BusDestroyer2000

Generated by Doxygen 1.8.17

Chapter 1

BUSDESTROYER 2000 (Programming 3 course project)

A shooter game developed with Qt Creator (C++) under open-source license. BusDestroyer 2000 has a game map of Tampere City Center and Nysse-buses and Passengers moving on the map.

The project was made by Miikka Mensio and fellow student/project partner in fall 2020 as a school project. We have developed our work under the StudentSide namespace so that there would be less confusion about which part is from CourseSide and which made by us. The project is open-source and is not in any way commercialized.

Doxygen documentation has been created inside the DoxygenDocumentation -folder (which can be found inside Documentation -folder) and inside there is index.html -file from which a documentation of this project can be viewed in a web browser.

All the relevant documentation of our project can be found inside the Documentation -folder, which is located in our repo root.

1.1 How to start the game

1) First clone this project to your local machine 2) Open Nysse.pro -file which is located in our repo root. 3) Run the program (CTRL + R)

1.2 DISCLAIMERS:

- We, the project team, do not take any credits for the code in directory "./Course" nor all the resources in "./Game/resources".
- Our contribution is in the code under the Game and StudentTest folders.
- This project was coded on top of the code provided by the Course Side.
- The software is provided as is.
- The software is not commercialized now nor in the future by the project team.

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 Interface Namespace Reference

All of the interfaces defined by the course are found in Interface namespace.

Classes

- class GameError

  Exception class that expresses errors ingame.
- class IActor

  ActorIF is an interface, which every single actor moving in the game implements.
- class ICity

  CityIF is an interface that every city in the game must fulfill. Kaupunki.
- class InitError

  Exception class that expresses errors during the initialization of the game.
- class IPassenger

  PassengerIF is an interface which every passenger in game implements.
- class IStatistics

  StatisticsIF is an interface, which defines an object that manages scoring statistics.
- class IStop

  StopIF is an interface that stops fulfill.
- class IVehicle

  VehicleIF is an interface that describes vehicles (nysse) in game.
- class Location

  Location is a class, which has methods dealing with the location of the objects.

Functions

- std::shared_ptr< ICity > createGame ()

  createGame creates the games city and return pointer to it.

6.1.1 Detailed Description

All of the interfaces defined by the course are found in Interface namespace.

6.1.2 Function Documentation

6.1.2.1 createGame()

```
std::shared_ptr<ICity> Interface::createGame ( )
```

createGame creates the games city and return pointer to it.

Precondition

-

Returns

pointer to the created city. (It is in initialization space)

Postcondition

Exception guaranteee: basic.

6.2 StudentSide Namespace Reference

All of the classes done by the student team are found in StudentSide namespace.

Classes

- class basicProjectile

  Defines a class for the projectile that player shoots in the game by pressing space key.
- class bonusItem

  Defines a class for the other unique object in the game: bonusItem's (small red diamonds) that appear randomly in the game map every 4 seconds. If the player manages to collect these bonus items either by moving into their location or by shooting them, 10 points are added to the player score.
- class gameCity

  Defines a class that inherits from ICity and implements its virtual functions. The class handles functionality regarding the actual city.
- class GameOverDialog

  Defines a QDialog that is shown to the player when the game is over.
- class gameStatistics

  Defines a class that inherits from IStatistics and implements its virtual functions. In addition class has some own functions and attributes. Class is responsible for the statistical bookkeeping for all the relevant statistics that the player generates ingame.
- class gameWindow

  Defines a class for the MainWindow where the actual playing happens. It has a game map background which depicts Tampere City Center. Game Window also shows the chosen player which can be moved with arrow keys. Shooting happens by pressing space bar. Also the CourseSide integration is visualized in the game window: Nysses and passengers move in real time on the map.
- class helpDialog

Defines a QDialog which offers help to a new player who possibly isn't aware of the game rules, settings or controls etc. The Dialog itself is accessible from MainMenu's "Help" -button.

- class initGame

  Defines a Class in which the CourseSide integration mostly happens. The class is also responsible for moving and showing all the actors offered by CourseSide. Furthermore, CreateGame -function is implemented here.

- class MainMenuDialog

  Defines a QDialog which is the first shown window to the user when the app is run. It is the main configuration dialog in which player sets a player nickname, chooses player and projectile type and possibly accesses settings or help.

- class Player

  Defines a class for the main player that the game's user controls in the game. The player can be in interaction with bonusItems, Nysses and passengers shown on the game map. Player movement happens via arrow key pressing, and shooting via space bar.

- class settingsDialog

  Defines a QDialog which offers a separate settings window where the player can set music on/off and alter the game duration. The Dialog itself is accessible from MainMenu's "Settings" -button. If the player does not make any changes to the settings (which isn't compulsory), the default settings are used: musics off and game duration 2 minutes. Beware that if you don't press the "Save" -button in the settingsDialog, your changes to the settings won't come into effect and the program will use the default settings!

- class statistisDialog

  Defines a QDialog which offers statisctics of the game played by the player. The Dialog itself is accessible from GameOverDialog's "Statistics" -button.

- class topHighScores

  Defines a Class which is responsible for saving and reading player name and SCALED points to and from a top10highscores.txt file. The class also implements a top10-highscore feature which gives the player information about all-time best players when the metric is the highest scaled points count.

Variables

- const QString **textFilePath** = "top10highscores.txt"

6.2.1 Detailed Description

All of the classes done by the student team are found in StudentSide namespace.

6.3 Ui Namespace Reference

Defines an interface that reperesents the GameOverDialog's User Interface (Ui).

6.3.1 Detailed Description

Defines an interface that reperesents the GameOverDialog's User Interface (Ui).

Defines an interface that reperesents the statistisDialog's User Interface (Ui).

Defines an interface that reperesents the settingsDialog's User Interface (Ui).

Defines an interface that reperesents the MainMenuDialog's User Interface (Ui).

Defines an interface that reperesents the helpDialogs's User Interface (Ui).

Defines an interface that reperesents the gameWindow's User Interface (Ui).

Chapter 7

Class Documentation

7.1 StudentSide::basicProjectile Class Reference

Defines a class for the projectile that player shoots in the game by pressing space key.

`#include <basicprojectile.h>`

Inherits QObject, and QGraphicsPixmapItem.

Public Slots

- void move ()

  move function is responsible for moving the bullet up in the game map. Also if the bullet encounters a bonus diamond, they both are deleted from the map and from the memory.

Public Member Functions

- basicProjectile (QGraphicsItem *parent=0)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.
- ~basicProjectile ()

  basicProjectile has a basic destructor.
- void setDimensions ()

  setDimensions sets the width and the height of a chosen projectile type.
- void setProjectilePicture ()

  setProjectilePicture sets the the chosen picture as the projectile icon in the game.
- bool removeShootedActors ()

  removeShootedActors is responsible for removing the Nysses and passengers (actors), that are in contact with the bullet, from the scene and memory.
- bool isClose (const Interface::Location &loc, int limit, int xCoord, int yCoord)

  isClose function is a courtesy from Course Side (core/logic) with slight modifications on parameters. It calculates the distance between two items given as a paremeters within specific limit which is also given as a parameter.

Private Attributes

- bool **_fireballChosen** = false
- bool **_missileChosen** = false
- bool **_laserChosen** = false
- int **_projectileVelocity** = 20
- int **_projectileHeight**
- int **_projectileWidth**
- QTimer * **_projectileTimer**
- int **_fireRate** = 50
- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()
- const QPixmap **_fireballPic** = QPixmap(":/images/fireball_16x16.png")
- const QPixmap **_missilePic** = QPixmap(":/images/missile_23x10.png")
- const QPixmap **_laserPic** = QPixmap(":/images/laser_32x32.png")

7.1.1 Detailed Description

Defines a class for the projectile that player shoots in the game by pressing space key.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 basicProjectile()

```
StudentSide::basicProjectile::basicProjectile (
            QGraphicsItem * parent = 0 )
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

basicProjectile is at initialization state.

7.1.3 Member Function Documentation

7.1.3.1 isClose()

```
bool StudentSide::basicProjectile::isClose (
            const Interface::Location & loc,
            int limit,
            int xCoord,
            int yCoord )
```

isClose function is a courtesy from Course Side (core/logic) with slight modifications on parameters. It calculates the distance between two items given as a paremeters within specific limit which is also given as a parameter.

Parameters

loc	is the location of map item 1, limit is the range in which the isClose boolean states true, and xCoord and yCoord are the coordinates of the second comparable item.

Precondition

-

Returns

boolean statement that basically tells if the two items are in close range to each other

Postcondition

A boolean (true or false) is returned which tells if the two items are close to each other in the game map. Exception guarantee: nothrow.

7.1.3.2 move

```
void StudentSide::basicProjectile::move ( )  [slot]
```

move function is responsible for moving the bullet up in the game map. Also if the bullet encounters a bonus diamond, they both are deleted from the map and from the memory.

Precondition

-

Postcondition

Bullet has moved in the map and is deleted from the scene and the memory if it encounters an object in its way or exits the game screen. Exception guarantee: nothrow.

7.1.3.3 removeShootedActors()

```
bool StudentSide::basicProjectile::removeShootedActors ( )
```

removeShootedActors is responsible for removing the Nysses and passengers (actors), that are in contact with the bullet, from the scene and memory.

Precondition

Projectile comes in contact with a Nysse or a passenger.

Postcondition

Shooted Nysse or passenger is deleted from the game map and memory or nothing happens. Exception guarantee: nothrow.

7.1.3.4 setDimensions()

```
void StudentSide::basicProjectile::setDimensions ( )
```

setDimensions sets the width and the height of a chosen projectile type.

Precondition

Chosen projectile type is existent and therefore has some kind of dimensions.

Postcondition

Dimensions for the chosen projectile type are successfully set. Exception guarantee: nothrow.

7.1.3.5 setProjectilePicture()

```
void StudentSide::basicProjectile::setProjectilePicture ( )
```

setProjectilePicture sets the the chosen picture as the projectile icon in the game.

Precondition

Projectile pictures provided are valid and they exist.

Postcondition

Projectile picture for the chosen projectile type is set. Exception guarantee: nothrow.

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

- Game/basicprojectile.h
- Game/basicprojectile.cpp

7.2 StudentSide::bonusItem Class Reference

Defines a class for the other unique object in the game: bonusItem's (small red diamonds) that appear randomly in the game map every 4 seconds. If the player manages to collect these bonus items either by moving into their location or by shooting them, 10 points are added to the player score.

```
#include <bonusitem.h>
```

Inherits QObject, and QGraphicsPixmapItem.

Public Slots

- void move ()

  move function is responsible for moving the bonusItem, that is the bonus diamond, downwards in the game map in a constant speed.

Public Member Functions

- bonusItem (QGraphicsItem ∗parent=0)

  Default constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Public Attributes

- int **currentWidth** = 800
- int **currentHeight** = 600

Private Attributes

- QTimer ∗ **_bonusTimer**
- int **_bonusInterval** = 50
- const QPixmap **_gemPic** = QPixmap(":/images/bonusGem_30x15.png")

7.2.1 Detailed Description

Defines a class for the other unique object in the game: bonusItem's (small red diamonds) that appear randomly in the game map every 4 seconds. If the player manages to collect these bonus items either by moving into their location or by shooting them, 10 points are added to the player score.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 bonusItem()

```
StudentSide::bonusItem::bonusItem (
            QGraphicsItem * parent = 0 )
```

Default constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

bonusItem is at initialization state.

7.2.3 Member Function Documentation

7.2.3.1 move

```
void StudentSide::bonusItem::move ( )  [slot]
```

move function is responsible for moving the bonusItem, that is the bonus diamond, downwards in the game map in a constant speed.

Precondition

-

Postcondition

The bonusItem has moved in the map and is deleted from the scene and the memory if it exits the game screen without being collected by the player. Exception guarantee: nothrow.

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

- Game/bonusitem.h
- Game/bonusitem.cpp

7.3 CourseSide::BusData Struct Reference

Public Attributes

- unsigned int **routeNumber**
- unsigned int **routeId**
- std::string **routeName**
- std::map< QTime, std::shared_ptr< Stop > > **stops**
- std::list< QTime > **schedule**
- std::vector< Interface::Location > **route**
- std::map< QTime, Interface::Location > **timeRoute**
- std::map< QTime, std::pair< Interface::Location, std::shared_ptr< Stop > > > **timeRoute2**

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

- Course/CourseLib/offlinereader.hh

7.4 StudentSide::gameCity Class Reference

Defines a class that inherits from ICity and implements its virtual functions. The class handles functionality regarding the actual city.

