Langmuir

Generated by Doxygen 1.8.1.2

Thu Jun 20 2013 16:24:02

Contents

Chapter 1

LANGMUIR

- This is the source code for the **Langmuir** engine for charge transfer simulations in organic electronics.

BUILD INSTRUCTIONS

- In order to build the Langmuir engine the following dependencies are required,
  - Qt4
  - Boost
  - CMake
- The following are optional:
  - OpenCL 1.1
  - OpenGL
  - Doxygen

SIMPLE BUILD

- mkdir build
- cd build
- cmake ../
- make -j 4
- make install
- make doc

NOTES

- ON MAC
  - cmake -DCMAKE_OSX_ARCHITECTURES=i386 ../
- OPENCL
  - cmake -DOPENCL_INCLUDE_DIRS=...
- Documentation
  - open doc/html/index.html in a web browser

CLANG SCAN-BUILD

- mkdir build
- cd build
- scan-build -v cmake ..
- scan-build -v -k -analyze-headers -stats -o . make -j 4
- scan-view scan-build-output-dir

NOTES

- scan-build-output-dir will be in the current directory and have the current date for its name

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 copy_to_frank Namespace Reference

Variables

- tuple work = os.getcwd()
- list exclude
- list found = [ ]
- copy_me = True

6.1.1 Variable Documentation

6.1.1.1 copy_to_frank.copy_me = True

6.1.1.2 list copy_to_frank.exclude

Initial value:

```
[
    re.compile(r’CMake.*’),
    re.compile(r’.*\.py’)
]
```

6.1.1.3 list copy_to_frank.found = [ ]

6.1.1.4 tuple copy_to_frank.work = os.getcwd()

6.2 Langmuir Namespace Reference

Classes

- class Agent

  A class that abstractly represents an object that can occupy grid sites.
- class ChargeAgent

  A class to represent moving charged particles.
- class ElectronAgent

  A class to represent moving negative charges.
- class HoleAgent

A class to represent moving positive charges.

- class CheckPointer

  A class to read and write checkpoint files.
- class Grid

  A class to hold Agents, calculate their positions, and store the background potential.
- class DrainAgent

  A class to remove charges.
- class ElectronDrainAgent

  A class to remove ElectronAgents.
- class HoleDrainAgent

  A class to remove HoleAgents.
- class RecombinationAgent

  A class to remove Excitons.
- class FluxAgent

  A class to change the number of carriers in the system.
- class KeyValueParser

  A class to read the parameters and store them in the correct place.
- class OpenClHelper

  A Class to run OpenCL calculations.
- class OutputInfo

  A class to generate file names using the SimulationParameters.
- class OutputStream

  A class to combine QFile, QTextStream and OutputInfo (QFileInfo).
- struct ConfigurationInfo

  A struct to temporarily store site IDs.
- struct SimulationParameters

  A struct to store all simulation options To add new variables, follow these steps:
- class PBSGPUParser
- class Potential

  A class to calculate the potential.
- class Random

  A class to generate random numbers.
- class Simulation

  A class to orchestrate the calculation.
- class SourceAgent

  A class to inject charges.
- class ElectronSourceAgent

  A class to inject ElectronAgents.
- class HoleSourceAgent

  A class to inject HoleAgents.
- class ExcitonSourceAgent

  A class to inject Excitons.
- class Tolerance

  A class to check if the simulation is converging.
- class Variable

  A class to map between variable names (keys) and locations (references)
- class TypedVariable

  A template class to map between variable names (keys) and locations (references)
- class World

  A class to hold all objects in a simulation.
- class XYZWriter

A class to output xyz files.

- class FluxWriter

  A class to output source and drain info.

- class CarrierWriter

  A class to output carrier stats (lifetime and pathlength)

- class ExcitonWriter

  A class to output exciton stats (lifetime and pathlength)

- class GridImage

  A class to draw images of the grid.

- class Logger

  A class that organizes output.

Functions

- QTextStream & operator<< (QTextStream &stream, const Agent::Type e)

  Output Agent type enum to stream.

- QDebug operator<< (QDebug dbg, const Agent::Type e)

  Output Agent type enum to debug information.

- static std::ostream & operator<< (std::ostream &stream, QString &string)
- static std::istream & operator>> (std::istream &stream, QString &string)
- QTextStream & operator<< (QTextStream &stream, const Grid::CubeFace e)

  Overload QTextStream for the Grid::CubeFace Enum.

- QDebug operator<< (QDebug dbg, const Grid::CubeFace e)

  Overload QDebug for the Grid::CubeFace Enum.

- std::ostream & operator<< (std::ostream &stream, const KeyValueParser &keyValueParser)
- void backupFile (const QString &name)

  Back up a file.

- void setCalculatedValues (SimulationParameters &par)

  sets parameters that depend upon other parameters

- void checkSimulationParameters (SimulationParameters &par)

  check the parameters, making sure they are valid

- QDataStream & operator<< (QDataStream &stream, Random &random)
- QDataStream & operator>> (QDataStream &stream, Random &random)
- QTextStream & operator<< (QTextStream &stream, Random &random)
- QTextStream & operator>> (QTextStream &stream, Random &random)
- std::ostream & operator<< (std::ostream &stream, Random &random)
- std::istream & operator>> (std::istream &stream, Random &random)
- QTextStream & operator<< (QTextStream &stream, const QDateTime &datetime)

  output QDateTime as qint64 mSecsSinceEpoch

- QTextStream & operator<< (QTextStream &stream, const Variable &variable)

  overload operator to write keyValue() to a stream

- QDebug operator<< (QDebug dbg, const Variable &variable)

  overload operator to write keyValue() to a QDebug

- std::ostream & operator<< (std::ostream &stream, Variable &variable)

  Operator overload to output to output 'key = value' to std::ostream.

6.2.1 Function Documentation

6.2.1.1 void Langmuir::backupFile (const QString & *name*)

Back up a file.

Back up the file using the current time and a revision number. The file is backed up as path/file.date.num, where num is determined by examing existing files in path with a similiar form (path/file.current_date.a_number). The file is renamed, not copied.

Parameters

name	a relative or absolute file name path

Warning

gives an error if a directory is passed instead of a file
gives an error if the file can not be renamed

6.2.1.2 void Langmuir::checkSimulationParameters (SimulationParameters & *par*) [inline]

check the parameters, making sure they are valid

6.2.1.3 QTextStream & Langmuir::operator<< (QTextStream & *stream*, const QDateTime & *datetime*) [inline]

output QDateTime as qint64 mSecsSinceEpoch

6.2.1.4 QDataStream& Langmuir::operator<< (QDataStream & *stream*, Random & *random*)

Warning

This may not quite be working correctly

6.2.1.5 QTextStream& Langmuir::operator<< (QTextStream & *stream*, const Agent::Type *e*) [inline]

Output Agent type enum to stream.

6.2.1.6 QDebug Langmuir::operator<< (QDebug *dbg*, const Agent::Type *e*) [inline]

Output Agent type enum to debug information.

