TransAmercia

v0.2.1

Generated by Doxygen 1.8.6

Sat Mar 8 2014 00:57:58

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

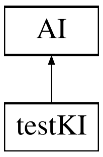
Chapter 3

Class Documentation

3.1 AI Class Reference

`#include <AI.h>`

Inheritance diagram for AI:



Public Member Functions

- **AI** (PLAYERCOLOR **playercolor**)

Public Attributes

- const PLAYERCOLOR **playercolor**
- string **owner**
- string **AIname**

Protected Member Functions

- virtual **Move doMove** (**State** &aktuell)=0
- virtual **Vector setPawn** (**State** &aktuell)=0
- virtual bool **countPoints** (**State** &currentState, std::vector$<$ **Connection** $*$ $>$ path)=0
- virtual void **gatherInformationEndOfRound** (const **RoundLogger** $*$currentInfos)=0

Protected Attributes

- const **City** $**$ **hand**

Friends

- class **Game**
- class **Round**

3.1.1 Detailed Description

If you want to create your own **AI** (p. 5), you have to implement this class as a kind of interface. You have to implement every abstract method.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 AI::AI (PLAYERCOLOR *playercolor*)

A constructor where you can set the name of your **AI** (p. 5) and your name, but not much more ;)

3.1.3 Member Function Documentation

3.1.3.1 virtual bool AI::countPoints (State & *currentState,* std::vector< Connection ∗ > *path*) `[protected]`, `[pure virtual]`

Here you can count your minus points at the end of each round. If you want to do so, you have to return true. For the beginning it is okay, if you just return false. Then the gamemaster will count the minuspoints. He counts in most cases the minimum of minuspoints you get, however in some cases the algorithm doesn't evaluate the best value.

Implemented in **testKI** (p. 16).

3.1.3.2 virtual Move AI::doMove (State & *aktuell*) `[protected]`, `[pure virtual]`

Inside this methode you calculate your next move in the game.

Implemented in **testKI** (p. 16).

3.1.3.3 virtual void AI::gatherInformationEndOfRound (const RoundLogger ∗ *currentInfos*) `[protected]`, `[pure virtual]`

At the end of each round you can take a look at the whole game and the playing cards of your opponents. This can be usefull, if you want to figure out there strategy and react to that over a simulation period.

Implemented in **testKI** (p. 16).

3.1.3.4 virtual Vector AI::setPawn (State & *aktuell*) `[protected]`, `[pure virtual]`

At the beginning of each round you have to define your starting position of your pawn.

Implemented in **testKI** (p. 17).

3.1.4 Member Data Documentation

3.1.4.1 string AI::AIname

Possibly the name of your **AI** (p. 5).

3.1.4.2 const City∗∗ AI::hand `[protected]`

This is just an array of City-Pointers, that represents your hand.

3.1.4.3 string AI::owner

Possibly your name.

3.1.4.4 const PLAYERCOLOR AI::playercolor

This represents your color during the game. You can try to change it, but hopefully you shouldn't be able.

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

- game/header/AI.h
- game/source/AI.cpp

3.2 Board Class Reference

Public Member Functions

- **Coordinate** $***$ **gitterAnlegen** () const
- **City** $**$ **stadtlisteAnlegen** () const
- **Connection** $****$ **kantenAnlegen** () const
- void **Ausgabe** () const
- **City** $*$const **durchsucheListe** (short xkoo, short ykoo) const
- **City** $*$ **getStadt** (short farbe, short nr) const

Public Attributes

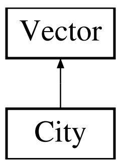
- const short **anzahlStaedte**
- **City** $*$const $*$const **Stadtliste**
- const **Coordinate** $*$const $*$const
  $*$const **Gitter**
- const **Connection** $*$const $*$const
  $*$const $*$const **Kanten**

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

- game/header/Board.h
- game/source/Board.cpp

3.3 City Class Reference

Inheritance diagram for City:



Public Member Functions

- **City** (string name, CITYCOLOUR cityColour, short number, **Vector** place)

Public Attributes

- string **name**
- CITYCOLOUR **cityColour**
- short **number**

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

- game/header/City.h
- game/source/City.cpp

3.4 Connection Class Reference

Public Member Functions

- **Connection** (const **Coordinate** &erste, const **Coordinate** &zweite, bool Hindernis)
- const **Connection** & **operator=** (const **Connection** &) const

Public Attributes

- const **Coordinate** & **first**
- const **Coordinate** & **second**
- const **Vector** **richtung**
- const bool **hindernis**