```
#include <gamecity.h>
```

Inherits Interface::ICity, and QGraphicsPixmapItem.

Public Types

- using **stopPtr** = std::shared_ptr< Interface::IStop >
- using **actorPtr** = std::shared_ptr< Interface::IActor >

Public Member Functions

- gameCity ()

  Basic constructor for gameCity.
- ~gameCity ()

  gameCity has a basic destructor.
- virtual void setBackground (QImage &basicbackground, QImage &bigbackground)

  setBackground sets the bitmap picture of the game area.
- virtual void setClock (QTime clock)

  setClock sets the time of the game clock.
- virtual void addStop (std::shared_ptr< Interface::IStop > stop)

  addStop adds a stop to the city.
- virtual void startGame ()

  startGame shofts city from init state to the gamestate.
- virtual void addActor (std::shared_ptr< Interface::IActor > newactor)

  addActor adds a new actor to the city.
- virtual void removeActor (std::shared_ptr< Interface::IActor > actor)

  removeActor removes the actor from the city.
- virtual void actorRemoved (std::shared_ptr< Interface::IActor > actor)

  actorRemoved tells the city that actor is removed ingame.
- virtual bool findActor (std::shared_ptr< Interface::IActor > actor) const

  findActor checks if the given actor is in the city.
- virtual void actorMoved (std::shared_ptr< Interface::IActor > actor)

  actorMoved is an operation that is used to tell wether certain actor has moved.
- virtual std::vector< std::shared_ptr< Interface::IActor > > getNearbyActors (Interface::Location loc) const

  getNearbyActors returns actors that are close to given position.
- virtual bool isGameOver () const

  isGameOver tells wether the game is overor not.

Public Attributes

- std::vector< stopPtr > **allStops**
- std::vector< actorPtr > **allActors**
- Interface::Location **startingLoc** = Interface::Location(6700000, 3500000)
- QTime **gameClock**
- bool **gameStateOn** = false
- bool **backgroundSet** = false
- bool **gameClockSet** = false

7.4.1 Detailed Description

Defines a class that inherits from ICity and implements its virtual functions. The class handles functionality regarding the actual city.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 gameCity()

```
StudentSide::gameCity::gameCity ( )
```

Basic constructor for gameCity.

Postcondition

gameCity is at initialization state.

7.4.3 Member Function Documentation

7.4.3.1 actorMoved()

```
void StudentSide::gameCity::actorMoved (
            std::shared_ptr< Interface::IActor > actor )  [virtual]
```

actorMoved is an operation that is used to tell wether certain actor has moved.

Parameters

actor	Actor that has moved.

Precondition

City is in gamestate. Given actor is found in the city.

Postcondition

Exception guarantee: basic.

Implements Interface::ICity.

7.4.3.2 actorRemoved()

```
void StudentSide::gameCity::actorRemoved (
            std::shared_ptr< Interface::IActor > actor )  [virtual]
```

actorRemoved tells the city that actor is removed ingame.

Parameters

actor	Actor that is set removed ingame.

Precondition

City is in gamestate. Given actor is found in the city. Actor has `actor.isRemoved() == true`.

Postcondition

Exception guarantee: strong.

Implements Interface::ICity.

7.4.3.3 addActor()

```
void StudentSide::gameCity::addActor (
            std::shared_ptr< Interface::IActor > newactor )  [virtual]
```

addActor adds a new actor to the city.

Parameters

newactor	actor to be added to the city that fulfills ActorIF.

Precondition

-

Postcondition

Actor is added to the city. Exception guarantee: basic.

Exceptions

GameError	Actor is already in the city.

Implements Interface::ICity.

7.4.3.4 addStop()

```
void StudentSide::gameCity::addStop (
            std::shared_ptr< Interface::IStop > stop )  [virtual]
```

addStop adds a stop to the city.

Parameters

stop	pointer to a stop object.

Precondition

City is in init state.

Postcondition

Stop is added to the city. Exception guarantee: basic

Exceptions

InitError	Stops position is not valid.

Implements Interface::ICity.

7.4.3.5 findActor()

```
bool StudentSide::gameCity::findActor (
            std::shared_ptr< Interface::IActor > actor ) const  [virtual]
```

findActor checks if the given actor is in the city.

Parameters

actor	Actor that that is looked for in the city.

Precondition

-

Returns

Boolean that tells wether the actor is in the city.

Postcondition

Exception guarantee: nothrow.

Implements Interface::ICity.

7.4.3.6 getNearbyActors()

```
std::vector< std::shared_ptr< Interface::IActor > > StudentSide::gameCity::getNearbyActors (
            Interface::Location loc ) const  [virtual]
```

getNearbyActors returns actors that are close to given position.

Parameters

loc	Location for getting the actors close to it.

Precondition

City is in gamestate.

Returns

Vector containing actors close to the location, that pass `getLocation().isClose(loc) == true`.

Postcondition

Exception guarantee: strong.

Implements Interface::ICity.

7.4.3.7 isGameOver()

```
bool StudentSide::gameCity::isGameOver ( ) const  [virtual]
```

isGameOver tells wether the game is overor not.

Precondition

City is in gamestate.

Returns

`true`, if game is over, else ´false´

Postcondition

Exception guarantee: nothrow.

Implements Interface::ICity.

7.4.3.8 removeActor()

```
void StudentSide::gameCity::removeActor (
            std::shared_ptr< Interface::IActor > actor )  [virtual]
```

removeActor removes the actor from the city.

Parameters

actor	Actor to be removed.

Precondition

City is in gamestate.

Postcondition

Actor is removed from the city. Exception guarantee: strong.

Exceptions

GameError	Actor not found in the city

Implements Interface::ICity.

7.4.3.9 setBackground()

```
void StudentSide::gameCity::setBackground (
            QImage & basicbackground,
            QImage & bigbackground )  [virtual]
```

setBackground sets the bitmap picture of the game area.

Parameters

basicbackground	Normal sized picture used as the game area. Bottom left position of the picture in pixelcoordinates can be found out using the offset()-method.
bigbackground	Background of the game that is bigger than normal. Used only if doing Scrolling map-expansion. Bottom left position of the picture in pixelcoordinates can be found out using the offset()-method.

Precondition

City is in init state.

Postcondition

Picture for the game area is set. Exception guarantee: basic.

Exceptions

InitError	Setting the picture was unsuccesful or the picture was invalid.

Implements Interface::ICity.

7.4.3.10 setClock()

```
void StudentSide::gameCity::setClock (
            QTime clock )  [virtual]
```

setClock sets the time of the game clock.

Parameters

clock	Game clock time at the function call.

Precondition

`kello.isValid() == true`.

Postcondition

Time is set. Exception guarantee: nothrow.

Implements Interface::ICity.

7.4.3.11 startGame()

```
void StudentSide::gameCity::startGame ( )  [virtual]
```

startGame shofts city from init state to the gamestate.

Precondition

City is in init state. setBackground() and setClock() have been called.

Postcondition

City is in gamestate. Exception guarantee: nothrow.

Implements Interface::ICity.

7.4.4 Member Data Documentation

7.4.4.1 allActors

```
std::vector< actorPtr> StudentSide::gameCity::allActors
```

Initial value:

```
=
            std::vector<actorPtr>()
```

7.4.4.2 allStops

```
std::vector<stopPtr> StudentSide::gameCity::allStops
```

Initial value:

```
=
            std::vector<stopPtr>()
```

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

- Game/gamecity.h
- Game/gamecity.cpp

7.5 Interface::GameError Class Reference

Exception class that expresses errors ingame.

```
#include <gameerror.hh>
```

Inherits exception.

Public Member Functions

- GameError ()
  Default constructor.
- GameError (const QString &message)
  Constructor.
- virtual ~GameError ()
  Destructor.
- virtual const char * what () const noexcept
  Implements std::exception interface.
- QString giveMessage () const
  giveMessage gives a message that clarifies the cause of the exception.

Private Attributes

- QString **message_**

7.5.1 Detailed Description

Exception class that expresses errors ingame.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 GameError() [1/2]

```
Interface::GameError::GameError ( )
```

Default constructor.

Precondition

-

Postcondition

Creates GameError without message.

7.5.2.2 GameError() [2/2]

```
Interface::GameError::GameError (
            const QString & message )
```

Constructor.

Parameters

message-	a message that clarifies what the exception is.

Precondition

-

Postcondition

Creates GameError that contains the message.

7.5.3 Member Function Documentation

7.5.3.1 giveMessage()

```
QString Interface::GameError::giveMessage ( ) const
```

giveMessage gives a message that clarifies the cause of the exception.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Returns

Message given in the constructor or empty string.

7.5.3.2 what()

```
const char * Interface::GameError::what ( ) const  [virtual], [noexcept]
```

Implements std::exception interface.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Returns

Name of the exception class.

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

- Course/CourseLib/errors/gameerror.hh
- Course/CourseLib/errors/gameerror.cc

7.6 StudentSide::GameOverDialog Class Reference

Defines a QDialog that is shown to the player when the game is over.

```
#include <gameoverdialog.h>
```

Inherits QDialog.

Public Member Functions

- GameOverDialog (QWidget *parent=nullptr)

  Basic constructor for GameOverDialog.
- ~GameOverDialog ()

  GameOverDialog has a basic destructor.
- void setToolTips ()

  setToolTips sets tool tips in the GameOverDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)
- void setPlayerPoints ()

  setPlayerPoints shows the NONSCALED (not scaled by the chosen game duration) points that the player got in the game.
- void initGameData ()

  initGameData initializes the player Statistics and Actors drawn in the game map in preparation for a new game session.

Private Slots

- void **on_gameOverCloseButton_clicked** ()
- void **on_playAgainButton_clicked** ()
- void **on_statsButton_clicked** ()

Private Attributes

- Ui::GameOverDialog * **ui**
- topHighScores * **_highScores**

7.6.1 Detailed Description

Defines a QDialog that is shown to the player when the game is over.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 GameOverDialog()

```
StudentSide::GameOverDialog::GameOverDialog (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor for GameOverDialog.

Postcondition

GameOverDialog is at initialization state.

7.6.3 Member Function Documentation

7.6.3.1 initGameData()

```
void StudentSide::GameOverDialog::initGameData ( )
```

initGameData initializes the player Statistics and Actors drawn in the game map in preparation for a new game session.

Precondition

-

Postcondition

the player Statistics and Actors drawn in the game map are initialized. Exception guarantee: nothrow.

7.6.3.2 setPlayerPoints()

```
void StudentSide::GameOverDialog::setPlayerPoints ( )
```

setPlayerPoints shows the NONSCALED (not scaled by the chosen game duration) points that the player got in the game.

Precondition

-

Postcondition

Player is shown the points in the topmost header in GameOverDialog. Exception guarantee: nothrow.

7.6.3.3 setToolTips()

```
void StudentSide::GameOverDialog::setToolTips ( )
```

setToolTips sets tool tips in the GameOverDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)

Precondition

-

Postcondition

ToolTips are shown to the user of the software when hovering mouse above buttons, labels or other items of the GUI. Exception guarantee: nothrow.

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

- Game/gameoverdialog.h
- Game/gameoverdialog.cpp

7.7 StudentSide::gameStatistics Class Reference

Defines a class that inherits from IStatistics and implements its virtual functions. In addition class has some own functions and attributes. Class is responsible for the statistical bookkeeping for all the relevant statistics that the player generates ingame.

`#include <gamestatistics.h>`

Inherits Interface::IStatistics.

Public Member Functions

- gameStatistics ()

  Basic constructor for gameStatistics.
- virtual int givePoints () const

  givePoints returns current score in the game.
- virtual void passengerDied (int num)

  passengerDied notifies, that the passanger is dead.
- virtual void morePassengers (int num)

  morePassengers notifies, that more passangers are added to the game.
- virtual void nysseRemoved ()

  nysseRemoved notifies, that the nysse is removed ingame.
- virtual void newNysse ()

  newNysse notifies, that a new nysse is added to the game.
- virtual void nysseLeft ()

  nysseLeft notifies, that a nysse has left the game.
- void addPoints ()

  addPoints function increases the player points by 10.
- void addCollectedDiamond ()

  addCollectedDiamond function increases the collected diamonds by 1.
- void passengerLeft ()

  passengerLeft function increases passengers that left ingame count by 1.
- void actorMoved ()

  actorMoved function increases the actors moved ingame count by 1.
- void initAllValues ()

  initAllValues function initializes all the gameStatistics class public attributes to zero.
- int giveCollectedDiamonds ()

  giveCollectedDiamonds returns the amount of collected diamonds by the player at the time when the function is called.
- int giveDestroyedNysses ()

  giveDestroyedNysses returns the amount of destroyed Nysse-buses by the player at the time when the function is called.
- int giveDestroyedPassengers ()

  giveDestroyedPassengers returns the amount of destroyed passengers by the player at the time when the function is called.

Public Attributes

- int **playerPoints** = 0
- int **passengersDead** = 0
- int **removedNysses** = 0
- int **totalPassengers** = 0
- int **totalNysses** = 0
- int **leftNysses** = 0
- int **leftPassengers** = 0
- int **collectedDiamonds** = 0
- int **movedActorsAmount** = 0

7.7.1 Detailed Description

Defines a class that inherits from IStatistics and implements its virtual functions. In addition class has some own functions and attributes. Class is responsible for the statistical bookkeeping for all the relevant statistics that the player generates ingame.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 gameStatistics()

```
StudentSide::gameStatistics::gameStatistics ( )
```

Basic constructor for gameStatistics.

Postcondition

gameStatistics is at initialization state.

7.7.3 Member Function Documentation

7.7.3.1 actorMoved()

```
void StudentSide::gameStatistics::actorMoved ( )
```

actorMoved function increases the actors moved ingame count by 1.

Precondition

-

Postcondition

Actors moved ingame count has been increased by 1. Exception guarantee: nothrow.

7.7.3.2 addCollectedDiamond()

```
void StudentSide::gameStatistics::addCollectedDiamond ( )
```

addCollectedDiamond function increases the collected diamonds by 1.

Precondition

-

Postcondition

Player's collected diamond count has been increased by 1. Exception guarantee: nothrow.

7.7.3.3 addPoints()

```
void StudentSide::gameStatistics::addPoints ( )  [inline]
```

addPoints function increases the player points by 10.

Precondition

-

Postcondition

Player's points have been increased by 10. Exception guarantee: nothrow.

7.7.3.4 giveCollectedDiamonds()

```
int StudentSide::gameStatistics::giveCollectedDiamonds ( )  [inline]
```

giveCollectedDiamonds returns the amount of collected diamonds by the player at the time when the function is called.

Precondition

-

Returns

Integer amount of collected diamonds by the player.

Postcondition

The amount of collected diamonds has been returned. Exception guarantee: nothrow.

7.7.3.5 giveDestroyedNysses()

```
int StudentSide::gameStatistics::giveDestroyedNysses ( )  [inline]
```

giveDestroyedNysses returns the amount of destroyed Nysse-buses by the player at the time when the function is called.

Precondition

-

Returns

Integer amount of destroyed Nysses by the player.

Postcondition

The amount of destroyed Nysses has been returned. Exception guarantee: nothrow.

7.7.3.6 giveDestroyedPassengers()

```
int StudentSide::gameStatistics::giveDestroyedPassengers ( )  [inline]
```

giveDestroyedPassengers returns the amount of destroyed passengers by the player at the time when the function is called.

Precondition

-

Returns

Integer amount of destroyed passengers by the player.

Postcondition

The amount of destroyed passengers has been returned. Exception guarantee: nothrow.

7.7.3.7 givePoints()

```
int StudentSide::gameStatistics::givePoints ( ) const  [virtual]
```

givePoints returns current score in the game.

Precondition

-

Returns

score

Postcondition

Exception guarantee: nothrow

7.7.3.8 initAllValues()

```
void StudentSide::gameStatistics::initAllValues ( )
```

initAllValues function initializes all the gameStatistics class public attributes to zero.

Precondition

-

Postcondition

gameStatistics class public attributes have all been set to zero. Exception guarantee: nothrow.

7.7.3.9 morePassengers()

```
void StudentSide::gameStatistics::morePassengers (
            int num )  [virtual]
```

morePassengers notifies, that more passangers are added to the game.

Parameters

num	how many new passangers are added.

Precondition

num $>$ 0

Postcondition

Exception guarantee: strong

Exceptions

GameError	number given as a paremeter is negative.