6.2.1.7 static std::ostream& Langmuir::operator<< (std::ostream & *stream*, QString & *string*) [inline], [static]

6.2.1.8 std::ostream& Langmuir::operator<< (std::ostream & *stream*, const KeyValueParser & *keyValueParser*)

6.2.1.9 QTextStream& Langmuir::operator<< (QTextStream & *stream*, Random & *random*)

Warning

This may not quite be working correctly

6.2.1.10 QTextStream& Langmuir::operator<< (QTextStream & *stream,* const Variable & *variable*) [inline]

overload operator to write keyValue() to a stream

Operator overload to output 'key = value' to QTextStream.

6.2.1.11 QDebug Langmuir::operator<< (QDebug *dbg,* const Variable & *variable*) [inline]

overload operator to write keyValue() to a QDebug

Operator overload to output 'key = value' to QDebug.

6.2.1.12 std::ostream& Langmuir::operator<< (std::ostream & *stream,* Variable & *variable*) [inline]

Operator overload to output to output 'key = value' to std::ostream.

Operator overload to output to output 'key = value' to std::ofstream.

6.2.1.13 std::ostream& Langmuir::operator<< (std::ostream & *stream,* Random & *random*)

6.2.1.14 QTextStream & Langmuir::operator<< (QTextStream & *stream,* const Grid::CubeFace *e*)

Overload QTextStream for the Grid::CubeFace Enum.

6.2.1.15 QDebug Langmuir::operator<< (QDebug *dbg,* const Grid::CubeFace *e*)

Overload QDebug for the Grid::CubeFace Enum.

6.2.1.16 QDataStream& Langmuir::operator>> (QDataStream & *stream,* Random & *random*)

Warning

> This may not quite be working correctly

6.2.1.17 static std::istream& Langmuir::operator>> (std::istream & *stream,* QString & *string*) [inline], [static]

6.2.1.18 QTextStream& Langmuir::operator>> (QTextStream & *stream,* Random & *random*)

Warning

> This may not quite be working correctly

6.2.1.19 std::istream& Langmuir::operator>> (std::istream & *stream,* Random & *random*)

6.2.1.20 void Langmuir::setCalculatedValues (SimulationParameters & *par*) [inline]

sets parameters that depend upon other parameters

Chapter 7

Class Documentation

7.1 Langmuir::Agent Class Reference

A class that abstractly represents an object that can occupy grid sites.

```
#include <agent.h>
```

Public Types

- enum Type {
  Empty = 0, Electron = 1, Hole = 2, Defect = 3,
  Source = 4, Drain = 5, SIZE = 6 }

  An identifier for the type of Agent.

Public Member Functions

- Agent (Type type, World &world, int site=0, QObject ∗parent=0)

  Create an Agent.
- virtual ∼Agent ()

  Destroy Agent.
- const QVector< int > & getNeighbors () const

  Get Agent neighbor list.
- void setNeighbors (QVector< int > neighbors)

  Set Agent neighbor list.
- int getCurrentSite () const

  Get Agent current site.
- int getFutureSite () const

  Get Agent future site.
- void setCurrentSite (int site)

  Set Agent current site.
- void setFutureSite (int site)

  Set Agent future site.
- Type getType () const

  Get Agent::Type enum.
- World & getWorld () const

  Get Langmuir::World reference.

Static Public Member Functions

- static QString toQString (const Agent::Type e)

  Convert Agent type enum to QString.

Protected Attributes

- int m_site

  Current site the Agent occupies.
- int m_fSite

  *Future site the Agent **will** occupy.*
- World & m_world

  Reference to World object.
- QVector< int > m_neighbors

  List fo neighboring site ids.
- Type m_type

  Agent Type enum.

7.1.1 Detailed Description

A class that abstractly represents an object that can occupy grid sites.

Agents can be Electrons, Holes, Defects, Sources, or Drains. The Agent class encodes basic information that all agents have, regardless of their type. For examples, all agents occupy a grid site, have knowledge of their neighboring sites, and know their own type.

7.1.2 Member Enumeration Documentation

7.1.2.1 enum Langmuir::Agent::Type

An identifier for the type of Agent.

Enumerator:

Empty Empty Grid site.

Electron ElectronAgent.

Hole HoleAgent.

Defect Defective Grid site.

Source SourceAgent.

Drain DrainAgent.

SIZE Number of Agent Types.

7.1.3 Constructor & Destructor Documentation

7.1.3.1 Langmuir::Agent::Agent (Type *type*, World & *world*, int *site* = 0, QObject ∗ *parent* = 0) [inline]

Create an Agent.

Parameters

type	identifier enum • for example: Electron, Hole, etc.
world	reference world object
site	grid site id Agent occupies
parent	parent QObject

7.1.3.2 Langmuir::Agent::∼Agent ( ) [inline], [virtual]

Destroy Agent.

7.1.4 Member Function Documentation

7.1.4.1 int Langmuir::Agent::getCurrentSite ( ) const [inline]

Get Agent current site.

7.1.4.2 int Langmuir::Agent::getFutureSite ( ) const [inline]

Get Agent future site.

7.1.4.3 const QVector< int > & Langmuir::Agent::getNeighbors ( ) const [inline]

Get Agent neighbor list.

7.1.4.4 Agent::Type Langmuir::Agent::getType ( ) const [inline]

Get Agent::Type enum.

7.1.4.5 World & Langmuir::Agent::getWorld ( ) const [inline]

Get Langmuir::World reference.

7.1.4.6 void Langmuir::Agent::setCurrentSite (int *site*) [inline]

Set Agent current site.

7.1.4.7 void Langmuir::Agent::setFutureSite (int *site*) [inline]

Set Agent future site.

7.1.4.8 void Langmuir::Agent::setNeighbors (QVector< int > *neighbors*) [inline]

Set Agent neighbor list.

7.1.4.9 QString Langmuir::Agent::toQString (const Agent::Type *e*) [inline], [static]

Convert Agent type enum to QString.

7.1.5 Member Data Documentation

7.1.5.1 int Langmuir::Agent::m_fSite [protected]

Future site the Agent **will** occupy.

7.1.5.2 QVector<int> Langmuir::Agent::m_neighbors [protected]

List fo neighboring site ids.

7.1.5.3 int Langmuir::Agent::m_site [protected]

Current site the Agent occupies.

7.1.5.4 Type Langmuir::Agent::m_type [protected]

Agent Type enum.

7.1.5.5 World& Langmuir::Agent::m_world [protected]

Reference to World object.