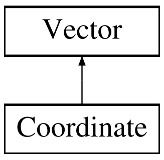
The documentation for this class was generated from the following files:

- game/header/Connection.h
- game/source/Connection.cpp

3.5 Coordinate Class Reference

```
#include <Coordinate.h>
```

Inheritance diagram for Coordinate:



Public Member Functions

- **Coordinate** (short x, short y)
- **Coordinate** (short x, short y, const **City** ∗const CityOnCoordinate)

Public Attributes

- const **City** ∗const **vorOrt**

3.5.1 Detailed Description

Objects of this class are coordiantes on the board, where the x-axis is parallel to the west-east direction and the y-axis parallel to the north-(south-west) direction. The (0,0) coordinate is the most upper left corner of the board. The range of the x-values is from 0 to MAX_X and the y-values from 0 to MAX_Y.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Coordinate::Coordinate (short *x*, short *y*)

This constructor creates a coordinate with no city on it.

3.5.2.2 Coordinate::Coordinate (short *x*, short *y*, const **City** ∗const *CityOnCoordinate*)

This constructor creates a coordinate with city on it.

3.5.3 Member Data Documentation

3.5.3.1 const **City**∗ const Coordinate::vorOrt

This pointer is zero, if there is no city on the coordinate, otherwise it's pointing to the city.

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

- game/header/Coordinate.h
- game/source/Coordinate.cpp

3.6 Counter Class Reference

Public Member Functions

- **Counter** (const **Counter** &copy)
- int **add** (**AI** ∗player, int counter)
- int **get** (**AI** ∗player) const
- **Counter operator+** (const **Counter** &rhs) const
- **Counter operator-** (const **Counter** &rhs) const
- **Counter operator=** (const **Counter** &copy)
- **Counter operator+=** (const **Counter** &rhs)
- **Counter operator-=** (const **Counter** &rhs)

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

- game/header/Counter.h
- game/source/Counter.cpp

3.7 Game Class Reference

Public Member Functions

- **Game** (**GameLogger** ∗gameLogger)
- void **play** ()

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

- game/header/Game.h
- game/source/Game.cpp

3.8 GameLogger Class Reference

Public Member Functions

- **GameLogger** (std::vector< **AI** ∗ > playerList, **Board** &board, **PlayingOrder** playingOrder, **AI** ∗gameStarting-Player)

Public Attributes

- **AI** ∗ **gameStartingPlayer**
- std::vector< **AI** ∗ > **playerList**
- **Board** & **board**
- **PlayingOrder playingOrder**
- int **deadLine**
- **Counter points**
- **Counter winnerPoints**
- std::vector< **RoundLogger** ∗ > **roundList**

3.8.1 Member Data Documentation

3.8.1.1 Counter GameLogger::points

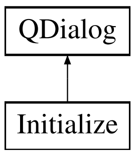
< if points equal or lower deadLine, you lose

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

- logger/header/GameLogger.h
- logger/source/GameLogger.cpp

3.9 Initialize Class Reference

Inheritance diagram for Initialize:



Public Member Functions

- **Initialize** (const QString &title, QWidget ∗parent)
- QString **name** ()
- int **numberOfGames** ()

Public Attributes

- std::vector< **AI** ∗ > **players**

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

- userinterface/source/initialize.h
- userinterface/source/initialize.cpp

3.10 PlayingOrder::iterator Class Reference

Public Member Functions

- **iterator** (PlayingOrderElement ∗cursor)
- **AI** ∗ **operator->** () const
- **AI** ∗ **operator∗** () const
- **iterator operator++** ()
- bool **operator!=** (const **PlayingOrder::iterator** &rhs) const

Public Attributes

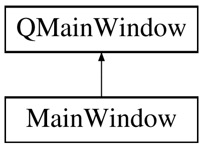
- PlayingOrderElement ∗ **cursor**

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

- game/header/PlayingOrder.h
- game/source/iterator.cpp

3.11 MainWindow Class Reference

Inheritance diagram for MainWindow:



Protected Member Functions

- void **mouseReleaseEvent** (QMouseEvent ∗event)

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

- userinterface/header/mainwindow.h
- userinterface/source/mainwindow.cpp

3.12 Move Class Reference

Public Member Functions

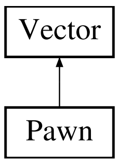
- **Move** (PLAYERCOLOR spielerfarbe, const **Connection** ∗belegt1, const **Connection** ∗belegt2)
- bool **valid** (**State**, PLAYERCOLOR)
- void **execute** (**State** &) const
- void **dump** () const
- **Move** & **operator=** (const **Move** &zuweisung)