Implements Interface::IStatistics.

7.7.3.10 newNysse()

```
void StudentSide::gameStatistics::newNysse ( )  [virtual]
```

newNysse notifies, that a new nysse is added to the game.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Implements Interface::IStatistics.

7.7.3.11 nysseLeft()

```
void StudentSide::gameStatistics::nysseLeft ( )  [virtual]
```

nysseLeft notifies, that a nysse has left the game.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Implements Interface::IStatistics.

7.7.3.12 nysseRemoved()

```
void StudentSide::gameStatistics::nysseRemoved ( )  [virtual]
```

nysseRemoved notifies, that the nysse is removed ingame.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Implements Interface::IStatistics.

7.7.3.13 passengerDied()

```
void StudentSide::gameStatistics::passengerDied (
            int num )  [virtual]
```

passengerDied notifies, that the passanger is dead.

Parameters

num	how many passangers eliminated.

Precondition

num $>$ 0

Postcondition

Exception guarantee: strong

Exceptions

GameError	number given as a paremeter is negative.

7.7.3.14 passengerLeft()

```
void StudentSide::gameStatistics::passengerLeft ( )
```

passengerLeft function increases passengers that left ingame count by 1.

Precondition

-

Postcondition

Passenger left ingame count has been increased by 1. Exception guarantee: nothrow.

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

- Game/gamestatistics.h
- Game/gamestatistics.cpp

7.8 StudentSide::gameWindow Class Reference

Defines a class for the MainWindow where the actual playing happens. It has a game map background which depicts Tampere City Center. Game Window also shows the chosen player which can be moved with arrow keys. Shooting happens by pressing space bar. Also the CourseSide integration is visualized in the game window: Nysses and passengers move in real time on the map.

`#include <gamewindow.h>`

Inherits QMainWindow.

Public Slots

- void updateCountDown ()

  updateCountDown substracts seconds from the game clock in 1000 ms intervals. Furthermore, it keeps track of time and shows GameOverDialog when the clock is zero.

Public Member Functions

- gameWindow (QWidget *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.
- ~gameWindow ()

  gameWindow has a basic destructor.
- void resizeEvent (QResizeEvent *event)

  resizeEvent is responsible for the resizing of the gameWindow and reacting to it appropriately.
- void setPicture (QImage img)

  setPicture sets the background picture for the game and scales it to the right dimensions if needed.
- void setLCDStyle ()

  setLCDStyle styles the gameWindows top-panel QLCDNumber widgets.
- void spawnBonusItem ()

  spawnBonusItem creates a new raw pointer to the bonusItem object and adds it to the QGraphicsScene.
- void addDataToLCD ()

  addDataToLCD is responsible for showing and updating the time-left count-down clock and player points on the QLCDNumber widgets.
- void setGameTime ()

  setGameTime checks the player's game duration setting set in the settingsDialog and sets the game duration according to that.
- void screenFrameUpdate ()

  screenFrameUpdate update QGraphicsView's viewport in predertimed timeout intervals.
- void stopTimers ()

  stopTimers stops all the QTimers used in gameWindow class.
- std::vector< int > getAvailableSize ()

  getAvailableSize returns player's current available screen size.

Public Attributes

- int **screenWidth** = 800
- int **screenHeight** = 600

Private Attributes

- Ui::gameWindow ∗ **ui**
- QTimer ∗ **_mainTimer**
- QTimer ∗ **_bonusTimer**
- QTimer ∗ **_gameTimer**
- QTimer ∗ **_labelTimer**
- MainMenuDialog ∗ **_mainMenu**
- QGraphicsScene ∗ **_scene**
- initGame ∗ **_newGame**
- Player ∗ **_player**
- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()
- bool **_largeMode** = false
- int **_gameDuration**
- int **_frameRate** = 20
- int **_interval** = 500
- int **_countDownInterval** = 1000
- int **_spawnBonusInterval** = 4000
- const QImage **_bkgndSmall** = QImage(":/offlinedata/offlinedata/kartta_pieni_500x500.png")
- const QImage **_bkgndBig** = QImage(":/offlinedata/offlinedata/kartta_iso_1095x592.png")

7.8.1 Detailed Description

Defines a class for the MainWindow where the actual playing happens. It has a game map background which depicts Tampere City Center. Game Window also shows the chosen player which can be moved with arrow keys. Shooting happens by pressing space bar. Also the CourseSide integration is visualized in the game window: Nysses and passengers move in real time on the map.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 gameWindow()

```
StudentSide::gameWindow::gameWindow (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.

Postcondition

gameWindow is at initialization state.

7.8.3 Member Function Documentation

7.8.3.1 addDataToLCD()

```
void StudentSide::gameWindow::addDataToLCD ( )
```

addDataToLCD is responsible for showing and updating the time-left count-down clock and player points on the QLCDNumber widgets.

Precondition

The QLCDNumber widgets exist.

Postcondition

The correct values have been updated to the gameWindow so that the player sees them. Exception guarantee: nothrow.

7.8.3.2 getAvailableSize()

```
std::vector< int > StudentSide::gameWindow::getAvailableSize ( )
```

getAvailableSize returns player's current available screen size.

Precondition

The game has started and gameWindow is active.

Returns

Vector containing the width (index 0) and the height (index 1) of the available screen.

Postcondition

Exception guarantee: nothrow.

7.8.3.3 resizeEvent()

```
void StudentSide::gameWindow::resizeEvent (
            QResizeEvent * event )
```

resizeEvent is responsible for the resizing of the gameWindow and reacting to it appropriately.

Parameters

Raw	pointer to the QResizeEvent.

Precondition

gameWindow's size is is in initialization state (800x600 pixels).

Postcondition

The new window size has been set and all the gamemap items scaled accordingly. Exception guarantee: nothrow.

7.8.3.4 screenFrameUpdate()

```
void StudentSide::gameWindow::screenFrameUpdate ( )
```

screenFrameUpdate update QGraphicsView's viewport in predertimed timeout intervals.

Precondition

QGraphicsView has been created.

Postcondition

Cumulated changes in the QGraphicsView's viewport have been updated. Exception guarantee: nothrow.

7.8.3.5 setGameTime()

```
void StudentSide::gameWindow::setGameTime ( )
```

setGameTime checks the player's game duration setting set in the settingsDialog and sets the game duration according to that.

Precondition

-

Postcondition

The correct game duration has been set for the game according to the player's choices. Exception guarantee: nothrow.

7.8.3.6 setLCDStyle()

```
void StudentSide::gameWindow::setLCDStyle ( )
```

setLCDStyle styles the gameWindows top-panel QLCDNumber widgets.

Precondition

The QLCDNumber widgets exist.

Postcondition

The desired styles have been set. Exception guarantee: nothrow.

7.8.3.7 setPicture()

```
void StudentSide::gameWindow::setPicture (
            QImage img )
```

setPicture sets the background picture for the game and scales it to the right dimensions if needed.

Parameters

QImage	picture of the big gamemap which is a courtesy from the Course Side.

Precondition

-

Postcondition

The backround picture of Tampere City Center has been set. Exception guarantee: nothrow.

7.8.3.8 spawnBonusItem()

```
void StudentSide::gameWindow::spawnBonusItem ( )
```

spawnBonusItem creates a new raw pointer to the bonusItem object and adds it to the QGraphicsScene.

Precondition

-

Postcondition

A new bonusItem has been created and added to the scene. Exception guarantee: nothrow.

7.8.3.9 stopTimers()

```
void StudentSide::gameWindow::stopTimers ( )
```

stopTimers stops all the QTimers used in gameWindow class.

Precondition

-

Postcondition

All the QTimers have been stopped. Exception guarantee: nothrow.

7.8.3.10 updateCountDown

```
void StudentSide::gameWindow::updateCountDown ( )  [slot]
```

updateCountDown substracts seconds from the game clock in 1000 ms intervals. Furthermore, it keeps track of time and shows GameOverDialog when the clock is zero.

Precondition

-

Postcondition

One second has been substracted from game time and/or the game is stopped and GameOverDialog is created and shown to the player. Exception guarantee: nothrow.

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

- Game/gamewindow.h
- Game/gamewindow.cpp

7.9 StudentSide::helpDialog Class Reference

Defines a QDialog which offers help to a new player who possibly isn't aware of the game rules, settings or controls etc. The Dialog itself is accessible from MainMenu's "Help" -button.

```
#include <helpdialog.h>
```

Inherits QDialog.

Public Member Functions

- helpDialog (QWidget *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.
- ~helpDialog ()

  helpDialog has a basic destructor.
- void setActionTips ()

  setActionTips sets tool tip in the helpDialog's GUI to guide the player (shown when user hovers mouse on top of the red close button on the top right corner).

Private Slots

- void **on_closeHelpButton_clicked** ()

Private Attributes

- Ui::helpDialog * **ui**

7.9.1 Detailed Description

Defines a QDialog which offers help to a new player who possibly isn't aware of the game rules, settings or controls etc. The Dialog itself is accessible from MainMenu's "Help" -button.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 helpDialog()

```
StudentSide::helpDialog::helpDialog (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.

Postcondition

helpDialog is at initialization state.

7.9.3 Member Function Documentation

7.9.3.1 setActionTips()

```
void StudentSide::helpDialog::setActionTips ( )
```

setActionTips sets tool tip in the helpDialog's GUI to guide the player (shown when user hovers mouse on top of the red close button on the top right corner).

Precondition

-

Postcondition

ToolTip is shown to the user of the software when hovering mouse above GUI's red close button. Exception guarantee: nothrow.

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

- Game/helpdialog.h
- Game/helpdialog.cpp

7.10 Interface::IActor Class Reference

ActorIF is an interface, which every single actor moving in the game implements.

`#include <iactor.hh>`

Inherited by Interface::IPassenger `[virtual]`, and Interface::IVehicle `[virtual]`.

Public Member Functions

- IActor ()=default

  Default constructor for the Interface (For documentation).
- virtual ~IActor ()=default

  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual Location giveLocation () const =0

  giveLocation returns the location of the actor.
- virtual void move (Location loc)=0

  move-method moves the actor to given location.
- virtual bool isRemoved () const =0

  isRemoved tells if the actor is removed ingame.
- virtual void remove ()=0

  remove marks the actor as removed.

7.10.1 Detailed Description

ActorIF is an interface, which every single actor moving in the game implements.

If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.10.2 Constructor & Destructor Documentation

7.10.2.1 IActor()

```
Interface::IActor::IActor ( )  [default]
```

Default constructor for the Interface (For documentation).

Postcondition

Actor isn't in removed-state by default. (Toimija ei alussa ole tuhottu-tilassa.)

7.10.3 Member Function Documentation

7.10.3.1 giveLocation()

```
virtual Location Interface::IActor::giveLocation ( ) const  [pure virtual]
```

giveLocation returns the location of the actor.

Precondition

-

Returns

Actors location.

Postcondition

Exception guarantee: strong.

Exceptions

GameError	- actor wasn't given a location.

Implemented in CourseSide::Nysse, and CourseSide::Passenger.

7.10.3.2 isRemoved()

```
virtual bool Interface::IActor::isRemoved ( ) const  [pure virtual]
```

isRemoved tells if the actor is removed ingame.

Precondition

-

Returns

`true`, if actor is removed ingame, otherwise `false`.

Postcondition

Exception guarantee: nothrow.

Implemented in CourseSide::Nysse, and CourseSide::Passenger.

7.10.3.3 move()

```
virtual void Interface::IActor::move (
            Location loc )  [pure virtual]
```

move-method moves the actor to given location.

Parameters

loc	Actors new location.

Precondition

-

Postcondition

Actors location is sij. Excaption guarantee: strong.

Exceptions

GameError	Location is not possible.

Implemented in CourseSide::Nysse, and CourseSide::Passenger.

7.10.3.4 remove()

```
virtual void Interface::IActor::remove ( )  [pure virtual]
```

remove marks the actor as removed.

Precondition

Actor is not removed already.

Postcondition

Actor is removed, after this isRemoved() returns `true`. Exception guarantee: basic.

Implemented in CourseSide::Nysse, and CourseSide::Passenger.

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

- Course/CourseLib/interfaces/iactor.hh

7.11 Interface::ICity Class Reference

CityIF is an interface that every city in the game must fulfill. Kaupunki.

`#include <icity.hh>`

Inherited by StudentSide::gameCity.

Public Member Functions

- ICity ()=default
  Default constructor of the interface (For documentation)
- virtual ~ICity ()=default
  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual void setBackground (QImage &basicbackground, QImage &bigbackground)=0
  setBackground sets the bitmap picture of the game area.
- virtual void setClock (QTime clock)=0
  setClock sets the time of the game clock.
- virtual void addStop (std::shared_ptr< IStop > stop)=0
  addStop adds a stop to the city.
- virtual void startGame ()=0
  startGame shofts city from init state to the gamestate.
- virtual void addActor (std::shared_ptr< IActor > newactor)=0
  addActor adds a new actor to the city.
- virtual void removeActor (std::shared_ptr< IActor > actor)=0
  removeActor removes the actor from the city.
- virtual void actorRemoved (std::shared_ptr< IActor > actor)=0
  actorRemoved tells the city that actor is removed ingame.
- virtual bool findActor (std::shared_ptr< IActor > actor) const =0
  findActor checks if the given actor is in the city.
- virtual void actorMoved (std::shared_ptr< IActor > actor)=0
  actorMoved is an operation that is used to tell wether certain actor has moved.
- virtual std::vector< std::shared_ptr< IActor > > getNearbyActors (Location loc) const =0
  getNearbyActors returns actors that are close to given position.
- virtual bool isGameOver () const =0
  isGameOver tells wether the game is overor not.

7.11.1 Detailed Description

CityIF is an interface that every city in the game must fulfill. Kaupunki.

If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.11.2 Constructor & Destructor Documentation

7.11.2.1 ICity()

```
Interface::ICity::ICity ( )  [default]
```

Default constructor of the interface (For documentation)

Postcondition

City is at initialization state.

7.11.3 Member Function Documentation

7.11.3.1 actorMoved()

```
virtual void Interface::ICity::actorMoved (
            std::shared_ptr< IActor > actor )  [pure virtual]
```

actorMoved is an operation that is used to tell wether certain actor has moved.

Parameters

actor	Actor that has moved.

Precondition

City is in gamestate. Given actor is found in the city.

Postcondition

Exception guarantee: basic.

Implemented in StudentSide::gameCity.