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

- agent.h

7.2 Langmuir::CarrierWriter Class Reference

A class to output carrier stats (lifetime and pathlength)

```
#include <writer.h>
```

Public Member Functions

- CarrierWriter (World &world, const QString &name, QObject *parent=0)

  constructs the writer, has the same parameters as OutputInfo
- void write (ChargeAgent &charge)

  write the charge carrier statistics to the stream

Protected Attributes

- World & m_world

  reference to the world object
- OutputStream m_stream

  output file stream

7.2.1 Detailed Description

A class to output carrier stats (lifetime and pathlength)

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Langmuir::CarrierWriter::CarrierWriter (World & *world,* const QString & *name,* QObject ∗ *parent* = 0)

constructs the writer, has the same parameters as OutputInfo

7.2.3 Member Function Documentation

7.2.3.1 void Langmuir::CarrierWriter::write (ChargeAgent & *charge*)

write the charge carrier statistics to the stream

7.2.4 Member Data Documentation

7.2.4.1 OutputStream Langmuir::CarrierWriter::m_stream [protected]

output file stream

7.2.4.2 World& Langmuir::CarrierWriter::m_world [protected]

reference to the world object

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

- writer.h
- writer.cpp

7.3 Langmuir::ChargeAgent Class Reference

A class to represent moving charged particles.

`#include <chargeagent.h>`

Public Member Functions

- ChargeAgent (Agent::Type getType, World &world, Grid &grid, int site, QObject ∗parent=0)

  Construct charge.
- virtual ∼ChargeAgent ()

  Destroy charge.
- int charge ()

  Get the charge of the ChargeAgent.
- void chooseFuture ()

  Propose a random site to move to.
- void decideFuture ()

  Decide what should happen, called after chooseFuture.
- void completeTick ()

  Perform action, called after decideFuture.
- bool removed ()

  True if decideFuture removed the charge from the grid.
- int lifetime ()

  Number of steps ChargeAgent has existed.

- int pathlength ()

  Number of sites ChargeAgent has traversed.
- void setOpenCLID (int id)

  Set the ChargeAgent OpenCL identifier.
- int getOpenCLID ()

  Get the ChargeAgent OpenCL identifier.
- double coulombInteraction ()

  Perform coulombCPU() or coulombGPU()
- void coulombCPU ()

  Calculate the Coulomb potential on the CPU.
- void coulombGPU ()

  ***Retrieve** the Coulomb potential from the GPU*
- void compareCoulomb ()

  compare results for CPU and GPU Coulomb (assumes kernel was called)
- Grid & getGrid ()

  Get the grid this ChargeAgent exists in.
- void setRemoved (const bool &status=true)

  Set the removed status of this ChargeAgent.
- virtual Agent::Type otherType ()=0

  Return the opposite ChargeAgent type relative to this ChargeAgent.
- virtual Grid & otherGrid ()=0

  Return the opposite Grid relative to this ChargeAgent's Agent::Type.

Protected Member Functions

- virtual double bindingPotential (int site)=0

  Calculate the exciton binding energy.

Protected Attributes

- int m_charge

  Charge of ChargeAgent (in units of e)
- bool m_removed

  Removed status of ChargeAgent.
- int m_lifetime

  Number of steps ChargeAgent as been in existance.
- int m_pathlength

  Number of grid spaces ChargeAgent has moved.
- Grid & m_grid

  The Grid the ChargeAgent lives in.
- int m_openClID

  The index of the Charge in the OpenCL vectors (see OpenClHelper)
- double m_de

  The difference in Coulomb potential between ChargeAgent::m_site and ChargeAgent::m_fSite.

Additional Inherited Members

7.3.1 Detailed Description

A class to represent moving charged particles.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Langmuir::ChargeAgent::ChargeAgent (Agent::Type *getType,* World & *world,* Grid & *grid,* int *site,* QObject ∗ *parent =* `0`)

Construct charge.

ChargeAgent

Parameters

getType	Agent type; must be Agent::Electron or Agent::Hole
world	reference to world
grid	reference to grid
site	site id in grid
parent	parent QObject

7.3.2.2 Langmuir::ChargeAgent::∼ChargeAgent ( ) `[virtual]`

Destroy charge.

7.3.3 Member Function Documentation

7.3.3.1 virtual double Langmuir::ChargeAgent::bindingPotential (int *site*) `[protected], [pure virtual]`

Calculate the exciton binding energy.

Parameters

site	the site to check in other Grid

Returns

- $+0.5$ eV if exciton
- 0 otherwise

Implemented in Langmuir::HoleAgent, and Langmuir::ElectronAgent.

7.3.3.2 int Langmuir::ChargeAgent::charge ( )

Get the charge of the ChargeAgent.

7.3.3.3 void Langmuir::ChargeAgent::chooseFuture ( )

Propose a random site to move to.

7.3.3.4 void Langmuir::ChargeAgent::compareCoulomb ( )

compare results for CPU and GPU Coulomb (assumes kernel was called)

7.3.3.5 void Langmuir::ChargeAgent::completeTick ( )

Perform action, called after decideFuture.

7.3.3.6 void Langmuir::ChargeAgent::coulombCPU ( )

Calculate the Coulomb potential on the CPU.

Note

The result is stored in m_de

7.3.3.7 void Langmuir::ChargeAgent::coulombGPU ( )

Retrieve the Coulomb potential from the GPU

Note

The result is stored in m_de

Warning

this function assumes:

- the openCL id set for the ChargeAgent is the correct one
- the openCL kernel has been executed

7.3.3.8 double Langmuir::ChargeAgent::coulombInteraction ( )

Perform coulombCPU() or coulombGPU()

depends upon SimulationParameters::useOpenCL and SimulationParameters::okCL

Returns

ChargeAgent::m_de

7.3.3.9 void Langmuir::ChargeAgent::decideFuture ( )

Decide what should happen, called after chooseFuture.

7.3.3.10 Grid & Langmuir::ChargeAgent::getGrid ( )

Get the grid this ChargeAgent exists in.

7.3.3.11 int Langmuir::ChargeAgent::getOpenCLID ( )

Get the ChargeAgent OpenCL identifier.

See Also

OpenClHelper

7.3.3.12 int Langmuir::ChargeAgent::lifetime ( )

Number of steps ChargeAgent has existed.

7.3.3.13 virtual Grid& Langmuir::ChargeAgent::otherGrid ( ) [pure virtual]

Return the opposite Grid relative to this ChargeAgent's Agent::Type.

Returns

World::holeGrid() if this chargeAgent is an Agent::Electron

Implemented in Langmuir::HoleAgent, and Langmuir::ElectronAgent.

7.3.3.14 virtual Agent::Type Langmuir::ChargeAgent::otherType ( ) [pure virtual]

Return the opposite ChargeAgent type relative to this ChargeAgent.

Returns

Agent::Hole if this ChargeAgent is an Agent::Electron

Implemented in Langmuir::HoleAgent, and Langmuir::ElectronAgent.

7.3.3.15 int Langmuir::ChargeAgent::pathlength ( )

Number of sites ChargeAgent has traversed.

7.3.3.16 bool Langmuir::ChargeAgent::removed ( )

True if decideFuture removed the charge from the grid.

7.3.3.17 void Langmuir::ChargeAgent::setOpenCLID (int *id*)

Set the ChargeAgent OpenCL identifier.