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

- game/header/Move.h
- game/source/Move.cpp

3.13 Pawn Class Reference

Inheritance diagram for Pawn:



Public Member Functions

- **Pawn** (PLAYERCOLOR colour, **Vector** pos)
- **Pawn** (const **Pawn** &copy)

Public Attributes

- short **schienennetznummer**
- const PLAYERCOLOR **spielerfarbe**

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

- game/header/Pawn.h
- game/source/Pawn.cpp

3.14 PlayingOrder Class Reference

Classes

- class **iterator**

Public Member Functions

- **PlayingOrder** (std::vector< **AI** ∗ > order)
- **PlayingOrder::iterator begin** (**AI** ∗player) const

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

- game/header/PlayingOrder.h
- game/source/PlayingOrder.cpp

3.15 Round Class Reference

Public Member Functions

- **Round** (**RoundLogger** ∗roundLogger)
- void **play** ()

Public Attributes

- **State currentState**

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

- game/header/Round.h
- game/source/Round.cpp

3.16 RoundLogger Class Reference

Public Member Functions

- **RoundLogger** (**PlayingOrder** &playingOrder, std::vector< **AI** ∗ > playerList, **Board** &board, **AI** ∗round-StartingPlayer)

Public Attributes

- **PlayingOrder** & **playingOrder**
- std::vector< **AI** ∗ > **playerList**
- **Board** & **board**
- **AI** ∗ **roundStartingPlayer**
- **City** ∗∗ **playingCards**
- **Counter lostPoints**
- **Pawn** ∗∗ **pawnList**
- std::vector< **Move** ∗ > **moveList**

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

- logger/header/RoundLogger.h
- logger/source/RoundLogger.cpp

3.17 Simulation Class Reference

Public Member Functions

- **Simulation** (**SimulationLogger** ∗simulationLogger)
- void **run** ()

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

- game/header/Simulation.h
- game/source/Simulation.cpp

3.18 SimulationLogger Class Reference

Public Member Functions

- **SimulationLogger** (std::vector< **AI** ∗ > playerList, **Board** &board, int numberOfPlayers, unsigned int seed=(unsigned) time(0))
- std::vector< **AI** ∗ > **getPlayingOrder** (int simulationNumber)

Public Attributes

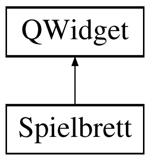
- std::vector< **AI** ∗ > **playerList**
- **Counter gamesWon**
- std::vector< **GameLogger** ∗ > **gameList**
- **Board board**
- unsigned int **seed**

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

- logger/header/SimulationLogger.h
- logger/source/SimulationLogger.cpp

3.19 Spielbrett Class Reference

Inheritance diagram for Spielbrett:



Public Slots

- void **zustandChanged** (int)
- void **drawCityChanged** (bool)

Public Member Functions

- **Spielbrett** (**Window** ∗parentalWindow)

Protected Member Functions

- void **paintEvent** (QPaintEvent ∗event)

3.19.1 Member Function Documentation

3.19.1.1 void Spielbrett::paintEvent (QPaintEvent ∗ *event*) `[protected]`

draws the Railway-System

draws the names of the citys

draw the city.gifs

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

- userinterface/header/spielbrett.h
- userinterface/source/spielbrett.cpp

3.20 State Class Reference

Public Member Functions

- **State** (**Board** &**Spielbrett**)
- **State** (const **State** &)
- **Pawn getPoeppel** (const PLAYERCOLOR spielerfarbe) const
- bool **schienenNetzNummerVon_Ist_** (const **Connection** &, const short schienennr) const
- short **getSchienenNetzNummer** (const **Vector** &koo) const
- void **setSchienenNetzNummer** (const **Coordinate** &koo, const short nr)
- void **setSchiene** (const **Connection** &)
- void **resetNr_ZuNr_** (const short, const short)
- void **schieneLegen** (const **Connection** &)
- const **Connection** ∗ **getVerbindung** (**Vector** a, **Vector** b) const
- void **addPawn** (**Pawn** insert)
- void **resetAll** ()
- unsigned short ∗∗ **evaluateBoard** (**Vector** target) const
- unsigned short **distance** (**Vector** target, const vector< **Vector** > &possibleStarts) const
- vector< **Vector** > **pointsBelongingToRailwaySystem** (PLAYERCOLOR playercolour) const
- void **aktAusgabe** () const
- void **setRound** (short x)
- void **setTurn** (short x)
- void **setPlayersTurn** (PLAYERCOLOR x)

Static Public Member Functions

- static short **RichtungsWert** (const **Vector** &)

