

Fisher sim
0.0.1

Generated by Doxygen 1.8.9.1

Sat May 9 2015 15:33:42

Contents

1	Fisher Sim - Introduction	1
2	Algorithms & Data Structures	3
3	Namespace Index	5
4	Hierarchical Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	Namespace Documentation	11
6.1	Ui Namespace Reference	11
7	Class Documentation	13
7.1	Agent Class Reference	13
7.1.1	Constructor & Destructor Documentation	14
7.1.1.1	Agent	14
7.1.2	Member Function Documentation	14
7.1.2.1	adaptNewStrat	14
7.1.2.2	calcThreshold	14
7.1.2.3	getAgentScore	14
7.1.2.4	getBoostScore	14
7.1.2.5	getCommunication	14
7.1.2.6	getDecision	15
7.1.2.7	getEarlyDecision	15
7.1.2.8	getFishingDuration	15
7.1.2.9	getHistory	15
7.1.2.10	getSkill	15
7.1.2.11	getStrat	15

7.1.2.12	getTemp	15
7.1.2.13	getThreshold	15
7.1.2.14	makeDecision	15
7.1.2.15	makeEarlyDecision	16
7.1.2.16	resetAgentScore	16
7.1.2.17	setCommunication	16
7.1.2.18	setFishingduration	16
7.1.2.19	setSkill	16
7.1.2.20	setStrategy	16
7.1.2.21	setTemp	16
7.1.2.22	updateAgentScore	16
7.1.2.23	updateBoostScore	16
7.1.2.24	updateHistory	17
7.1.2.25	updateStrategyScore	17
7.1.3	Member Data Documentation	17
7.1.3.1	agentscore	17
7.1.3.2	boostscore	17
7.1.3.3	communication	17
7.1.3.4	decision	17
7.1.3.5	earlydecision	17
7.1.3.6	fishingduration	17
7.1.3.7	history	17
7.1.3.8	skill	17
7.1.3.9	strat	17
7.1.3.10	temp	17
7.1.3.11	threshold	17
7.2	Dlocation Class Reference	18
7.2.1	Constructor & Destructor Documentation	18
7.2.1.1	Dlocation	18
7.2.2	Member Function Documentation	19
7.2.2.1	getPop	19
7.2.2.2	paintEvent	19
7.2.2.3	setCenter	19
7.2.2.4	setPop	19
7.3	Drawing Class Reference	19
7.3.1	Detailed Description	19
7.3.2	Constructor & Destructor Documentation	19

7.3.2.1	Drawing	19
7.3.3	Member Function Documentation	19
7.3.3.1	DrawLocationPop	20
7.3.3.2	DrawPerson	20
7.3.3.3	getLocationCenterX	21
7.3.3.4	getLocationCenterY	21
7.3.3.5	ReDraw	21
7.3.3.6	SetDay	22
7.3.3.7	SetLocationPop	22
7.3.3.8	SetNumberOfLocations	23
7.3.4	Member Data Documentation	23
7.3.4.1	CurrentDay	23
7.3.4.2	locationPop	23
7.3.4.3	NumberOfLocations	23
7.3.4.4	scene	23
7.3.4.5	view	23
7.4	Graphview Class Reference	23
7.4.1	Detailed Description	25
7.4.2	Constructor & Destructor Documentation	25
7.4.2.1	Graphview	25
7.4.2.2	~Graphview	25
7.4.3	Member Function Documentation	25
7.4.3.1	on_actionInsert_Plot_triggered	25
7.4.3.2	on_actionSave_Document_triggered	25
7.4.3.3	printSettings	26
7.4.3.4	setupPlot	26
7.4.4	Member Data Documentation	26
7.4.4.1	ui	26
7.5	MainWindow Class Reference	27
7.5.1	Constructor & Destructor Documentation	28
7.5.1.1	MainWindow	28
7.5.1.2	~MainWindow	29
7.5.2	Member Function Documentation	29
7.5.2.1	calculateSpot	29
7.5.2.2	log	29
7.5.2.3	on_fishers_valueChanged	30
7.5.2.4	on_fishpop_valueChanged	30

7.5.2.5	on_fishtypes_valueChanged	30
7.5.2.6	on_lineEdit_0_textEdited	30
7.5.2.7	on_lineEdit_1_textEdited	30
7.5.2.8	on_lineEdit_2_textEdited	30
7.5.2.9	on_lineEdit_3_textEdited	30
7.5.2.10	on_lineEdit_4_textEdited	30
7.5.2.11	on_locations_valueChanged	31
7.5.2.12	on_reportButton_clicked	31
7.5.2.13	on_runtime_valueChanged	31
7.5.2.14	on_simulateButton_clicked	31
7.5.2.15	on_weather_clicked	32
7.5.2.16	ResetRealTimeDisplay	32
7.5.2.17	startSimulate	32
7.5.2.18	update	33
7.5.3	Friends And Related Function Documentation	33
7.5.3.1	getSpotppp	33
7.5.3.2	Graphview	33
7.5.4	Member Data Documentation	33
7.5.4.1	drawing_scene	33
7.5.4.2	RealTime	33
7.5.4.3	settings	33
7.5.4.4	simulated	33
7.5.4.5	timer	33
7.5.4.6	ui	33
7.6	Spot Class Reference	33
7.6.1	Detailed Description	34
7.6.2	Constructor & Destructor Documentation	34
7.6.2.1	Spot	34
7.6.3	Member Function Documentation	34
7.6.3.1	getAgentNum	34
7.6.3.2	getAgents	35
7.6.3.3	getSpotCapacity	35
7.6.3.4	setAgentNum	35
7.6.3.5	setCap	36
7.6.4	Member Data Documentation	36
7.6.4.1	agents	36
7.6.4.2	maxcapacity	36

7.6.4.3	numAgent	36
7.7	Strategy Class Reference	36
7.7.1	Constructor & Destructor Documentation	36
7.7.1.1	Strategy	36
7.7.2	Member Function Documentation	36
7.7.2.1	clearLose	36
7.7.2.2	clearWin	37
7.7.2.3	getDecisionPattern	37
7.7.2.4	getLose	37
7.7.2.5	getScore	37
7.7.2.6	getWin	38
7.7.2.7	updateScore	38
7.7.3	Member Data Documentation	38
7.7.3.1	decisionPattern	38
7.7.3.2	lose	38
7.7.3.3	score	38
7.7.3.4	win	38
7.8	UserSettings Class Reference	38
7.8.1	Detailed Description	39
7.8.2	Constructor & Destructor Documentation	39
7.8.2.1	UserSettings	39
7.8.3	Member Function Documentation	39
7.8.3.1	getfisherNum	39
7.8.3.2	getfishLoc	40
7.8.3.3	getfishPop	40
7.8.3.4	getfishTemp	40
7.8.3.5	getfishType	41
7.8.3.6	getRuntime	41
7.8.4	Member Data Documentation	41
7.8.4.1	fisherNum	41
7.8.4.2	fishLoc	41
7.8.4.3	fishPop	41
7.8.4.4	fishTemp	41
7.8.4.5	fishType	42
7.8.4.6	runtime	42

Chapter 1

Fisher Sim - Introduction

Introduction

Fisher Sim is being developed as part of a Software Engineering project at Rutgers University for the spring semester of 2015.