See Also

OpenClHelper

7.3.3.18 void Langmuir::ChargeAgent::setRemoved (const bool & *status* = `true`)

Set the removed status of this ChargeAgent.

Note

Removed charges are not actually removed until completeTick() is called

7.3.4 Member Data Documentation

7.3.4.1 int Langmuir::ChargeAgent::m_charge [protected]

Charge of ChargeAgent (in units of e)

7.3.4.2 double Langmuir::ChargeAgent::m_de [protected]

The difference in Coulomb potential between ChargeAgent::m_site and ChargeAgent::m_fSite.

7.3.4.3 Grid& Langmuir::ChargeAgent::m_grid [protected]

The Grid the ChargeAgent lives in.

7.3.4.4 int Langmuir::ChargeAgent::m_lifetime [protected]

Number of steps ChargeAgent as been in existance.

7.3.4.5 int Langmuir::ChargeAgent::m_openClID [protected]

The index of the Charge in the OpenCL vectors (see OpenClHelper)

7.3.4.6 int Langmuir::ChargeAgent::m_pathlength [protected]

Number of grid spaces ChargeAgent has moved.

7.3.4.7 bool Langmuir::ChargeAgent::m_removed [protected]

Removed status of ChargeAgent.

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

- chargeagent.h
- chargeagent.cpp

7.4 Langmuir::CheckPointer Class Reference

A class to read and write checkpoint files.

`#include <checkpointer.h>`

Public Types

- enum Section {
  Parameters, Electrons, Holes, Defects,
  Traps, TrapPotentials, RandomState, FluxState }

  A way to identify different sections in the input file.

Public Member Functions

- CheckPointer (World &world, QObject ∗parent=0)

  Create the checkpointer object.
- void load (const QString &fileName, ConfigurationInfo &configInfo)

  load simulation information
- void save (const QString &fileName="%stub.chk")

  save simulation information
- void checkStream (std::istream &stream, const QString &message="")

  check to see if input stream has failed

Private Member Functions

- std::istream & loadElectrons (std::istream &stream, ConfigurationInfo &configInfo)

  load electrons sites from input file
- std::istream & loadHoles (std::istream &stream, ConfigurationInfo &configInfo)

  load hole sites from input file
- std::istream & loadDefects (std::istream &stream, ConfigurationInfo &configInfo)

  load defect sites from input file
- std::istream & loadTraps (std::istream &stream, ConfigurationInfo &configInfo)

  load trap sites from input file
- std::istream & loadTrapPotentials (std::istream &stream, ConfigurationInfo &configInfo)

  load trap energies from input file
- std::istream & loadFluxState (std::istream &stream, ConfigurationInfo &configInfo)

  load flux state from input file
- std::istream & loadParameters (std::istream &stream)

  load parameter from input file
- std::istream & loadRandomState (std::istream &stream)

  load random number generator state from input file
- std::ostream & saveElectrons (std::ostream &stream)

  save electron site ids to output file
- std::ostream & saveHoles (std::ostream &stream)

  save hole site ids to output file
- std::ostream & saveDefects (std::ostream &stream)

  save defect site ids to output file
- std::ostream & saveTraps (std::ostream &stream)

  save trap site ids to output file
- std::ostream & saveTrapPotentials (std::ostream &stream)

  save trap energies to output file
- std::ostream & saveFluxState (std::ostream &stream)

  save flux states to output file
- std::ostream & saveParameters (std::ostream &stream)

  save parameters to output file
- std::ostream & saveRandomState (std::ostream &stream)

  save random number generator state to output file

Private Attributes

- World & m_world

  reference to world object

7.4.1 Detailed Description

A class to read and write checkpoint files.

Checkpoint files are essentially the same as input files

7.4.2 Member Enumeration Documentation

7.4.2.1 enum Langmuir::CheckPointer::Section

A way to identify different sections in the input file.

Enumerator:

Parameters

Electrons

Holes

Defects

Traps

TrapPotentials

RandomState

FluxState

7.4.3 Constructor & Destructor Documentation

7.4.3.1 Langmuir::CheckPointer::CheckPointer (World & *world*, QObject ∗ *parent* = 0) [explicit]

Create the checkpointer object.

Parameters

world	reference world object
parent	parent QObject

7.4.4 Member Function Documentation

7.4.4.1 void Langmuir::CheckPointer::checkStream (std::istream & *stream*, const QString & *message* = "")

check to see if input stream has failed

Parameters

stream	input stream
message	the error message to output if stream failed

7.4.4.2 void Langmuir::CheckPointer::load (const QString & *fileName*, ConfigurationInfo & *configInfo*)

load simulation information

Parameters

fileName	name of input file
configInfo	temporary storage for electrons, holes, etc

7.4.4.3 std::istream & Langmuir::CheckPointer::loadDefects (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load defect sites from input file

Parameters

stream	the input stream
configInfo	temporary storage for site ids

7.4.4.4 std::istream & Langmuir::CheckPointer::loadElectrons (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load electrons sites from input file

Parameters

stream	the input stream
configInfo	temporary storage for site ids

7.4.4.5 std::istream & Langmuir::CheckPointer::loadFluxState (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load flux state from input file

Parameters

stream	the input stream
configInfo	temporary storage for flux state

7.4.4.6 std::istream & Langmuir::CheckPointer::loadHoles (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load hole sites from input file

Parameters

stream	the input stream
configInfo	temporary storage for site ids

7.4.4.7 std::istream & Langmuir::CheckPointer::loadParameters (std::istream & *stream*) [private]

load parameter from input file

Parameters

stream	the input stream

7.4.4.8 std::istream & Langmuir::CheckPointer::loadRandomState (std::istream & *stream*) [private]

load random number generator state from input file

Parameters

stream	the input stream

7.4.4.9 std::istream & Langmuir::CheckPointer::loadTrapPotentials (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load trap energies from input file

Parameters

stream	the input stream
configInfo	temporary storage for site energies

7.4.4.10 std::istream & Langmuir::CheckPointer::loadTraps (std::istream & *stream*, ConfigurationInfo & *configInfo*) [private]

load trap sites from input file

Parameters

stream	the input stream
configInfo	temporary storage for site ids

7.4.4.11 void Langmuir::CheckPointer::save (const QString & *fileName* = "%stub.chk")

save simulation information

Parameters

fileName	name of output file

7.4.4.12 std::ostream & Langmuir::CheckPointer::saveDefects (std::ostream & *stream*) [private]

save defect site ids to output file

Parameters

stream	output stream

7.4.4.13 std::ostream & Langmuir::CheckPointer::saveElectrons (std::ostream & *stream*) [private]

save electron site ids to output file

Parameters

stream	output stream

7.4.4.14 std::ostream & Langmuir::CheckPointer::saveFluxState (std::ostream & *stream*) [private]

save flux states to output file

Parameters

stream	output stream

7.4.4.15 std::ostream & Langmuir::CheckPointer::saveHoles (std::ostream & *stream*) [private]

save hole site ids to output file

Parameters

stream	output stream

7.4.4.16 std::ostream & Langmuir::CheckPointer::saveParameters (std::ostream & *stream*) [private]