Public Attributes

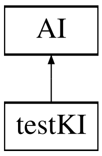
- short ∗∗ **schienenNetzNummer**
- bool ∗∗∗ **schieneGelegt**
- short **anzahlPoeppel**
- const **Board** & **Spielbrett**
- std::vector< **Pawn** ∗ > **unsortedPawns**

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

- game/header/State.h
- game/source/State.cpp

3.21 testKI Class Reference

Inheritance diagram for testKI:



Public Member Functions

- **testKI** (PLAYERCOLOR farbe)
- **Move doMove** (**State** &aktuell)
- **Vector setPawn** (**State** &aktuell)
- bool **countPoints** (**State** &, std::vector< **Connection** ∗ >)
- void **gatherInformationEndOfRound** (const **RoundLogger** ∗)
- **Vector getNaechsterPunktZu** (**Vector**, **State**) const

Static Public Member Functions

- static short **sign** (short)

Additional Inherited Members

3.21.1 Member Function Documentation

3.21.1.1 bool testKI::countPoints (State & *currentState,* std::vector< Connection ∗ > *path*) `[inline]`, `[virtual]`

Here you can count your minus points at the end of each round. If you want to do so, you have to return true. For the beginning it is okay, if you just return false. Then the gamemaster will count the minuspoints. He counts in most cases the minimum of minuspoints you get, however in some cases the algorithm doesn't evaluate the best value.

Implements **AI** (p. 6).

3.21.1.2 Move testKI::doMove (State & *aktuell*) `[virtual]`

Inside this methode you calculate your next move in the game.

Implements **AI** (p. 6).

3.21.1.3 void testKI::gatherInformationEndOfRound (const RoundLogger ∗ *currentInfos*) `[inline]`, `[virtual]`

At the end of each round you can take a look at the whole game and the playing cards of your opponents. This can be usefull, if you want to figure out there strategy and react to that over a simulation period.

Implements **AI** (p. 6).

3.21.1.4 Vector testKI::setPawn (State & *aktuell*) `[virtual]`

At the beginning of each round you have to define your starting position of your pawn.

Implements **AI** (p. 6).

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

- ai/testKI.h
- ai/testKI.cpp

3.22 UIEXEC Class Reference

Public Member Functions

- void **simulateSimulation** (int games=1)

Public Attributes

- **Window** ∗ **wp**
- **Board** ∗ **board**
- **SimulationLogger** ∗ **simulationLogger**
- **Simulation** ∗ **simulation**

3.22.1 Member Function Documentation

3.22.1.1 void UIEXEC::simulateSimulation (int *games* = `1`)

window implementation

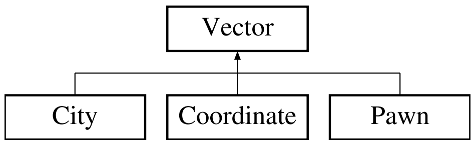
The documentation for this class was generated from the following files:

- userinterface/header/uiexec.h
- userinterface/source/uiexec.cpp

3.23 Vector Class Reference

```
#include <Vector.h>
```

Inheritance diagram for Vector:



Public Member Functions

- **Vector** (short x, short y)
- **Vector operator-** (const **Vector**) const
- **Vector operator+** (const **Vector**) const
- short **distance** () const
- void **dump** () const

Public Attributes

- short **x**
- short **y**

3.23.1 Detailed Description

An object of the class **Vector** (p. 17) represents a 2-dimensional vector with two integer values. It is a vector referenced to the board of the game.

3.23.2 Member Function Documentation

3.23.2.1 short Vector::distance ( ) const

Determines a non-negative integer, that represents the distance on the board in terms of steps. : This distance doesn't represent the number of steps, because it doesn't take care of bridges/tunnels!

3.23.2.2 void Vector::dump ( ) const [inline]

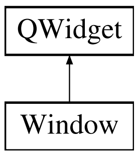
Dumps the values of the vector on the standard stream.

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

- game/header/Vector.h
- game/source/Vector.cpp

3.24 Window Class Reference

Inheritance diagram for Window:



Public Member Functions

- **Window** (**SimulationLogger** ∗game=0)
- void **setsimulationp** (**SimulationLogger** ∗game)

Friends

- class **Spielbrett**
- class **UIEXEC**
- class **MainWindow**

3.24.1 Constructor & Destructor Documentation

3.24.1.1 Window::Window (**SimulationLogger** ∗ *game* = 0)

Aukommentiert um Zusulassen, dass game = NULL

Layout-Design

Connect-Implementationen

Eventuell SimulationLogger∗ setzten

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

- userinterface/header/window.h
- userinterface/source/window.cpp

Index