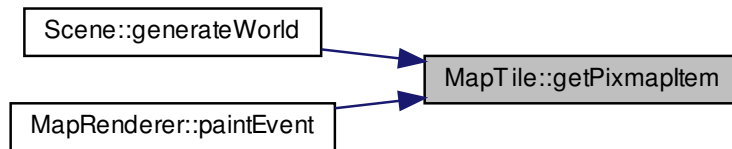
```
QGraphicsPixmapItem * MapTile::getPixmapItem ( )
```

`MapTile::getPixmapItem` Liefert das Pixmap Item.

Rückgabe

Das Pixmap Item.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.13.3.4 getRotation()**

```
int MapTile::getRotation ( )
```

[MapTile::getRotation](#) Liefert die aktuelle Rotation. (Himmelsrichtung)

Rückgabe

Die aktuelle Rotation (0-3)

7.13.3.5 getRotationDeg()

```
int MapTile::getRotationDeg ( )
```

7.13.3.6 getType()

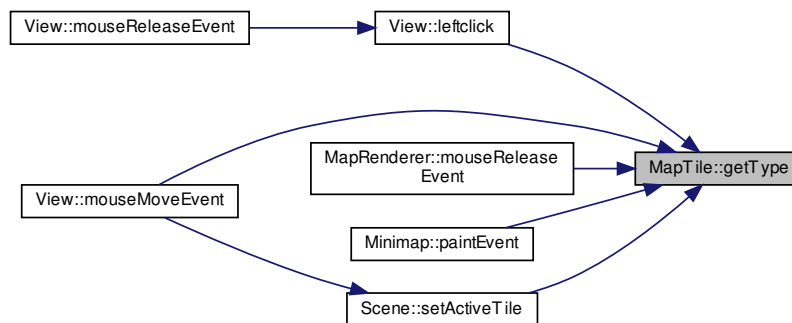
```
MapTile::TYPE MapTile::getType ( )
```

[MapTile::getType](#) Liefert den Typ des Quadranten.

Rückgabe

Den Typ.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.13.3.7 getX()**

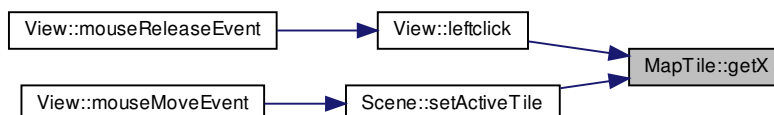
```
int MapTile::getX ( )
```

[MapTile::getX](#).

Rückgabe

Der X Index des Quadranten.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.8 getY()

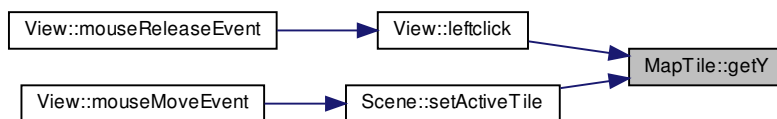
```
int MapTile::getY ( )
```

[MapTile::getY](#).

Rückgabe

Der Y Index des Quadranten.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.9 isRail()

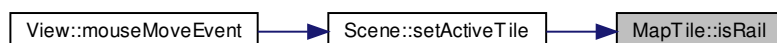
```
bool MapTile::isRail ( )
```

[MapTile::isRail](#) Checkt ob die Kachel eine Schiene ist.

Rückgabe

Ob die Kachel eine Schiene ist.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.10 isRiver()

```
bool MapTile::isRiver ( )
```

[MapTile::isRiver](#) Checkt ob die Kachel ein Fluss ist.

Rückgabe

Ob die Kachel ein Fluss ist.

7.13.3.11 setCity()

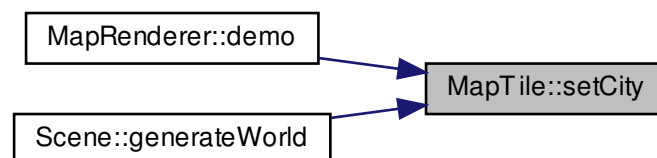
```
void MapTile::setCity (
    City * pCity )
```

[MapTile::setCity](#).

Parameter

| | |
|--------------|--|
| <i>pCity</i> | Fügt dem Quadranten Daten über eine Stadt hinzu. |
|--------------|--|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.12 setPosition()

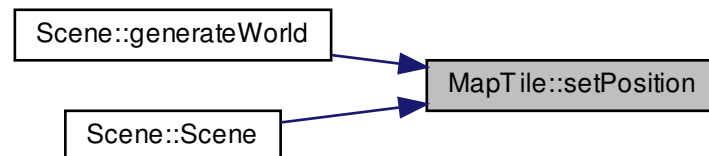
```
void MapTile::setPosition (
    int posX,
    int posY )
```

[MapTile::setPosition](#) Setzt die Position der Kachel. (In Pixeln)

Parameter

| | |
|-------------|-------------------|
| <i>posX</i> | Die X Koordinate. |
| <i>posY</i> | Die Y Koordinate. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

7.13.3.13 `setRotation()`

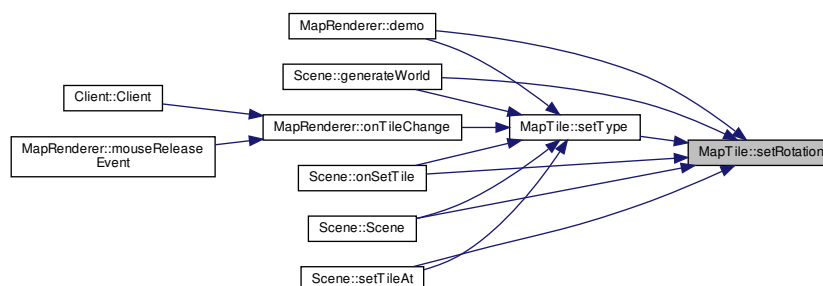
```
void MapTile::setRotation (
    int pRotation )
```

[MapTile::setRotation](#) Hilfsfunktion zur Rotation im Quadrat.

Parameter

| | |
|------------------|--|
| <i>pRotation</i> | 0=Ursprung 1=90° Grad 2=180° Grad 3=270° |
|------------------|--|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.3.14 setType()

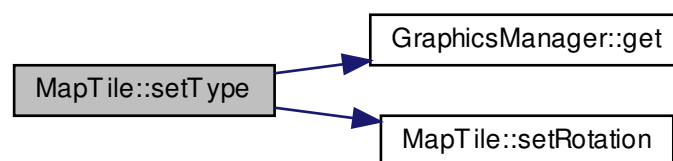
```
void MapTile::setType (
    MapTile::TYPE pType )
```

[MapTile::setType](#) Setzt den Typ der Kachel.

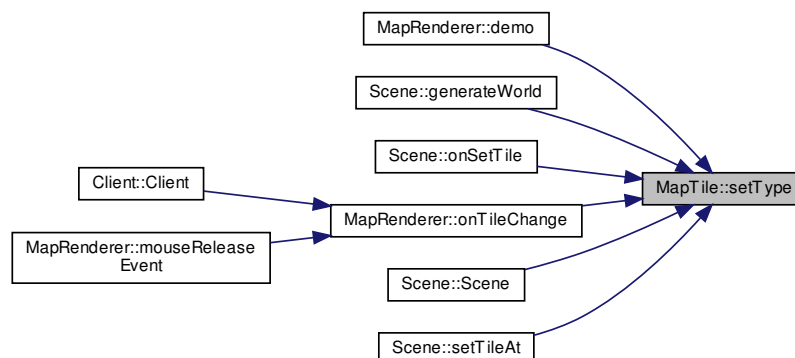
Parameter

| | |
|--------------|----------|
| <i>pType</i> | Der Typ. |
|--------------|----------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.13.4 Dokumentation der Datenelemente

7.13.4.1 city

```
City* MapTile::city [private]
```

7.13.4.2 currentRotation

```
int MapTile::currentRotation {0} [private]
```

7.13.4.3 graphicsManager

```
GraphicsManager* MapTile::graphicsManager [private]
```

7.13.4.4 pixmapItem

```
QGraphicsPixmapItem* MapTile::pixmapItem [private]
```

7.13.4.5 type

```
TYPE MapTile::type [private]
```

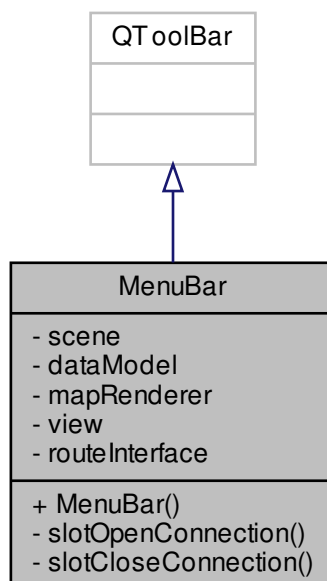
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/maptile.h](#)
- [src/application_server/maptile.cpp](#)

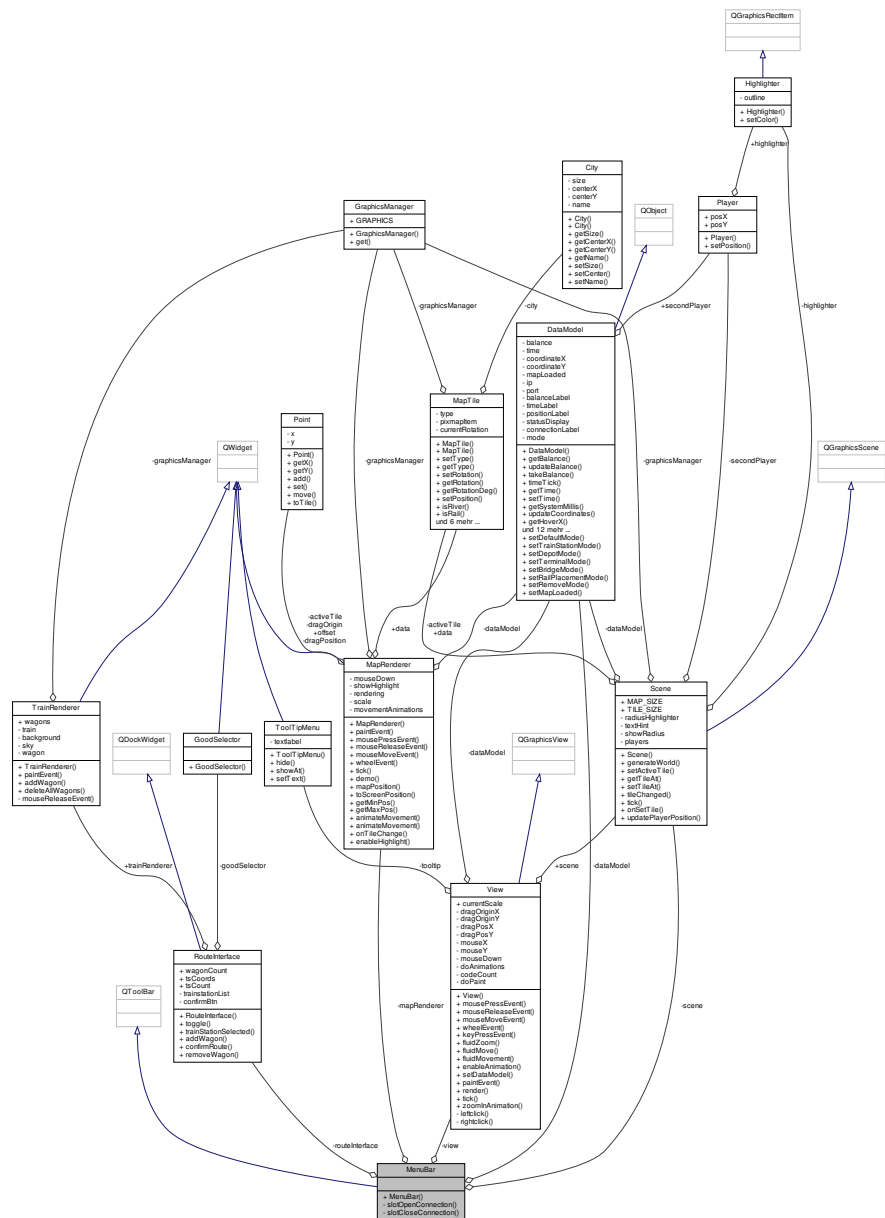
7.14 MenuBar Klassenreferenz

```
#include <menubar.h>
```

Klassendiagramm für MenuBar:



Zusammengehörigkeiten von MenuBar:



Öffentliche Methoden

- **MenuBar** (Scene *pScene, **MapRenderer** *pRenderer, **DataModel** *pDataModel, **View** *pView, **RouteInterface** *)

MenuBar::MenuBar Erzeugt Menüstruktur.

Private Slots

- void slotOpenConnection ()

MenuBar::openConnection Öffnet Input-Dialog für IP-Adresse und initiiert *Client*.

- void slotCloseConnection ()

MenuBar::closeConnection Schließt die aktuelle Verbindung mit dem Server.

Private Attribute

- [Scene](#) * [scene](#)
- [DataModel](#) * [dataModel](#)
- [MapRenderer](#) * [mapRenderer](#)
- [View](#) * [view](#)
- [RouteInterface](#) * [routeInterface](#)

7.14.1 Beschreibung der Konstruktoren und Destruktoren

7.14.1.1 MenuBar()

```
MenuBar::MenuBar (
    Scene * pScene,
    MapRenderer * pRenderer,
    DataModel * pDataModel,
    View * pView,
    RouteInterface * pRouteInterface )
```

[MenuBar::MenuBar](#) Erzeugt Menüstruktur.

7.14.2 Dokumentation der Elementfunktionen

7.14.2.1 slotCloseConnection

```
void MenuBar::slotCloseConnection ( ) [private], [slot]
```

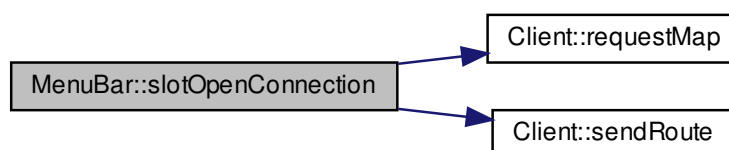
[MenuBar::closeConnection](#) Schließt die aktuelle Verbindung mit dem Server.

7.14.2.2 slotOpenConnection

```
void MenuBar::slotOpenConnection ( ) [private], [slot]
```

[MenuBar::openConnection](#) Öffnet Input-Dialog für IP-Adresse und initiiert [Client](#).

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.14.3 Dokumentation der Datenelemente

7.14.3.1 dataModel

```
DataModel* MenuBar::dataModel [private]
```

7.14.3.2 mapRenderer

```
MapRenderer* MenuBar::mapRenderer [private]
```

7.14.3.3 routeInterface

```
RouteInterface* MenuBar::routeInterface [private]
```

7.14.3.4 scene

```
Scene* MenuBar::scene [private]
```

7.14.3.5 view

```
View* MenuBar::view [private]
```

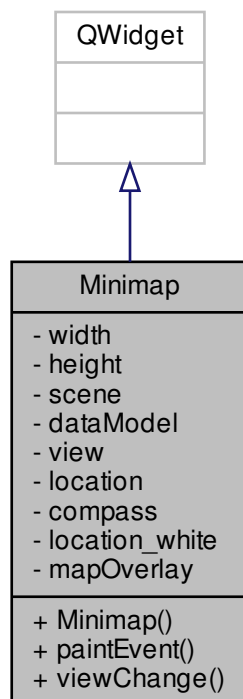
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/menubar.h](#)
- [src/application_server/menubar.cpp](#)

7.15 Minimap Klassenreferenz

```
#include <minimap.h>
```

Klassendiagramm für Minimap:



Öffentliche Methoden

- [Minimap](#) (int, int, [Scene](#) *, [View](#) *, [DataModel](#) *)
[Minimap::Minimap](#) Erzeugt eine neue [Minimap](#) Komponente.
- void [paintEvent](#) (QPaintEvent *event) override
[Minimap::paintEvent](#) Rendert die [Minimap](#).

Private Attribute

- int [width](#)
- int [height](#)
- [Scene](#) * [scene](#)
- [DataModel](#) * [dataModel](#)
- [View](#) * [view](#)
- QImage [location](#)
- QImage [compass](#)
- QImage [location_white](#)
- QImage [mapOverlay](#)

7.15.1 Beschreibung der Konstruktoren und Destruktoren

7.15.1.1 Minimap()

```
Minimap::Minimap (
    int pWidth,
    int pHeight,
    Scene * pScene,
    View * pView,
    DataModel * pDataModel )
```

[Minimap::Minimap](#) Erzeugt eine neue [Minimap](#) Komponente.

Parameter

| | |
|----------------------------|--|
| pWidth | Die Breite der Minimap in Pixeln |
| pHeight | Die Höhe der Minimap in Pixeln |
| pScene | Die Szene |
| pDataModel | Das DataModel |

7.15.2 Dokumentation der Elementfunktionen

7.15.2.1 paintEvent()

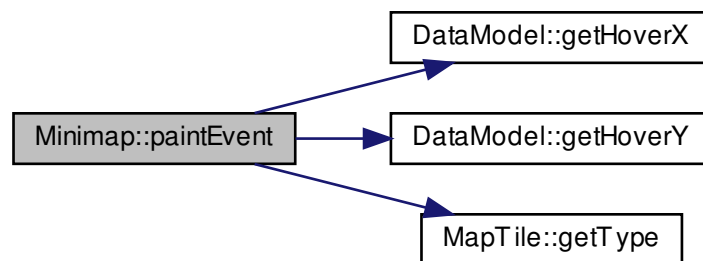
```
void Minimap::paintEvent (
    QPaintEvent * event ) [override]
```

[Minimap::paintEvent](#) Rendert die [Minimap](#).

Parameter

| | |
|--------------|-----------------------|
| <i>event</i> | Das zugehörige Event. |
|--------------|-----------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.15.2.2 viewChange

```
void Minimap::viewChange ( ) [slot]
```

[Minimap::viewChange](#) Slot der aufgerufen wird wenn die [Minimap](#) komplett neu gezeichnet werden soll.