Group 12

Team members:

- Matthew Chatten
- Ameer Fiqri Barahim
- Vicent Vindel Dura
- Alexander Hill
- David Lazaar
- Orielle Joy Yu

Project Goals

The Fisher Sim project seeks to build off of the classic El Farol Bar problem in game theory. In the El Farol Bar problem models for decisions that a based on others are examined. In the original formulation, the question is whether or not to go to a bar. Going to the bar is a good decision only if most people decide it is a bad decision, and vice versa.

Fisher Sim adds additional metrics to this problem in an attempt to better understand and predict people's disision to go fishing.

Compiling the software

Fisher sim currently consists of two separate programs. The primary component is located under the CrowdAnalysys folder in in the project root directory. This folder contains the main project as a QT application along with the technical documentation (this file). The other components of the Fisher sim program are located under the /spot and /Agent folders. These folders contain work on the simulation engine and contain basic console c++ applications. They are currently separated from the primary GUI application in order to simplify debugging.

To build the primary application you will need a working installation of the QT creator framework. The community edition obtained for free from their website located here: <https://www.qt.io/download/> In addition to QT creator, you will need a c++ compiler for your system. If you do not already have a compiler installed and are on a Windows system then a suitable compiler can be obtained by installing a version of Microsoft's visual studio express. On Debian Linux systems, a c++ compiler can be installed by installing the buildutils package from your package manager.

Updating Documentation

Technical documentation is maintained through the Doxygen tool by loading the Doxyfile located under /Crowd↵ Analysys/docs. Using Doxygen allows for the documentation to be included along with the code which can assist in keeping things up to date. When changes to the code / documentation are made the Doxygen tool must be run to rebuild the Technical Documentation. This will create an additional 2 folders in the docs folder each one containing an html edition and the other containing a Latex / pdf version.

If you wish to build the pdf version you will need an installation of latex on your system and to have its binaries in your system path. Linux editions of latex can be installed through the package manager and a windows edition can be obtained from the Miktex project located at <http://miktex.org/>. In order to generate class relation images your system will need GraphViz installed.

Tools needed summery

Software Build

- MSVS or GNU Build system
- Qt Creator

Documentation Build

- Doxygen
- Latex
- GraphViz

Adding Documentation

Documentation can be added in two general styles. Most documentation will mostly be general explanations for programming constructs which can be added as explained <http://www.stack.nl/~dimitri/doxygen/manual/docblocks.↵html>.

More extensive comments can take advantage of Markdown formatting and Latex style mathematical expressions. Supported markdown formatting can be seen here: <http://www.stack.nl/~dimitri/doxygen/manual/markdown.↵html>.

Chapter 2

Algorithms & Data Structures

Algorithms

Decision Making

The algorithm is made to compute a unique decision for every agent. The decision is either to go fishing (denoted as 1) or stay at home (denoted as -1). At first every decision of an agent is randomly chosen from a random strategy. Then, every decision may change by the percentage of influence threshold, p . The decision is determined using the logic below:

```
if p < 70
    decision that is made by the strategy is kept.
else if p > 70
    decision will be change to 1-go to fishing.
```

The value of influence threshold depends on the factors below:

- Skill and experience rank
- Frequency of communication
- Amount of each type of fish
- Fishing duration
- Weather pattern

Since some of the factors above are unique for each agents, it will be able to preserve the uniqueness of every decision. Every factors will contribute 20% to the influence threshold.

Strategy

Every agent will have a short-term memory and a long-term memory. Short-term memory is limited to 3 previous outcomes of the agent winning and losing. Long-term memory is the strategy that is used by the agent to make the initial decision before taking into account of influence threshold.

Since there are 8 possible outcomes from the short-term memory, the strategy that can be generated from these outcome is 256. Every agent is allow to have 3 strategies, this will result in 2,763,520 different combinations of strategies. Every agent will get a random combination of 3 strategies and it will be likely that every combination is unique.

The process to make the early decision is shown below: strategy=choose the strategy that has a higher score

```

for i=1 →8
    if history==strategy history [i]
        early decision=strategy action [i]
    return early decision

```

At the beginning of every simulation, all the strategies' score are zero. So, it can be conclude that the initial strategy of every agent is random. If the agent won the round the strategy score will increase by one point. Conversely, every losing round the strategy score is lowered by one. The early decision will be passed to the decision making where the influence threshold of the agent will be calculated and the early decision may be changed.

Overall process

Below is the overall process of how every decision of an agent being made:

```

strategy=choose the strategy that has a higher score
for i=1 →8
    if history==strategy history [i]
        early decision=strategy action [i]
if p<70
    decision=early decision
else if p>70
    decision=1

```

Strategy score will be calculated when all the decisions have been made. Plus for a strategy to earn the score the decision must not be changed by the influence threshold. The logic is shown below:

```

if p<70
    if majority go to fishing and decision == -1
        strategy score increase by one point
    else strategy score lower by one point
    if majority stay at home and decision==1
        strategy score increase by one point
    else strategy score lower by one point

```

Chapter 3

Namespace Index

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Agent	13
Drawing	19
QMainWindow	
Graphview	23
MainWindow	27
QWidget	
Dlocation	18
Spot	33
Strategy	36
UserSettings	38
MainWindow	27

Chapter 5

Class Index

5.1 Class List

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

Agent	13
Dlocation	18
Drawing	
Provides a method of drawing locations and the number of people in each location	19
Graphview	
View that shows the collected graphs and allows them to be inserted into a report	23
MainWindow	27
Spot	
Spot is used to create a location and calculate how crowded it is	33
Strategy	36
UserSettings	
Users global simulation parameters	38

Chapter 6

Namespace Documentation

6.1 Ui Namespace Reference

Chapter 7

Class Documentation

7.1 Agent Class Reference

Public Member Functions

- [Agent](#) (vector< [Strategy](#) * > newStrat, int newSkill, int newCom)
- void [updateStrategyScore](#) (int majorityScore)
- void [calcThreshold](#) ()
- void [makeEarlyDecision](#) ()
- void [makeDecision](#) ()
- void [updateHistory](#) ()
- void [updateBoostScore](#) (int index, int score)
- void [adaptNewStrat](#) ()
- void [updateAgentScore](#) (int majorityScore)
- void [resetAgentScore](#) ()
- void [setTemp](#) (float newTemp)
- void [setSkill](#) (int newskill)
- void [setFishingduration](#) (int newFishingDuration)
- void [setCommunication](#) (int newCommunication)
- void [setStrategy](#) (vector< [Strategy](#) * > newStrat)
- vector< int > [getHistory](#) ()
- int [getDecision](#) ()
- int [getCommunication](#) ()
- int [getSkill](#) ()
- float [getTemp](#) ()
- int [getFishingDuration](#) ()
- int [getEarlyDecision](#) ()
- float [getThreshold](#) ()
- int [getBoostScore](#) (int index)
- int [getAgentScore](#) (int index)
- vector< [Strategy](#) * > [getStrat](#) ()

Private Attributes

- vector< Strategy * > strat
- vector< int > history
- int decision
- int earlydecision
- int skill
- int fishingduration
- float temp
- int communication
- float threshold
- int boostscore [2]
- int agentscore [2]