7.11.3.2 actorRemoved()

```
virtual void Interface::ICity::actorRemoved (
            std::shared_ptr< IActor > actor )  [pure virtual]
```

actorRemoved tells the city that actor is removed ingame.

Parameters

actor	Actor that is set removed ingame.

Precondition

City is in gamestate. Given actor is found in the city. Actor has `actor.isRemoved() == true`.

Postcondition

Exception guarantee: strong.

Implemented in StudentSide::gameCity.

7.11.3.3 addActor()

```
virtual void Interface::ICity::addActor (
            std::shared_ptr< IActor > newactor )  [pure virtual]
```

addActor adds a new actor to the city.

Parameters

newactor	actor to be added to the city that fulfills ActorIF.

Precondition

-

Postcondition

Actor is added to the city. Exception guarantee: basic.

Exceptions

GameError	Actor is already in the city.

Implemented in StudentSide::gameCity.

7.11.3.4 addStop()

```
virtual void Interface::ICity::addStop (
            std::shared_ptr< IStop > stop )  [pure virtual]
```

addStop adds a stop to the city.

Parameters

stop	pointer to a stop object.

Precondition

City is in init state.

Postcondition

Stop is added to the city. Exception guarantee: basic

Exceptions

InitError	Stops position is not valid.

Implemented in StudentSide::gameCity.

7.11.3.5 findActor()

```
virtual bool Interface::ICity::findActor (
            std::shared_ptr< IActor > actor ) const  [pure virtual]
```

findActor checks if the given actor is in the city.

Parameters

actor	Actor that that is looked for in the city.

Precondition

-

Returns

Boolean that tells wether the actor is in the city.

Postcondition

Exception guarantee: nothrow.

Implemented in StudentSide::gameCity.

7.11.3.6 getNearbyActors()

```
virtual std::vector<std::shared_ptr<IActor> > Interface::ICity::getNearbyActors (
            Location loc ) const  [pure virtual]
```

getNearbyActors returns actors that are close to given position.

Parameters

loc	Location for getting the actors close to it.

Precondition

City is in gamestate.

Returns

Vector containing actors close to the location, that pass `getLocation().isClose(loc) == true`.

Postcondition

Exception guarantee: strong.

Implemented in StudentSide::gameCity.

7.11.3.7 isGameOver()

```
virtual bool Interface::ICity::isGameOver ( ) const  [pure virtual]
```

isGameOver tells wether the game is overor not.

Precondition

City is in gamestate.

Returns

`true`, if game is over, else ´false´

Postcondition

Exception guarantee: nothrow.

Implemented in StudentSide::gameCity.

7.11.3.8 removeActor()

```
virtual void Interface::ICity::removeActor (
            std::shared_ptr< IActor > actor )  [pure virtual]
```

removeActor removes the actor from the city.

Parameters

actor	Actor to be removed.

Precondition

City is in gamestate.

Postcondition

Actor is removed from the city. Exception guarantee: strong.

Exceptions

GameError	Actor not found in the city

Implemented in StudentSide::gameCity.

7.11.3.9 setBackground()

```
virtual void Interface::ICity::setBackground (
            QImage & basicbackground,
            QImage & bigbackground )  [pure virtual]
```

setBackground sets the bitmap picture of the game area.

Parameters

basicbackground	Normal sized picture used as the game area. Bottom left position of the picture in pixelcoordinates can be found out using the offset()-method.
bigbackground	Background of the game that is bigger than normal. Used only if doing Scrolling map-expansion. Bottom left position of the picture in pixelcoordinates can be found out using the offset()-method.

Precondition

City is in init state.

Postcondition

Picture for the game area is set. Exception guarantee: basic.

Exceptions

InitError	Setting the picture was unsuccesful or the picture was invalid.

Implemented in StudentSide::gameCity.

7.11.3.10 setClock()

```
virtual void Interface::ICity::setClock (
            QTime clock )  [pure virtual]
```

setClock sets the time of the game clock.

Parameters

clock	Game clock time at the function call.

Precondition

`kello.isValid() == true`.

Postcondition

Time is set. Exception guarantee: nothrow.

Implemented in StudentSide::gameCity.

7.11.3.11 startGame()

```
virtual void Interface::ICity::startGame ( )  [pure virtual]
```

startGame shofts city from init state to the gamestate.

Precondition

City is in init state. setBackground() and setClock() have been called.

Postcondition

City is in gamestate. Exception guarantee: nothrow.

Implemented in StudentSide::gameCity.

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

- Course/CourseLib/interfaces/icity.hh

7.12 Interface::InitError Class Reference

Exception class that expresses errors during the initialization of the game.

`#include <initerror.hh>`

Inherits exception.

Public Member Functions

- InitError ()
  Default constructor.
- InitError (const QString &message)
  Constructor.
- virtual ~InitError ()
  Destructor.
- virtual const char * what () const noexcept
  Implements std::exception interface.
- QString giveMessage () const
  giveMessage gives a message that clarifies the cause of the exception.

Private Attributes

- QString **message_**

7.12.1 Detailed Description

Exception class that expresses errors during the initialization of the game.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 InitError() [1/2]

```
Interface::InitError::InitError ( )
```

Default constructor.

Postcondition

Creates InitError without a message.

7.12.2.2 InitError() [2/2]

```
Interface::InitError::InitError (
            const QString & message )
```

Constructor.

Parameters

message	a message that clarifies the error.

Precondition

-

Postcondition

Creates InitError that constains the message.

7.12.3 Member Function Documentation

7.12.3.1 giveMessage()

```
QString Interface::InitError::giveMessage ( ) const
```

giveMessage gives a message that clarifies the cause of the exception.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Returns

Message given in the constructor or empty string.

7.12.3.2 what()

```
const char * Interface::InitError::what ( ) const  [virtual], [noexcept]
```

Implements std::exception interface.

Precondition

-

Postcondition

Exception guarantee: nothrow.

Returns

Name of the exception class.

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

- Course/CourseLib/errors/initerror.hh
- Course/CourseLib/errors/initerror.cc

7.13 StudentSide::initGame Class Reference

Defines a Class in which the CourseSide integration mostly happens. The class is also responsible for moving and showing all the actors offered by CourseSide. Furthermore, CreateGame -function is implemented here.

`#include <initgame.h>`

Inherits QObject, and QGraphicsPixmapItem.

Public Member Functions

- initGame ()

  Basic constructor for initGame.
- ~initGame ()

  initGame has a basic destructor.
- void drawStops (std::shared_ptr< gameCity > currCity, QGraphicsScene *scene)

  drawStops adds all the bus stops that fit in the game mip to the QGraphicsScene and also set the picture for each stop (red flag).
- void drawActorItems (QGraphicsScene *scene)

  drawActorItems adds all the Nysse-buses and passengers that fit in the game mip to the QGraphicsScene and also set the picture for each Nysse and passenger.
- void readActors (std::shared_ptr< gameCity > currCity)

  readActors separates the actors for one another: Nysses and passengers are being pushed to their own vectors.
- void setActorPic (QPixmap pic, QGraphicsPixmapItem *actorItem, int w, int h)

  setActorPic sets an icon picture for desired actor.
- void setActorPos (int newX, int newY, QGraphicsPixmapItem *actorItem)

  setActorPos sets the position of the actor (x- and y-coordinates).
- void initLogic (QGraphicsScene *scene)

  initLogic initializes the Logic so that that the CourseSide integration would be possible.
- std::shared_ptr< gameCity > createGame ()

  createGame creates the games city (gameCity object) and returns a shared pointer to it.
- void moveSceneActors ()

  moveSceneActors moves the actors in the scene (sets new positions according to their current Location).
- void endGame ()

  endGame calls the gameCity's function isGameOver.

Public Attributes

- int **screenWidth** = 800
- int **screenHeight** = 600

Private Attributes

- QTimer * **_updateTimer**
- std::map< std::shared_ptr< Interface::IStop >, QGraphicsPixmapItem * > **_stopMap**
- std::map< std::shared_ptr< Interface::IActor >, QGraphicsPixmapItem * > **_nysseMap**
- std::map< std::shared_ptr< Interface::IActor >, QGraphicsPixmapItem * > **_passengerMap**
- std::map< std::shared_ptr< Interface::IActor >, QGraphicsPixmapItem * > **_actorsMap**
- std::vector< std::shared_ptr< Interface::IActor > > **_nysseVec**
- std::vector< std::shared_ptr< Interface::IActor > > **_passengerVec**
- std::vector< std::shared_ptr< Interface::IStop > > **_stopsVec**
- std::vector< std::shared_ptr< Interface::IActor > > **_actorsVec**
- std::shared_ptr< gameCity > **_newCity**
- std::shared_ptr< CourseSide::Logic > **_gameLogic**
- const QPixmap **_stopPic** = QPixmap(":/images/stop_15x25.png")
- const QPixmap **_busPic** = QPixmap(":/images/bus_10x20.png")
- const QPixmap **_passengerPic** = QPixmap(":/images/passenger_20x15.png")

7.13.1 Detailed Description

Defines a Class in which the CourseSide integration mostly happens. The class is also responsible for moving and showing all the actors offered by CourseSide. Furthermore, CreateGame -function is implemented here.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 initGame()

```
StudentSide::initGame::initGame ( )
```

Basic constructor for initGame.

Postcondition

initGame is at initialization state.

7.13.3 Member Function Documentation

7.13.3.1 createGame()

```
std::shared_ptr< gameCity > StudentSide::initGame::createGame ( )
```

createGame creates the games city (gameCity object) and returns a shared pointer to it.

Precondition

-

Returns

pointer to the created city which is in initialization state.

Postcondition

Exception guarantee: basic.

7.13.3.2 drawActorItems()

```
void StudentSide::initGame::drawActorItems (
            QGraphicsScene * scene )
```

drawActorItems adds all the Nysse-buses and passengers that fit in the game mip to the QGraphicsScene and also set the picture for each Nysse and passenger.

Parameters

Raw	pointer to the game's main QGraphicsScene.

Precondition

QGraphicsScene has been created and is active.

Postcondition

Nysses and passengers have been added to the game map and they are depicted with a proper icon. Exception guarantee: nothrow.

7.13.3.3 drawStops()

```
void StudentSide::initGame::drawStops (
            std::shared_ptr< gameCity > currCity,
            QGraphicsScene * scene )
```

drawStops adds all the bus stops that fit in the game mip to the QGraphicsScene and also set the picture for each stop (red flag).

Parameters

Shared	pointer to the gameCity and a raw pointer to the game's main QGraphicsScene.

Precondition

QGraphicsScene has been created and is active.

Postcondition

Stops have been added to the game map and they are depicted with a red flag. Exception guarantee: nothrow.

7.13.3.4 endGame()

```
void StudentSide::initGame::endGame ( )
```

endGame calls the gameCity's function isGameOver.

Precondition

-

Postcondition

ICity's function isGameOver has been called. Exception guarantee: nothrow.

7.13.3.5 initLogic()

```
void StudentSide::initGame::initLogic (
            QGraphicsScene * scene )
```

initLogic initializes the Logic so that that the CourseSide integration would be possible.

Parameters

Raw	pointer to the game's main QGraphicsScene.

Precondition

QGraphicsScene has been created and is active.

Postcondition

New Logic object has been created and properly initialized and finally finalizeGameStart method of Logic is being called to start the back-end functionality. Exception guarantee: nothrow.

7.13.3.6 moveSceneActors()

```
void StudentSide::initGame::moveSceneActors ( )
```

moveSceneActors moves the actors in the scene (sets new positions according to their current Location).

Precondition

-

Postcondition

If actor has moved (it has a different Location than before), then the method updates the actor's new position to the scene. Exception guarantee: nothrow.

7.13.3.7 readActors()

```
void StudentSide::initGame::readActors (
            std::shared_ptr< gameCity > currCity )
```

readActors separates the actors for one another: Nysses and passengers are being pushed to their own vectors.

Parameters

Shared	pointer to the gameCity.

Precondition

-

Postcondition

Nysses and passengers are added into their own separate vectors. Exception guarantee: nothrow.

7.13.3.8 setActorPic()

```
void StudentSide::initGame::setActorPic (
            QPixmap pic,
            QGraphicsPixmapItem * actorItem,
            int w,
            int h )
```

setActorPic sets an icon picture for desired actor.

Parameters

QPixmap	pic (picture to be set), QGraphicsPixmapItem raw pointer to the actor, width and the height of the to-be-set picture.

Precondition

-

Postcondition

Picture for the actor has been set. Exception guarantee: nothrow.

7.13.3.9 setActorPos()

```
void StudentSide::initGame::setActorPos (
            int newX,
            int newY,
            QGraphicsPixmapItem * actorItem )
```

setActorPos sets the position of the actor (x- and y-coordinates).

Parameters

newX	x-coordinate, newY y-coordinate and QGraphicsPixmapItem raw pointer to the actor whose position we want to set.

Precondition

-

Postcondition

Position for the actor has been set. Exception guarantee: nothrow.

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

- Game/initgame.h
- Game/initgame.cpp

7.14 Interface::IPassenger Class Reference

PassengerIF is an interface which every passenger in game implements.

`#include <ipassenger.hh>`

Inherits Interface::IActor.

Inherited by CourseSide::Passenger.

Public Member Functions

- IPassenger ()=default

  Default constructor for the Interface.(For documentation).
- virtual ~IPassenger ()=default

  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual bool isInVehicle () const =0

  isInVehicle tells if passenger is in any vehicle currently.
- virtual std::shared_ptr< IVehicle > getVehicle () const =0

  getVehicle returns the vehicle passenger is in.

7.14.1 Detailed Description

PassengerIF is an interface which every passenger in game implements.

PassengerIF is inherited from ActorIF interface. If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.14.2 Constructor & Destructor Documentation

7.14.2.1 IPassenger()

```
Interface::IPassenger::IPassenger ( )  [default]
```

Default constructor for the Interface.(For documentation).

Postcondition

Passenger is not in any vehicle by default. Passengers destination is set.

7.14.3 Member Function Documentation

7.14.3.1 getVehicle()

```
virtual std::shared_ptr<IVehicle> Interface::IPassenger::getVehicle ( ) const  [pure virtual]
```

getVehicle returns the vehicle passenger is in.

Precondition

-

Returns

Vechicle where passenger is in. Empty pointer if passenger is not in any vehicle.

Postcondition

Exception guarantee: nothrow.

Implemented in CourseSide::Passenger.

7.14.3.2 isInVehicle()

```
virtual bool Interface::IPassenger::isInVehicle ( ) const  [pure virtual]
```

isInVehicle tells if passenger is in any vehicle currently.