7.15.3 Dokumentation der Datenelemente

7.15.3.1 compass

```
QImage Minimap::compass [private]
```

7.15.3.2 dataModel

```
DataModel* Minimap::dataModel [private]
```

7.15.3.3 height

```
int Minimap::height [private]
```

7.15.3.4 location

```
QImage Minimap::location [private]
```

7.15.3.5 location_white

```
QImage Minimap::location_white [private]
```

7.15.3.6 mapOverlay

```
QImage Minimap::mapOverlay [private]
```

7.15.3.7 scene

```
Scene* Minimap::scene [private]
```

7.15.3.8 view

```
View* Minimap::view [private]
```

7.15.3.9 width

```
int Minimap::width [private]
```

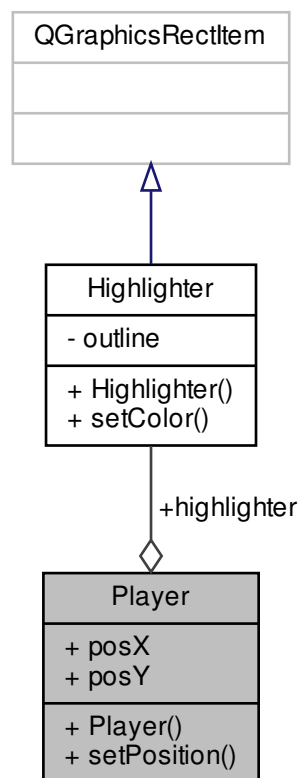
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/minimap.h](#)
- [src/application_server/minimap.cpp](#)

7.16 Player Klassenreferenz

```
#include <player.h>
```

Zusammengehörigkeiten von Player:



Öffentliche Methoden

- [Player](#) ()
- void [setPosition](#) (int pX, int pY)
[Player::setPosition](#) Updated die Position des Spielers.

Öffentliche Attribute

- [Highlighter](#) * [highlighter](#)
- int [posX](#)
- int [posY](#)

7.16.1 Beschreibung der Konstruktoren und Destruktoren

7.16.1.1 Player()

```
Player::Player ( )
```

7.16.2 Dokumentation der Elementfunktionen

7.16.2.1 setPosition()

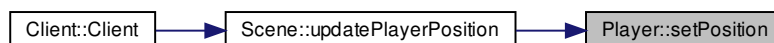
```
void Player::setPosition (
    int pX,
    int pY )
```

[Player::setPosition](#) Updated die Position des Spielers.

Parameter

| | |
|-----------|-------------|
| <i>pX</i> | Der X-Index |
| <i>pY</i> | Der Y-Index |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.16.3 Dokumentation der Datenelemente

7.16.3.1 highlighter

```
Highlighter* Player::highlighter
```

7.16.3.2 posX

```
int Player::posX
```

7.16.3.3 posY

```
int Player::posY
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/player.h](#)
- [src/application_server/player.cpp](#)

7.17 Point Klassenreferenz

```
#include <point.h>
```

Zusammengehörigkeiten von Point:

| Point |
|---|
| - x - y |
| + Point() + getX() + getY() + add() + set() + move() + toTile() |

Öffentliche Methoden

- `Point` (int, int)
`Point::Point` Erzeugt einen 2D-Punkt.
- int `getX` ()
`Point::getX` Gibt die X-Koordinate des Punktes.
- int `getY` ()
`Point::getY` Gibt die Y-Koordinate des Punktes.
- `Point add` (int, int)
`Point::add` Addiert Werte auf den Punkt.
- void `set` (int, int)
`Point::set` Ändert die Koordinaten.
- void `move` (int, int)
`Point::move` Verschiebt den Punkt.
- `Point toTile` ()
`Point::toTile` Gibt den Punkt als Tile-Index zurück.

Private Attribute

- int `x`
- int `y`

7.17.1 Beschreibung der Konstruktoren und Destruktoren

7.17.1.1 `Point()`

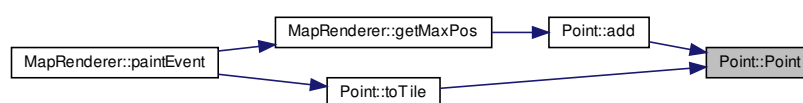
```
Point::Point (
    int pX,
    int pY )
```

`Point::Point` Erzeugt einen 2D-Punkt.

Parameter

| | |
|-----------------|-------------------|
| <code>pX</code> | Die X-Koordinate. |
| <code>pY</code> | Die Y-Koordinate. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.17.2 Dokumentation der Elementfunktionen

7.17.2.1 add()

```
Point Point::add (
    int ax,
    int ay )
```

Point::add Addiert Werte auf den Punkt.

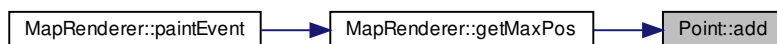
Rückgabe

Ein neuer Punkt.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.17.2.2 getX()

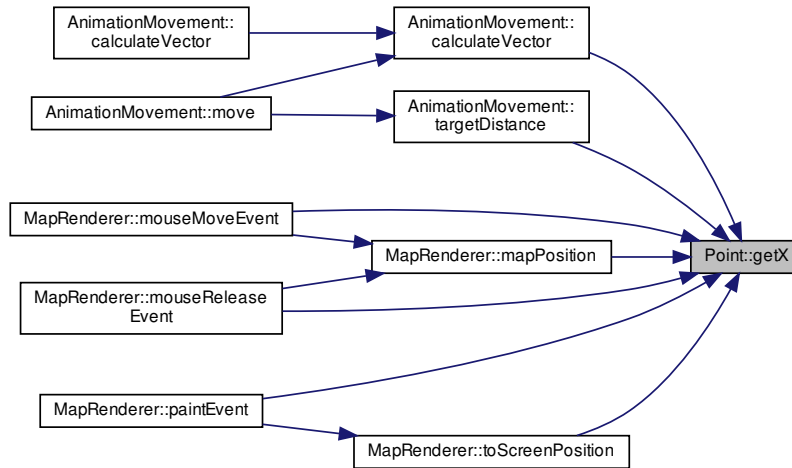
```
int Point::getX ( )
```

Point::getX Gibt die X-Koordinate des Punktes.

Rückgabe

Die X-Koordinate.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

**7.17.2.3 getY()**

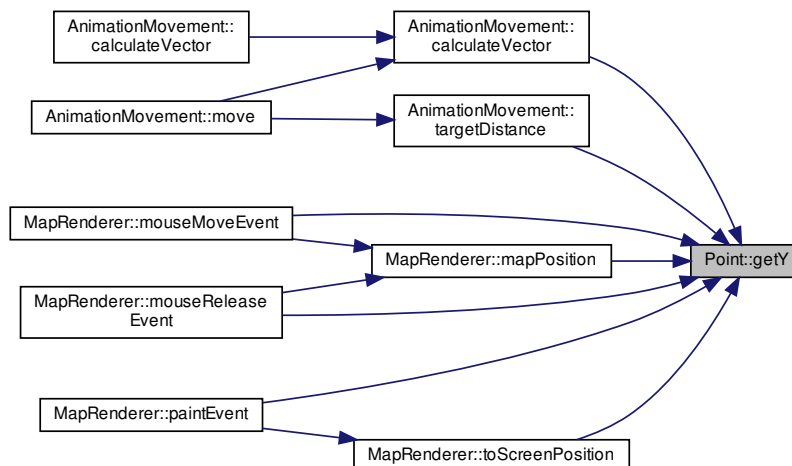
```
int Point::getY ( )
```

[Point::getX](#) Gibt die Y-Koordinate des Punktes.

Rückgabe

Die Y-Koordinate.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

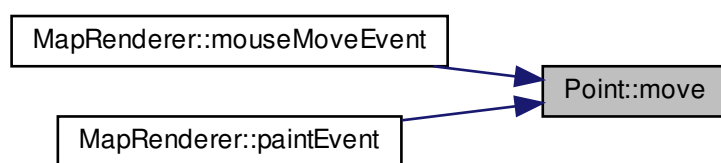


7.17.2.4 move()

```
void Point::move (
    int mx,
    int my )
```

[Point::move](#) Verschiebt den Punkt.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

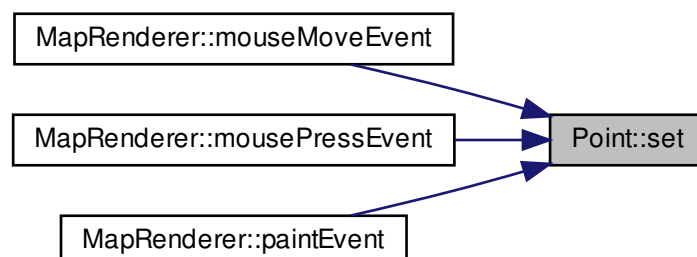


7.17.2.5 set()

```
void Point::set (
    int nx,
    int ny )
```

[Point::set](#) Ändert die Koordinaten.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.17.2.6 toTile()

```
Point Point::toTile ( )
```

[Point::toTile](#) Gibt den Punkt als Tile-Index zurück.

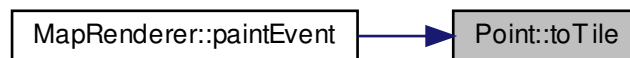
Rückgabe

Ein neuer Punkt.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.17.3 Dokumentation der Datenelemente

7.17.3.1 x

```
int Point::x [private]
```

7.17.3.2 y

```
int Point::y [private]
```

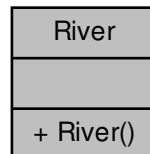
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/point.h](#)
- [src/application_server/point.cpp](#)

7.18 River Klassenreferenz

```
#include <river.h>
```

Zusammengehörigkeiten von River:



Öffentliche Methoden

- [River \(\)](#)

7.18.1 Beschreibung der Konstruktoren und Destruktoren

7.18.1.1 River()

```
River::River ( )
```

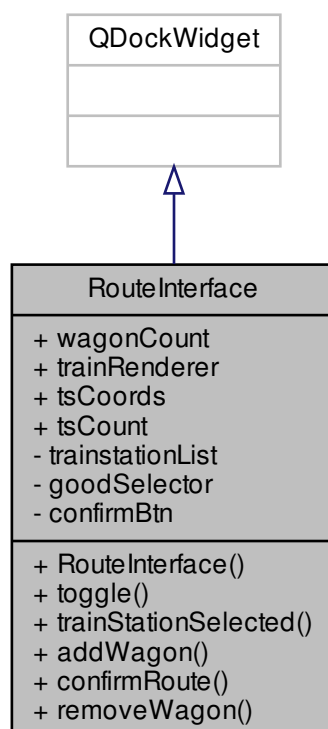
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/river.h](#)
- [src/application_server/river.cpp](#)

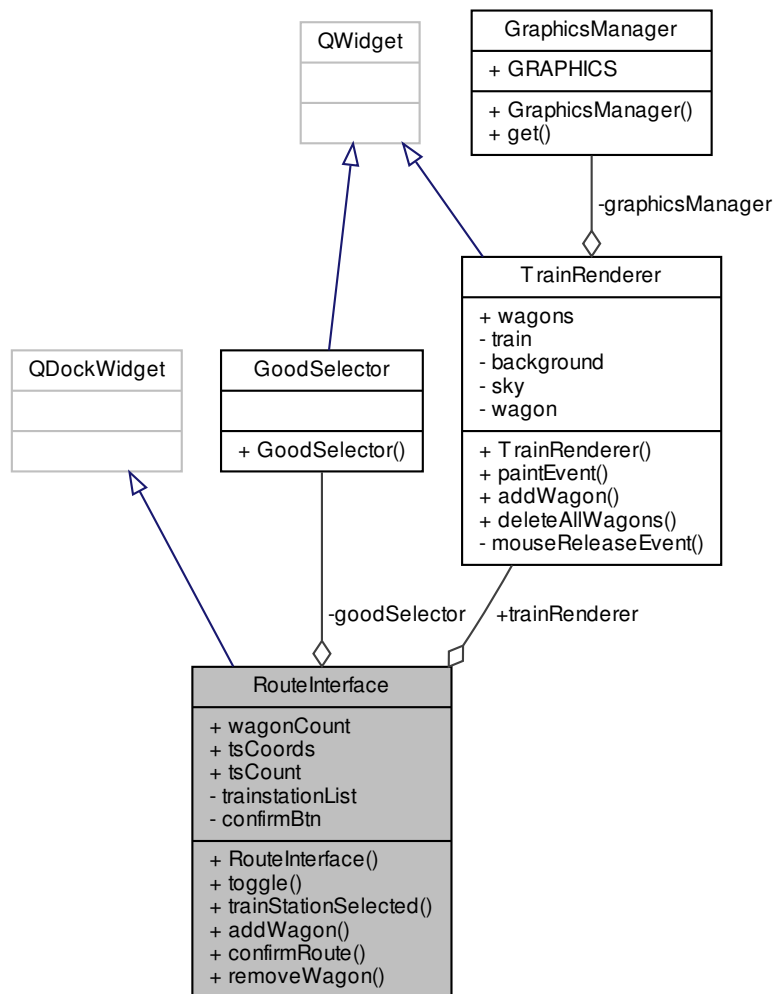
7.19 RouteInterface Klassenreferenz

```
#include <routeinterface.h>
```

Klassendiagramm für RouteInterface:



Zusammengehörigkeiten von RouteInterface:



Öffentliche Slots

- void `toggle` ()
RouteInterface::toggle Wechselt die Sichtbarkeit des Widgets.
- void `trainStationSelected` (int, int)
RouteInterface::trainStationSelected Wird ausgeführt wenn ein Bahnhof angeklickt wurde.
- void `addWagon` (QString *)
RouteInterface::addWagon Slot zum hinzufügen eines Wagons zum aktuellen Renderer.
- void `confirmRoute` ()
- void `removeWagon` ()

Signale

- void `sendConfirmRoute` (QString)

Öffentliche Methoden

- [RouteInterface](#) ([GraphicsManager](#) *)
[RouteInterface::RouteInterface](#) Erzeugt das Routerinterface.

Öffentliche Attribute

- int [wagonCount](#)
- [TrainRenderer](#) * [trainRenderer](#)
- QPoint [tsCoords](#) [9]
- int [tsCount](#)

Private Attribute

- QListWidget * [trainstationList](#)
- [GoodSelector](#) * [goodSelector](#)
- QPushButton * [confirmBtn](#)

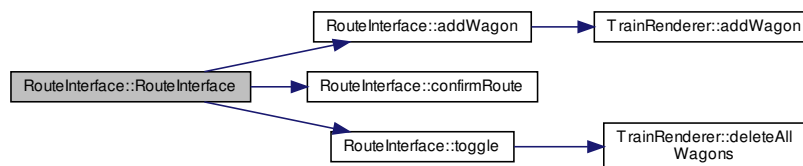
7.19.1 Beschreibung der Konstruktoren und Destruktoren

7.19.1.1 RouteInterface()

```
RouteInterface::RouteInterface (
    GraphicsManager * gm )
```

[RouteInterface::RouteInterface](#) Erzeugt das Routerinterface.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.19.2 Dokumentation der Elementfunktionen

7.19.2.1 addWagon

```
void RouteInterface::addWagon (
    QString * name ) [slot]
```

[RouteInterface::addWagon](#) Slot zum hinzufügen eines Wagons zum aktuellen Renderer.

Parameter

| | |
|-------------|----------------------|
| <i>name</i> | Der Name des Wagens. |
|-------------|----------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.19.2.2 confirmRoute

```
void RouteInterface::confirmRoute ( ) [slot]
```

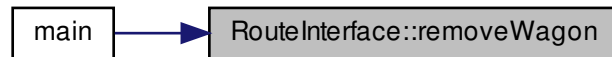
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.19.2.3 removeWagon

```
void RouteInterface::removeWagon ( ) [slot]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.19.2.4 sendConfirmRoute

```
void RouteInterface::sendConfirmRoute (
    QString ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.19.2.5 toggle

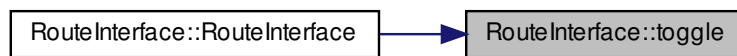
```
void RouteInterface::toggle ( ) [slot]
```

[RouteInterface::toggle](#) Wechselt die Sichtbarkeit des Widgets.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

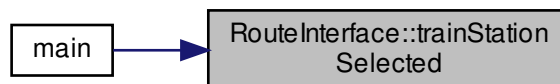


7.19.2.6 trainStationSelected

```
void RouteInterface::trainStationSelected (  
    int px,  
    int py ) [slot]
```

[RouteInterface::trainStationSelected](#) Wird ausgeführt wenn ein Bahnhof angeklickt wurde.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.19.3 Dokumentation der Datenelemente

7.19.3.1 confirmBtn

```
QPushButton* RouteInterface::confirmBtn [private]
```

7.19.3.2 goodSelector

```
GoodSelector* RouteInterface::goodSelector [private]
```

7.19.3.3 trainRenderer