save parameters to output file

Parameters

stream	output stream

7.4.4.17 std::ostream & Langmuir::CheckPointer::saveRandomState (std::ostream & *stream*) [private]

save random number generator state to output file

Parameters

stream	output stream

7.4.4.18 std::ostream & Langmuir::CheckPointer::saveTrapPotentials (std::ostream & *stream*) [private]

save trap energies to output file

Parameters

stream	output stream

7.4.4.19 std::ostream & Langmuir::CheckPointer::saveTraps (std::ostream & *stream*) [private]

save trap site ids to output file

Parameters

stream	output stream

7.4.5 Member Data Documentation

7.4.5.1 World& Langmuir::CheckPointer::m_world [private]

reference to world object

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

- checkpointer.h
- checkpointer.cpp

7.5 Langmuir::ConfigurationInfo Struct Reference

A struct to temporarily store site IDs.

```
#include <parameters.h>
```

Public Attributes

- QList< qint32 > electrons

  a list of current electron site IDs
- QList< qint32 > holes

  a list of current holes site IDs
- QList< qint32 > defects

  a list of current defects site IDs
- QList< qint32 > traps

  a list of current traps site IDs
- QList< qreal > trapPotentials

  a list of current traps site IDs
- QList< quint64 > fluxInfo

  a list of flux attempt, success values

7.5.1 Detailed Description

A struct to temporarily store site IDs.

7.5.2 Member Data Documentation

7.5.2.1 QList<qint32> Langmuir::ConfigurationInfo::defects

a list of current defects site IDs

7.5.2.2 QList<qint32> Langmuir::ConfigurationInfo::electrons

a list of current electron site IDs

7.5.2.3 QList<quint64> Langmuir::ConfigurationInfo::fluxInfo

a list of flux attempt, success values

7.5.2.4 QList<qint32> Langmuir::ConfigurationInfo::holes

a list of current holes site IDs

7.5.2.5 QList<qreal> Langmuir::ConfigurationInfo::trapPotentials

a list of current traps site IDs

7.5.2.6 QList<qint32> Langmuir::ConfigurationInfo::traps

a list of current traps site IDs

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

- parameters.h

7.6 Langmuir::DrainAgent Class Reference

A class to remove charges.

`#include <drainagent.h>`

Public Member Functions

- DrainAgent (World &world, Grid &grid, QObject *parent=0)

  create a DrainAgent
- virtual bool tryToAccept (ChargeAgent *charge)

  accept charge with constant probability

Additional Inherited Members

7.6.1 Detailed Description

A class to remove charges.

Unlike SourceAgents, the algorithms for removing charges currently reside in the Simulation class. This should be fixed. The DrainAgent is keeping track of drain statistics and transport probability.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Langmuir::DrainAgent::DrainAgent (World & *world,* Grid & *grid,* QObject * *parent =* `0`)

create a DrainAgent

7.6.3 Member Function Documentation

7.6.3.1 bool Langmuir::DrainAgent::tryToAccept (ChargeAgent * *charge*) `[virtual]`

accept charge with constant probability

Reimplemented in Langmuir::RecombinationAgent.

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

- drainagent.h
- drainagent.cpp

7.7 Langmuir::ElectronAgent Class Reference

A class to represent moving negative charges.

`#include <chargeagent.h>`

Public Member Functions

- ElectronAgent (World &world, int site, QObject ∗parent=0)

  Construct ElectronAgent.

Protected Member Functions

- virtual double bindingPotential (int site)

  Calculate Exciton Binding Energy.
- virtual Agent::Type otherType ()

  Return other Agent::Type.
- virtual Grid & otherGrid ()

  Return other Grid.

Additional Inherited Members

7.7.1 Detailed Description

A class to represent moving negative charges.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 Langmuir::ElectronAgent::ElectronAgent (World & *world,* int *site,* QObject ∗ *parent* = 0)

Construct ElectronAgent.

7.7.3 Member Function Documentation

7.7.3.1 double Langmuir::ElectronAgent::bindingPotential (int *site*) [protected], [virtual]

Calculate Exciton Binding Energy.

Parameters

site	the site to check in other Grid

Returns

- $+0.5$ eV if exciton
- 0 otherwise

Implements Langmuir::ChargeAgent.

7.7.3.2 Grid & Langmuir::ElectronAgent::otherGrid ( ) [protected], [virtual]

Return other Grid.

Returns

World::holeGrid

Implements Langmuir::ChargeAgent.

7.7.3.3 Agent::Type Langmuir::ElectronAgent::otherType ( ) [protected], [virtual]

Return other Agent::Type.

Returns

Agent::Hole

Implements Langmuir::ChargeAgent.

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

- chargeagent.h
- chargeagent.cpp

7.8 Langmuir::ElectronDrainAgent Class Reference

A class to remove ElectronAgents.

`#include <drainagent.h>`

Public Member Functions

- ElectronDrainAgent (World &world, int site, QObject ∗parent=0)

  create an ElectronDrainAgent at a specific site
- ElectronDrainAgent (World &world, Grid::CubeFace cubeFace, QObject ∗parent=0)

  create a ElectronDrainAgent at a specific Grid::CubeFace

Protected Member Functions

- virtual double energyChange (int fSite)

  same as FluxAgent::energyChange(), but specialized for ElectronAgents.

Additional Inherited Members

7.8.1 Detailed Description

A class to remove ElectronAgents.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Langmuir::ElectronDrainAgent::ElectronDrainAgent (World & *world*, int *site*, QObject ∗ *parent* = 0)

create an ElectronDrainAgent at a specific site

7.8.2.2 Langmuir::ElectronDrainAgent::ElectronDrainAgent (World & *world*, Grid::CubeFace *cubeFace*, QObject ∗ *parent* = 0)

create a ElectronDrainAgent at a specific Grid::CubeFace

7.8.3 Member Function Documentation

7.8.3.1 double Langmuir::ElectronDrainAgent::energyChange (int *fSite*) [protected], [virtual]

same as FluxAgent::energyChange(), but specialized for ElectronAgents.

Note really used because the default FluxAgent::shouldTransport() behavoir, which is to use a simple constant probability, has not been reimplemented for DrainAgents.

Reimplemented from Langmuir::FluxAgent.

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

- drainagent.h
- drainagent.cpp

7.9 Langmuir::ElectronSourceAgent Class Reference

A class to inject ElectronAgents.

`#include <sourceagent.h>`

Public Member Functions

- ElectronSourceAgent (World &world, int site, QObject ∗parent=0)

  create an ElectronSourceAgent at a specific site
- ElectronSourceAgent (World &world, Grid::CubeFace cubeFace, QObject ∗parent=0)

  create an ElectronSourceAgent at a specific Grid::CubeFace

Protected Member Functions

- virtual bool validToInject (int site)

  same as SourceAgent::validToInject(), but specialized for ElectronAgents.
- virtual double energyChange (int site)

  same as FluxAgent::energyChange(), but specialized for ElectronAgents.
- virtual void inject (int site)

  same as SourceAgent::inject(), but specialized for ElectronAgents.