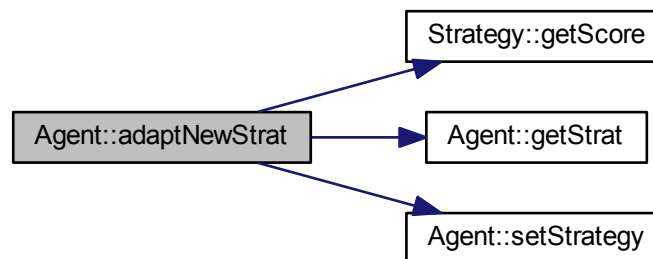
7.1.1 Constructor & Destructor Documentation

7.1.1.1 Agent::Agent (vector< Strategy * > newStrat, int newSkill, int newCom)

7.1.2 Member Function Documentation

7.1.2.1 void Agent::adaptNewStrat ()

Here is the call graph for this function:



7.1.2.2 void Agent::calcThreshold ()

7.1.2.3 int Agent::getAgentScore (int index)

7.1.2.4 int Agent::getBoostScore (int index)

7.1.2.5 int Agent::getCommunication ()

7.1.2.6 int Agent::getDecision ()

Here is the caller graph for this function:



7.1.2.7 int Agent::getEarlyDecision ()

7.1.2.8 int Agent::getFishingDuration ()

7.1.2.9 vector< int > Agent::getHistory ()

7.1.2.10 int Agent::getSkill ()

7.1.2.11 vector< Strategy * > Agent::getStrat ()

Here is the caller graph for this function:



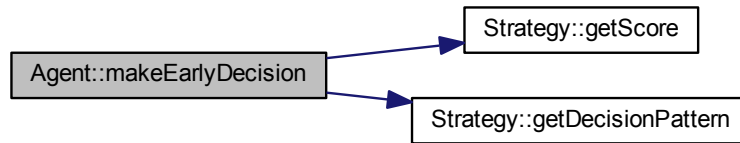
7.1.2.12 float Agent::getTemp ()

7.1.2.13 float Agent::getThreshold ()

7.1.2.14 void Agent::makeDecision ()

7.1.2.15 void Agent::makeEarlyDecision ()

Here is the call graph for this function:



7.1.2.16 void Agent::resetAgentScore ()

7.1.2.17 void Agent::setCommunication (int *newCommunication*)

7.1.2.18 void Agent::setFishingduration (int *newFishingDuration*)

7.1.2.19 void Agent::setSkill (int *newskill*)

7.1.2.20 void Agent::setStrategy (vector< Strategy * > *newStrat*)

Here is the caller graph for this function:



7.1.2.21 void Agent::setTemp (float *newTemp*)

Sets the temperature of the water

Parameters

<i>newTemp</i>	the new temperature in degrees celsius
----------------	--

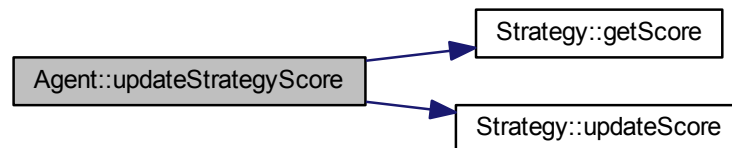
7.1.2.22 void Agent::updateAgentScore (int *majorityScore*)

7.1.2.23 void Agent::updateBoostScore (int *index*, int *score*)

7.1.2.24 `void Agent::updateHistory ()`

7.1.2.25 `void Agent::updateStrategyScore (int majorityScore)`

Here is the call graph for this function:



7.1.3 Member Data Documentation

7.1.3.1 `int Agent::agentscore[2] [private]`

7.1.3.2 `int Agent::boostscore[2] [private]`

7.1.3.3 `int Agent::communication [private]`

7.1.3.4 `int Agent::decision [private]`

7.1.3.5 `int Agent::earlydecison [private]`

7.1.3.6 `int Agent::fishingduration [private]`

7.1.3.7 `vector<int> Agent::history [private]`

7.1.3.8 `int Agent::skill [private]`

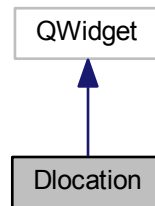
7.1.3.9 `vector<Strategy*> Agent::strat [private]`

7.1.3.10 `float Agent::temp [private]`

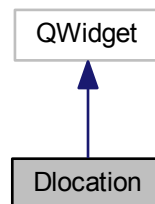
7.1.3.11 `float Agent::threshold [private]`

7.2 Dlocation Class Reference

Inheritance diagram for Dlocation:



Collaboration diagram for Dlocation:



Public Member Functions

- [Dlocation](#) (QWidget *parent)
- bool [setPop](#) (int number)
- int [getPop](#) (void)
- bool [setCenter](#) (int x, int y)

Protected Member Functions

- void [paintEvent](#) (QPaintEvent *event) Q_DECL_OVERRIDE

7.2.1 Constructor & Destructor Documentation

7.2.1.1 Dlocation::Dlocation (QWidget * parent)

7.2.2 Member Function Documentation

7.2.2.1 `int Dlocation::getPop (void)`

7.2.2.2 `void Dlocation::paintEvent (QPaintEvent * event)` [protected]

7.2.2.3 `bool Dlocation::setCenter (int x, int y)`

7.2.2.4 `bool Dlocation::setPop (int number)`

7.3 Drawing Class Reference

Provides a method of drawing locations and the number of people in each location.

Public Member Functions

- [Drawing](#) (QGraphicsView *graphics_view, QGraphicsScene *drawing_scene)
- void [SetNumberOfLocations](#) (int number)
- void [DrawPerson](#) (int x, int y)
- double [getLocationCenterX](#) (int location)
- double [getLocationCenterY](#) (int location)
- void [SetLocationPop](#) (int location, int pop)
- void [DrawLocationPop](#) (int location)
- void [ReDraw](#) ()
- void [SetDay](#) (int day)

Public Attributes

- QGraphicsScene * [scene](#)
- QGraphicsView * [view](#)
- int [NumberOfLocations](#)
- int [locationPop](#) [10]
- int [CurrentDay](#)

7.3.1 Detailed Description

Provides a method of drawing locations and the number of people in each location.

[Drawing](#) is used by the main window to create the real time display. It provides methods to get and set the number of locations and the number of people at each location. Setting different values is not updated on the screen without calling [ReDraw](#).