Precondition

-

Returns

Boolean, tells wether passenger is in any vehicle.

Postcondition

Exception guarantee: nothrow

Implemented in CourseSide::Passenger.

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

- Course/CourseLib/interfaces/ipassenger.hh

7.15 Interface::IStatistics Class Reference

StatisticsIF is an interface, which defines an object that manages scoring statistics.

`#include <istatistics.hh>`

Inherited by StudentSide::gameStatistics.

Public Member Functions

- IStatistics ()=default

  Default constructor for the Interface.(For documentation).
- virtual ~IStatistics ()=default

  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual void morePassengers (int num)=0

  morePassengers notifies, that more passangers are added to the game.
- virtual void nysseRemoved ()=0

  nysseRemoved notifies, that the nysse is removed ingame.
- virtual void newNysse ()=0

  newNysse notifies, that a new nysse is added to the game.
- virtual void nysseLeft ()=0

  nysseLeft notifies, that a nysse has left the game.

7.15.1 Detailed Description

StatisticsIF is an interface, which defines an object that manages scoring statistics.

If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.15.2 Constructor & Destructor Documentation

7.15.2.1 IStatistics()

```
Interface::IStatistics::IStatistics ( )  [default]
```

Default constructor for the Interface.(For documentation).

Postcondition

Scores are reset by default.

7.15.3 Member Function Documentation

7.15.3.1 morePassengers()

```
virtual void Interface::IStatistics::morePassengers (
            int num )  [pure virtual]
```

morePassengers notifies, that more passangers are added to the game.

Parameters

num	how many new passangers are added.

Precondition

num $>$ 0

Postcondition

Exception guarantee: strong

Implemented in StudentSide::gameStatistics.

7.15.3.2 newNysse()

```
virtual void Interface::IStatistics::newNysse ( )  [pure virtual]
```

newNysse notifies, that a new nysse is added to the game.

Precondition

-

Postcondition

Exception guarantee: strong

Implemented in StudentSide::gameStatistics.

7.15.3.3 nysseLeft()

```
virtual void Interface::IStatistics::nysseLeft ( )  [pure virtual]
```

nysseLeft notifies, that a nysse has left the game.

Precondition

-

Postcondition

Exception guarantee: strong

Implemented in StudentSide::gameStatistics.

7.15.3.4 nysseRemoved()

```
virtual void Interface::IStatistics::nysseRemoved ( )  [pure virtual]
```

nysseRemoved notifies, that the nysse is removed ingame.

Precondition

-

Postcondition

Exception guarantee: strong

Implemented in StudentSide::gameStatistics.

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

- Course/CourseLib/interfaces/istatistics.hh

7.16 Interface::IStop Class Reference

StopIF is an interface that stops fulfill.

```
#include <istop.hh>
```

Inherited by CourseSide::Stop.

Public Member Functions

- IStop ()=default
  Default constructor for the interface. (For documentation)
- virtual ~IStop ()=default
  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual Location getLocation () const =0
  getLocation returns the location of the stop.
- virtual QString getName () const =0
  getName returns the name of the stop.
- virtual unsigned int getId () const =0
  getId returns the id of the stop (the stop number).
- virtual std::vector< std::shared_ptr< Interface::IPassenger > > getPassengers () const =0
  getPassengers returns all passangers in the stop.

7.16.1 Detailed Description

StopIF is an interface that stops fulfill.

If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.16.2 Constructor & Destructor Documentation

7.16.2.1 IStop()

```
Interface::IStop::IStop ( )  [default]
```

Default constructor for the interface. (For documentation)

Postcondition

Stop has no passengers by default.

7.16.3 Member Function Documentation

7.16.3.1 getId()

```
virtual unsigned int Interface::IStop::getId ( ) const  [pure virtual]
```

getId returns the id of the stop (the stop number).

Precondition

-

Returns

stop number

Postcondition

Exception guarantee: nothrow

Implemented in CourseSide::Stop.

7.16.3.2 getLocation()

```
virtual Location Interface::IStop::getLocation ( ) const  [pure virtual]
```

getLocation returns the location of the stop.

Precondition

-

Returns

Stops location

Postcondition

Exception guarantee: nothrow

Implemented in CourseSide::Stop.

7.16.3.3 getName()

```
virtual QString Interface::IStop::getName ( ) const  [pure virtual]
```

getName returns the name of the stop.

Precondition

-

Returns

Stops name

Postcondition

Exception guarantee: nothrow

Implemented in CourseSide::Stop.

7.16.3.4 getPassengers()

```
virtual std::vector<std::shared_ptr<Interface::IPassenger> > Interface::IStop::getPassengers
( ) const  [pure virtual]
```

getPassengers returns all passangers in the stop.

Precondition

-

Returns

Vector that constains all passangers in the stop.

Postcondition

Exception guarantee: strong

Implemented in CourseSide::Stop.

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

- Course/CourseLib/interfaces/istop.hh

7.17 Interface::IVehicle Class Reference

VehicleIF is an interface that describes vehicles (nysse) in game.

```
#include <ivehicle.hh>
```

Inherits Interface::IActor.

Inherited by CourseSide::Nysse.

Public Member Functions

- IVehicle ()=default
  Default constructor (For documentation).
- virtual ~IVehicle ()=default
  Interface has default virtual destructor (base class needs to have a virtual destructor).
- virtual std::string getName () const =0
  getName returns the name of the vehicle(might not be unique).
- virtual std::vector< std::shared_ptr< IPassenger > > getPassengers () const =0
  getPassengers returns all passengers in the vehicle.
- virtual void addPassenger (std::shared_ptr< IPassenger > passenger)=0
  addPassenger adds a new passenger to the vehicle.
- virtual void removePassenger (std::shared_ptr< IPassenger > passenger)=0
  removePassenger removes the passenger from the vehicle.

7.17.1 Detailed Description

VehicleIF is an interface that describes vehicles (nysse) in game.

VehicleIF is inherited from ActorIF-interface. If class method doesn't have exception guarantee of nothrow, method can leak out exception `std::bad_alloc` (out of memory)

7.17.2 Constructor & Destructor Documentation

7.17.2.1 IVehicle()

```
Interface::IVehicle::IVehicle ( )  [default]
```

Default constructor (For documentation).

Postcondition

vehicle has no passengers by default

7.17.3 Member Function Documentation

7.17.3.1 addPassenger()

```
virtual void Interface::IVehicle::addPassenger (
            std::shared_ptr< IPassenger > passenger )  [pure virtual]
```

addPassenger adds a new passenger to the vehicle.

Parameters

passenger	an passenger object to be added to the Vehicle.

Precondition

`matkustaja.onkoKulkuneuvossa() == false`.

Postcondition

Passenger is added into the vehicle. Exception guarantee: basic.

Implemented in CourseSide::Nysse.

7.17.3.2 getName()

```
virtual std::string Interface::IVehicle::getName ( ) const  [pure virtual]
```

getName returns the name of the vehicle(might not be unique).

Precondition

-

Returns

name of the vehicle

Postcondition

Exception guarantee: strong

Implemented in CourseSide::Nysse.

7.17.3.3 getPassengers()

```
virtual std::vector<std::shared_ptr<IPassenger> > Interface::IVehicle::getPassengers ( )
const  [pure virtual]
```

getPassengers returns all passengers in the vehicle.

Precondition

-

Returns

Vector containing all passengers in the vehicle.

Postcondition

Exception guarantee: strong.

Implemented in CourseSide::Nysse.

7.17.3.4 removePassenger()

```
virtual void Interface::IVehicle::removePassenger (
            std::shared_ptr< IPassenger > passenger )  [pure virtual]
```

removePassenger removes the passenger from the vehicle.

Parameters

passenger	Passenger to be removed from the vehicle.

Precondition

-

Postcondition

Passenger is removed from the vehicle. Exception guarantee: basic.

Exceptions

GameError	Passenger is not in the vehicle.

Implemented in CourseSide::Nysse.

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

- Course/CourseLib/interfaces/ivehicle.hh

7.18 Interface::Location Class Reference

Location is a class, which has methods dealing with the location of the objects.

`#include <location.hh>`

Public Member Functions

- Location ()
  Default constructor.
- Location (int northcoord, int eastcoord)
  Constructor that defines a location in certain map coordinate.
- int giveX () const
  giveX returns the x-coordinate of the location in the pixel grid of the game ui.
- int giveY () const
  giveY returns the y-coordinate of the location in the pixel grid of the game ui.
- void setXY (int x, int y)
  setXY moves the location to a new point.(In pixel grid)
- double giveNorthernCoord () const
  giveNorthernCoord returns the location of the northern coordinate from map grid.
- double giveEasternCoord () const
  giveEasternCoord returns the location of the eastern coordinate from map grid.
- void setNorthEast (int northcoord, int eastcoord)
  setNorthEast moves the location to a new coordinate in map grid.
- bool isClose (Location const &loc, int limit=10) const
  isClose tells if given location is close to this location.
- void printBoth ()
  printBoth prints (for debugging purposes) both map and pixel coordinates of the location.
- bool **operator==** (const Location &location)

Static Public Member Functions

- static double calcDistance (Location a, Location b)

  calcDistance calculates the distance between two locations in map grid.
- static Location calcBetween (Location a, Location b, double distance)

  calcBetween calculates wanted position between two locations.

Static Private Member Functions

- static int **xFromEast** (int eastcoord)
- static int **yFromNorth** (int northcoord)
- static int **EastFromX** (int x)
- static int **NorthFromY** (int y)

Private Attributes

- int **northcoord_**
- int **eastcoord_**
- int **x_**
- int **y_**

7.18.1 Detailed Description

Location is a class, which has methods dealing with the location of the objects.

The class provides transformation from map coordinates to pixel coordinates and back, calculation of the distnce and the possibility of generating points between two coordinates.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Location() [1/2]

```
Interface::Location::Location ( )
```

Default constructor.

Postcondition

Location is set to north=6700000, east=3500000.

7.18.2.2 Location() [2/2]

```
Interface::Location::Location (
            int northcoord,
            int eastcoord )
```

Constructor that defines a location in certain map coordinate.

Parameters

northcoord	northern coordinate of the location
eastcoord	eastern coordinate of the location

Precondition

-

Postcondition

Location is given.

7.18.3 Member Function Documentation

7.18.3.1 calcBetween()

```
Location Interface::Location::calcBetween (
            Location a,
            Location b,
            double distance )  [static]
```

calcBetween calculates wanted position between two locations.

Parameters

a	first location
b	second location
distance	ratio where point between a and b is. 0.0=at a, 1.0=at b, 0.5=in the middle.

Precondition

-

Returns

Location of the ration between a and b

Postcondition

Exception guarantee: nothrow.

7.18.3.2 calcDistance()

```
double Interface::Location::calcDistance (
            Location a,
            Location b )  [static]
```

calcDistance calculates the distance between two locations in map grid.

Parameters

a	first location
b	second location

Precondition

-

Returns

distance between two locations

Postcondition

Exception guarantee: nothrow.

7.18.3.3 giveEasternCoord()

```
double Interface::Location::giveEasternCoord ( ) const
```

giveEasternCoord returns the location of the eastern coordinate from map grid.

Precondition

-

Returns

Eastern coordinate

Postcondition

Exception guarantee: nothrow.

7.18.3.4 giveNorthernCoord()

```
double Interface::Location::giveNorthernCoord ( ) const
```

giveNorthernCoord returns the location of the northern coordinate from map grid.

Precondition

-

Returns

Northern coordinate

Postcondition

Exception guarantee: nothrow.

7.18.3.5 giveX()

```
int Interface::Location::giveX ( ) const
```

giveX returns the x-coordinate of the location in the pixel grid of the game ui.

Precondition

-

Returns

X-pixel coordinate.

Postcondition

Exception guarantee: nothrow.

7.18.3.6 giveY()

```
int Interface::Location::giveY ( ) const
```

giveY returns the y-coordinate of the location in the pixel grid of the game ui.

Precondition

-

Returns

Y-pixel coordinate.

Postcondition

Exception guarantee: nothrow.

7.18.3.7 isClose()

```
bool Interface::Location::isClose (
            Location const & loc,
            int limit = 10 ) const
```

isClose tells if given location is close to this location.

Returns true if in function call `s1.isClose(s2)` positions `s1` and `s2` are close enough to affect each other in game.

Parameters

loc	location which closeness ic checked.

Precondition

-

Returns

Boolean which tells wether loc is close to this location.

Postcondition

Exception guarantee: nothrow.

7.18.3.8 printBoth()

```
void Interface::Location::printBoth ( )
```

printBoth prints (for debugging purposes) both map and pixel coordinates of the location.

Precondition

-

Postcondition

Prints coordinates. Exception guarantee: strong.

7.18.3.9 setNorthEast()

```
void Interface::Location::setNorthEast (
            int northcoord,
            int eastcoord )
```

setNorthEast moves the location to a new coordinate in map grid.

Parameters

northcoord	northern coordinate of the new location
eastcoord	eastern coordinate of the new location

Precondition

-

Postcondition

Location is updated. Exception guarantee: strong.

7.18.3.10 setXY()

```
void Interface::Location::setXY (
            int x,
            int y )
```

setXY moves the location to a new point.(In pixel grid)

Parameters

x	x-pixel coordinate of the new location
y	y-pixel coordinate of the new location

Precondition

-

Postcondition

Location is updated. Exception guarantee: strong.

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

- Course/CourseLib/core/location.hh
- Course/CourseLib/core/location.cc

7.19 CourseSide::Logic Class Reference

The Logic class.

```
#include <logic.hh>
```

Inherits QObject.