```
TrainRenderer* RouteInterface::trainRenderer
```

7.19.3.4 trainstationList

```
QWidget* RouteInterface::trainstationList [private]
```

7.19.3.5 tsCoords

```
QPoint RouteInterface::tsCoords[9]
```

7.19.3.6 tsCount

```
int RouteInterface::tsCount
```

7.19.3.7 wagonCount

```
int RouteInterface::wagonCount
```

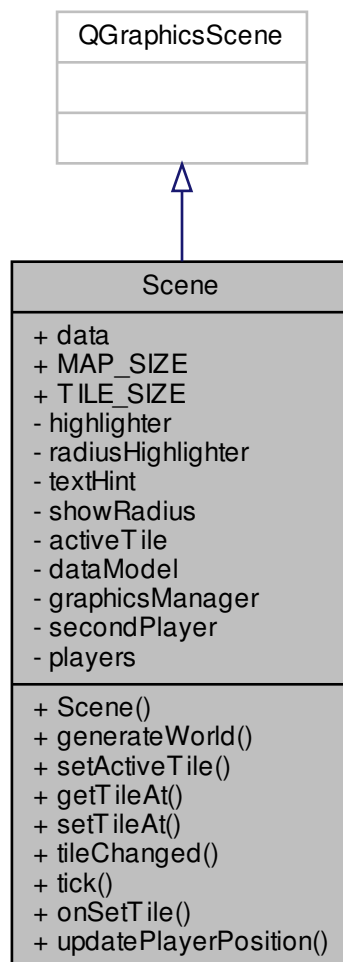
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/routeinterface.h](#)
- [src/application_server/routeinterface.cpp](#)

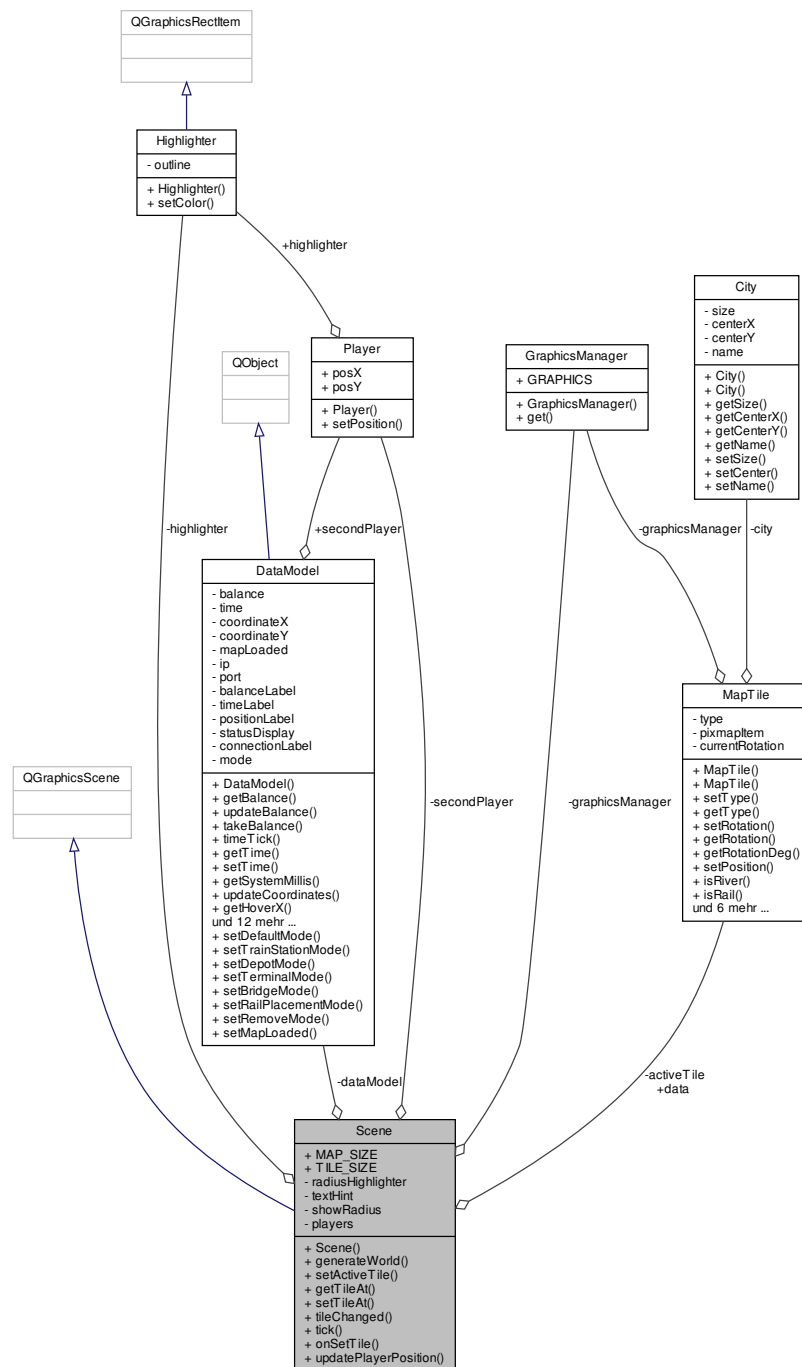
7.20 Scene Klassenreferenz

```
#include <scene.h>
```

Klassendiagramm für Scene:



Zusammengehörigkeiten von Scene:



Öffentliche Slots

- void [onSetTile](#) (int, int, int, int)

Scene::onSetTile Ändert ein [MapTile](#) ohne ein Signal an den Server. Notwendig zum Empfangen von Änderungen.

- void [updatePlayerPosition](#) (int, int)

Scene::updatePlayerPosition Slot zum Updaten eines Spielers.

Signale

- void `tileUpdate` (int, int, int, int)

Öffentliche Methoden

- `Scene` (`GraphicsManager` *pGraphicsManager, `DataModel` *pDataModel)
`Scene::Scene` Konstruktor.
- void `generateWorld` ()
`Scene::generateWorld` Diese Methode generiert eine neue Welt.
- void `setActiveTile` (QGraphicsItem *pItem)
`Scene::setActiveTile` Setzt den `MapTile` über dem die Maus gerade ist. Wird von view aufgerufen.
- `MapTile` * `getTileAt` (int posX, int posY, bool isPixelCoordinate=false)
`Scene::getTileAt` Liefert ein `MapTile` anhand der Pixel-Koordinaten oder der Indizes.
- void `setTileAt` (int, int, int, int)
`Scene::setTileAt` Setzt ein `MapTile` anhand der Pixel-Koordinaten oder der Indizes.
- void `tileChanged` (int, int)
`Scene::tileChanged` Meldet das sich ein `MapTile` geändert hat.
- void `tick` ()
`Scene::tick` Asynchrone Tickfunktion. Wird alle 20ms aufgerufen.

Öffentliche Attribute

- `MapTile` data [`Scene::MAP_SIZE`][`Scene::MAP_SIZE`]

Statische öffentliche Attribute

- const static int `MAP_SIZE` {300}
- const static int `TILE_SIZE` {64}

Private Attribute

- `Highlighter` * `highlighter`
- QGraphicsEllipseItem * `radiusHighlighter`
- QGraphicsTextItem * `textHint`
- bool `showRadius`
- `MapTile` * `activeTile`
- `DataModel` * `dataModel`
- `GraphicsManager` * `graphicsManager`
- `Player` * `secondPlayer`
- std::vector< `Player` > `players`

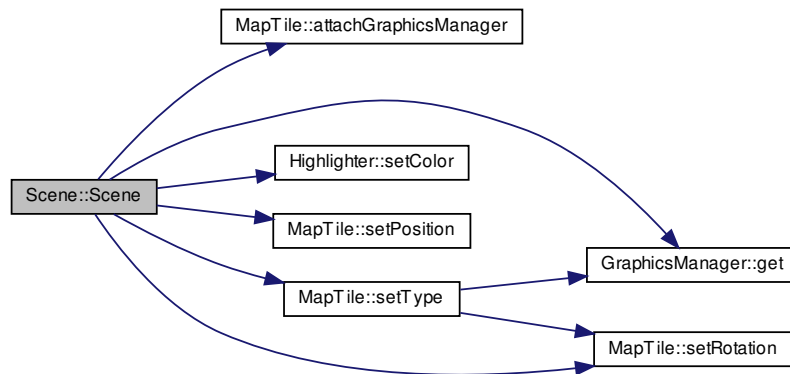
7.20.1 Beschreibung der Konstruktoren und Destruktoren

7.20.1.1 Scene()

```
Scene::Scene (
    GraphicsManager * pGraphicsManager,
    DataModel * pDataModel )
```

[Scene::Scene](#) Konstruktor.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



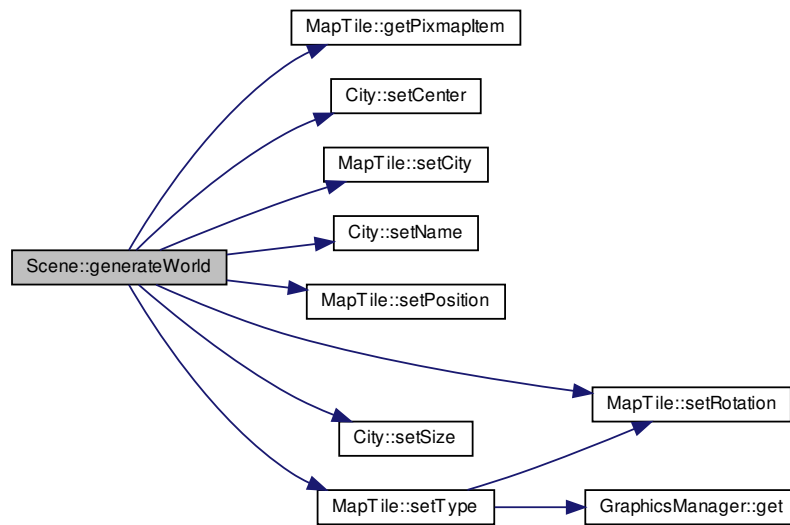
7.20.2 Dokumentation der Elementfunktionen

7.20.2.1 generateWorld()

```
void Scene::generateWorld ( )
```

[Scene::generateWorld](#) Diese Methode generiert eine neue Welt.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.20.2.2 `getTileAt()`

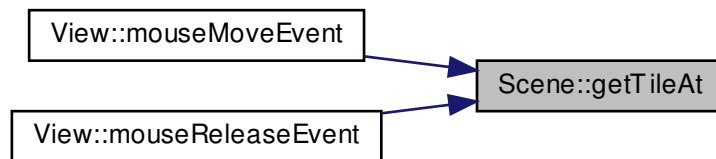
```
MapTile * Scene::getTileAt (
    int posX,
    int posY,
    bool isPixelCoordinate = false )
```

`Scene::getTileAt` liefert ein `MapTile` anhand der Pixel-Koordinaten oder der Indizes.

Parameter

| | |
|-------------------|------------------|
| <code>posX</code> | Die X-Koordinate |
| <code>posY</code> | Die Y-Koordinate |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.20.2.3 onSetTile

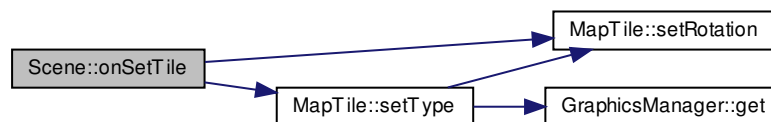
```
void Scene::onSetTile (
    int pX,
    int pY,
    int pType,
    int pRotation ) [slot]
```

[Scene::onSetTile](#) Ändert ein [MapTile](#) ohne ein Signal an den Server. Notwendig zum Empfangen von Änderungen.

Parameter

| | |
|------------------|-------------------|
| <i>pX</i> | Die X-Koordinate. |
| <i>pY</i> | Die Y-Koordinate. |
| <i>pType</i> | Der Typ. |
| <i>pRotation</i> | Die Rotation. |

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.20.2.4 setActiveTile()

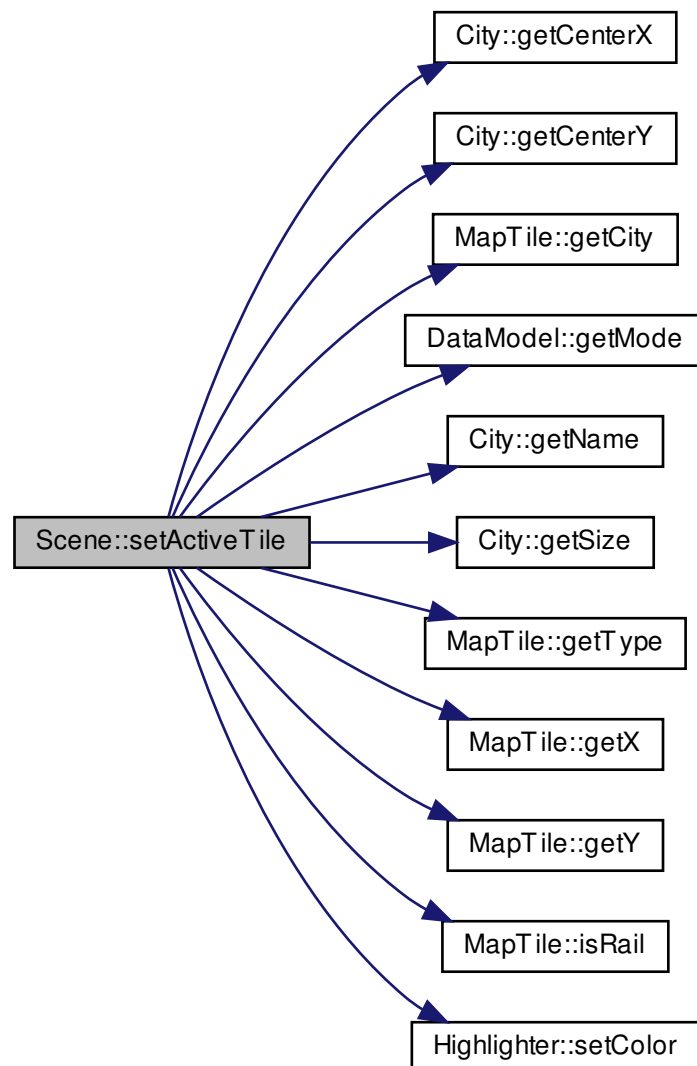
```
void Scene::setActiveTile (
    QGraphicsItem * pItem )
```

`Scene::setActiveTile` Setzt den `MapTile` über dem die Maus gerade ist. Wird von view aufgerufen.

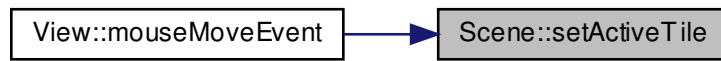
Parameter

| | |
|--------------------|---|
| <code>pltem</code> | Ein Grafikitem zu dem die Methode den zugehörigen Maptile bestimmt. |
|--------------------|---|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.20.2.5 setTileAt()

```

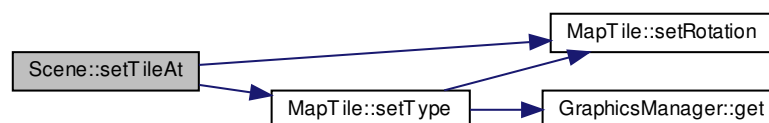
void Scene::setTileAt (
    int pX,
    int pY,
    int pType,
    int pRotation )
  
```

[Scene::setTileAt](#) Setzt ein [MapTile](#) anhand der Pixel-Koordinaten oder der Indizes.

Parameter

| | |
|--------------------------|--|
| <i>posX</i> | |
| <i>posY</i> | |
| <i>isPixelCoordinate</i> | |

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.20.2.6 tick()

```

void Scene::tick ( )
  
```

[Scene::tick](#) Asynchrone Tickfunktion. Wird alle 20ms aufgerufen.

7.20.2.7 tileChanged()

```
void Scene::tileChanged (
    int pX,
    int pY )
```

[Scene::tileChanged](#) Meldet das sich ein [MapTile](#) geändert hat.

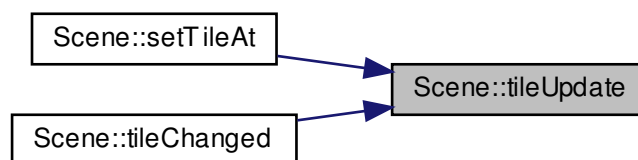
Parameter

| | |
|-----------|-------------------|
| <i>pX</i> | Die X-Koordinate. |
| <i>pY</i> | Die Y-Koordinate. |

7.20.2.8 tileUpdate

```
void Scene::tileUpdate (
    int ,
    int ,
    int ,
    int ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

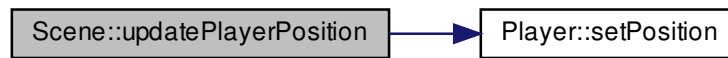


7.20.2.9 updatePlayerPosition

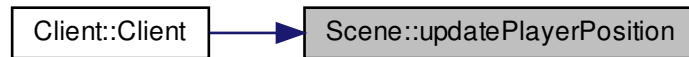
```
void Scene::updatePlayerPosition (
    int pX,
    int pY ) [slot]
```

[Scene::updatePlayerPosition](#) Slot zum Updaten eines Spielers.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.20.3 Dokumentation der Datenelemente

7.20.3.1 activeTile

```
MapTile* Scene::activeTile [private]
```

7.20.3.2 data

```
MapTile Scene::data[Scene::MAP_SIZE][Scene::MAP_SIZE]
```

7.20.3.3 dataModel

```
DataModel* Scene::dataModel [private]
```

7.20.3.4 graphicsManager

```
GraphicsManager* Scene::graphicsManager [private]
```

7.20.3.5 highlighter

```
Highlighter* Scene::highlighter [private]
```

7.20.3.6 MAP_SIZE

```
const static int Scene::MAP_SIZE {300} [static]
```

7.20.3.7 players

```
std::vector<Player> Scene::players [private]
```

7.20.3.8 radiusHighlighter

```
QGraphicsEllipseItem* Scene::radiusHighlighter [private]
```

7.20.3.9 secondPlayer

```
Player* Scene::secondPlayer [private]
```

7.20.3.10 showRadius

```
bool Scene::showRadius [private]
```

7.20.3.11 textHint

```
QGraphicsTextItem* Scene::textHint [private]
```

7.20.3.12 TILE_SIZE

```
const static int Scene::TILE_SIZE {64} [static]
```

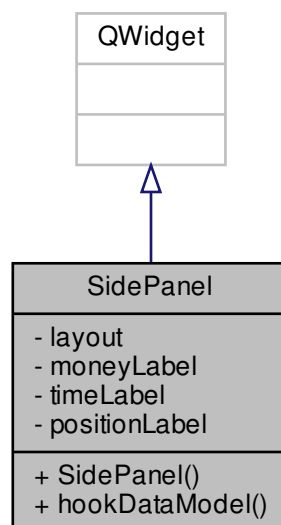
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/scene.h](#)
- [src/application_server/scene.cpp](#)

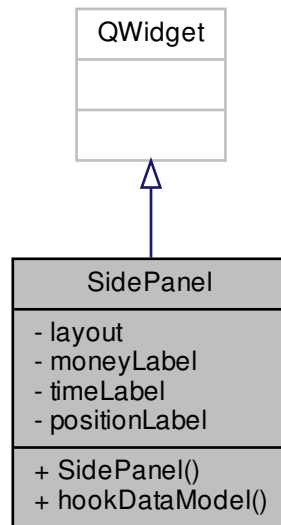
7.21 SidePanel Klassenreferenz

```
#include <sidepanel.h>
```

Klassendiagramm für SidePanel:



Zusammengehörigkeiten von SidePanel:



Öffentliche Methoden

- [SidePanel \(\)](#)
SidePanel::SidePanel Erzeugt ein neues Side-Panel (Menü)
- void [hookDataModel \(DataModel *pModel\)](#)
SidePanel::hookDataModel Verknüpft ein Datenmodell mit der Anzeige. Dadurch können dann Textfelder etc. aktualisiert werden.

Private Attribute

- QGridLayout * [layout](#)
- QLabel * [moneyLabel](#)
- QLabel * [timeLabel](#)
- QLabel * [positionLabel](#)

7.21.1 Beschreibung der Konstruktoren und Destruktoren

7.21.1.1 SidePanel()

```
SidePanel::SidePanel ( )
```

[SidePanel::SidePanel](#) Erzeugt ein neues Side-Panel (Menü)

Parameter

| | |
|----------------|--------------------|
| <i>pParent</i> | Das Parent-Element |
|----------------|--------------------|

7.21.2 Dokumentation der Elementfunktionen

7.21.2.1 hookDataModel()

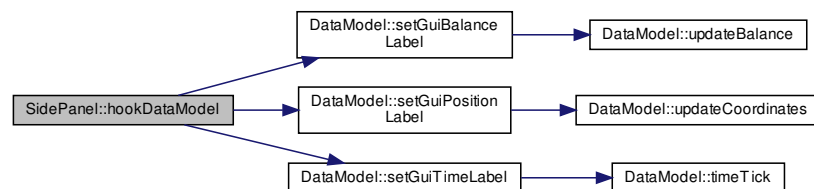
```
void SidePanel::hookDataModel (
    DataModel * pModel )
```

[SidePanel::hookDataModel](#) Verknüpft ein Datenmodell mit der Anzeige. Dadurch können dann Textfelder etc. aktualisiert werden.