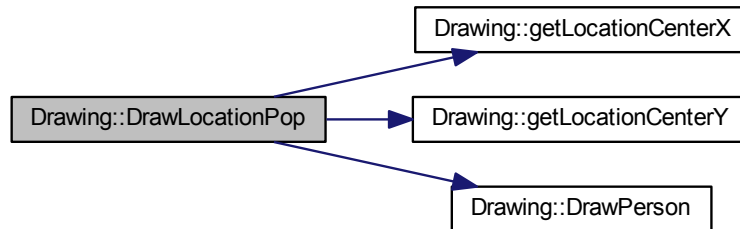
7.3.2 Constructor & Destructor Documentation

7.3.2.1 `Drawing::Drawing (QGraphicsView * graphics_view, QGraphicsScene * drawing_scene)`

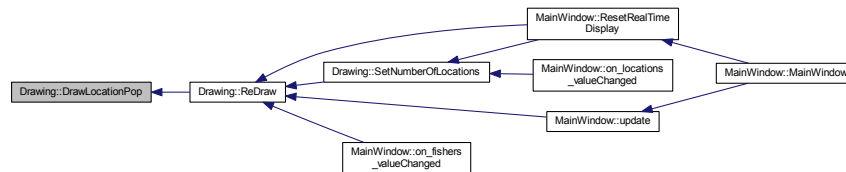
7.3.3 Member Function Documentation

7.3.3.1 void Drawing::DrawLocationPop (int *location*)

Here is the call graph for this function:

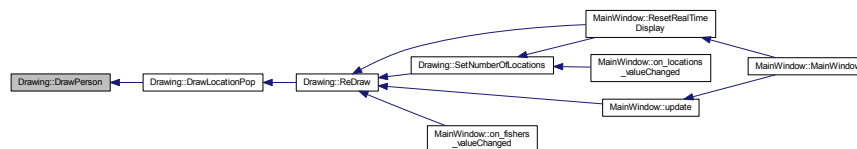


Here is the caller graph for this function:



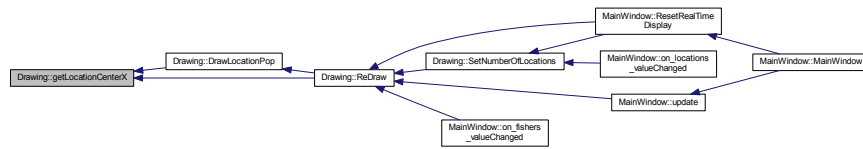
7.3.3.2 void Drawing::DrawPerson (int *x*, int *y*)

Here is the caller graph for this function:



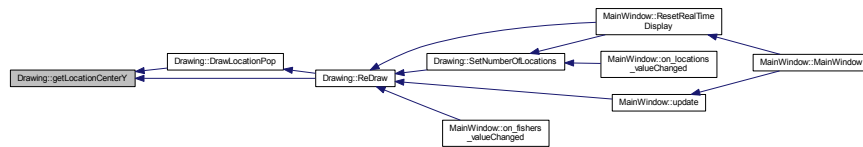
7.3.3.3 double Drawing::getLocationCenterX (int *location*)

Here is the caller graph for this function:



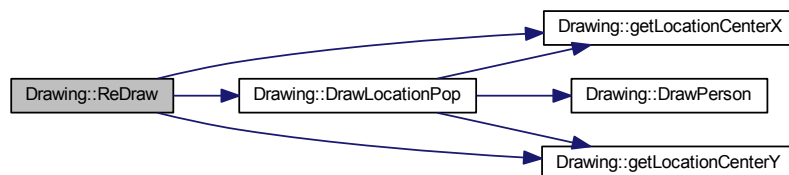
7.3.3.4 double Drawing::getLocationCenterY (int *location*)

Here is the caller graph for this function:

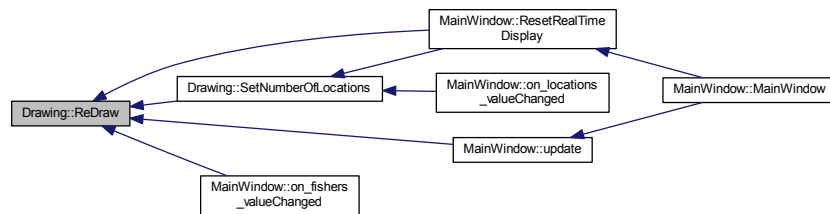


7.3.3.5 void Drawing::ReDraw ()

Here is the call graph for this function:

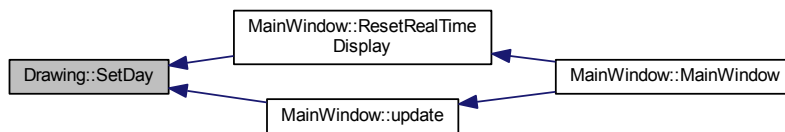


Here is the caller graph for this function:



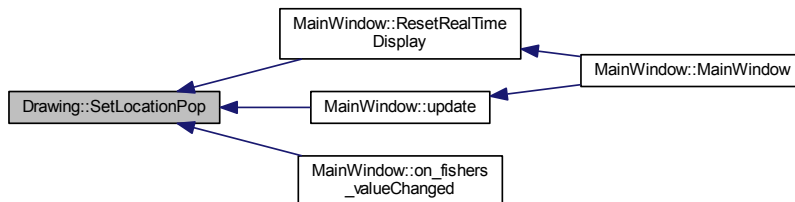
7.3.3.6 void Drawing::SetDay (int day)

Here is the caller graph for this function:



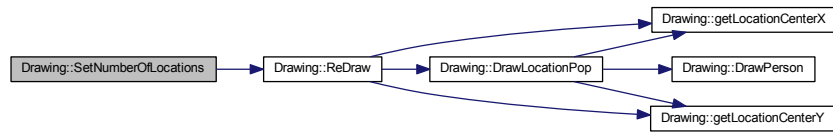
7.3.3.7 void Drawing::SetLocationPop (int location, int pop)

Here is the caller graph for this function:

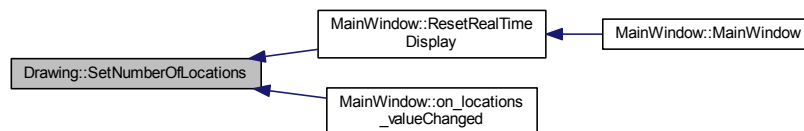


7.3.3.8 void Drawing::SetNumberOfLocations (int *number*)

Here is the call graph for this function:



Here is the caller graph for this function:



7.3.4 Member Data Documentation

7.3.4.1 int Drawing::CurrentDay

7.3.4.2 int Drawing::locationPop[10]

7.3.4.3 int Drawing::NumberOfLocations

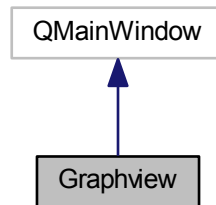
7.3.4.4 QGraphicsScene* Drawing::scene

7.3.4.5 QGraphicsView* Drawing::view

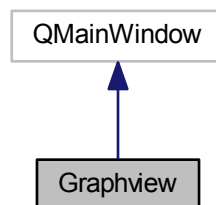
7.4 Graphview Class Reference

provides a view that shows the collected graphs and allows them to be inserted into a report.

Inheritance diagram for Graphview:



Collaboration diagram for Graphview:



Public Member Functions

- [Graphview](#) (`QWidget *parent=0`)
constructor for the [Graphview](#) class
- [~Graphview](#) ()
- void [setupPlot](#) ()
setupPlot
- void [printSettings](#) (`QString s`)

Private Slots

- void [on_actionInsert_Plot_triggered](#) ()
[Graphview::on_actionInsert_Plot_triggered](#).
- void [on_actionSave_Document_triggered](#) ()

Private Attributes

- `Ui::Graphview * ui`

7.4.1 Detailed Description

provides a view that shows the collected graphs and allows them to be inserted into a report.