Additional Inherited Members

7.9.1 Detailed Description

A class to inject ElectronAgents.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 Langmuir::ElectronSourceAgent::ElectronSourceAgent (World & *world,* int *site,* QObject ∗ *parent =* 0)

create an ElectronSourceAgent at a specific site

7.9.2.2 Langmuir::ElectronSourceAgent::ElectronSourceAgent (World & *world,* Grid::CubeFace *cubeFace,* QObject ∗ *parent =* 0)

create an ElectronSourceAgent at a specific Grid::CubeFace

7.9.3 Member Function Documentation

7.9.3.1 double Langmuir::ElectronSourceAgent::energyChange (int *site*) [protected], [virtual]

same as FluxAgent::energyChange(), but specialized for ElectronAgents.

Reimplemented from Langmuir::FluxAgent.

7.9.3.2 void Langmuir::ElectronSourceAgent::inject (int *site*) [protected], [virtual]

same as SourceAgent::inject(), but specialized for ElectronAgents.

Implements Langmuir::SourceAgent.

7.9.3.3 bool Langmuir::ElectronSourceAgent::validToInject (int *site*) [protected], [virtual]

same as SourceAgent::validToInject(), but specialized for ElectronAgents.

Implements Langmuir::SourceAgent.

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

- sourceagent.h
- sourceagent.cpp

7.10 Langmuir::ExcitonSourceAgent Class Reference

A class to inject Excitons.

`#include <sourceagent.h>`

Public Member Functions

- ExcitonSourceAgent (World &world, QObject ∗parent=0)

  create an ExcitonSourceAgent

Protected Member Functions

- virtual bool validToInject (int site)

  checks both grids if its ok to inject charges
- virtual double energyChange (int site)

  currently implemented as zero and not really used
- virtual bool shouldTransport (int site)

  uses the simple constant probability method
- virtual int chooseSite ()

  choose a site to inject to
- virtual void inject (int site)

  similar to SourceAgent::inject(), but injects both a HoleAgent and an ElectronAgent

Additional Inherited Members

7.10.1 Detailed Description

A class to inject Excitons.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 Langmuir::ExcitonSourceAgent::ExcitonSourceAgent (World & *world*, QObject ∗ *parent* = 0)

create an ExcitonSourceAgent

7.10.3 Member Function Documentation

7.10.3.1 int Langmuir::ExcitonSourceAgent::chooseSite ( ) [protected], [virtual]

choose a site to inject to

reimplemented to chose a site at any grid site

Reimplemented from Langmuir::SourceAgent.

7.10.3.2 double Langmuir::ExcitonSourceAgent::energyChange (int *site*) [protected], [virtual]

currently implemented as zero and not really used

Reimplemented from Langmuir::FluxAgent.

7.10.3.3 void Langmuir::ExcitonSourceAgent::inject (int *site*) [protected], [virtual]

similar to SourceAgent::inject(), but injects both a HoleAgent and an ElectronAgent

Implements Langmuir::SourceAgent.

7.10.3.4 bool Langmuir::ExcitonSourceAgent::shouldTransport (int *site*) [protected], [virtual]

uses the simple constant probability method

Reimplemented from Langmuir::SourceAgent.

7.10.3.5 bool Langmuir::ExcitonSourceAgent::validToInject (int *site*) [protected], [virtual]

checks both grids if its ok to inject charges

Implements Langmuir::SourceAgent.

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

- sourceagent.h
- sourceagent.cpp

7.11 Langmuir::ExcitonWriter Class Reference

A class to output exciton stats (lifetime and pathlength)

`#include <writer.h>`

Public Member Functions

- ExcitonWriter (World &world, const QString &name, QObject ∗parent=0)

  constructs the writer, has the same parameters as OutputInfo

- void write (ChargeAgent &charge1, ChargeAgent &charge2, bool recombined=false)

  write the exciton statistics to the stream

Protected Attributes

- World & m_world

  reference to the world object
- OutputStream m_stream

  output file stream

7.11.1 Detailed Description

A class to output exciton stats (lifetime and pathlength)

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Langmuir::ExcitonWriter::ExcitonWriter (World & *world,* const QString & *name,* QObject ∗ *parent* = 0)

constructs the writer, has the same parameters as OutputInfo

7.11.3 Member Function Documentation

7.11.3.1 void Langmuir::ExcitonWriter::write (ChargeAgent & *charge1,* ChargeAgent & *charge2,* bool *recombined* = false)

write the exciton statistics to the stream

7.11.4 Member Data Documentation

7.11.4.1 OutputStream Langmuir::ExcitonWriter::m_stream [protected]

output file stream

7.11.4.2 World& Langmuir::ExcitonWriter::m_world [protected]

reference to the world object

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

- writer.h
- writer.cpp

7.12 Langmuir::FluxAgent Class Reference

A class to change the number of carriers in the system.

```
#include <fluxagent.h>
```

Public Member Functions

- FluxAgent (Agent::Type type, World &world, Grid &grid, QObject ∗parent=0)

  Create the flux agent.
- ∼FluxAgent ()

  unregisters FluxAgent from the grid
- void setPotential (double potential)

  set the FluxAgent's potential
- double potential () const

  get the FluxAgent's potential
- void setRate (double rate)

  set the FluxAgent's rate
- void setRateSmartly (double rate, double dflt)

  set the FluxAgent's rate
- double rate () const

  get the FluxAgent's rate
- void setAttempts (unsigned long int value)

  set the FluxAgent's attempt counter
- unsigned long int attempts () const

  get the FluxAgent's attempt counter
- void setSuccesses (unsigned long int value)

  set the FluxAgent's success counter
- unsigned long int successes () const

  get the FluxAgent's success counter
- void storeLast ()

  set the value of last to the value of successes, and store the current step
- unsigned long int successesSinceLast () const

  get the number of successes since storeLast() was called
- unsigned long int attemptsSinceLast () const

  get the number of attempts since storeLast() was called
- unsigned long int stepsSinceLast () const

  get the number of steps since storeLast() was called
- double successProbability () const

  calculate and return the current probabilty of success
- double successRate () const

  calculate and return the current rate of success
- double successProbabilitySinceLast () const

  calculate and return the probabilty of success since storeLast() was called
- double successRateSinceLast () const

  calculate and return the rate of success since storeLast() was called
- void resetCounters ()

  set the attempt and success counters to zero
- Grid::CubeFace face () const

  get the Grid:CubeFace this FluxAgent is assigned to
- Grid & grid () const

  get the Grid this FluxAgent belongs to

Protected Member Functions

- void initializeSite (int site)

  assign the FluxAgent to a specific site in the grid
- void initializeSite (Grid::CubeFace cubeFace)

  assign the FluxAgent to a specific Grid::CubeFace
- virtual bool shouldTransport (int site)

  decide if the FluxAgent should transport a carrier to/from a given site
- virtual double energyChange (int site)

  The energy change associated with moving a carrier from the FluxAgent to a site.
- QString faceToLetter ()

  convert the Grid::CubeFace to a single letter

Protected Attributes

- unsigned long int m_attempts

  the number of times the FluxAgent has tried to transport.
- unsigned long int m_successes

  the number of times the FluxAgent was successful in transporting.
- unsigned long int m_lastSuccesses

  storage to note the number of successes at some step
- unsigned long int m_lastAttempts

  storage to note the number of successes at some step
- unsigned long int m_lastStep

  the step at which last was noted
- double m_probability

  the constant probability used in the default behavoir of shouldTransport().
- double m_potential

  the potential that is (possibly) used when calculating an energy change
- Grid & m_grid

  the grid this FluxAgent resides in
- Grid::CubeFace m_face

  the face of the grid this FluxAgent occupies

Additional Inherited Members

7.12.1 Detailed Description

A class to change the number of carriers in the system.