Parameter

| | |
|---------------|------------------|
| <i>pModel</i> | Ein Datenmodell. |
|---------------|------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.21.3 Dokumentation der Datenelemente

7.21.3.1 layout

```
QGridLayout* SidePanel::layout [private]
```

7.21.3.2 moneyLabel

```
QLabel* SidePanel::moneyLabel [private]
```

7.21.3.3 positionLabel

```
QLabel* SidePanel::positionLabel [private]
```

7.21.3.4 timeLabel

```
QLabel* SidePanel::timeLabel [private]
```

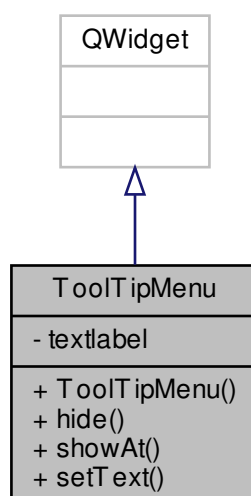
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/sidepanel.h](#)
- [src/application_server/sidepanel.cpp](#)

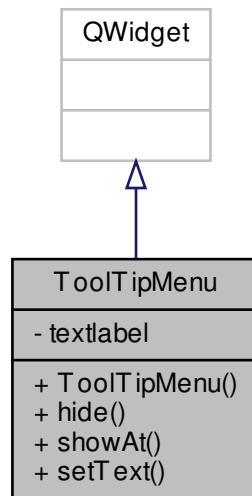
7.22 ToolTipMenu Klassenreferenz

```
#include <tooltipmenu.h>
```

Klassendiagramm für ToolTipMenu:



Zusammengehörigkeiten von ToolTipMenu:



Öffentliche Methoden

- [ToolTipMenu \(\)](#)
[ToolTipMenu::ToolTipMenu](#) Erzeugt ein Tool-Tip Menü das absolut positioniert werden kann.
- void [hide \(\)](#)
[ToolTipMenu::hide](#) Blendet das Menü aus.
- void [showAt](#) (int x, int y)
[ToolTipMenu::showAt](#) Blendet das Menü an einer bestimmten Stelle (relativ zum Parent) ein.
- void [setText](#) (QString pText)
[ToolTipMenu::setText](#) Setzt den Text. (HTML-Fähig)

Private Attribute

- QLabel * [textlabel](#)

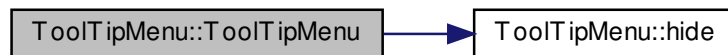
7.22.1 Beschreibung der Konstruktoren und Destruktoren

7.22.1.1 ToolTipMenu()

```
ToolTipMenu::ToolTipMenu ( )
```

[ToolTipMenu::ToolTipMenu](#) Erzeugt ein Tool-Tip Menü das absolut positioniert werden kann.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



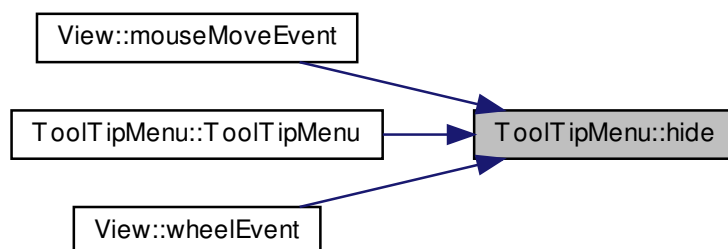
7.22.2 Dokumentation der Elementfunktionen

7.22.2.1 hide()

```
void ToolTipMenu::hide ( )
```

[ToolTipMenu::hide](#) Blendet das Menü aus.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.22.2.2 setText()

```
void ToolTipMenu::setText (
    QString pText )
```

[ToolTipMenu::setText](#) Setzt den Text. (HTML-Fähig)

Parameter

| | |
|--------------|-----------------------|
| <i>pText</i> | Der Text als QString. |
|--------------|-----------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.22.2.3 showAt()

```
void ToolTipMenu::showAt (
    int x,
    int y )
```

[ToolTipMenu::showAt](#) Blendet das Menü an einer bestimmten Stelle (relativ zum Parent) ein.

Parameter

| | |
|----------|-------------------|
| <i>x</i> | Die X-Koordinate. |
| <i>y</i> | Die Y-Koordinate. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.22.3 Dokumentation der Datenelemente

7.22.3.1 textlabel

```
QLabel* ToolTipMenu::textlabel [private]
```

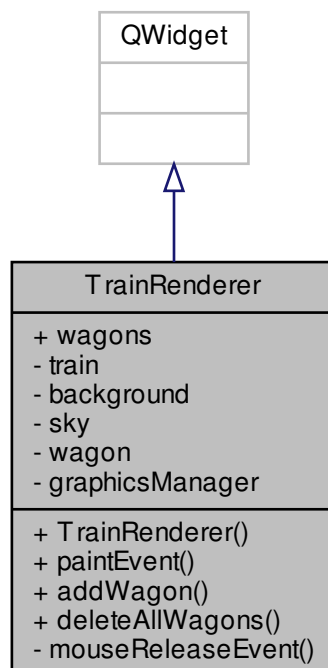
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/tooltipmenu.h](#)
- [src/application_server/tooltipmenu.cpp](#)

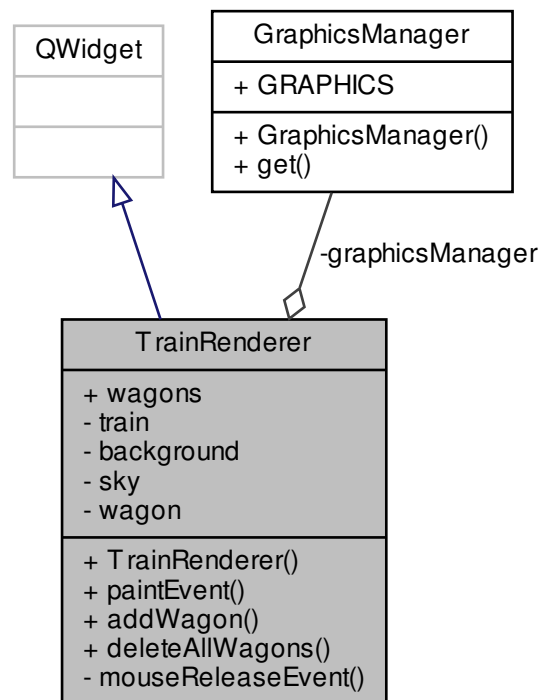
7.23 TrainRenderer Klassenreferenz

```
#include <trainrenderer.h>
```

Klassendiagramm für TrainRenderer:



Zusammengehörigkeiten von TrainRenderer:



Signale

- void [triggerRemoveWagon](#) ()

Öffentliche Methoden

- [TrainRenderer](#) ([GraphicsManager](#) *)
- void [paintEvent](#) (QPaintEvent *event) override
- void [addWagon](#) (std::string name)
- void [deleteAllWagons](#) ()

Öffentliche Attribute

- std::map< std::string, int > [wagons](#)

Private Slots

- void [mouseReleaseEvent](#) (QMouseEvent *event) override

Private Attribute

- QImage [train](#)
- QImage [background](#)
- QImage [sky](#)
- QImage [wagon](#)
- [GraphicsManager](#) * [graphicsManager](#)

7.23.1 Beschreibung der Konstruktoren und Destruktoren

7.23.1.1 TrainRenderer()

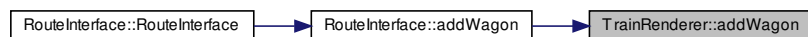
```
TrainRenderer::TrainRenderer (
    GraphicsManager * pGm )
```

7.23.2 Dokumentation der Elementfunktionen

7.23.2.1 addWagon()

```
void TrainRenderer::addWagon (
    std::string name )
```

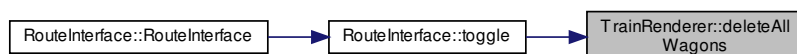
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.23.2.2 deleteAllWagons()

```
void TrainRenderer::deleteAllWagons ( )
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.23.2.3 mouseReleaseEvent

```
void TrainRenderer::mouseReleaseEvent (
    QMouseEvent * event ) [override], [private], [slot]
```

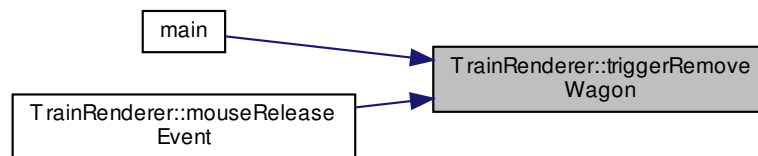
7.23.2.4 paintEvent()

```
void TrainRenderer::paintEvent (
    QPaintEvent * event ) [override]
```

7.23.2.5 triggerRemoveWagon

```
void TrainRenderer::triggerRemoveWagon ( ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.23.3 Dokumentation der Datenelemente

7.23.3.1 background

```
QImage TrainRenderer::background [private]
```

7.23.3.2 graphicsManager

```
GraphicsManager* TrainRenderer::graphicsManager [private]
```

7.23.3.3 sky

```
QImage TrainRenderer::sky [private]
```

7.23.3.4 train

```
QImage TrainRenderer::train [private]
```

7.23.3.5 wagon

```
QImage TrainRenderer::wagon [private]
```

7.23.3.6 wagons

```
std::map<std::string, int> TrainRenderer::wagons
```

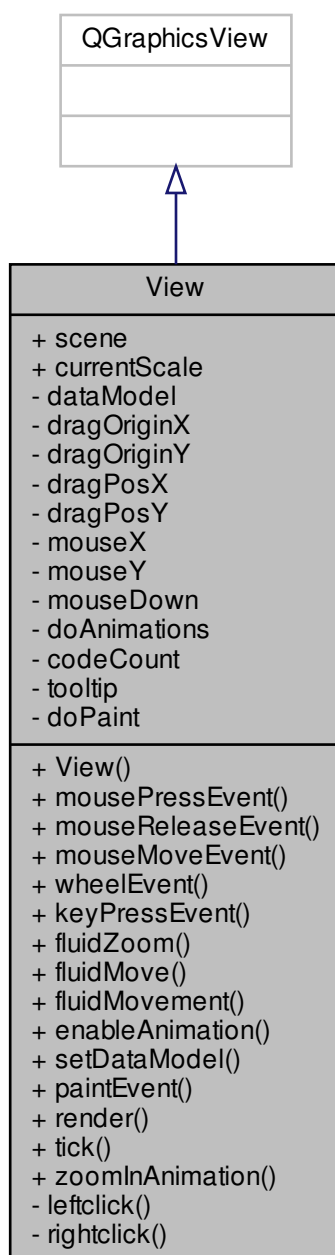
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/trainrenderer.h](#)
- [src/application_server/trainrenderer.cpp](#)

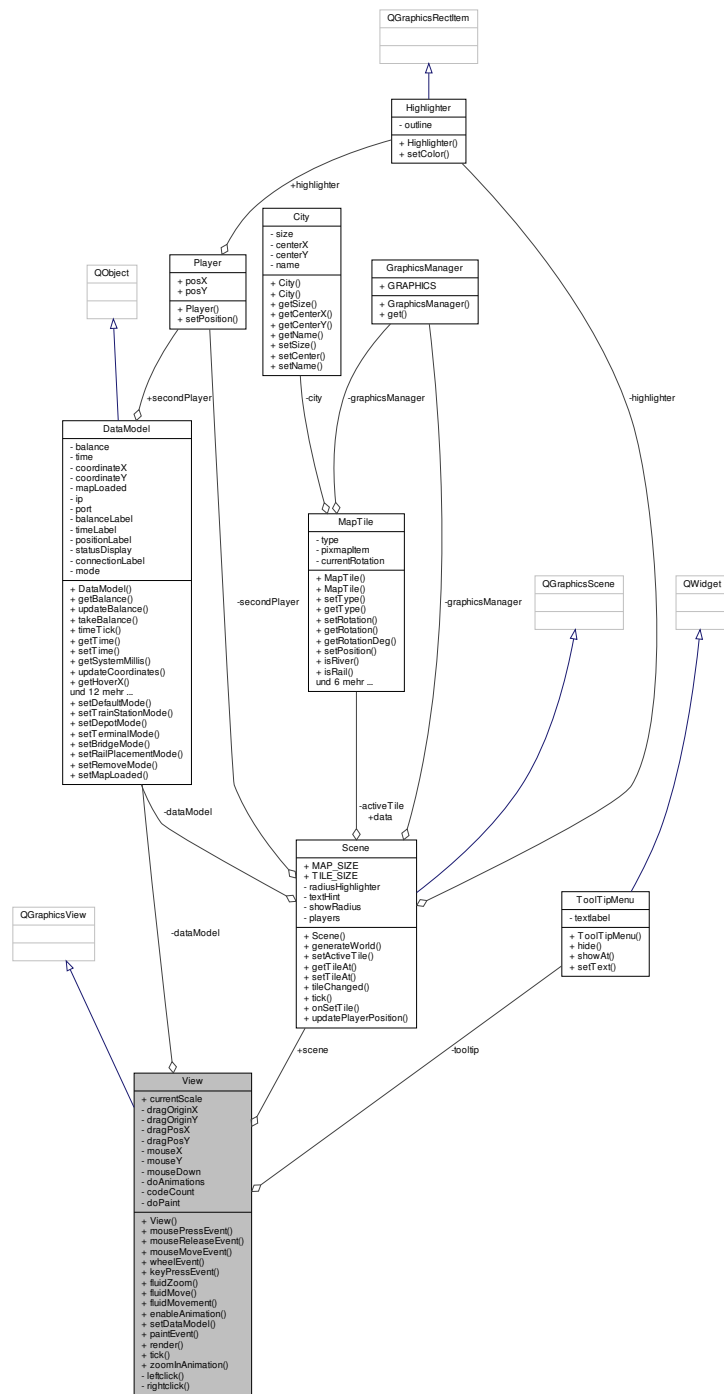
7.24 View Klassenreferenz

```
#include <view.h>
```

Klassendiagramm für View:



Zusammengehörigkeiten von View:



Öffentliche Slots

- void zoomInAnimation ()

View::zoomInAnimation Slot der nach dem Laden der Karte aufgerufen wird.

Signale

- void [onLeftclick](#) ()
- void [onRightclick](#) ()
- void [onTrainStationClick](#) (int, int)

Öffentliche Methoden

- [View](#) ([Scene](#) *pScene, [ToolTipMenu](#) *pToolTip)
[View::View](#) Konstruktor. Versteckt u.a. die Scrollbars und aktiviert Mousetracking.
- void [mousePressEvent](#) (QMouseEvent *event) override
[View::mousePressEvent](#) QT Methode. Wird aufgerufen wenn die Maus gedrückt wurde.
- void [mouseReleaseEvent](#) (QMouseEvent *event) override
[View::mouseReleaseEvent](#) QT Methode. Wird aufgerufen wenn die Maus losgelassen wird.
- void [mouseMoveEvent](#) (QMouseEvent *event) override
[View::mouseMoveEvent](#) QT Methode. Wird aufgerufen wenn die Maus bewegt wird.
- void [wheelEvent](#) (QWheelEvent *event) override
[View::wheelEvent](#) QT Methode. Wird aufgerufen wenn das Mousrad gedreht wird.
- void [keyPressEvent](#) (QKeyEvent *event) override
[View::keyPressEvent](#) QT Methode. Wird Aufgerufen wenn eine Taste gedrückt wird.
- void [fluidZoom](#) (double target, bool in)
[View::fluidZoom](#) Startet eine Zoom-Animation. Zuvor muss doAnimations=true gesetzt sein. Bsp: fluidZoom(3, true) zoomt 3x in die Karte hinein.
- void [fluidMove](#) (int vX, int vY)
[View::fluidMove](#) Verschiebt die Karte animiert und relativ zur aktuellen Position.
- void [fluidMovement](#) (int pX, int pY)
[View::fluidMovement](#) Verschiebt die Karte animiert an zu einer absoluten Koordinate.
- void [enableAnimation](#) ()
[View::enableAnimation](#) Aktiviert animationen bis zum nächsten Event.
- void [setDataModel](#) ([DataModel](#) *pModel)
[View::setDataModel](#) Setzt das Datenmodell. An dieses wird dann kontinuierlich die aktuelle Position weitergegeben.
- void [paintEvent](#) (QPaintEvent *event) override
[View::paintEvent](#) Überschreibt das PaintEvent des Views für eigene Zeichenanweisungen.
- void [render](#) ()
- void [tick](#) ()
[View::tick](#) Asynchroner Tick. Wird alle 20MS von [GameLoop](#) aufgerufen.

Öffentliche Attribute

- [Scene](#) * scene
- double [currentScale](#) {1.0}

Private Methoden

- void [leftclick](#) (QMouseEvent *, [MapTile](#) *)
[View::leftclick](#) Führt einen Linksklick aus.
- void [rightclick](#) (QMouseEvent *, [MapTile](#) *)
[View::leftclick](#) Führt einen Rechtsklick aus.

Private Attribute

- [DataModel](#) * [dataModel](#)
- int [dragOriginX](#)
- int [dragOriginY](#)
- int [dragPosX](#)
- int [dragPosY](#)
- int [mouseX](#)
- int [mouseY](#)
- bool [mouseDown](#)
- bool [doAnimations](#)
- int [codeCount](#)
- [ToolTipMenu](#) * [tooltip](#)
- bool [doPaint](#) {false}

7.24.1 Beschreibung der Konstruktoren und Destruktoren

7.24.1.1 View()

```
View::View (
    Scene * pScene,
    ToolTipMenu * pToolTip )
```

[View::View](#) Konstruktor. Versteckt u.a. die Scrollbars und aktiviert Mousetracking.