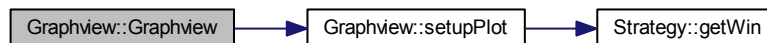
[Graphview](#) is intended to be used after the simulation has finished. It will accept data from the simulation module defining plots and display them to the users. There is also a report view on the left side that allows users to insert selected graphs to compile a final report.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `Graphview::Graphview (QWidget * parent = 0) [explicit]`

constructor for the [Graphview](#) class

Here is the call graph for this function:



7.4.2.2 `Graphview::~~Graphview ()`

7.4.3 Member Function Documentation

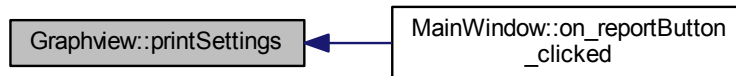
7.4.3.1 `void Graphview::on_actionInsert_Plot_triggered () [private], [slot]`

[Graphview::on_actionInsert_Plot_triggered.](#)

7.4.3.2 `void Graphview::on_actionSave_Document_triggered () [private], [slot]`

7.4.3.3 void Graphview::printSettings (QString s)

Here is the caller graph for this function:



7.4.3.4 void Graphview::setupPlot ()

`setupPlot`

configures the plots

Here is the call graph for this function:



Here is the caller graph for this function:

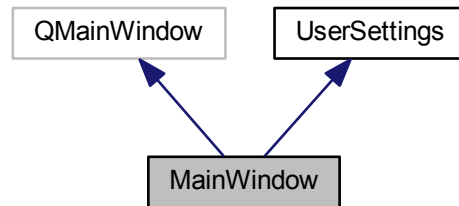


7.4.4 Member Data Documentation

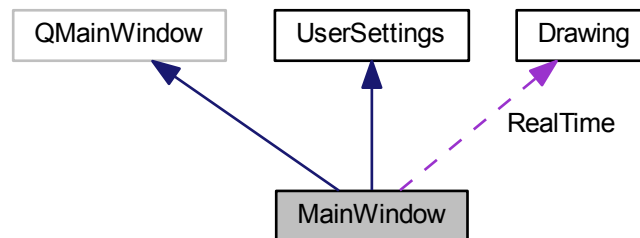
7.4.4.1 Ui::Graphview* Graphview::ui [private]

7.5 MainWindow Class Reference

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Member Functions

- `MainWindow` (`QWidget *parent=0`)
- `~MainWindow` ()
- `void log` (`const QString &text`)
Sends a string to the simulation log.

Protected Attributes

- `Ui::MainWindow * ui`
- `QString settings`
- `bool simulated = false`
- `QTimer * timer`

Private Slots

- void [on_fishers_valueChanged](#) (int value)
- void [on_locations_valueChanged](#) (int value)
- void [on_fishtypes_valueChanged](#) (int value)
- void [on_fishpop_valueChanged](#) (int value)
- void [on_runtime_valueChanged](#) (int value)
- void [on_lineEdit_0_textEdited](#) (const QString &arg1)
- void [on_lineEdit_1_textEdited](#) (const QString &arg1)
- void [on_lineEdit_2_textEdited](#) (const QString &arg1)
- void [on_lineEdit_3_textEdited](#) (const QString &arg1)
- void [on_lineEdit_4_textEdited](#) (const QString &arg1)
- void [on_weather_clicked](#) ()
- void [on_reportButton_clicked](#) ()
- void [on_simulateButton_clicked](#) ()
- void [startSimulate](#) (int fisherNum, int fishLoc, int fishType, int fishPop, int fishTemp, int runtime)
- void [calculateSpot](#) (int fisherNum, int fishLoc)
- void [update](#) ()
- void [ResetRealTimeDisplay](#) (void)

Private Attributes

- QGraphicsScene * [drawing_scene](#)
- [Drawing](#) * [RealTime](#)

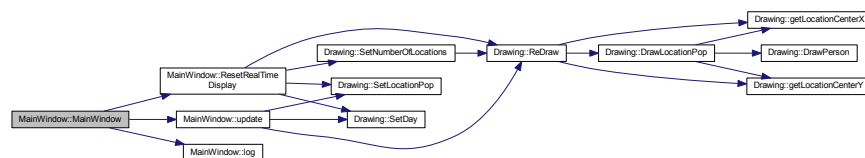
Friends

- class [Graphview](#)
- double [getSpotppp](#) ()

7.5.1 Constructor & Destructor Documentation

7.5.1.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

Here is the call graph for this function:

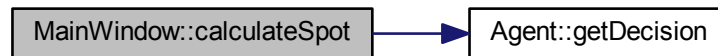


7.5.1.2 MainWindow::~MainWindow ()

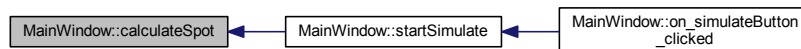
7.5.2 Member Function Documentation

7.5.2.1 void MainWindow::calculateSpot (int *fisherNum*, int *fishLoc*) [private],[slot]

Here is the call graph for this function:



Here is the caller graph for this function:

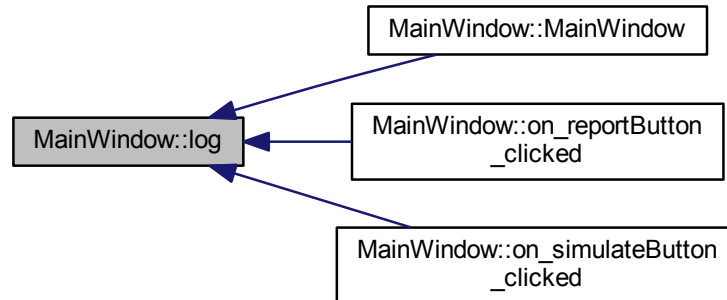
7.5.2.2 void MainWindow::log (const QString & *text*)

Sends a string to the simulation log.

Parameters

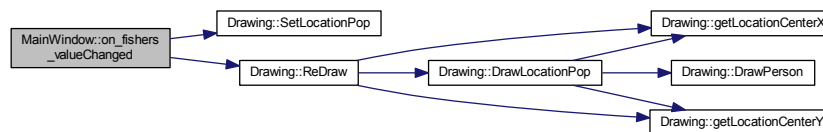
<i>text</i>	to display in the log.
-------------	------------------------

Here is the caller graph for this function:



7.5.2.3 `void MainWindow::on_fishers_valueChanged (int value)` [private],[slot]

Here is the call graph for this function:



7.5.2.4 `void MainWindow::on_fishpop_valueChanged (int value)` [private],[slot]

7.5.2.5 `void MainWindow::on_fishtypes_valueChanged (int value)` [private],[slot]

7.5.2.6 `void MainWindow::on_lineEdit_0_textEdited (const QString & arg1)` [private],[slot]

7.5.2.7 `void MainWindow::on_lineEdit_1_textEdited (const QString & arg1)` [private],[slot]

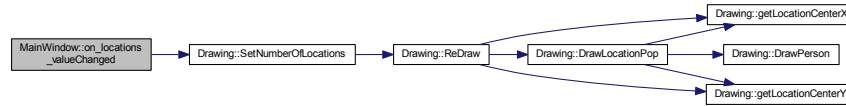
7.5.2.8 `void MainWindow::on_lineEdit_2_textEdited (const QString & arg1)` [private],[slot]