Public Slots

- void advance ()

  advance handles the movement and removal of buses and passengers. Gets called every timeout by increaseTime
- void configChanged (QTime time, bool debug)

  configChanged handles possible config parameters and calls fileconfig
- void increaseTime ()

  increaseTime gets called when timer_ timeouts and increases time when game is not over, calls advance to move buses to next position.
- void addNewBuses ()

  addNewBuses adds new buses to traffic from offlinedata
- void addNewPassengers (std::shared_ptr< Stop > stop, unsigned int no)

  addNewPassengers adds passengers to given stop

Public Member Functions

- Logic (QObject *parent=0)

  Default constructor.
- bool readOfflineData (const QString &buses, const QString &stops)

  readOfflineData uses OfflineReader class to read given offlinedata-files
- void finalizeGameStart ()

  finalizeGameStart calls to add buses, stops and passengers, calls cityif_ to start the game and starts timer to update buses movement
- void fileConfig (QString stops=DEFAULT_STOPS_FILE, QString buses=DEFAULT_BUSES_FILE)

  fileConfig calls to read offlinedata
- void setTime (unsigned short hr, unsigned short min)

  setTime sets time_ to given time
- bool takeCity (std::shared_ptr< Interface::ICity > city)

  *takeCity sets given parameter as cityif_*

Private Member Functions

- bool **calculateNewLocationForBus** (std::shared_ptr< Nysse > bussi)
- void **addBuses** ()
- void **addStopsAndPassengers** ()
- std::map< QTime, std::weak_ptr< Stop > > **calculateStopTimes** (std::map< QTime, std::shared_ptr< Stop >> &stops, QTime &departure_time)
- void **createBus** (std::shared_ptr< BusData > bus, QTime departure_time)

Private Attributes

- std::shared_ptr< Interface::ICity > **cityif_**
- std::list< std::shared_ptr< Passenger > > **passengers_**
- std::list< std::shared_ptr< Nysse > > **buses_**
- std::vector< std::shared_ptr< Stop > > **stops_**
- std::shared_ptr< OfflineData > **offlinedata_**
- QString **busfile_**
- QString **stopfile_**
- bool **debugstate_**
- bool **gamestarted_**
- QTime **time_**
- QTimer **timer_**
- QTimer **animationtimer_**
- QTimer **departuretimer_**
- int **busSID_**

Static Private Attributes

- static const int **TIME_SPEED** = 10
- static const int **UPDATE_INTERVAL_MS** = 100

7.19.1 Detailed Description

The Logic class.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Logic()

```
CourseSide::Logic::Logic (
            QObject * parent = 0 )
```

Default constructor.

Parameters

parent	is a QObject pointer that defaults to 0

7.19.3 Member Function Documentation

7.19.3.1 addNewPassengers

```
void CourseSide::Logic::addNewPassengers (
            std::shared_ptr< Stop > stop,
            unsigned int no )  [slot]
```

addNewPassengers adds passengers to given stop

Parameters

stop	pointer to stop that passengers are added
no	number of added passengers

7.19.3.2 configChanged

```
void CourseSide::Logic::configChanged (
            QTime time,
            bool debug )  [slot]
```

configChanged handles possible config parameters and calls fileconfig

Parameters

time	given time in QTime
debug	if debugmode is true

7.19.3.3 fileConfig()

```
void CourseSide::Logic::fileConfig (
            QString stops = DEFAULT_STOPS_FILE,
            QString buses = DEFAULT_BUSES_FILE )
```

fileConfig calls to read offlinedata

Parameters

stops	datafile for stops, defaults to constant
buses	datafile for buses, defaults to constant

7.19.3.4 finalizeGameStart()

```
void CourseSide::Logic::finalizeGameStart ( )
```

finalizeGameStart calls to add buses, stops and passengers, calls cityif_ to start the game and starts timer to update buses movement

Precondition

takeCity and fileConfig must be called

7.19.3.5 readOfflineData()

```
bool CourseSide::Logic::readOfflineData (
            const QString & buses,
            const QString & stops )
```

readOfflineData uses OfflineReader class to read given offlinedata-files

Parameters

buses	filepath for busfile
stops	filepath for stopfile

Returns

returns true if data files are read correctly, else false

7.19.3.6 setTime()

```
void CourseSide::Logic::setTime (
            unsigned short hr,
            unsigned short min )
```

setTime sets time_ to given time

Parameters

hr	time in hours
min	time in minutes

7.19.3.7 takeCity()

```
bool CourseSide::Logic::takeCity (
            std::shared_ptr< Interface::ICity > city )
```

takeCity sets given parameter as cityif_

Parameters

city	pointer of a class that is derived from ICity interface in StudentSide

Returns

true

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

- Course/CourseLib/core/logic.hh
- Course/CourseLib/core/logic.cc

7.20 StudentSide::MainMenuDialog Class Reference

Defines a QDialog which is the first shown window to the user when the app is run. It is the main configuration dialog in which player sets a player nickname, chooses player and projectile type and possibly accesses settings or help.

`#include <mainmenudialog.h>`

Inherits QDialog.

Public Types

- enum **playerConfig** {
  spaceshipOption, **tankOption**, **ufoOption**, **fireballOption**,
  missileOption, **laserOption**, **musicStateOn**, **musicStateOff**,
  fireballSound, **missileSound**, **blasterSound** }

Public Member Functions

- MainMenuDialog (QWidget *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.
- ~MainMenuDialog ()

  MainMenuDialog has a basic destructor.
- void setToolTips ()

  setToolTips sets tool tips in the MainMenuDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)
- void setStartGameText (QString text, QString color="red")

  setStartGameText sets a string to QLabel in desired color.
- bool nameIsEmpty (const QString playerName) const

  nameIsEmpty checks and validates a QString which is set in the MainMenuDialog

Private Slots

- void **on_startButton_clicked** ()
- void **on_exitButton_clicked** ()
- void **on_playerNameEdit_editingFinished** ()
- void **on_spaceshipButton_clicked** ()
- void **on_tankButton_clicked** ()
- void **on_ufoButton_clicked** ()
- void **on_fireballButton_clicked** ()
- void **on_missileButton_clicked** ()
- void **on_laserButton_clicked** ()
- void **on_settingsButton_clicked** ()
- void **on_helpButton_clicked** ()

Private Attributes

- Ui::MainMenuDialog * **ui**
- QSize **_menuDialogSize**
- const QPixmap **_fireballPic** = QPixmap(":/images/fireball_16x16.png")
- const QPixmap **_missilePic** = QPixmap(":/images/missile_23x10.png")
- const QPixmap **_laserPic** = QPixmap(":/images/laser_32x32.png")
- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()
- bool **_tank** = false
- bool **_spaceShip** = false
- bool **_ufo** = false
- bool **_fireball** = false
- bool **_missile** = false
- bool **_laser** = false
- bool **_musicsOn** = false
- QString **_playerAlias**

7.20.1 Detailed Description

Defines a QDialog which is the first shown window to the user when the app is run. It is the main configuration dialog in which player sets a player nickname, chooses player and projectile type and possibly accesses settings or help.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 MainMenuDialog()

```
StudentSide::MainMenuDialog::MainMenuDialog (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QWidget.

Postcondition

MainMenuDialog is at initialization state.

7.20.3 Member Function Documentation

7.20.3.1 nameIsEmpty()

```
bool StudentSide::MainMenuDialog::nameIsEmpty (
            const QString playerName ) const
```

nameIsEmpty checks and validates a QString which is set in the MainMenuDialog

Parameters

String	in QString format.

Precondition

-

Returns

Returns true if the string given as a parameter contains ONLY whitespace characters, otherwise returns false.

Postcondition

Boolean has been returned to tell if the string contains only whitespace or not. Exception guarantee: nothrow.

7.20.3.2 setStartGameText()

```
void StudentSide::MainMenuDialog::setStartGameText (
            QString text,
            QString color = "red" )
```

setStartGameText sets a string to QLabel in desired color.

Parameters

String	to-be-shown in the QLabel and the color of the string.

Precondition

The QLabel where the text is shown, exists.

Postcondition

Desired string is set to the QLabel. Exception guarantee: nothrow.

7.20.3.3 setToolTips()

```
void StudentSide::MainMenuDialog::setToolTips ( )
```

setToolTips sets tool tips in the MainMenuDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)

Precondition

-

Postcondition

ToolTips are shown to the user of the software when hovering mouse above buttons, labels or other items of the GUI. Exception guarantee: nothrow.

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

- Game/mainmenudialog.h
- Game/mainmenudialog.cpp

7.21 CourseSide::Nysse Class Reference

Inherits Interface::IVehicle.

Public Member Functions

- **Nysse** (unsigned int line)
- std::string getName () const

  getName returns the name of the vehicle(might not be unique).
- Interface::Location giveLocation () const

  giveLocation returns the location of the actor.
- void move (Interface::Location loc)

  move-method moves the actor to given location.
- void remove ()

  remove marks the actor as removed.
- bool isRemoved () const

  isRemoved tells if the actor is removed ingame.
- std::vector< std::shared_ptr< Interface::IPassenger > > getPassengers () const

  getPassengers returns all passengers in the vehicle.
- void addPassenger (std::shared_ptr< Interface::IPassenger > passenger)

  addPassenger adds a new passenger to the vehicle.
- void removePassenger (std::shared_ptr< Interface::IPassenger > passenger)

  removePassenger removes the passenger from the vehicle.
- std::weak_ptr< Stop > **getFinalStop** () const
- std::map< QTime, std::pair< Interface::Location, std::weak_ptr< Stop > > > **getTimeRoute** () const
- void **setRoute** (const std::map< QTime, std::pair< Interface::Location, std::shared_ptr< Stop > > > &timeroute, QTime &departuretime)
- unsigned int **getLine** ()
- std::weak_ptr< Stop > **getStop** ()
- Interface::Location **moveToNextPosition** (QTime time)
- void **calcStartingPos** (QTime time)
- void **setCity** (std::shared_ptr< Interface::ICity > city)
- int **getSID** () const
- void **setSID** (int sid)

Private Attributes

- unsigned int **line_**
- std::string **name_**
- std::shared_ptr< Interface::ICity > **city_**
- Interface::Location **location_**
- std::vector< std::shared_ptr< Interface::IPassenger > > **passengers_**
- std::map< QTime, std::pair< Interface::Location, std::weak_ptr< Stop > > > **timeroute2_**
- std::map< QTime, std::pair< Interface::Location, std::weak_ptr< Stop > > >::iterator **ar2Iterator_**
- std::weak_ptr< Stop > **stop_**
- bool **removed_**
- int **SID_**

7.21.1 Member Function Documentation

7.21.1.1 addPassenger()

```
void CourseSide::Nysse::addPassenger (
            std::shared_ptr< Interface::IPassenger > passenger )  [virtual]
```

addPassenger adds a new passenger to the vehicle.

Parameters

passenger	an passenger object to be added to the Vehicle.

Precondition

`matkustaja.onkoKulkuneuvossa() == false`.

Postcondition

Passenger is added into the vehicle. Exception guarantee: basic.

Implements Interface::IVehicle.

7.21.1.2 getName()

```
std::string CourseSide::Nysse::getName ( ) const  [virtual]
```

getName returns the name of the vehicle(might not be unique).

Precondition

-

Returns

name of the vehicle

Postcondition

Exception guarantee: strong

Implements Interface::IVehicle.

7.21.1.3 getPassengers()

```
std::vector< std::shared_ptr< Interface::IPassenger > > CourseSide::Nysse::getPassengers ( )
const  [virtual]
```

getPassengers returns all passengers in the vehicle.

Precondition

-

Returns

Vector containing all passengers in the vehicle.

Postcondition

Exception guarantee: strong.

Implements Interface::IVehicle.

7.21.1.4 giveLocation()

```
Interface::Location CourseSide::Nysse::giveLocation ( ) const  [virtual]
```

giveLocation returns the location of the actor.

Precondition

-

Returns

Actors location.

Postcondition

Exception guarantee: strong.

Exceptions

GameError	- actor wasn't given a location.

Implements Interface::IActor.

7.21.1.5 isRemoved()

```
bool CourseSide::Nysse::isRemoved ( ) const  [virtual]
```

isRemoved tells if the actor is removed ingame.

Precondition

-

Returns

`true`, if actor is removed ingame, otherwise `false`.

Postcondition

Exception guarantee: nothrow.

Implements Interface::IActor.

7.21.1.6 move()

```
void CourseSide::Nysse::move (
            Interface::Location loc )  [virtual]
```

move-method moves the actor to given location.

Parameters

loc	Actors new location.

Precondition

-

Postcondition

Actors location is sij. Excaption guarantee: strong.

Exceptions

GameError	Location is not possible.

Implements Interface::IActor.

7.21.1.7 remove()

```
void CourseSide::Nysse::remove ( )  [virtual]
```

remove marks the actor as removed.

Precondition

Actor is not removed already.

Postcondition

Actor is removed, after this isRemoved() returns `true`. Exception guarantee: basic.

Implements Interface::IActor.

7.21.1.8 removePassenger()

```
void CourseSide::Nysse::removePassenger (
            std::shared_ptr< Interface::IPassenger > passenger )  [virtual]
```

removePassenger removes the passenger from the vehicle.

Parameters

passenger	Passenger to be removed from the vehicle.

Precondition

-

Postcondition

Passenger is removed from the vehicle. Exception guarantee: basic.

Exceptions

GameError	Passenger is not in the vehicle.

Implements Interface::IVehicle.

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

- Course/CourseLib/actors/nysse.hh
- Course/CourseLib/actors/nysse.cc

7.22 CourseSide::OfflineData Struct Reference

Public Attributes

- std::vector< std::shared_ptr< Stop > > **stops**
- std::list< std::shared_ptr< BusData > > **buses**

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

- Course/CourseLib/offlinereader.hh

7.23 CourseSide::OfflineReader Class Reference

Public Member Functions

- std::shared_ptr< OfflineData > **readFiles** (const QString &busfile, const QString &stopfile)

Private Member Functions

- void **readBusFile** (const QString &busfile)
- void **readStopFile** (const QString &stopfile)
- void **readDepartureTimes** (const QJsonArray &timearray, BusData *bus)
- void **readRoute** (std::shared_ptr< BusData > bus, QJsonObject &o)
- std::shared_ptr< Stop > **findStops** (int id)
- QTime **calculateQTime** (int time)

Private Attributes

- std::shared_ptr< OfflineData > **offlinedata_**

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

- Course/CourseLib/offlinereader.hh
- Course/CourseLib/offlinereader.cc

7.24 CourseSide::Passenger Class Reference

Inherits Interface::IPassenger.

Public Member Functions

- **Passenger** (std::weak_ptr< Interface::IStop > destination)
- Interface::Location giveLocation () const

  giveLocation returns the location of the actor.
- void move (Interface::Location loc)

  move-method moves the actor to given location.
- void remove ()

  remove marks the actor as removed.
- bool isRemoved () const

  isRemoved tells if the actor is removed ingame.
- bool isInVehicle () const

  isInVehicle tells if passenger is in any vehicle currently.
- std::shared_ptr< Interface::IVehicle > getVehicle () const

  getVehicle returns the vehicle passenger is in.
- std::shared_ptr< Interface::IStop > **getStop** () const
- virtual bool **wantToEnterNysse** (std::weak_ptr< Nysse > nysse) const
- void **enterNysse** (std::weak_ptr< Nysse > nysse)
- virtual bool **wantToEnterVehicle** (std::weak_ptr< Interface::IVehicle > vehicle) const
- void **enterVehicle** (std::weak_ptr< Interface::IVehicle > vehicle)
- virtual bool **wantToEnterStop** (std::weak_ptr< Interface::IStop > stop) const
- void **enterStop** (std::weak_ptr< Interface::IStop > stop)

Protected Attributes

- bool **removed_**
- std::weak_ptr< Interface::IStop > **destination_**

Private Attributes

- std::weak_ptr< Interface::IVehicle > **nyssep_**
- std::weak_ptr< Interface::IStop > **stopp_**

7.24.1 Member Function Documentation

7.24.1.1 getVehicle()