Parameter

| | |
|------------------------|------------------------------|
| pScene | Das Zugehörige Szenenobjekt. |
|------------------------|------------------------------|

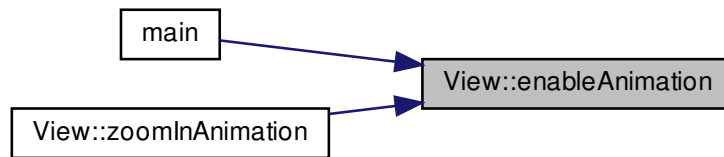
7.24.2 Dokumentation der Elementfunktionen

7.24.2.1 enableAnimation()

```
void View::enableAnimation ( )
```

[View::enableAnimation](#) Aktiviert animationen bis zum nächsten Event.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.2 fluidMove()

```
void View::fluidMove (
    int vX,
    int vY )
```

[View::fluidMove](#) Verschiebt die Karte animiert und relativ zur aktuellen Position.

Parameter

| | |
|----|-----------------------------|
| vX | Verschiebung in X-Richtung. |
| vY | Verschiebung in Y-Richtung. |

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.3 fluidMovement()

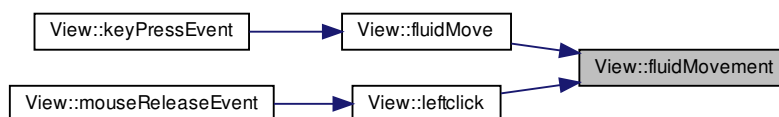
```
void View::fluidMovement (
    int pX,
    int pY )
```

[View::fluidMovement](#) Verschiebt die Karte animiert an zu einer absoluten Koordinate.

Parameter

| | |
|-----------|-------------------|
| <i>pX</i> | Die X-Koordinate. |
| <i>pY</i> | Die Y-Koordinate. |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.4 fluidZoom()

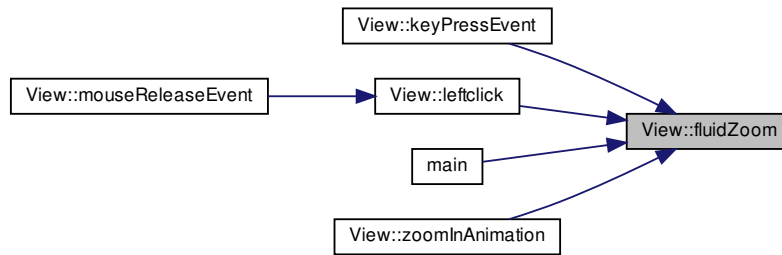
```
void View::fluidZoom (
    double target,
    bool in )
```

[View::fluidZoom](#) Startet eine Zoom-Animation. Zuvor muss `doAnimations=true` gesetzt sein. Bsp: `fluidZoom(3, true)` zoomt 3x in die Karte hinein.

Parameter

| | |
|---------------|--|
| <i>target</i> | Die angestrebte Skalierung. |
| <i>in</i> | Ob vergrößert oder verkleinert werden soll. (true = reinzoomen, false=rauszoomen). |

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.5 keyPressEvent()

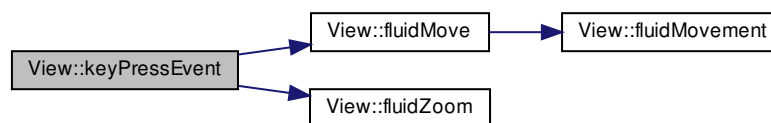
```
void View::keyPressEvent (
    QKeyEvent * event ) [override]
```

[View::keyPressEvent](#) QT Methode. Wird Aufgerufen wenn eine Taste gedrückt wird.

Parameter

| | |
|--------------|--|
| <i>event</i> | Event mit Informationen. Wichtig: <code>event->text()</code> : Text der Taste und <code>event->key()</code> : Id der Taste |
|--------------|--|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

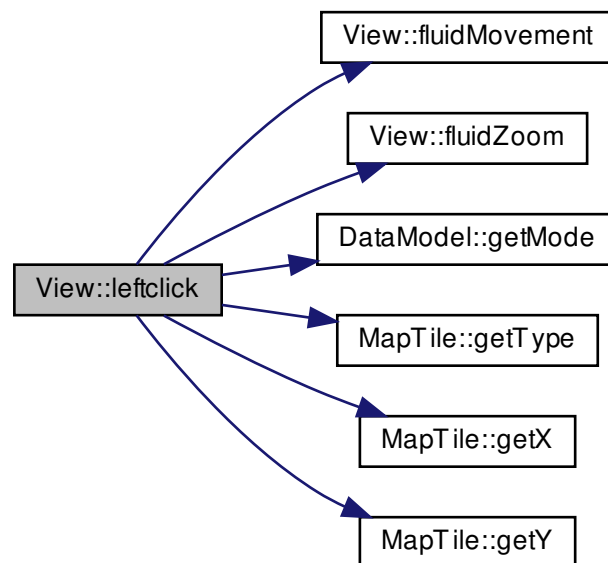


7.24.2.6 leftclick()

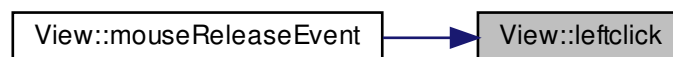
```
void View::leftclick (
    QMouseEvent * pEvent,
    MapTile * pTile ) [private]
```

[View::leftclick](#) Führt einen Linksklick aus.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.7 mouseMoveEvent()

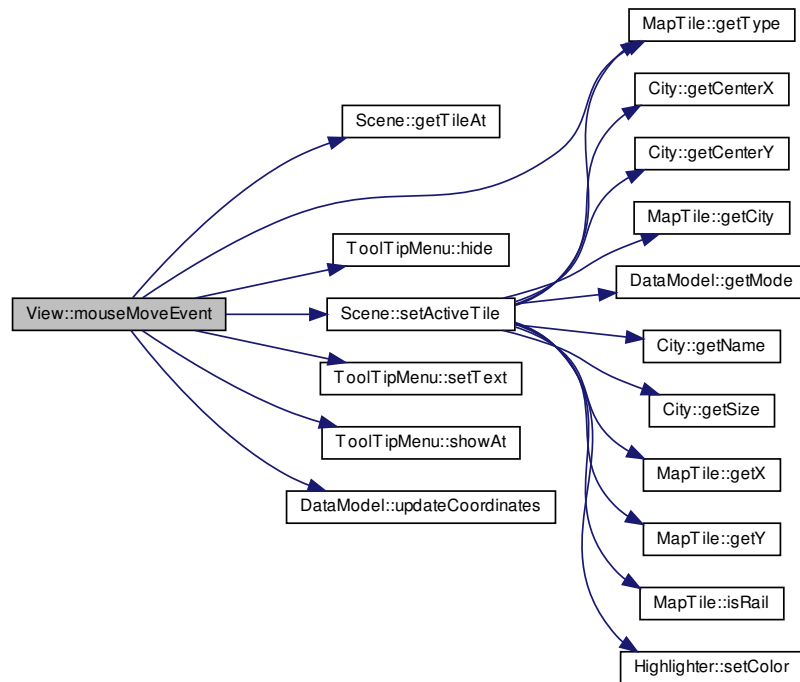
```
void View::mouseMoveEvent (
    QMouseEvent * event ) [override]
```

[View::mouseMoveEvent](#) QT Methode. Wird aufgerufen wenn die Maus bewegt wird.

Parameter

| | |
|--------------------|--------------------------------------|
| <code>event</code> | Informationen über Position der Maus |
|--------------------|--------------------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.24.2.8 mousePressEvent()

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

[View::mousePressEvent](#) QT Methode. Wird aufgerufen wenn die Maus gedrückt wurde.

Parameter

| | |
|--------------|--|
| <i>event</i> | Enthält Informationen über die Taste und Position. |
|--------------|--|

7.24.2.9 mouseReleaseEvent()

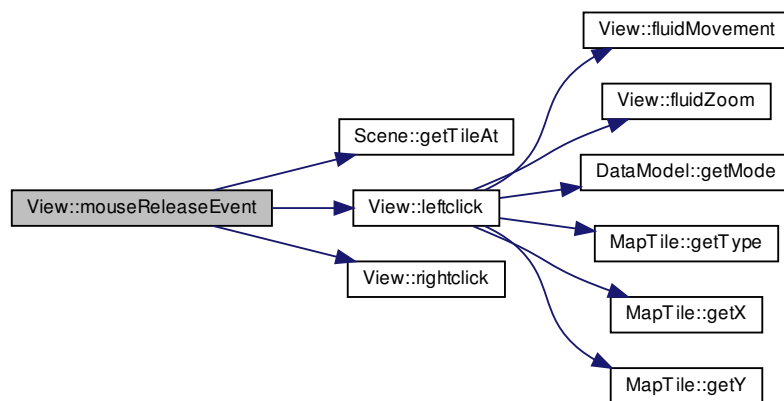
```
void View::mouseReleaseEvent (
    QMouseEvent * event ) [override]
```

[View::mouseReleaseEvent](#) QT Methode. Wird aufgerufen wenn die Maus losgelassen wird.

Parameter

| | |
|--------------------|---------------------------------------|
| <code>event</code> | Informationen über Position und Taste |
|--------------------|---------------------------------------|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.24.2.10 onLeftclick

```
void View::onLeftclick ( ) [signal]
```

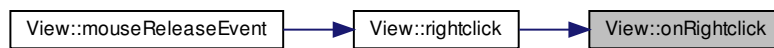
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.11 onRightclick

```
void View::onRightclick ( ) [signal]
```

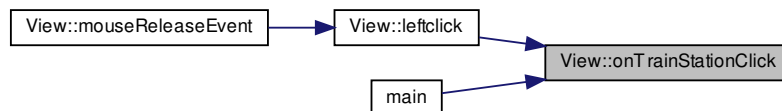
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.12 onTrainStationClick

```
void View::onTrainStationClick (
    int ,
    int ) [signal]
```

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.13 paintEvent()

```
void View::paintEvent (
    QPaintEvent * event ) [override]
```

[View::paintEvent](#) überschreibt das `PaintEvent` des Views für eigene Zeichenanweisungen.

Parameter

| | |
|--------------|---|
| <i>event</i> | Das zugehörige <code>QPaintEvent</code> . |
|--------------|---|

7.24.2.14 render()

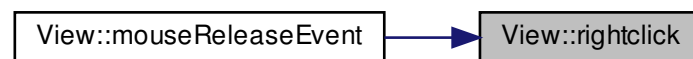
```
void View::render ( )
```

7.24.2.15 rightclick()

```
void View::rightclick (
    QMouseEvent * pEvent,
    MapTile * pTile ) [private]
```

[View::leftclick](#) Führt einen Rechtsklick aus.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.16 setDataModel()

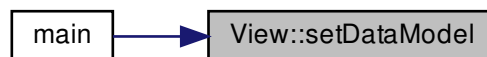
```
void View::setDataModel (
    DataModel * pModel )
```

[View::setDataModel](#) Setzt das Datenmodell. An dieses wird dann kontinuierlich die aktuelle Position weitergegeben.

Parameter

| | |
|---------------------|------------------|
| <code>pModel</code> | Ein Datenmodell. |
|---------------------|------------------|

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.24.2.17 tick()

```
void View::tick ( )
```

[View::tick](#) Asynchroner Tick. Wird alle 20MS von [GameLoop](#) aufgerufen.

7.24.2.18 wheelEvent()

```
void View::wheelEvent (
    QWheelEvent * event ) [override]
```

[View::wheelEvent](#) QT Methode. Wird aufgerufen wenn das Mausevent gedreht wird.

Parameter

| | |
|--------------|--|
| <i>event</i> | Eventobjekt mit Infos. Wichtig: <code>event->delta()</code> : Positiv oder negativ je nachdem in welche Richtung gedreht wurde. |
|--------------|--|

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

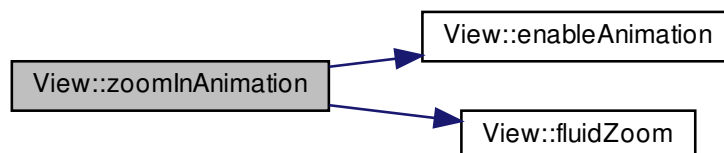


7.24.2.19 zoomInAnimation

```
void View::zoomInAnimation ( ) [slot]
```

[View::zoomInAnimation](#) Slot der nach dem Laden der Karte aufgerufen wird.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.24.3 Dokumentation der Datenelemente

7.24.3.1 codeCount

```
int View::codeCount [private]
```

7.24.3.2 currentScale

```
double View::currentScale {1.0}
```

7.24.3.3 dataModel

```
DataModel* View::dataModel [private]
```

7.24.3.4 doAnimations

```
bool View::doAnimations [private]
```

7.24.3.5 doPaint

```
bool View::doPaint {false} [private]
```

7.24.3.6 dragOriginX

```
int View::dragOriginX [private]
```

7.24.3.7 dragOriginY

```
int View::dragOriginY [private]
```

7.24.3.8 dragPosX

```
int View::dragPosX [private]
```

7.24.3.9 dragPosY

```
int View::dragPosY [private]
```

7.24.3.10 mouseDown

```
bool View::mouseDown [private]
```

7.24.3.11 mouseX

```
int View::mouseX [private]
```

7.24.3.12 mouseY

```
int View::mouseY [private]
```

7.24.3.13 scene

```
Scene* View::scene
```

7.24.3.14 tooltip

```
ToolTipMenu* View::tooltip [private]
```

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- [src/application_server/view.h](#)
- [src/application_server/view.cpp](#)

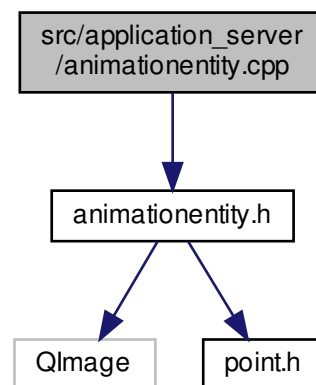
Kapitel 8

Datei-Dokumentation

8.1 src/application_server/animationentity.cpp-Dateireferenz

```
#include "animationentity.h"
```

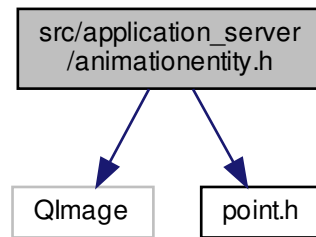
Include-Abhängigkeitsdiagramm für animationentity.cpp:



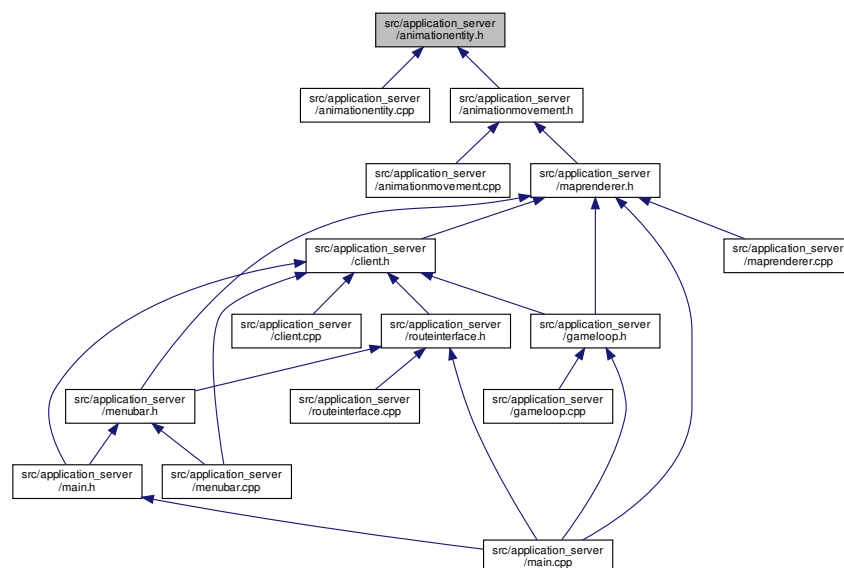
8.2 src/application_server/animationentity.h-Dateireferenz

```
#include <QImage>  
#include "point.h"
```

Include-Abhängigkeitsdiagramm für animationentity.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [AnimationEntity](#)

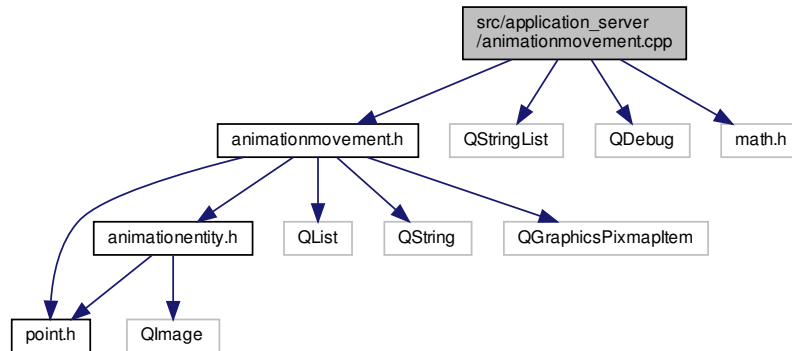
8.3 src/application_server/animationmovement.cpp-Dateireferenz

```

#include "animationmovement.h"
#include <QStringList>
#include <QDebug>
  
```

```
#include <math.h>
```

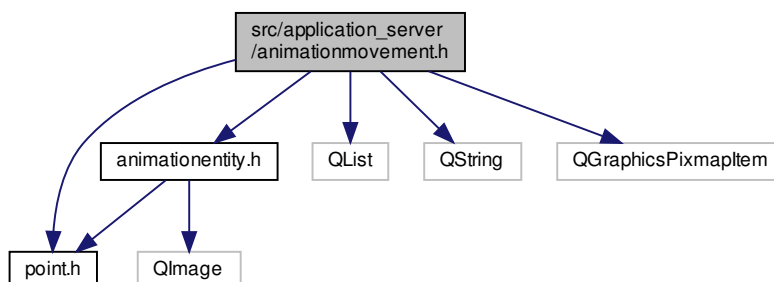
Include-Abhängigkeitsdiagramm für animationmovement.cpp:



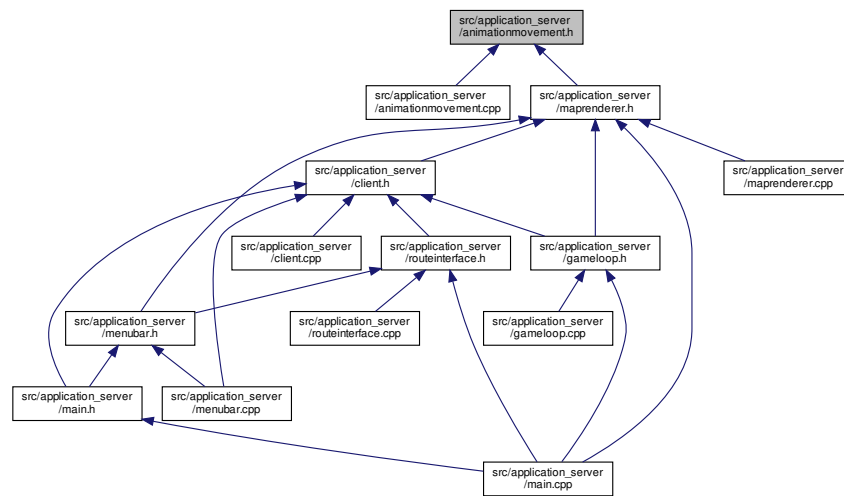
8.4 src/application_server/animationmovement.h-Dateireferenz

```
#include "point.h"
#include "animationentity.h"
#include <QList>
#include <QString>
#include <QGraphicsPixmapItem>
```

Include-Abhängigkeitsdiagramm für animationmovement.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



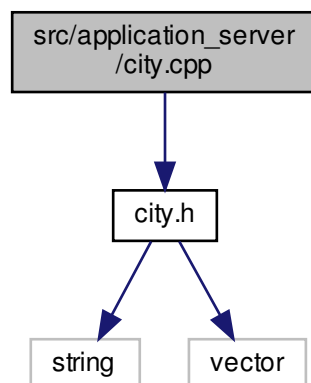
Klassen

- class [AnimationMovement](#)

8.5 src/application_server/city.cpp-Dateireferenz

```
#include "city.h"
```

Include-Abhängigkeitsdiagramm für city.cpp:

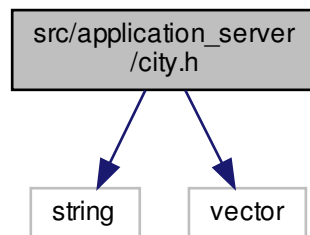


8.6 src/application_server/city.h-Dateireferenz

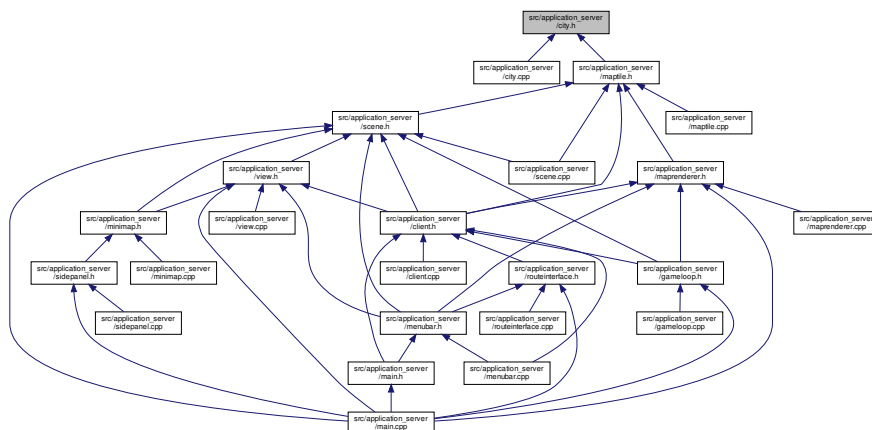
```
#include <string>
```

```
#include <vector>
```

Include-Abhängigkeitsdiagramm für city.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class `City`

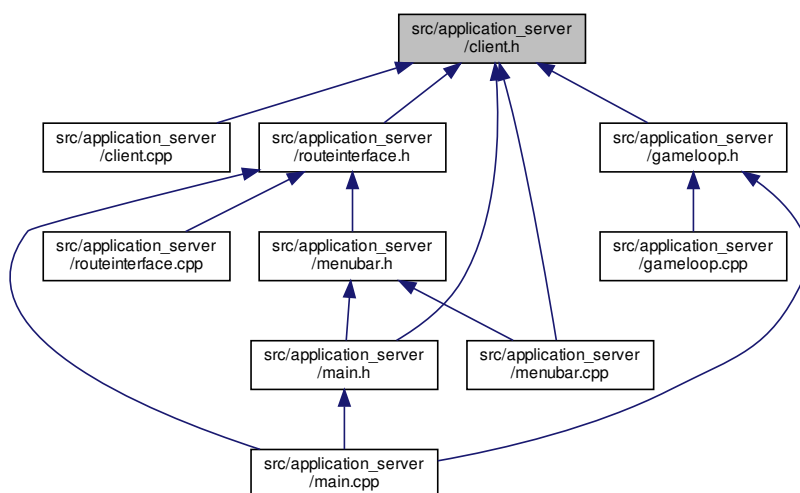
8.7 src/application_server/client.cpp-Dateireferenz

```
#include "client.h"
```

```
#include <QTimer>
```

```
#include <QDebug>
```


Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



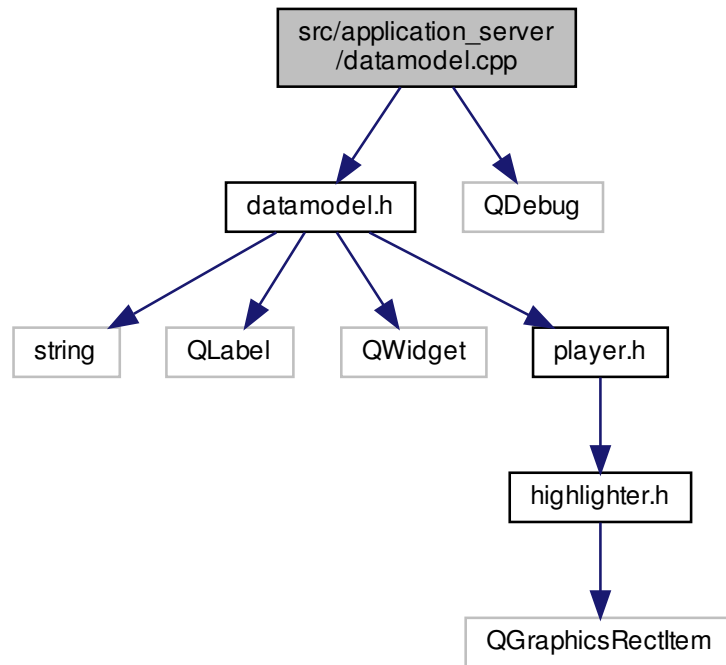
Klassen

- class Client

8.9 src/application_server/datamodel.cpp-Dateireferenz

```
#include "datamodel.h"
#include <QDebug>
```

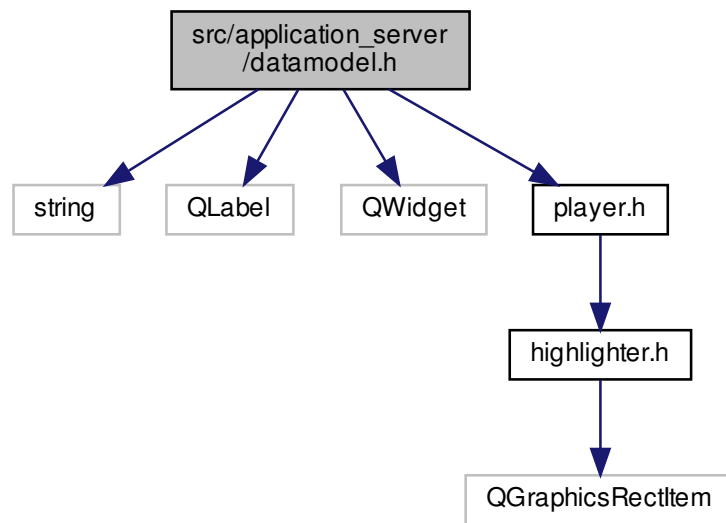
Include-Abhängigkeitsdiagramm für datamodel.cpp:



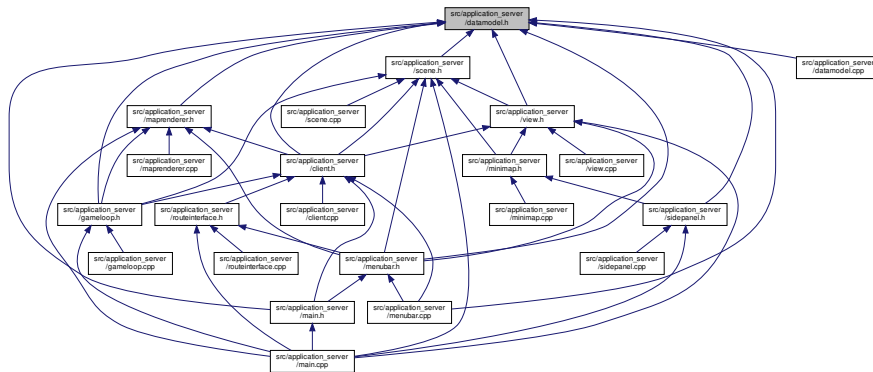
8.10 src/application_server/datamodel.h-Dateireferenz

```
#include <string>
#include <QLabel>
#include <QWidget>
#include "player.h"
```

Include-Abhängigkeitsdiagramm für datamodel.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [DataModel](#)

8.11 src/application_server/gameloop.cpp-Dateireferenz

```

#include "gameloop.h"
#include <iostream>
#include <ctime>
  
```

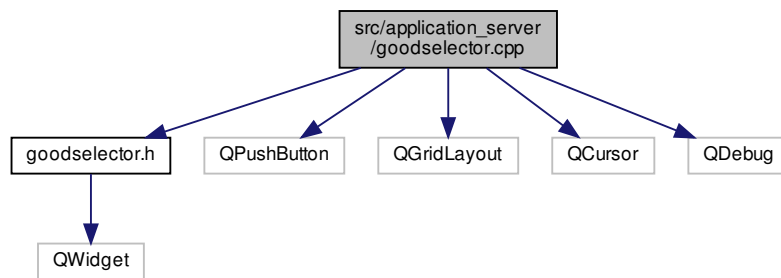

Klassen

- class [GameLoop](#)

8.13 src/application_server/goodselector.cpp-Dateireferenz

```
#include "goodselector.h"  
#include <QPushButton>  
#include <QGridLayout>  
#include <QCursor>  
#include <QDebug>
```

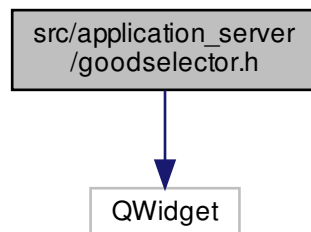
Include-Abhängigkeitsdiagramm für goodselector.cpp:



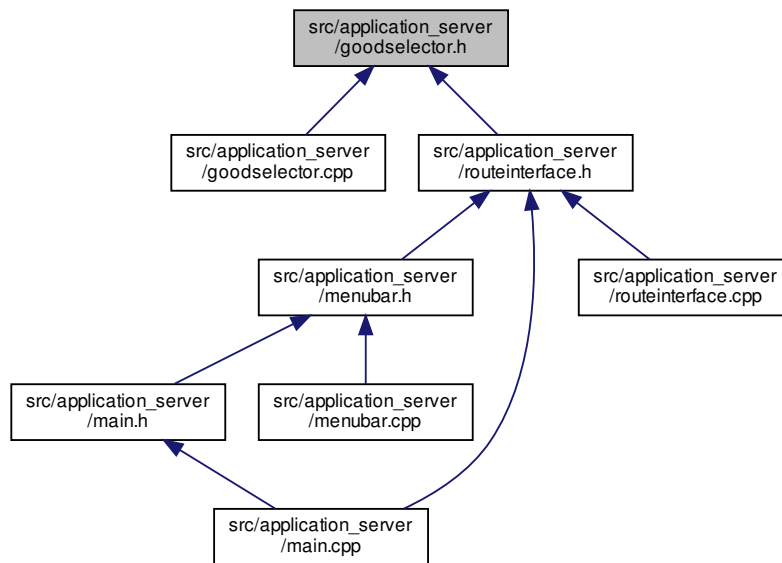
8.14 src/application_server/goodselector.h-Dateireferenz

```
#include <QWidget>
```

Include-Abhängigkeitsdiagramm für goodselector.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



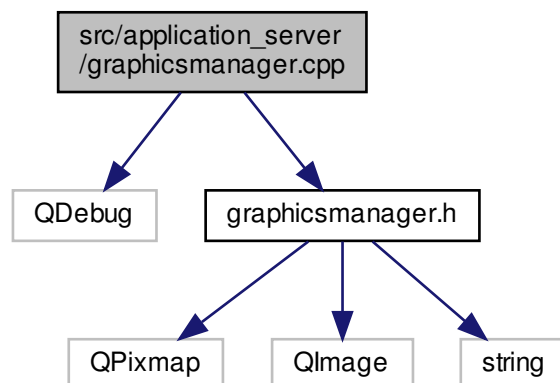
Klassen

- class [GoodSelector](#)

8.15 src/application_server/graphicsmanager.cpp-Dateireferenz

```
#include <QDebug>
#include "graphicsmanager.h"
```

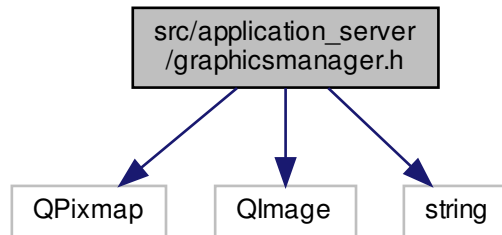
Include-Abhängigkeitsdiagramm für `graphicsmanager.cpp`:



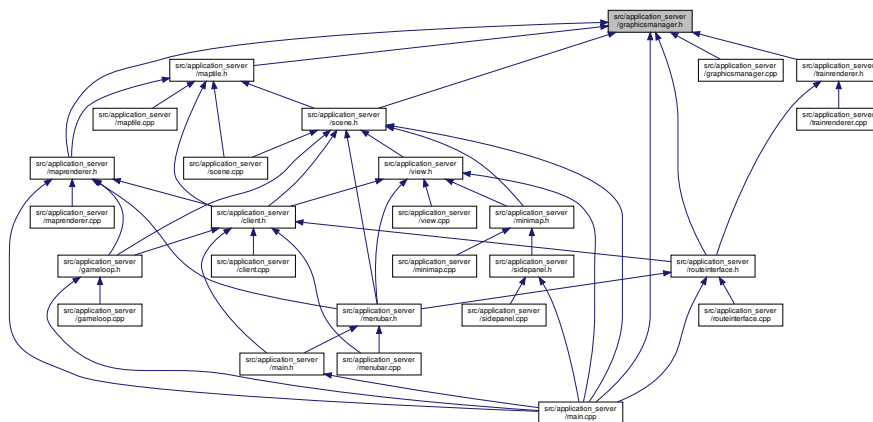
8.16 src/application_server/graphicsmanager.h-Dateireferenz

```
#include <QPixmap>
#include <QImage>
#include <string>
```

Include-Abhängigkeitsdiagramm für graphicsmanager.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



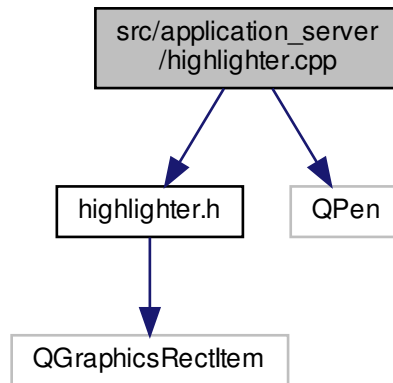
Klassen

- class `GraphicsManager`

8.17 src/application_server/highlighter.cpp-Dateireferenz

```
#include "highlighter.h"
#include <QPen>
```

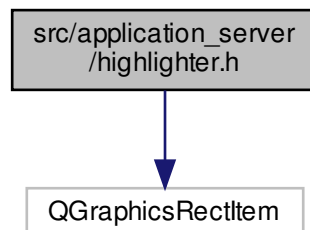
Include-Abhängigkeitsdiagramm für highlighter.cpp:



8.18 src/application_server/highlighter.h-Dateireferenz

```
#include <QGraphicsRectItem>
```

Include-Abhängigkeitsdiagramm für highlighter.h:

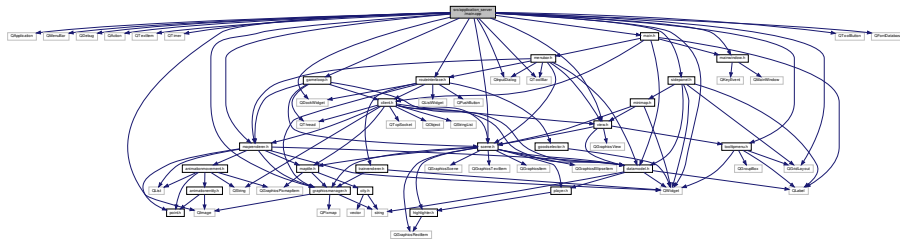


- class **Highlighter**

```
#include <QApplication>
#include <QMenuBar>
#include <QDebug>
#include <QAction>
#include <QTextItem>
#include <QTimer>
#include <QDockWidget>
#include <QWidget>
#include <QGridLayout>
#include <QInputDialog>
#include <QToolBar>
#include <QToolButton>
#include <QFontDatabase>
#include <QImage>
#include "mainwindow.h"
#include "main.h"
#include "view.h"
#include "scene.h"
#include "graphicsmanager.h"
#include "sidepanel.h"
#include "tooltipmenu.h"
#include "gameloop.h"
#include "routeinterface.h"
```

```
#include "maprender.h"
```

Include-Abhängigkeitsdiagramm für main.cpp:



Funktionen

- void `timeTicker` ()
- int `main` (int argc, char *argv[])
main Startmethode.

Variablen

- `GraphicsManager` * `graphics`
- `MainWindow` * `mainWindow`
- `DataModel` * `dataModel`
- bool `gameRunning` = true
- `View` * `view`
- `Scene` * `scene`
- `SidePanel` * `sidePanel`
- `Client` * `client`
- `MapRenderer` * `mapRenderer`

8.19.1 Dokumentation der Funktionen

8.19.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

main Startmethode.