7.5.2.9 `void MainWindow::on_lineEdit_3_textEdited (const QString & arg1)` [private],[slot]

7.5.2.10 `void MainWindow::on_lineEdit_4_textEdited (const QString & arg1)` [private],[slot]

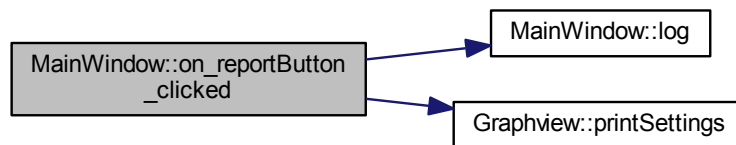
7.5.2.11 void MainWindow::on_locations_valueChanged (int value) [private],[slot]

Here is the call graph for this function:



7.5.2.12 void MainWindow::on_reportButton_clicked () [private],[slot]

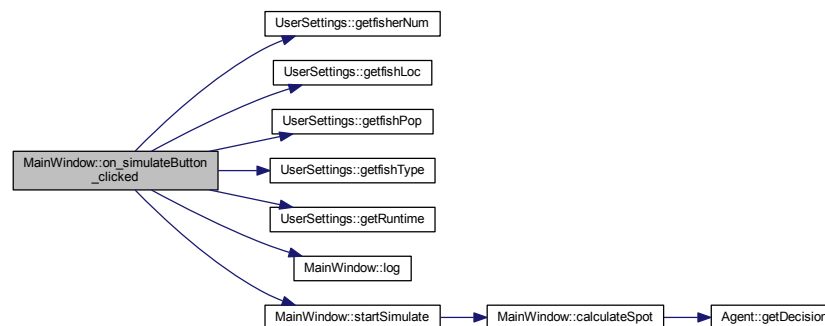
Here is the call graph for this function:



7.5.2.13 void MainWindow::on_runtime_valueChanged (int value) [private],[slot]

7.5.2.14 void MainWindow::on_simulateButton_clicked () [private],[slot]

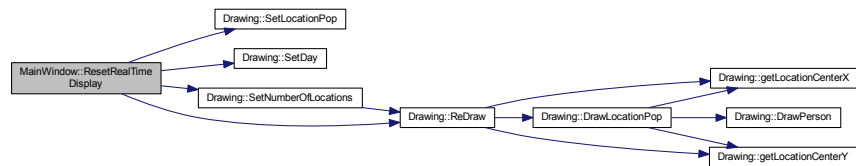
Here is the call graph for this function:



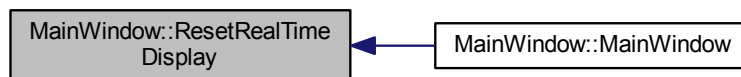
7.5.2.15 `void MainWindow::on_weather_clicked () [private],[slot]`

7.5.2.16 `void MainWindow::ResetRealTimeDisplay (void) [private],[slot]`

Here is the call graph for this function:

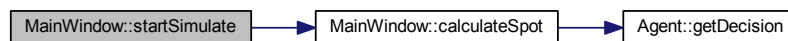


Here is the caller graph for this function:

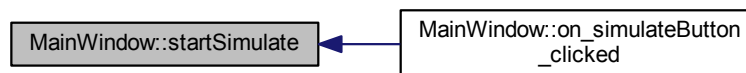


7.5.2.17 `void MainWindow::startSimulate (int fisherNum, int fishLoc, int fishType, int fishPop, int fishTemp, int runtime) [private],[slot]`

Here is the call graph for this function:

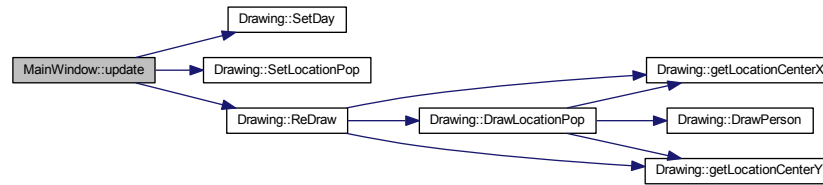


Here is the caller graph for this function:



7.5.2.18 void MainWindow::update () [private],[slot]

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3 Friends And Related Function Documentation

7.5.3.1 double getSpotppp () [friend]

7.5.3.2 friend class Graphview [friend]

7.5.4 Member Data Documentation

7.5.4.1 QGraphicsScene* MainWindow::drawing_scene [private]

7.5.4.2 Drawing* MainWindow::RealTime [private]

7.5.4.3 QString MainWindow::settings [protected]

7.5.4.4 bool MainWindow::simulated = false [protected]

7.5.4.5 QTimer* MainWindow::timer [protected]

7.5.4.6 Ui::MainWindow* MainWindow::ui [protected]

7.6 Spot Class Reference

[Spot](#) is used to create a location and calculate how crowded it is.

Public Member Functions

- [Spot](#) (list< [Agent](#) * > newAgents)
- void [setCap](#) (double cap)
- double [getSpotCapacity](#) ()
- void [setAgentNum](#) (int fisherNum)
- int [getAgentNum](#) ()
- list< [Agent](#) * > [getAgents](#) ()

Private Attributes

- double [maxcapacity](#)
max agents per spot
- int [numAgent](#)
number of agents possibly going fishing per spo
- list< [Agent](#) * > [agents](#)
list of agents

7.6.1 Detailed Description

[Spot](#) is used to create a location and calculate how crowded it is.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 [Spot::Spot](#) (list< [Agent](#) * > *newAgents*)

Constructor for a spot

Precondition

A list of agents

Postcondition

numAgent and maxcapacity is initialized to zero, sets a new list of agents

Returns

none

7.6.3 Member Function Documentation

7.6.3.1 [int Spot::getAgentNum](#) ()

Get back total number agents

Precondition

Number of agents is already set

Postcondition

none

Returns

Integer number of total agents (numAgent)

7.6.3.2 list< Agent * > Spot::getAgents ()**Precondition**

A [Spot](#) wa initialized

Postcondition

Creates a list of agents

Returns

List of agents

7.6.3.3 double Spot::getSpotCapacity ()

Get the maxcapacity

Precondition

A capacity is already set

Postcondition

none

Returns

The number of maxcapacity

7.6.3.4 void Spot::setAgentNum (int *fisherNum*)

Set the number of agents possibly going to spot

Precondition

Give the number of agents called fisherNum

Postcondition

The numAgent is equal to given fisherNum

Returns

none

7.6.3.5 void Spot::setCap (double *cap*)

7.6.4 Member Data Documentation

7.6.4.1 list<Agent *> Spot::agents [private]

list of agents

7.6.4.2 double Spot::maxcapacity [private]

max agents per spot

7.6.4.3 int Spot::numAgent [private]

number of agents possibly going fishing per spo

7.7 Strategy Class Reference

Public Member Functions

- [Strategy](#) (vector< int > randDecision)
- void [updateScore](#) (int point)
records the seccess of this strategy
- vector< int > [getDecisionPattern](#) ()
- int [getScore](#) ()
returns the score This value represents the number of wins that an agent has made using this strategy
- int [getWin](#) ()
- int [getLose](#) ()
- void [clearWin](#) ()
- void [clearLose](#) ()

Private Attributes

- int [score](#)
- int [win](#)
- int [lose](#)
- vector< int > [decisionPattern](#)
brief represents a strategy for determining the conditions of going fishing.