```
std::shared_ptr< Interface::IVehicle > CourseSide::Passenger::getVehicle ( ) const  [virtual]
```

getVehicle returns the vehicle passenger is in.

Precondition

-

Returns

Vechicle where passenger is in. Empty pointer if passenger is not in any vehicle.

Postcondition

Exception guarantee: nothrow.

Implements Interface::IPassenger.

7.24.1.2 giveLocation()

```
Interface::Location CourseSide::Passenger::giveLocation ( ) const  [virtual]
```

giveLocation returns the location of the actor.

Precondition

-

Returns

Actors location.

Postcondition

Exception guarantee: strong.

Exceptions

GameError	- actor wasn't given a location.

Implements Interface::IActor.

7.24.1.3 isInVehicle()

```
bool CourseSide::Passenger::isInVehicle ( ) const  [virtual]
```

isInVehicle tells if passenger is in any vehicle currently.

Precondition

-

Returns

Boolean, tells wether passenger is in any vehicle.

Postcondition

Exception guarantee: nothrow

Implements Interface::IPassenger.

7.24.1.4 isRemoved()

```
bool CourseSide::Passenger::isRemoved ( ) const  [virtual]
```

isRemoved tells if the actor is removed ingame.

Precondition

-

Returns

`true`, if actor is removed ingame, otherwise `false`.

Postcondition

Exception guarantee: nothrow.

Implements Interface::IActor.

7.24.1.5 move()

```
void CourseSide::Passenger::move (
            Interface::Location loc )  [virtual]
```

move-method moves the actor to given location.

Parameters

loc	Actors new location.

Precondition

-

Postcondition

Actors location is sij. Excaption guarantee: strong.

Exceptions

GameError	Location is not possible.

Implements Interface::IActor.

7.24.1.6 remove()

```
void CourseSide::Passenger::remove ( )  [virtual]
```

remove marks the actor as removed.

Precondition

Actor is not removed already.

Postcondition

Actor is removed, after this isRemoved() returns `true`. Exception guarantee: basic.

Implements Interface::IActor.

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

- Course/CourseLib/actors/passenger.hh
- Course/CourseLib/actors/passenger.cc

7.25 CourseSide::Place Struct Reference

Public Attributes

- bool **stop**
- std::shared_ptr< Stop > **stopPtr**
- std::shared_ptr< Interface::Location > **locationPtr**

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

- Course/CourseLib/offlinereader.hh

7.26 StudentSide::Player Class Reference

Defines a class for the main player that the game's user controls in the game. The player can be in interaction with bonusItems, Nysses and passengers shown on the game map. Player movement happens via arrow key pressing, and shooting via space bar.

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

Inherits QObject, and QGraphicsPixmapItem.

Public Member Functions

- Player (QGraphicsItem ∗parent=0)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.
- ∼Player ()

  Player has a basic destructor.
- void keyPressEvent (QKeyEvent ∗event)

  keyPressEvent checks which keys are being pressed during the game in MainWindow to enable smooth player transitions in the game map.
- void keyReleaseEvent (QKeyEvent ∗event)

  keyReleaseEvent checks which keys are being released during the game in MainWindow to enable smooth player transitions in the game map.
- void movePlayer ()

  movePlayer function is responsible for moving the player in the game map.
- void setDimensions ()

  setDimensions sets the width and the height of a chosen player type.
- void changePlayerSpeed (QKeyEvent ∗speedEvent)

  changePlayerSpeed increases (if "+"-key is pressed) or decreases (if "-"-key is pressed) player speed during the game within reasonable limits. Each increase or decrease is at 5 % delta.
- void addPlayerSprite ()

  addPlayerSprite sets the the chosen picture as the player icon in the game.
- void initMusic ()

  initMusic initializes the QSoundEffect object according to the projectile type chosen by the player. Every porjectile tupe has an unique sound: fireball has a magic cast, missile an explosion and laser a blaster sound effect.
- void configureMusic ()

  configureMusic checks is the sound is being played ingame at the time of the function call. It plays the music if needed to and also sets the music volume to a player-friendly level.
- void setMusicChoice ()

  setMusicChoice checks the music setting and acts accordingly: if the music has been opted in, then it calls the configureMusic() -method or otherwise it does not do anything.
- void savePlayerName ()

  savePlayerName saves player name (in std::string format) chosen by the player into Player-class attribute.
- void removeCollidingItem ()

  removeCollidingItem is responsible for removing the bonus diamonds that are in contact with the player from the scene and memory. It also increases player points and updates the changes to the gameStatistics if a bonus diamond is collected by the player.
- std::vector< int > getPlayerOrigin (int width, int height)

  getPlayerOrigin gets chosen player type's center coordinates and returns them in a vector.

Public Attributes

- int **playerHeight**
- int **playerWidth**
- int **screenWidth** = 800
- int **screenHeight** = 600
- std::string **playerName**

Private Attributes

- double **_spaceshipVelocity** = 23.0
- QSoundEffect * **_projectileSound**
- QTimer * **_moveTimer**
- int **_interval** = 50
- bool **_keyLeft** = false
- bool **_keyRight** = false
- bool **_keyUp** = false
- bool **_keyDown** = false
- bool **_keySpace** = false
- bool **_tankChosen** = false
- bool **_spaceshipChosen** = false
- bool **_ufoChosen** = false
- bool **_fireballChosen** = false
- bool **_missileChosen** = false
- bool **_laserChosen** = false
- bool **_musicsOn** = false
- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()
- const QPixmap **_spaceshipPic** = QPixmap(":/images/spaceship_45x31.png")
- const QPixmap **_tankPic** = QPixmap(":/images/tank_sprite_26x50.png")
- const QPixmap **_ufoPic** = QPixmap(":/images/ufo_sprite_50x50.png")
- const QUrl **_blasterSound** = QUrl("qrc:/sounds/blaster_sound.wav")
- const QUrl **_fireballSound** = QUrl("qrc:/sounds/fireballSound.wav")
- const QUrl **_missileSound** = QUrl("qrc:/sounds/missileSound.wav")

7.26.1 Detailed Description

Defines a class for the main player that the game's user controls in the game. The player can be in interaction with bonusItems, Nysses and passengers shown on the game map. Player movement happens via arrow key pressing, and shooting via space bar.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Player()

```
StudentSide::Player::Player (
            QGraphicsItem * parent = 0 )
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

Player is at initialization state.

7.26.3 Member Function Documentation

7.26.3.1 addPlayerSprite()

```
void StudentSide::Player::addPlayerSprite ( )
```

addPlayerSprite sets the the chosen picture as the player icon in the game.

Precondition

Player pictures provided are valid and they exist.

Postcondition

Player picture for the chosen player type is set. Exception guarantee: nothrow.

7.26.3.2 changePlayerSpeed()

```
void StudentSide::Player::changePlayerSpeed (
            QKeyEvent * speedEvent )
```

changePlayerSpeed increases (if "+"-key is pressed) or decreases (if "-"-key is pressed) player speed during the game within reasonable limits. Each increase or decrease is at 5 % delta.

Parameters

QKeyEvent	when player presses "+" or "-" key ingame.

Precondition

The game has started and gameWindow is active and responsive.

Postcondition

Player speed has been changed if certain keys have been pressed ingame. Exception guarantee: nothrow.

7.26.3.3 configureMusic()

```
void StudentSide::Player::configureMusic ( )
```

configureMusic checks is the sound is being played ingame at the time of the function call. It plays the music if needed to and also sets the music volume to a player-friendly level.

Precondition

Music has been opted in by the player.

Postcondition

Exception guarantee: nothrow.

7.26.3.4 getPlayerOrigin()

```
std::vector< int > StudentSide::Player::getPlayerOrigin (
            int width,
            int height )
```

getPlayerOrigin gets chosen player type's center coordinates and returns them in a vector.

Parameters

Player	icon's width and height.

Precondition

-

Returns

Vector containing player icon's origin coordinates: x-coordinate is at index 0 and y-coordinate is at index 1.

Postcondition

Exception guarantee: nothrow.

7.26.3.5 initMusic()

```
void StudentSide::Player::initMusic ( )
```

initMusic initializes the QSoundEffect object according to the projectile type chosen by the player. Every porjectile tupe has an unique sound: fireball has a magic cast, missile an explosion and laser a blaster sound effect.

Precondition

-

Postcondition

QSoundEffect has been initialized according to the chosen projectile type. Exception guarantee: nothrow.

7.26.3.6 keyPressEvent()

```
void StudentSide::Player::keyPressEvent (
            QKeyEvent * event )
```

keyPressEvent checks which keys are being pressed during the game in MainWindow to enable smooth player transitions in the game map.

Parameters

QKeyEvent	when player presses arrow or space keys ingame.

Precondition

The game has started and gameWindow is active and responsive.

Postcondition

Keys that are pressed ingame are set to "true". Exception guarantee: nothrow.

7.26.3.7 keyReleaseEvent()

```
void StudentSide::Player::keyReleaseEvent (
            QKeyEvent * event )
```

keyReleaseEvent checks which keys are being released during the game in MainWindow to enable smooth player transitions in the game map.

Parameters

QKeyEvent	when player releases arrow or space keys ingame.

Precondition

The game has started and gameWindow is active and responsive.

Postcondition

Keys that are released ingame are set to "false". Exception guarantee: nothrow.

7.26.3.8 movePlayer()

```
void StudentSide::Player::movePlayer ( )
```

movePlayer function is responsible for moving the player in the game map.

Precondition

-

Postcondition

Bullet has moved in the map and is deleted from the scene and the memory if it encounters a object in its way or exits the game screen. Exception guarantee: nothrow.

7.26.3.9 removeCollidingItem()

```
void StudentSide::Player::removeCollidingItem ( )
```

removeCollidingItem is responsible for removing the bonus diamonds that are in contact with the player from the scene and memory. It also increases player points and updates the changes to the gameStatistics if a bonus diamond is collected by the player.

Precondition

Player comes in contact with a bonus diamond in the game map.

Postcondition

Collected diamond is deleted from the game map and memory or nothing happens. Exception guarantee: nothrow.

7.26.3.10 savePlayerName()

```
void StudentSide::Player::savePlayerName ( )
```

savePlayerName saves player name (in std::string format) chosen by the player into Player-class attribute.

Precondition

-

Postcondition

Player name has been saved correctly. Exception guarantee: nothrow.

7.26.3.11 setDimensions()

```
void StudentSide::Player::setDimensions ( )
```

setDimensions sets the width and the height of a chosen player type.

Precondition

Chosen player type is existent and therefore has some kind of dimensions.

Postcondition

Dimensions for the player projectile type are successfully set. Exception guarantee: nothrow.

7.26.3.12 setMusicChoice()

```
void StudentSide::Player::setMusicChoice ( )
```

setMusicChoice checks the music setting and acts accordingly: if the music has been opted in, then it calls the configureMusic() -method or otherwise it does not do anything.

Precondition

-

Postcondition

Exception guarantee: nothrow.

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

- Game/player.h
- Game/player.cpp

7.27 StudentSide::settingsDialog Class Reference

Defines a QDialog which offers a separate settings window where the player can set music on/off and alter the game duration. The Dialog itself is accessible from MainMenu's "Settings" -button. If the player does not make any changes to the settings (which isn't compulsory), the default settings are used: musics off and game duration 2 minutes. Beware that if you don't press the "Save" -button in the settingsDialog, your changes to the settings won't come into effect and the program will use the default settings!

```
#include <settingsdialog.h>
```

Inherits QDialog.

Public Types

- enum **musicState** { **musicStateOn**, **musicStateOff** }
- enum **gameTime** { **gameTime1**, **gameTime2**, **gameTime3** }

Public Member Functions

- settingsDialog (QWidget *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.
- ~settingsDialog ()

  settingsDialog has a basic destructor.
- void setActionToolTips ()

  setActionToolTips sets tool tips in the settingsDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)
- void setCorrectMusicState ()

  setCorrectMusicState saves the music setting chosen by the player.
- void setWantedGameTime ()

  setWantedGameTime saves the game duration chosen by the player.

Private Slots

- void **on_backToMainButton_clicked** ()
- void **onMusicsOnClicked** ()
- void **on_saveSettingsButton_clicked** ()

Private Attributes

- Ui::settingsDialog * **ui**
- bool **_musicsOn** = false
- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()

7.27.1 Detailed Description

Defines a QDialog which offers a separate settings window where the player can set music on/off and alter the game duration. The Dialog itself is accessible from MainMenu's "Settings" -button. If the player does not make any changes to the settings (which isn't compulsory), the default settings are used: musics off and game duration 2 minutes. Beware that if you don't press the "Save" -button in the settingsDialog, your changes to the settings won't come into effect and the program will use the default settings!