A flux agent can inject carriers (Agent::Source) or accept carriers (Agent::Drain)

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Langmuir::FluxAgent::FluxAgent (**Agent::Type** *type,* **World &** *world,* **Grid &** *grid,* **QObject** ∗ *parent* = 0)

Create the flux agent.

Parameters

type	either a Agent::Source or Agent::Drain
world	reference to world object
grid	reference to grid
parent	parent QObject

7.12.2.2 Langmuir::FluxAgent::∼FluxAgent ( )

unregisters FluxAgent from the grid

7.12.3 Member Function Documentation

7.12.3.1 unsigned long int Langmuir::FluxAgent::attempts ( ) const

get the FluxAgent's attempt counter

7.12.3.2 unsigned long int Langmuir::FluxAgent::attemptsSinceLast ( ) const

get the number of attempts since storeLast() was called

7.12.3.3 double Langmuir::FluxAgent::energyChange (int *site*) [protected], [virtual]

The energy change associated with moving a carrier from the FluxAgent to a site.

Parameters

site	the site involved

Returns

the energy change

Reimplemented in Langmuir::ExcitonSourceAgent, Langmuir::HoleSourceAgent, Langmuir::ElectronSourceAgent, Langmuir::HoleDrainAgent, and Langmuir::ElectronDrainAgent.

7.12.3.4 Grid::CubeFace Langmuir::FluxAgent::face ( ) const

get the Grid:CubeFace this FluxAgent is assigned to

7.12.3.5 QString Langmuir::FluxAgent::faceToLetter ( ) [protected]

convert the Grid::CubeFace to a single letter

For example, Grid::Left would return **L**. This is used in the output file titles.

7.12.3.6 Grid & Langmuir::FluxAgent::grid ( ) const

get the Grid this FluxAgent belongs to

7.12.3.7 void Langmuir::FluxAgent::initializeSite (int *site*) [protected]

assign the FluxAgent to a specific site in the grid

Parameters

site	the site in the grid

7.12.3.8 void Langmuir::FluxAgent::initializeSite (Grid::CubeFace *cubeFace*) [protected]

assign the FluxAgent to a specific Grid::CubeFace

Parameters

cubeFace	the face of a cubic grid; for example Grid::Left

When assigning to a specific Grid::CubeFace, the FluxAgent is considered to be a special agent, and thus resides in the sites reserved by the grid for special agents.

7.12.3.9 double Langmuir::FluxAgent::potential ( ) const

get the FluxAgent's potential

7.12.3.10 double Langmuir::FluxAgent::rate ( ) const

get the FluxAgent's rate

7.12.3.11 void Langmuir::FluxAgent::resetCounters ( )

set the attempt and success counters to zero

7.12.3.12 void Langmuir::FluxAgent::setAttempts (unsigned long int *value*)

set the FluxAgent's attempt counter

Parameters

potential	the value of the attempt counter

Warning

also calls storeLast()

7.12.3.13 void Langmuir::FluxAgent::setPotential (double *potential*)

set the FluxAgent's potential

Parameters

potential	the value of the potential

7.12.3.14 void Langmuir::FluxAgent::setRate (double *rate*)

set the FluxAgent's rate

Parameters

potential	the value of the rate

7.12.3.15 void Langmuir::FluxAgent::setRateSmartly (double *rate,* double *dflt*)

set the FluxAgent's rate

Parameters

potential	the value of the rate
dflt	the default value to set the rate to

If rate is negative, uses the default rate instead

7.12.3.16 void Langmuir::FluxAgent::setSuccesses (unsigned long int *value*)

set the FluxAgent's success counter

Parameters

potential	the value of the counter

Warning

also calls storeLast()

7.12.3.17 bool Langmuir::FluxAgent::shouldTransport (int *site*) `[protected], [virtual]`

decide if the FluxAgent should transport a carrier to/from a given site

Parameters

site	the site involved

Returns

true if the FluxAgent should transport to/from the site

The default behavoir is for the FluxAgent to use a simple constant probabilty to make this decision. However, classes derived from FluxAgent can reimplement this function. For example, one might want to use a Metropolis criterion to make this decision.

Reimplemented in Langmuir::ExcitonSourceAgent, and Langmuir::SourceAgent.

7.12.3.18 unsigned long int Langmuir::FluxAgent::stepsSinceLast ( ) const

get the number of steps since storeLast() was called

7.12.3.19 void Langmuir::FluxAgent::storeLast ( )

set the value of last to the value of successes, and store the current step

7.12.3.20 unsigned long int Langmuir::FluxAgent::successes ( ) const

get the FluxAgent's success counter

7.12.3.21 unsigned long int Langmuir::FluxAgent::successesSinceLast ( ) const

get the number of successes since storeLast() was called

7.12.3.22 double Langmuir::FluxAgent::successProbability ( ) const

calculate and return the current probabilty of success

This is the number of successes divided by the number of attempts (x100). Ideally, this number should approach probability() as the simulation progresses, if shouldTransport() uses the simple constant probability method.

7.12.3.23 double Langmuir::FluxAgent::successProbabilitySinceLast ( ) const

calculate and return the probabilty of success since storeLast() was called

This is the number of successesSinceLast() divided by the number of attemptsSinceLast() (x100).

7.12.3.24 double Langmuir::FluxAgent::successRate ( ) const

calculate and return the current rate of success

This is the number of successes divided by the number of simulation steps. The current is related to the rate.

7.12.3.25 double Langmuir::FluxAgent::successRateSinceLast ( ) const

calculate and return the rate of success since storeLast() was called

This is the number of successesSinceLast() divided by the number of stepsSinceLast(). The current is related to the rate.

7.12.4 Member Data Documentation

7.12.4.1 unsigned long int Langmuir::FluxAgent::m_attempts [protected]

the number of times the FluxAgent has tried to transport.

7.12.4.2 Grid::CubeFace Langmuir::FluxAgent::m_face [protected]

the face of the grid this FluxAgent occupies

It may be Grid::NoFace is the FluxAgent occupies an actual site.