7.7.1 Constructor & Destructor Documentation

7.7.1.1 Strategy::Strategy (vector< int > *randDecision*)

7.7.2 Member Function Documentation

7.7.2.1 void Strategy::clearLose ()

7.7.2.2 void Strategy::clearWin ()

7.7.2.3 vector< int > Strategy::getDecisionPattern ()

Returns the decision pattern used by this strategy

return a vector containing the decision pattern used. 0 represents staying home and 1 going fishing.

Here is the caller graph for this function:



7.7.2.4 int Strategy::getLose ()

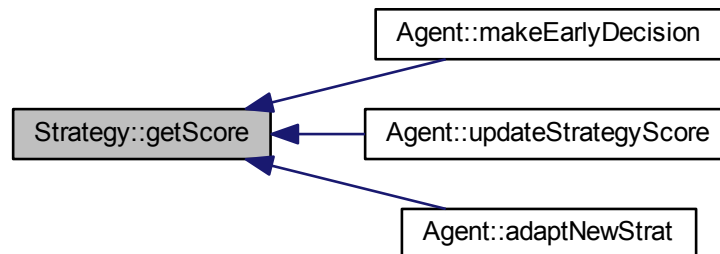
7.7.2.5 int Strategy::getScore ()

returns the score This value represents the number of wins that an agent has made using this strategy

Returns

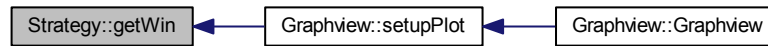
the strategy score

Here is the caller graph for this function:



7.7.2.6 `int Strategy::getWin ()`

Here is the caller graph for this function:



7.7.2.7 `void Strategy::updateScore (int point)`

records the secess of this strategy

Here is the caller graph for this function:



7.7.3 Member Data Documentation

7.7.3.1 `vector<int> Strategy::decisionPattern [private]`

brief represents a strategy for determining the conditions of going fishing.

7.7.3.2 `int Strategy::lose [private]`

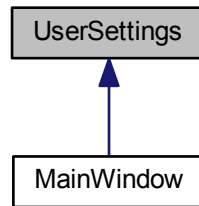
7.7.3.3 `int Strategy::score [private]`

7.7.3.4 `int Strategy::win [private]`

7.8 UserSettings Class Reference

contains the users global simulation parameters.

Inheritance diagram for UserSettings:



Public Member Functions

- [UserSettings](#) ()
- [int getfisherNum](#) ()
- [int getfishLoc](#) ()
- [int getfishType](#) ()
- [int getfishPop](#) ()
- [int getfishTemp](#) ()
- [int getRuntime](#) ()

Protected Attributes

- [int fisherNum](#)
- [int fishLoc](#)
- [int fishType](#)
- [int fishPop](#)
- [int fishTemp](#)
- [int runtime](#)

7.8.1 Detailed Description

contains the users global simulation parameters.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 [UserSettings::UserSettings](#) ()

7.8.3 Member Function Documentation

7.8.3.1 [int UserSettings::getfisherNum](#) ()

Returns the number of Fishers to use in the simulation

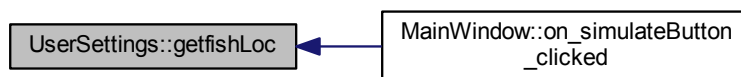
Here is the caller graph for this function:



7.8.3.2 `int UserSettings::getfishLoc ()`

Returns the number of different locations

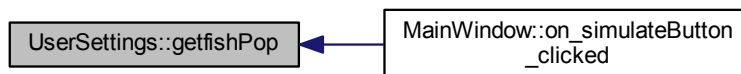
Here is the caller graph for this function:



7.8.3.3 `int UserSettings::getfishPop ()`

Returns the initial population of fish when the simulation starts.

Here is the caller graph for this function:



7.8.3.4 `int UserSettings::getfishTemp ()`

Returns the conditions: overcast, snow, rain.

7.8.3.5 int UserSettings::getfishType ()

Returns the number of fish types.

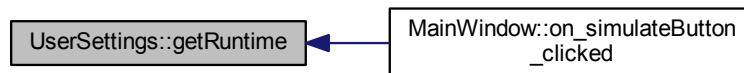
Here is the caller graph for this function:



7.8.3.6 int UserSettings::getRuntime ()

Returns the number of days to run the simulation.

Here is the caller graph for this function:



7.8.4 Member Data Documentation

7.8.4.1 int UserSettings::fisherNum [protected]

The number of Fishers to use in the simulation

7.8.4.2 int UserSettings::fishLoc [protected]

The number of different locations

7.8.4.3 int UserSettings::fishPop [protected]

The initial population of fish when the simulation starts.

7.8.4.4 int UserSettings::fishTemp [protected]

The conditions: overcast, snow, rain

7.8.4.5 `int UserSettings::fishType` `[protected]`

The number of fish types.

7.8.4.6 `int UserSettings::runtime` `[protected]`

The number of days to run the simulation

Index

- ~Graphview
 - Graphview, [25](#)
- ~MainWindow
 - MainWindow, [28](#)
- adaptNewStrat
 - Agent, [14](#)
- Agent, [13](#)
 - adaptNewStrat, [14](#)
 - Agent, [14](#)
 - agentscore, [17](#)
 - boostscore, [17](#)
 - calcThreshold, [14](#)
 - communication, [17](#)
 - decision, [17](#)
 - earlydecision, [17](#)
 - fishingduration, [17](#)
 - getAgentScore, [14](#)
 - getBoostScore, [14](#)
 - getCommunication, [14](#)
 - getDecision, [14](#)
 - getEarlyDecision, [15](#)
 - getFishingDuration, [15](#)
 - getHistory, [15](#)
 - getSkill, [15](#)
 - getStrat, [15](#)
 - getTemp, [15](#)
 - getThreshold, [15](#)
 - history, [17](#)
 - makeDecision, [15](#)
 - makeEarlyDecision, [15](#)
 - resetAgentScore, [16](#)
 - setCommunication, [16](#)
 - setFishingduration, [16](#)
 - setSkill, [16](#)
 - setStrategy, [16](#)
 - setTemp, [16](#)
 - skill, [17](#)
 - strat, [17](#)
 - temp, [17](#)
 - threshold, [17](#)
 - updateAgentScore, [16](#)
 - updateBoostScore, [16](#)
 - updateHistory, [16](#)
 - updateStrategyScore, [17](#)
- agents

- Spot, [36](#)
- agentscore
 - Agent, [17](#)
- boostscore
 - Agent, [17](#)
- calcThreshold
 - Agent, [14](#)
- calculateSpot
 - MainWindow, [29](#)
- clearLose
 - Strategy, [36](#)
- clearWin
 - Strategy, [36](#)
- communication
 - Agent, [17](#)
- CurrentDay
 - Drawing, [23](#)
- decision
 - Agent, [17](#)
- decisionPattern
 - Strategy, [38](#)
- Dlocation, [18](#)
 - Dlocation, [18](#)
 - getPop, [19](#)
 - paintEvent, [19](#)
 - setCenter, [19](#)
 - setPop, [19](#)
- DrawLocationPop
 - Drawing, [19](#)
- DrawPerson
 - Drawing, [20](#)
- Drawing, [19](#)
 - CurrentDay, [23](#)
 - DrawLocationPop, [19](#)
 - DrawPerson, [20](#)
 - Drawing, [19](#)
 - getLocationCenterX, [20](#)
 - getLocationCenterY, [21](#)
 - locationPop, [23](#)
 - NumberOfLocations, [23](#)
 - ReDraw, [21](#)
 - scene, [23](#)
 - SetDay, [22](#)