7.27.2 Constructor & Destructor Documentation

7.27.2.1 settingsDialog()

```
StudentSide::settingsDialog::settingsDialog (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

settingsDialog is at initialization state.

7.27.3 Member Function Documentation

7.27.3.1 setActionToolTips()

```
void StudentSide::settingsDialog::setActionToolTips ( )
```

setActionToolTips sets tool tips in the settingsDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)

Precondition

-

Postcondition

ToolTips are shown to the user of the software when hovering mouse above buttons, labels or other items of the GUI. Exception guarantee: nothrow.

7.27.3.2 setCorrectMusicState()

```
void StudentSide::settingsDialog::setCorrectMusicState ( )
```

setCorrectMusicState saves the music setting chosen by the player.

Precondition

-

Postcondition

Chosen music setting has been saved. Exception guarantee: nothrow.

7.27.3.3 setWantedGameTime()

```
void StudentSide::settingsDialog::setWantedGameTime ( )
```

setWantedGameTime saves the game duration chosen by the player.

Precondition

-

Postcondition

Chosen game time duration setting has been saved. Exception guarantee: nothrow.

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

- Game/settingsdialog.h
- Game/settingsdialog.cpp

7.28 CourseSide::SimpleActorItem Class Reference

Inherits QGraphicsItem.

Public Member Functions

- **SimpleActorItem** (int x, int y, int type=0)
- QRectF **boundingRect** () const
- void **paint** (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)
- void **setCoord** (int x, int y)

Private Attributes

- int **x_**
- int **y_**
- int **type_**

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

- Course/CourseLib/graphics/simpleactoritem.hh
- Course/CourseLib/graphics/simpleactoritem.cpp

7.29 CourseSide::SimpleMainWindow Class Reference

Inherits QMainWindow.

Signals

- void **gameStarted** ()

Public Member Functions

- **SimpleMainWindow** (QWidget ∗parent=0)
- void **setSize** (int w, int h)
- void **setTick** (int t)
- virtual void **addActor** (int locX, int locY, int type=0)
- void **updateCoords** (int nX, int nY)
- void **setPicture** (QImage &img)

Private Slots

- void **on_startButton_clicked** ()

Private Attributes

- Ui::SimpleMainWindow ∗ **ui**
- QGraphicsScene ∗ **map**
- QTimer ∗ **timer**
- QVector< QGraphicsItem ∗ > **actors_**
- SimpleActorItem ∗ **last_**
- int **width_** = 500
- int **height_** = 500
- int **tick_** = 500

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

- Course/CourseLib/graphics/simplemainwindow.hh
- Course/CourseLib/graphics/simplemainwindow.cpp

7.30 statisticsTest Class Reference

Unit tests for the gameStatistics class which inherits from Course Side's iStatistics.

Inherits QObject.

Public Member Functions

- statisticsTest ()
  Basic constructor for statisticsTest.
- ~statisticsTest ()
  statisticsTest has a basic destructor.

Private Slots

- void testGivePoints ()
  Testing give points -functionality.
- void testPassengerDied ()
  Testing passenger died -functionality.
- void testNysseRemoved ()
  Testing Nysse removed -functionality.
- void testMorePassengers ()
  Testing more passengers -functionality.
- void testNewNysse ()
  Testing new Nysse addition -functionality.
- void testNysseLeft ()
  Testing Nysse left -functionality.
- void addCollectedDiamond ()
  Testing adding collected diamond -functionality.
- void testActorMoved ()
  Testing actor removed -functionality.
- void testPassengerLeft ()
  Testing passenger left -functionality.
- void testAddPoints ()
  Testing adding points -functionality.

7.30.1 Detailed Description

Unit tests for the gameStatistics class which inherits from Course Side's iStatistics.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 statisticsTest()

```
statisticsTest::statisticsTest ( )
```

Basic constructor for statisticsTest.

Postcondition

statisticsTest is at initialization state.

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

- StudentSideTests/tst_statisticstest.cpp

7.31 StudentSide::statistisDialog Class Reference

Defines a QDialog which offers statisctics of the game played by the player. The Dialog itself is accessible from GameOverDialog's "Statistics" -button.

```
#include <statistisdialog.h>
```

Inherits QDialog.

Public Member Functions

- statistisDialog (QWidget *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.
- ~statistisDialog ()

  statistisDialog has a basic destructor.
- void setToolTips ()

  setToolTips sets tool tips in the statistisDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)
- void generateStatsString ()

  generateStatsString creates and adds player's statistical data of a single game into the QTextBrowser in statistics↩ Dialog.

Private Slots

- void **on_backButton_clicked** ()

Private Attributes

- Ui::statistisDialog * **ui**

7.31.1 Detailed Description

Defines a QDialog which offers statisctics of the game played by the player. The Dialog itself is accessible from GameOverDialog's "Statistics" -button.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 statistisDialog()

```
StudentSide::statistisDialog::statistisDialog (
            QWidget * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

statistisDialog is at initialization state.

7.31.3 Member Function Documentation

7.31.3.1 generateStatsString()

```
void StudentSide::statistisDialog::generateStatsString ( )
```

generateStatsString creates and adds player's statistical data of a single game into the QTextBrowser in statistics↩ Dialog.

Precondition

-

Postcondition

Player game statisctics are shown to the user of the software in the statisticsDialog. Exception guarantee: nothrow.

7.31.3.2 setToolTips()

```
void StudentSide::statistisDialog::setToolTips ( )
```

setToolTips sets tool tips in the statistisDialog's GUI to guide the player (shown when player hovers mouse on top of a button or a label etc.)

Precondition

-

Postcondition

ToolTips are shown to the user of the software when hovering mouse above buttons, labels or other items of the GUI. Exception guarantee: nothrow.

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

- Game/statistisdialog.h
- Game/statistisdialog.cpp

7.32 CourseSide::Stop Class Reference

Inherits Interface::IStop.

Public Member Functions

- **Stop** (const Interface::Location &location, const QString &name, unsigned int id)
- Interface::Location getLocation () const

  getLocation returns the location of the stop.
- QString getName () const

  getName returns the name of the stop.
- unsigned int getId () const

  getId returns the id of the stop (the stop number).
- std::vector< std::shared_ptr< Interface::IPassenger > > getPassengers () const

  getPassengers returns all passangers in the stop.
- void **setLocation** (const Interface::Location &location)
- void **setName** (const QString &name)
- void **setId** (unsigned int id)
- void **addPassenger** (const std::weak_ptr< Interface::IPassenger > passenger)
- void **removePassenger** (const std::weak_ptr< Interface::IPassenger > passenger)

Private Attributes

- Interface::Location **location_**
- QString **name_**
- unsigned int **id_**
- std::vector< std::shared_ptr< Interface::IPassenger > > **passengers_**

7.32.1 Member Function Documentation

7.32.1.1 getId()

```
unsigned int CourseSide::Stop::getId ( ) const  [virtual]
```

getId returns the id of the stop (the stop number).

Precondition

-

Returns

stop number

Postcondition

Exception guarantee: nothrow

Implements Interface::IStop.

7.32.1.2 getLocation()

```
Interface::Location CourseSide::Stop::getLocation ( ) const  [virtual]
```

getLocation returns the location of the stop.

Precondition

-

Returns

Stops location

Postcondition

Exception guarantee: nothrow

Implements Interface::IStop.

7.32.1.3 getName()

```
QString CourseSide::Stop::getName ( ) const  [virtual]
```

getName returns the name of the stop.

Precondition

-

Returns

Stops name

Postcondition

Exception guarantee: nothrow

Implements Interface::IStop.

7.32.1.4 getPassengers()

```
std::vector< std::shared_ptr< Interface::IPassenger > > CourseSide::Stop::getPassengers ( )
const  [virtual]
```

getPassengers returns all passangers in the stop.

Precondition

-

Returns

Vector that constains all passangers in the stop.

Postcondition

Exception guarantee: strong

Implements Interface::IStop.

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

- Course/CourseLib/actors/stop.hh
- Course/CourseLib/actors/stop.cc

7.33 StudentSide::topHighScores Class Reference

Defines a Class which is responsible for saving and reading player name and SCALED points to and from a top10highscores.txt file. The class also implements a top10-highscore feature which gives the player information about all-time best players when the metric is the highest scaled points count.

```
#include <tophighscores.h>
```

Inherits QObject.

Public Member Functions

- topHighScores (QObject *parent=nullptr)

  Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.
- void readFile (QString filename=textFilePath)

  readFile reads data from a textfile into map.
- void writeFile (QString filename=textFilePath)

  writeFile writes data into a textfile (player name and his points scaled according to the chosen game duration).
- void sortAndDisplay (std::map< QString, int > mapToBeSorted)

  sortAndDisplay sorts the players according to their points into a top10 -subset and appends these top10 -players into a QString which is shown to the player later in GameOverDialog's QTextBrowser.
- int getDuration ()

  getDuration reads the game duration setting saved before and returns it as an integer which represents game duration in minutes.

Public Attributes

- std::map< QString, int > **scores**
- QString **scoreStream**

Private Attributes

- std::shared_ptr< QSettings > **_playerSettings** = std::make_shared<QSettings>()

7.33.1 Detailed Description

Defines a Class which is responsible for saving and reading player name and SCALED points to and from a top10highscores.txt file. The class also implements a top10-highscore feature which gives the player information about all-time best players when the metric is the highest scaled points count.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 topHighScores()

```
StudentSide::topHighScores::topHighScores (
            QObject * parent = nullptr )  [explicit]
```

Basic constructor of the class. As a default, parent is set to a nullpointer to QGraphicsItem.

Postcondition

topHighScores is at initialization state.

7.33.3 Member Function Documentation

7.33.3.1 getDuration()

```
int StudentSide::topHighScores::getDuration ( )
```

getDuration reads the game duration setting saved before and returns it as an integer which represents game duration in minutes.

Precondition

-

Returns

Chosen game duration as an integer which represents game duration in minutes. If the player has not chosen game durations, it is by default 2 minutes.

Postcondition

Game duration has been returned. Exception guarantee: nothrow.

7.33.3.2 readFile()

```
void StudentSide::topHighScores::readFile (
            QString filename = textFilePath )
```

readFile reads data from a textfile into map.

Parameters

Default	path to the top10highscores.txt -file where the data is ultimately stored.

Precondition

The textfile exists in the path set.

Postcondition

Data from textfile (playerName: scaledPoints) has been read into a map. Exception guarantee: nothrow.

7.33.3.3 sortAndDisplay()

```
void StudentSide::topHighScores::sortAndDisplay (
            std::map< QString, int > mapToBeSorted )
```

sortAndDisplay sorts the players according to their points into a top10 -subset and appends these top10 -players into a QString which is shown to the player later in GameOverDialog's QTextBrowser.

Parameters

Unsorted	map that contains all-time player data.

Precondition

-

Postcondition

Map is sorted and its contents are appended into a QString. Exception guarantee: nothrow.

7.33.3.4 writeFile()

```
void StudentSide::topHighScores::writeFile (
            QString filename = textFilePath )
```

writeFile writes data into a textfile (player name and his points scaled according to the chosen game duration).

Parameters

Default	path to the top10highscores.txt -file where the data is ultimately stored.

Precondition

-

Postcondition

Data has been written to the textfile (playerName: scaledPoints). Exception guarantee: nothrow.

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

- Game/tophighscores.h
- Game/tophighscores.cpp

Chapter 8

File Documentation

8.1 Course/CourseLib/core/location.hh File Reference

Defines a class that contains methods for handling location. (coordinates)

Classes

- class Interface::Location

  Location is a class, which has methods dealing with the location of the objects.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.1.1 Detailed Description

Defines a class that contains methods for handling location. (coordinates)

8.2 Course/CourseLib/core/logic.hh File Reference

Defines a class that handles the courseside gamelogic.

```
#include "actors/passenger.hh"
#include "actors/nysse.hh"
#include "offlinereader.hh"
#include "interfaces/icity.hh"
#include <list>
#include <QTime>
#include <QTimer>
```

Classes

- class CourseSide::Logic

  The Logic class.

Variables

- const QString **CourseSide::DEFAULT_STOPS_FILE** = ":/offlinedata/offlinedata/full_stations_kkj3.json"
- const QString **CourseSide::DEFAULT_BUSES_FILE** = ":/offlinedata/offlinedata/final_bus_liteN.json"

8.2.1 Detailed Description

Defines a class that handles the courseside gamelogic.

8.3 Course/CourseLib/creategame.hh File Reference

Defines a function that creates the city (Students implement it).

```
#include "interfaces/icity.hh"
#include <memory>
```

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

Functions

- std::shared_ptr< ICity > Interface::createGame ()

  createGame creates the games city and return pointer to it.

8.3.1 Detailed Description

Defines a function that creates the city (Students implement it).

8.4 Course/CourseLib/errors/gameerror.hh File Reference

Defines an exception class for errors ingame.

```
#include <exception>
#include <QString>
```

Classes

- class Interface::GameError

  Exception class that expresses errors ingame.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.4.1 Detailed Description

Defines an exception class for errors ingame.

8.5 Course/CourseLib/errors/initerror.hh File Reference

Defines an exception class for initialization errors.

```
#include <exception>
#include <QString>
```

Classes

- class Interface::InitError

  Exception class that expresses errors during the initialization of the game.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.5.1 Detailed Description

Defines an exception class for initialization errors.

8.6 Course/CourseLib/interfaces/iactor.hh File Reference

Defines a single actor (= an object acting in the game), operations describe the interface.

```
#include "core/location.hh"
```

Classes

- class Interface::IActor

  ActorIF is an interface, which every single actor moving in the game implements.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.6.1 Detailed Description

Defines a single actor (= an object acting in the game), operations describe the interface.

8.7 Course/CourseLib/interfaces/icity.hh File Reference

Defines an interface that reperesents the city's operations.

```
#include "iactor.hh"
#include "istop.hh"
#include <memory>
#include <vector>
#include <QImage>
```

Classes

- class Interface::ICity

  CityIF is an interface that every city in the game must fulfill. Kaupunki.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.7.1 Detailed Description

Defines an interface that reperesents the city's operations.

8.8 Course/CourseLib/interfaces/ipassenger.hh File Reference

Defines interface that represents the passengers operations.

```
#include "iactor.hh"
#include "ivehicle.hh"
```

Classes

- class Interface::IPassenger

  PassengerIF is an interface which every passenger in game implements.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.8.1 Detailed Description

Defines interface that represents the passengers operations.

8.9 Course/CourseLib/interfaces/istatistics.hh File Reference

Defines an interface for scoring statistics.

Classes

- class Interface::IStatistics

  StatisticsIF is an interface, which defines an object that manages scoring statistics.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.9.1 Detailed Description

Defines an interface for scoring statistics.

8.10 Course/CourseLib/interfaces/istop.hh File Reference

Defines an interface that describes stops operations.

```
#include "ipassenger.hh"
#include <memory>
#include <QString>
```

Classes

- class Interface::IStop

  StopIF is an interface that stops fulfill.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.10.1 Detailed Description

Defines an interface that describes stops operations.

8.11 Course/CourseLib/interfaces/ivehicle.hh File Reference

Defines an interface that desribes operations of the vericle.

```
#include "iactor.hh"
#include <string>
#include <vector>
#include <memory>
```

Classes

- class Interface::IVehicle

  VehicleIF is an interface that describes vehicles (nysse) in game.

Namespaces

- Interface

  All of the interfaces defined by the course are found in Interface namespace.

8.11.1 Detailed Description

Defines an interface that desribes operations of the vericle.