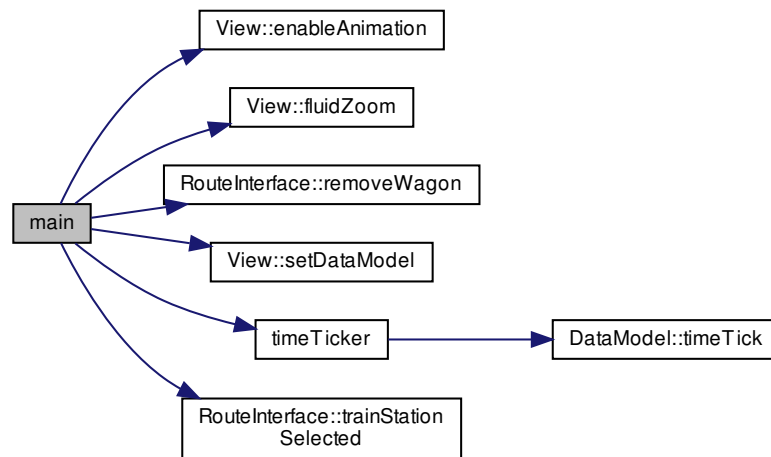
Parameter

| | |
|-------------------|----------------------|
| <code>argc</code> | Anzahl der Parameter |
| <code>argv</code> | Startparameter |

Rückgabe

Exit-Code (0=Alles gut)

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.19.1.2 timeTicker()

```
void timeTicker ( )
```

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.19.2 Variablen-Dokumentation

8.19.2.1 client

`Client*` client

8.19.2.2 dataModel

`DataModel*` dataModel

8.19.2.3 gameRunning

`bool` gameRunning = true

8.19.2.4 graphics

`GraphicsManager*` graphics

8.19.2.5 mainWindow

`MainWindow*` mainWindow

8.19.2.6 mapRenderer

`MapRenderer*` mapRenderer

8.19.2.7 scene

`Scene*` scene

Variablen

- `MainWindow * mainWindow`
- `bool gameRunning`
- `DataModel * dataModel`
- `Client * client`

8.20.1 Variablen-Dokumentation

8.20.1.1 client

```
Client* client
```

8.20.1.2 dataModel

```
DataModel* dataModel
```

8.20.1.3 gameRunning

```
bool gameRunning
```

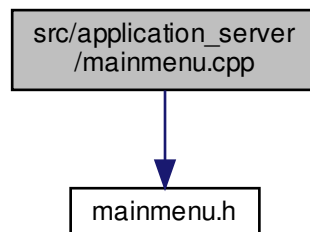
8.20.1.4 mainWindow

```
MainWindow* mainWindow
```

8.21 src/application_server/mainmenu.cpp-Dateireferenz

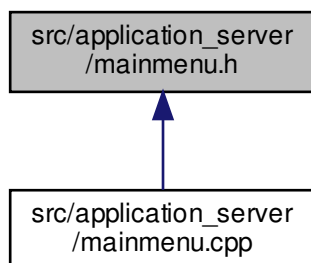
```
#include "mainmenu.h"
```

Include-Abhängigkeitsdiagramm für mainmenu.cpp:



8.22 src/application_server/mainmenu.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



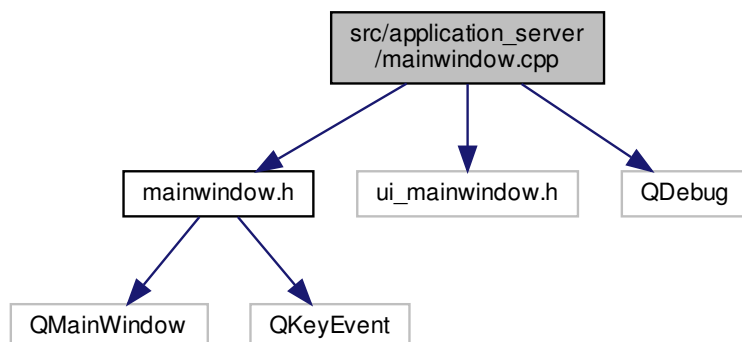
Klassen

- class [MainMenu](#)

8.23 src/application_server/mainwindow.cpp-Dateireferenz

```
#include "mainwindow.h"  
#include "ui_mainwindow.h"  
#include <QDebug>
```

Include-Abhängigkeitsdiagramm für mainwindow.cpp:

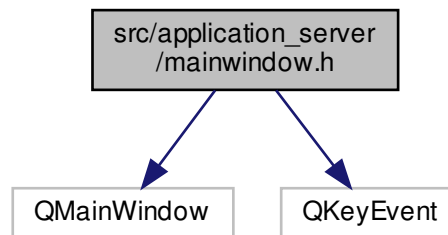


8.24 src/application_server/mainwindow.h-Dateireferenz

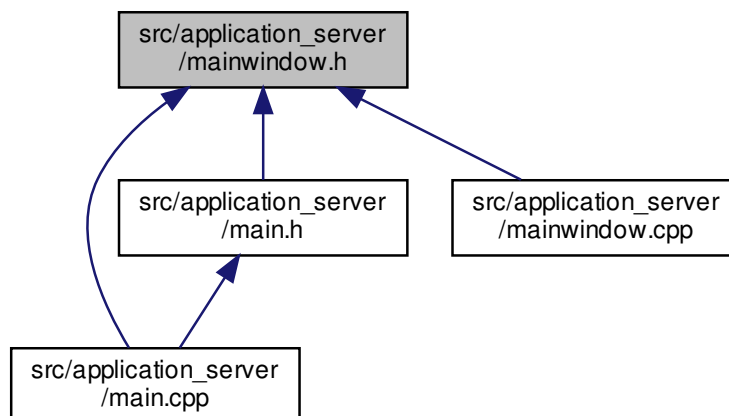
```
#include <QMainWindow>
```

```
#include <QKeyEvent>
```

Include-Abhängigkeitsdiagramm für mainwindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [MainWindow](#)

Namensbereiche

- [Ui](#)

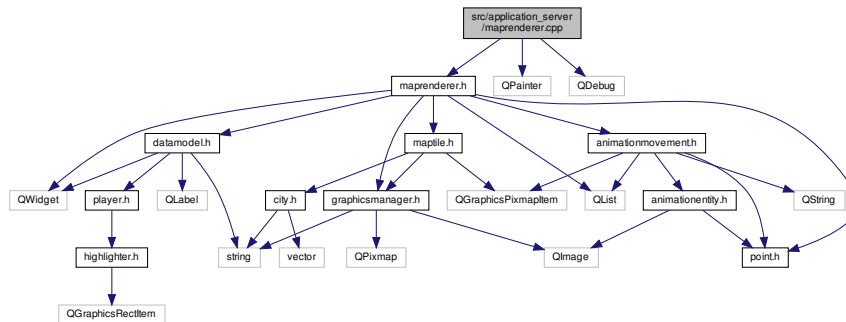
8.25 src/application_server/maprender.cpp-Dateireferenz

```
#include "maprender.h"
```

```
#include <QPainter>
```

```
#include <QDebug>
```

Include-Abhängigkeitsdiagramm für maprender.cpp:



8.26 src/application_server/maprender.h-Dateireferenz

```
#include <QWidget>
```

```
#include <QList>
```

```
#include "graphicsmanager.h"
```

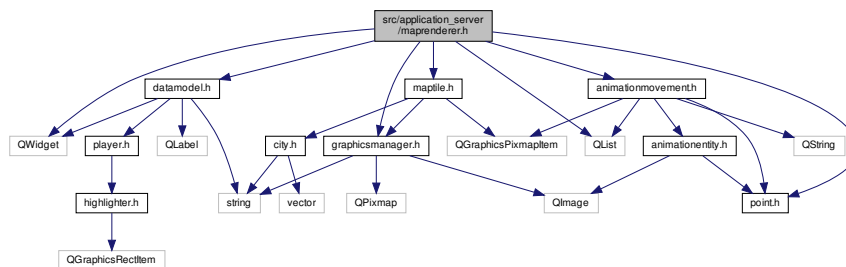
```
#include "datamodel.h"
```

```
#include "maptile.h"
```

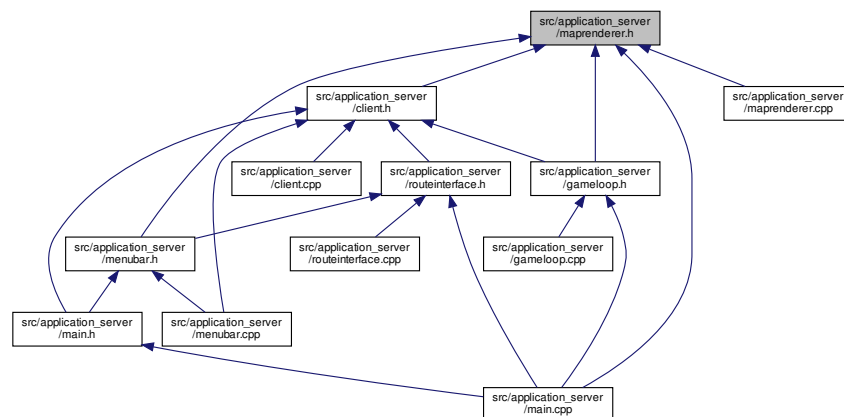
```
#include "point.h"
```

```
#include "animationmovement.h"
```

Include-Abhängigkeitsdiagramm für maprender.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



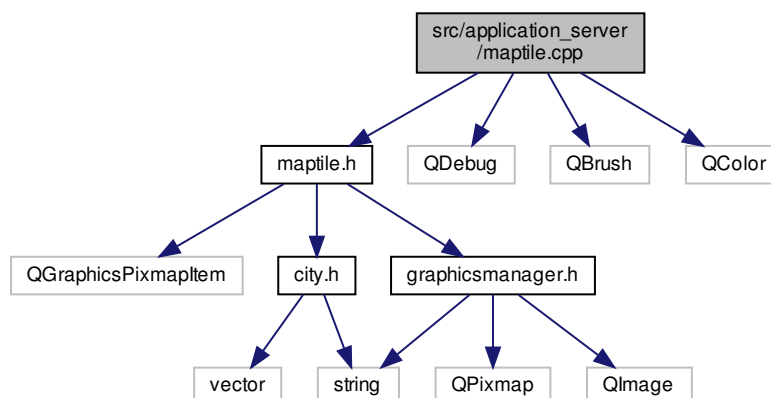
Klassen

- class [MapRenderer](#)

8.27 src/application_server/maptile.cpp-Dateireferenz

```
#include "maptile.h"
#include <QDebug>
#include <QBrush>
#include <QColor>
```

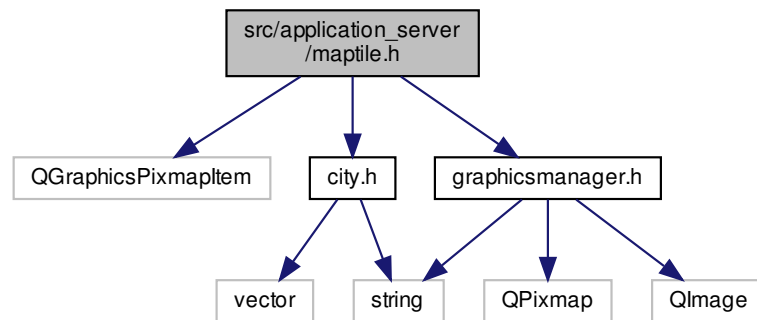
Include-Abhängigkeitsdiagramm für maptile.cpp:



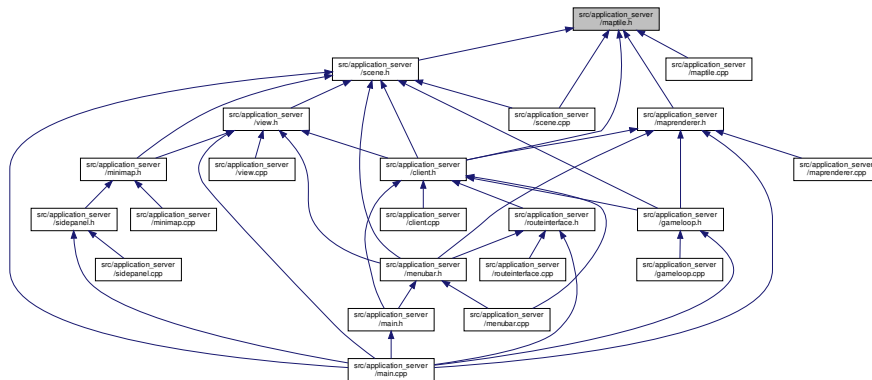
8.28 src/application_server/maptile.h-Dateireferenz

```
#include <QGraphicsPixmapItem>
#include "city.h"
#include "graphicsmanager.h"
```

Include-Abhängigkeitsdiagramm für maptile.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



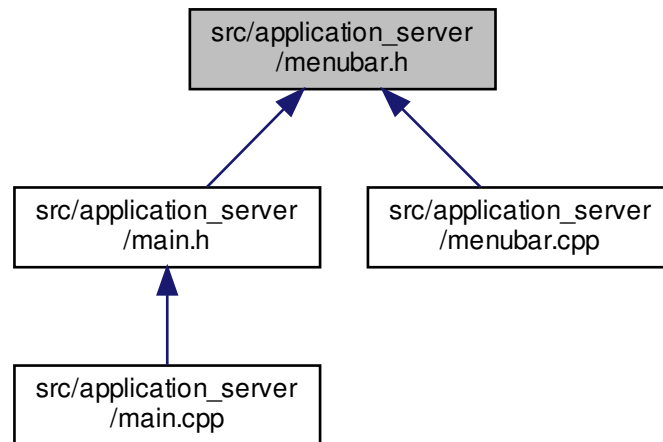
Klassen

- class MapTile

8.29 src/application_server/menubar.cpp-Dateireferenz

```
#include "menubar.h"
#include <QDebug>
#include "client.h"
#include "datamodel.h"
```


Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [MenuBar](#)

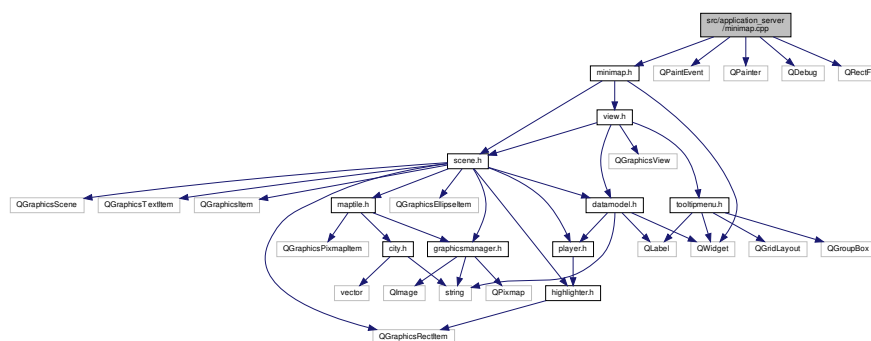
8.31 src/application_server/minimap.cpp-Dateireferenz

```

#include "minimap.h"
#include <QPaintEvent>
#include <QPainter>
#include <QDebug>
#include <QRectF>

```

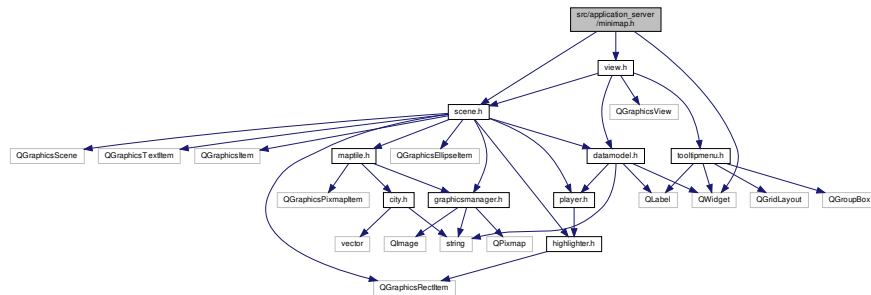
Include-Abhängigkeitsdiagramm für minimap.cpp:



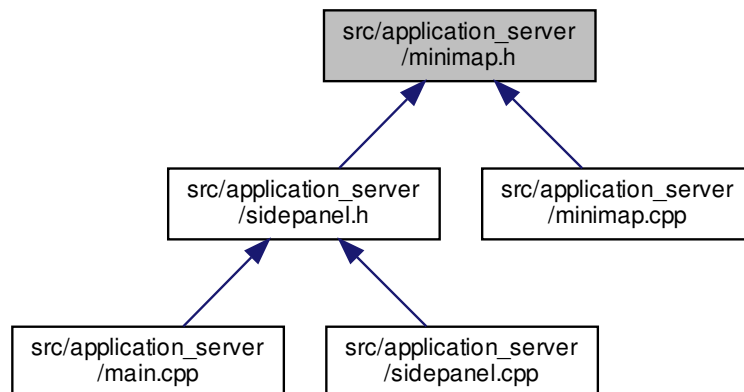
8.32 src/application_server/minimap.h-Dateireferenz

```
#include <QWidget>
#include "scene.h"
#include "view.h"
```

Include-Abhängigkeitsdiagramm für minimap.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



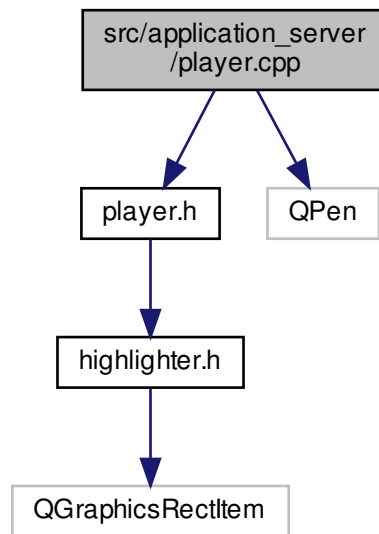
Klassen

- class [Minimap](#)

8.33 src/application_server/player.cpp-Dateireferenz

```
#include "player.h"
#include <QPen>
```

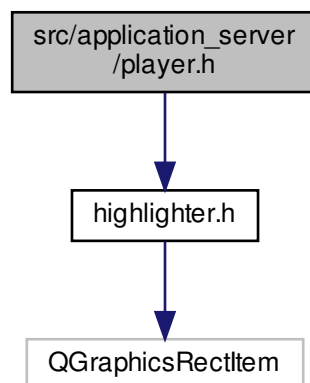
Include-Abhängigkeitsdiagramm für player.cpp:



8.34 src/application_server/player.h-Dateireferenz

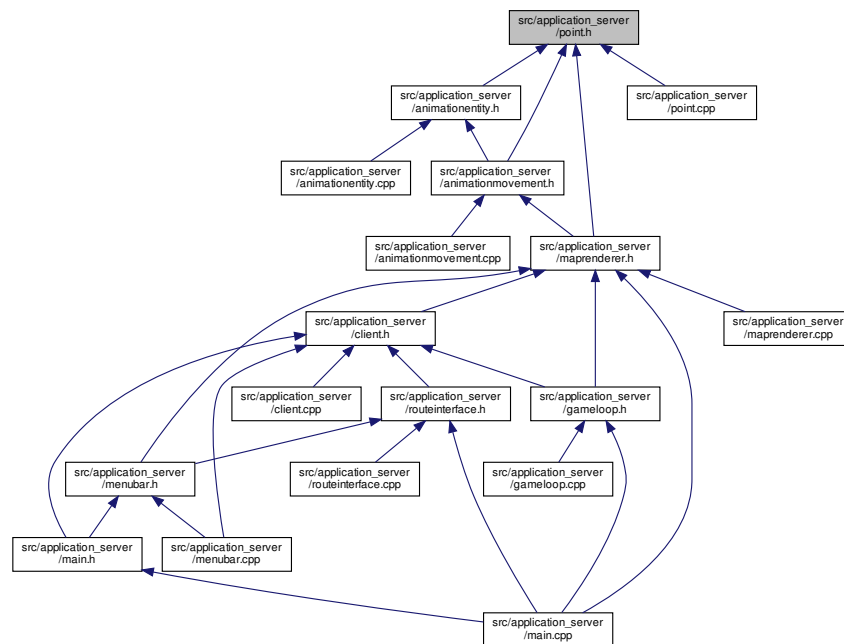
```
#include "highlighter.h"
```

Include-Abhängigkeitsdiagramm für player.h:



8.36 src/application_server/point.h-Dateireferenz

Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

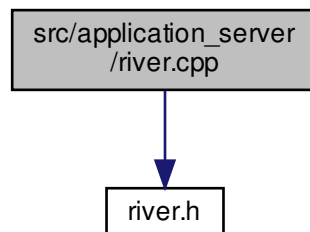
- class [Point](#)

8.37 src/application_server/README.md-Dateireferenz

8.38 src/application_server/river.cpp-Dateireferenz

```
#include "river.h"
```

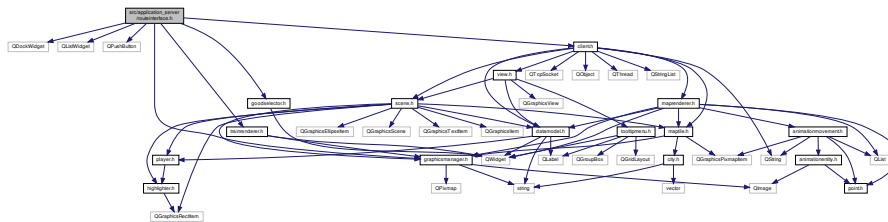
Include-Abhängigkeitsdiagramm für river.cpp:



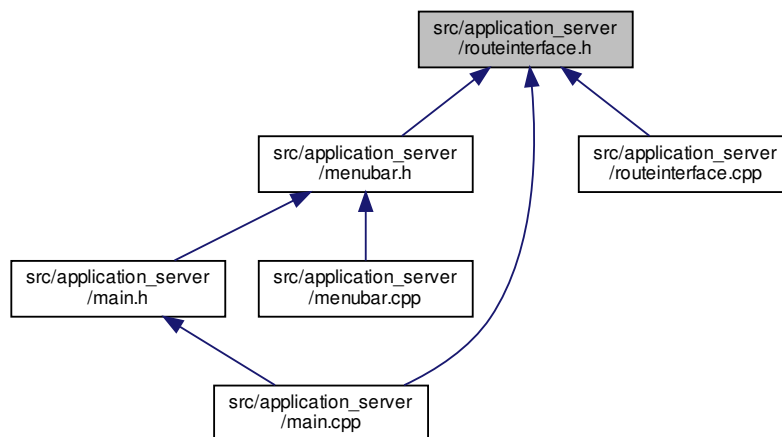
8.41 src/application_server/routeinterface.h-Dateireferenz

```
#include <QDockWidget>
#include <QListWidget>
#include <QPushButton>
#include "trainrenderer.h"
#include "graphicsmanager.h"
#include "goodselector.h"
#include "client.h"
```

Include-Abhängigkeitsdiagramm für routeinterface.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



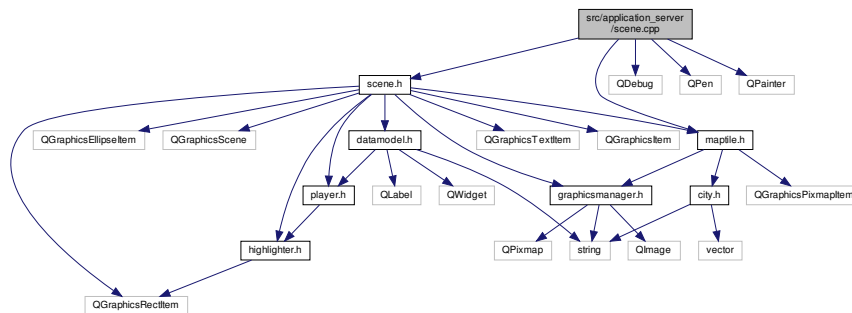
Klassen

- class [RouteInterface](#)

8.42 src/application_server/scene.cpp-Dateireferenz

```
#include "scene.h"
#include "maptile.h"
#include <QDebug>
```

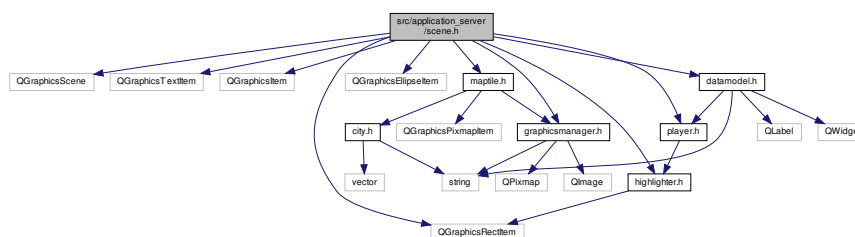
```
#include <QPen>
#include <QPainter>
Include-Abhängigkeitsdiagramm für scene.cpp:
```



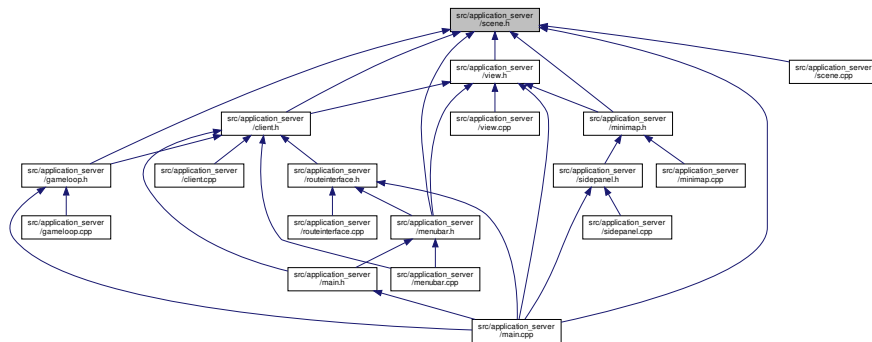
8.43 src/application_server/scene.h-Dateireferenz

```
#include <QGraphicsScene>
#include <QGraphicsTextItem>
#include <QGraphicsItem>
#include <QGraphicsRectItem>
#include <QGraphicsEllipseItem>
#include "maptile.h"
#include "graphicsmanager.h"
#include "player.h"
#include "highlighter.h"
#include "datamodel.h"
```

Include-Abhängigkeitsdiagramm für scene.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



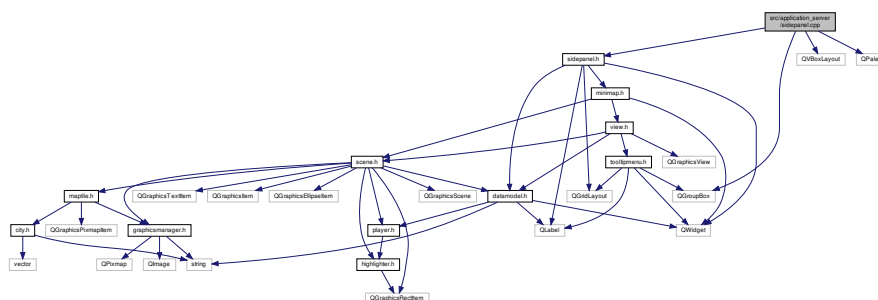
Klassen

- class [Scene](#)

8.44 src/application_server/sidepanel.cpp-Dateireferenz

```
#include "sidepanel.h"
#include <QGroupBox>
#include <QVBoxLayout>
#include <QPalette>
```

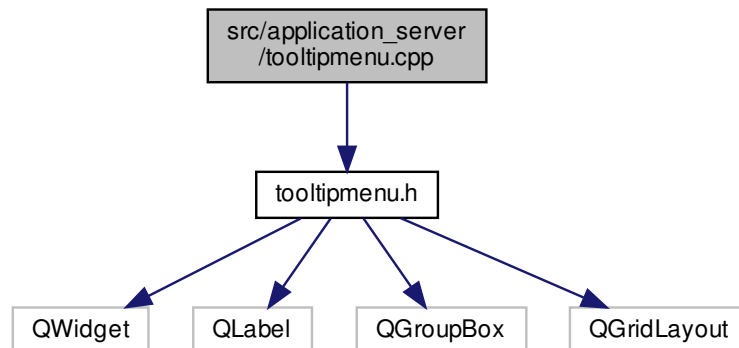
Include-Abhängigkeitsdiagramm für sidepanel.cpp:



8.45 src/application_server/sidepanel.h-Dateireferenz

```
#include <QWidget>
#include <QGridLayout>
#include <QLabel>
#include "datamodel.h"
```

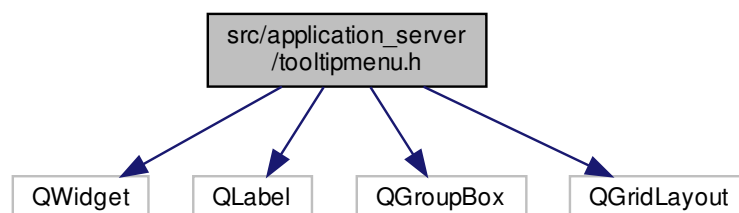

Include-Abhängigkeitsdiagramm für tooltipmenu.cpp:



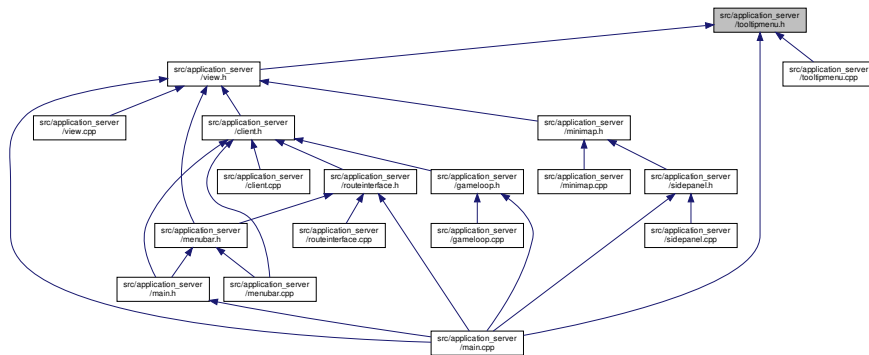
8.47 src/application_server/tooltipmenu.h-Dateireferenz

```
#include <QWidget>  
#include <QLabel>  
#include <QGroupBox>  
#include <QGridLayout>
```

Include-Abhängigkeitsdiagramm für tooltipmenu.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



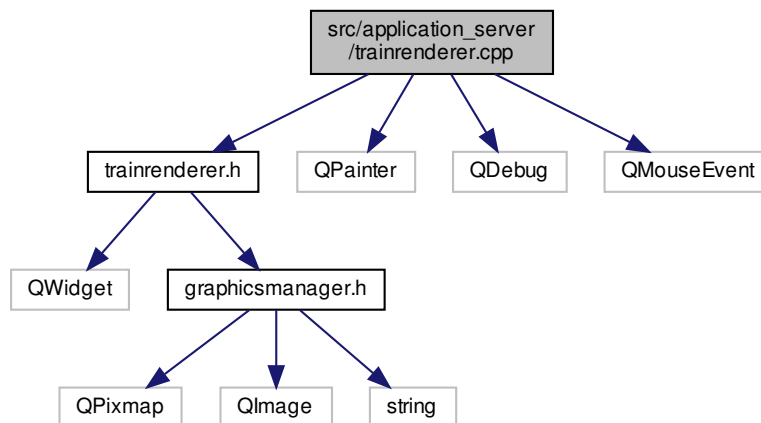
Klassen

- class [ToolTipMenu](#)

8.48 src/application_server/trainrenderer.cpp-Dateireferenz

```
#include "trainrender.h"
#include <QPainter>
#include <QDebug>
#include <QMouseEvent>
```

Include-Abhängigkeitsdiagramm für trainrenderer.cpp:



Funktionen

- void [triggerRemoveWagon](#) ()

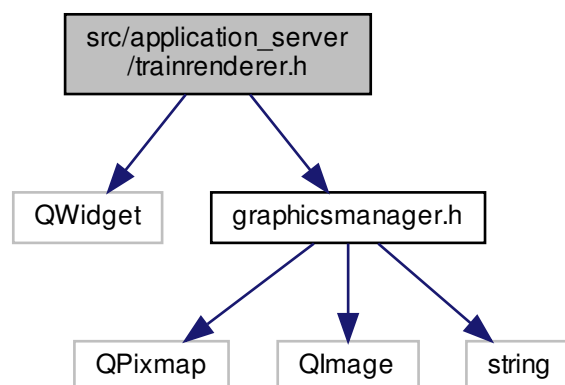
8.48.1 Dokumentation der Funktionen

8.48.1.1 triggerRemoveWagon()

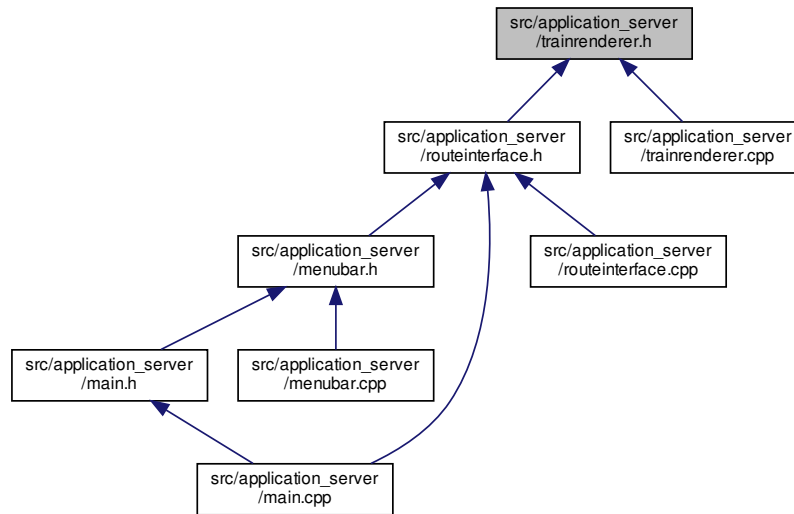
```
void triggerRemoveWagon ( )
```

8.49 src/application_server/trainrender.h-Dateireferenz

```
#include <QWidget>  
#include "graphicsmanager.h"  
Include-Abhängigkeitsdiagramm für trainrender.h:
```



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class [TrainRenderer](#)

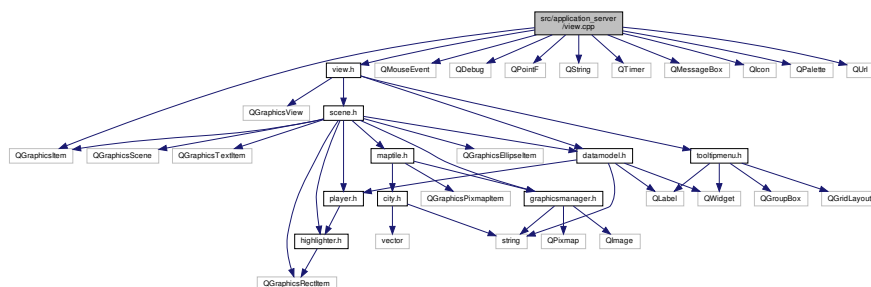
8.50 src/application_server/view.cpp-Dateireferenz

```

#include "view.h"
#include <QMouseEvent>
#include <QDebug>
#include <QPointF>
#include <QString>
#include <QGraphicsItem>
#include <QTimer>
#include <QMessageBox>
#include <QIcon>
#include <QPalette>
#include <QUrl>

```

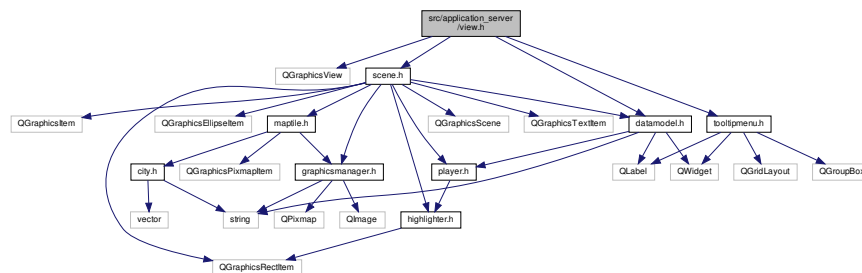
Include-Abhängigkeitsdiagramm für view.cpp:



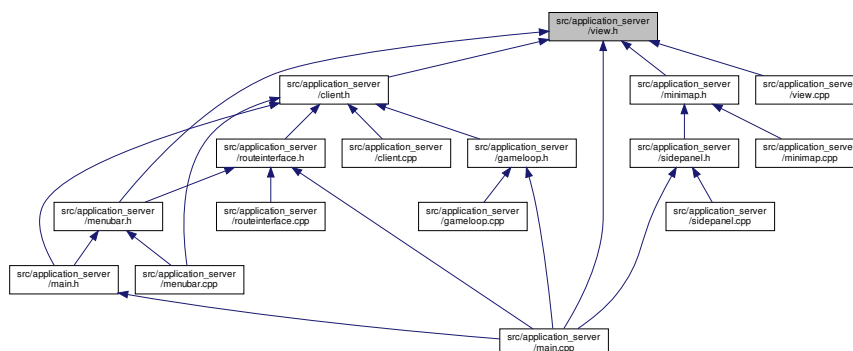
8.51 src/application_server/view.h-Dateireferenz

```
#include <QGraphicsView>
#include "scene.h"
#include "datamodel.h"
#include "tooltipmenu.h"
```

Include-Abhängigkeitsdiagramm für view.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class **View**