- SetLocationPop, 22
 - SetNumberOfLocations, 22
 - view, 23
- drawing_scene
 - MainWindow, 33
- earlydecision
 - Agent, 17
- fishLoc
 - UserSettings, 41
- fishPop
 - UserSettings, 41
- fishTemp
 - UserSettings, 41
- fishType
 - UserSettings, 41
- fisherNum
 - UserSettings, 41
- fishingduration
 - Agent, 17
- getAgentNum
 - Spot, 34
- getAgentScore
 - Agent, 14
- getAgents
 - Spot, 35
- getBoostScore
 - Agent, 14
- getCommunication
 - Agent, 14
- getDecision
 - Agent, 14
- getDecisionPattern
 - Strategy, 37
- getEarlyDecision
 - Agent, 15
- getFishingDuration
 - Agent, 15
- getHistory
 - Agent, 15
- getLocationCenterX
 - Drawing, 20
- getLocationCenterY
 - Drawing, 21
- getLose
 - Strategy, 37
- getPop
 - Dlocation, 19
- getRuntime
 - UserSettings, 41
- getScore
 - Strategy, 37
- getSkill
 - Agent, 15
- getSpotCapacity
 - Spot, 35
- getSpotppp
 - MainWindow, 33
- getStrat
 - Agent, 15
- getTemp
 - Agent, 15
- getThreshold
 - Agent, 15
- getWin
 - Strategy, 37
- getfishLoc
 - UserSettings, 40
- getfishPop
 - UserSettings, 40
- getfishTemp
 - UserSettings, 40
- getfishType
 - UserSettings, 40
- getfisherNum
 - UserSettings, 39
- Graphview, 23
 - ~Graphview, 25
 - Graphview, 25
 - MainWindow, 33
 - on_actionInsert_Plot_triggered, 25
 - on_actionSave_Document_triggered, 25
 - printSettings, 25
 - setupPlot, 26
 - ui, 26
- history
 - Agent, 17
- locationPop
 - Drawing, 23
- log
 - MainWindow, 29
- lose
 - Strategy, 38
- MainWindow, 27
 - ~MainWindow, 28
 - calculateSpot, 29
 - drawing_scene, 33
 - getSpotppp, 33
 - Graphview, 33
 - log, 29
 - MainWindow, 28
 - on_fishers_valueChanged, 30
 - on_fishpop_valueChanged, 30
 - on_fishtypes_valueChanged, 30
 - on_lineEdit_0_textEdited, 30

- on_lineEdit_1_textEdited, 30
- on_lineEdit_2_textEdited, 30
- on_lineEdit_3_textEdited, 30
- on_lineEdit_4_textEdited, 30
- on_locations_valueChanged, 30
- on_reportButton_clicked, 31
- on_runtime_valueChanged, 31
- on_simulateButton_clicked, 31
- on_weather_clicked, 31
- RealTime, 33
- ResetRealTimeDisplay, 32
- settings, 33
- simulated, 33
- startSimulate, 32
- timer, 33
- ui, 33
- update, 32
- makeDecision
 - Agent, 15
- makeEarlyDecision
 - Agent, 15
- maxcapacity
 - Spot, 36
- numAgent
 - Spot, 36
- NumberOfLocations
 - Drawing, 23
- on_actionInsert_Plot_triggered
 - Graphview, 25
- on_actionSave_Document_triggered
 - Graphview, 25
- on_fishers_valueChanged
 - MainWindow, 30
- on_fishpop_valueChanged
 - MainWindow, 30
- on_fishtypes_valueChanged
 - MainWindow, 30
- on_lineEdit_0_textEdited
 - MainWindow, 30
- on_lineEdit_1_textEdited
 - MainWindow, 30
- on_lineEdit_2_textEdited
 - MainWindow, 30
- on_lineEdit_3_textEdited
 - MainWindow, 30
- on_lineEdit_4_textEdited
 - MainWindow, 30
- on_locations_valueChanged
 - MainWindow, 30
- on_reportButton_clicked
 - MainWindow, 31
- on_runtime_valueChanged
 - MainWindow, 31
- on_simulateButton_clicked
 - MainWindow, 31
- on_weather_clicked
 - MainWindow, 31
- paintEvent
 - Dlocation, 19
- printSettings
 - Graphview, 25
- ReDraw
 - Drawing, 21
- RealTime
 - MainWindow, 33
- resetAgentScore
 - Agent, 16
- ResetRealTimeDisplay
 - MainWindow, 32
- runtime
 - UserSettings, 42
- scene
 - Drawing, 23
- score
 - Strategy, 38
- setAgentNum
 - Spot, 35
- setCap
 - Spot, 35
- setCenter
 - Dlocation, 19
- setCommunication
 - Agent, 16
- SetDay
 - Drawing, 22
- setFishingduration
 - Agent, 16
- SetLocationPop
 - Drawing, 22
- SetNumberOfLocations
 - Drawing, 22
- setPop
 - Dlocation, 19
- setSkill
 - Agent, 16
- setStrategy
 - Agent, 16
- setTemp
 - Agent, 16
- settings
 - MainWindow, 33
- setupPlot
 - Graphview, 26
- simulated
 - MainWindow, 33

- skill
 - Agent, 17
- Spot, 33
 - agents, 36
 - getAgentNum, 34
 - getAgents, 35
 - getSpotCapacity, 35
 - maxcapacity, 36
 - numAgent, 36
 - setAgentNum, 35
 - setCap, 35
 - Spot, 34
- startSimulate
 - MainWindow, 32
- strat
 - Agent, 17
- Strategy, 36
 - clearLose, 36
 - clearWin, 36
 - decisionPattern, 38
 - getDecisionPattern, 37
 - getLose, 37
 - getScore, 37
 - getWin, 37
 - lose, 38
 - score, 38
 - Strategy, 36
 - updateScore, 38
 - win, 38
- temp
 - Agent, 17
- threshold
 - Agent, 17
- timer
 - MainWindow, 33
- Ui, 11
- ui
 - Graphview, 26
 - MainWindow, 33
- update
 - MainWindow, 32
- updateAgentScore
 - Agent, 16
- updateBoostScore
 - Agent, 16
- updateHistory
 - Agent, 16
- updateScore
 - Strategy, 38
- updateStrategyScore
 - Agent, 17
- UserSettings, 38
 - fishLoc, 41
 - fishPop, 41
 - fishTemp, 41
 - fishType, 41
 - fisherNum, 41
 - getRuntime, 41
 - getfishLoc, 40
 - getfishPop, 40
 - getfishTemp, 40
 - getfishType, 40
 - getfisherNum, 39
 - runtime, 42
 - UserSettings, 39
- view
 - Drawing, 23
- win
 - Strategy, 38