7.12.4.3 Grid& Langmuir::FluxAgent::m_grid [protected]

the grid this FluxAgent resides in

7.12.4.4 unsigned long int Langmuir::FluxAgent::m_lastAttempts [protected]

storage to note the number of successes at some step

7.12.4.5 unsigned long int Langmuir::FluxAgent::m_lastStep [protected]

the step at which last was noted

7.12.4.6 unsigned long int Langmuir::FluxAgent::m_lastSuccesses [protected]

storage to note the number of successes at some step

7.12.4.7 double Langmuir::FluxAgent::m_potential [protected]

the potential that is (possibly) used when calculating an energy change

The energy change can be used in the shouldTransport() function.

7.12.4.8 double Langmuir::FluxAgent::m_probability [protected]

the constant probability used in the default behavoir of shouldTransport().

7.12.4.9 unsigned long int Langmuir::FluxAgent::m_successes [protected]

the number of times the FluxAgent was successful in transporting.

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

- fluxagent.h
- fluxagent.cpp

7.13 Langmuir::FluxWriter Class Reference

A class to output source and drain info.

`#include <writer.h>`

Public Member Functions

- FluxWriter (World &world, const QString &name, QObject *parent=0)

  constructs the writer, has the same parameters as OutputInfo
- void write ()

  write the flux statistics of the current step to the stream

Protected Attributes

- World & m_world

  reference to the world object
- OutputStream m_stream

  output file stream

7.13.1 Detailed Description

A class to output source and drain info.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Langmuir::FluxWriter::FluxWriter (World & *world,* const QString & *name,* QObject * *parent* = 0)

constructs the writer, has the same parameters as OutputInfo

7.13.3 Member Function Documentation

7.13.3.1 void Langmuir::FluxWriter::write ( )

write the flux statistics of the current step to the stream

7.13.4 Member Data Documentation

7.13.4.1 OutputStream Langmuir::FluxWriter::m_stream [protected]

output file stream

7.13.4.2 World& Langmuir::FluxWriter::m_world [protected]

reference to the world object

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

- writer.h
- writer.cpp

7.14 Langmuir::Grid Class Reference

A class to hold Agents, calculate their positions, and store the background potential.

`#include <cubicgrid.h>`

Public Types

- enum CubeFace {
  Left = 0, Right = 1, Top = 2, Bottom = 3,
  Front = 4, Back = 5, NoFace = 6 }

  A way to indicate the faces of a cube.

Public Member Functions

- Grid (World &world, QObject ∗parent=0)

  Create a grid.
- ∼Grid ()

  Destroy the grid.
- int xSize ()

  Get the number of sites along the x-direction.
- int ySize ()

  Get the number of sites along the y-direction.
- int zSize ()

  Get the number of sites along the z-direction.
- int xyPlaneArea ()

  Get the number of sites in the xy-plane.
- int volume ()

  Get the total number of sites.
- double totalDistance (int site1, int site2)

Get the distance between two sites.

- double xDistance (int site1, int site2)

  Get the distance along the x-direction between two sites.
- double yDistance (int site1, int site2)

  Get the distance along the y-direction between two sites.
- double zDistance (int site1, int site2)

  Get the distance along the z-direction between two sites.
- double xImageDistance (int site1, int site2)

  Get the image distance along the x-direction between two sites.
- double yImageDistance (int site1, int site2)

  Get the image distance along the y-direction between two sites.
- double zImageDistance (int site1, int site2)

  Get the image distance along the z-direction between two sites.
- int xDistancei (int site1, int site2)

  Get the ***integer*** *distance along the x-direction between two sites.*
- int yDistancei (int site1, int site2)

  Get the ***integer*** *distance along the y-direction between two sites.*
- int zDistancei (int site1, int site2)

  Get the ***integer*** *distance along the z-direction between two sites.*
- int xImageDistancei (int site1, int site2)

  Get the ***integer*** *image distance along the x-direction between two sites.*
- int yImageDistancei (int site1, int site2)

  Get the ***integer*** *image distance along the y-direction between two sites.*
- int zImageDistancei (int site1, int site2)

  Get the ***integer*** *image distance along the z-direction between two sites.*
- int getIndexS (int xIndex, int yIndex, int zIndex=0)

  Get the serial site ID.
- int getIndexY (int site)

  Get the "y-site ID" from the "s-site ID".
- int getIndexX (int site)

  Get the "x-site ID" from the "s-site ID".
- int getIndexZ (int site)

  Get the "z-site ID" from the "s-site ID".
- double getPositionY (int site)

  Get the y-position from the "s-site ID".
- double getPositionX (int site)

  Get the x-position from the "s-site ID".
- double getPositionZ (int site)

  Get the z-position from the "s-site ID".
- Agent $*$ agentAddress (int site)

  Get a pointer to the Agent at a site.
- Agent::Type agentType (int site)

  Get the type of Agent at a site.
- void addToPotential (int site, double potential)

  Add some value to the background potential at a site.
- void setPotential (int site, double potential)

  Set the background potential at a site to some value.
- double potential (int site)

  Get the background potential at some site.
- QVector$<$ int $>$ neighborsSite (int site, int hoppingRange=1)

  Calculate the neighboring sites of a given site.

- QVector< int > neighborsFace (Grid::CubeFace cubeFace)

  Calculate the neighboring sites of a given face of the Grid.
- QVector< int > sliceIndex (int xi, int xf, int yi, int yf, int zi, int zf)

  Calculate the list of sites occupying a given range.
- void registerAgent (Agent *agent)

  Assign an Agent to a site in the Grid.
- void registerSpecialAgent (Agent *agent, Grid::CubeFace cubeFace)

  Assign an Agent to a special location.
- void unregisterAgent (Agent *agent)

  Remove an Agent from the Grid.
- void unregisterSpecialAgent (Agent *agent, Grid::CubeFace cubeFace)

  Remove an Agent from the special list of Agents in the Grid.
- void unregisterDefect (int site)

  Remove a defect from the Grid.
- void registerDefect (int site)

  Assign a site to be Agent::Defect.
- int specialAgentCount ()

  The total number of special Agents.
- QList< Agent * > & getSpecialAgentList (Grid::CubeFace cubeFace)

  Get a list of special Agents assigned to a specific Grid::CubeFace.

Static Public Member Functions

- static QString toQString (const Grid::CubeFace e)

Protected Attributes

- World & m_world

  Reference to the World object.
- QVector< Agent * > m_agents

  1D list of Agent pointers, the size of which is the volume of the Grid + the max number of special Agents.
- QVector< double > m_potentials

  1D list of site potentials, the size of which is the volume of the Grid + the max number of special Agents.
- QVector< Agent::Type > m_agentType

  1D list of Agent types, the size of which is the volume of the Grid + the max number of special Agents.
- QList< QList< Agent * > > m_specialAgents

  A list of lists of special agents, where each sub-list is for a different Grid::CubeFace.
- int m_specialAgentReserve

  The max number of special Agents allowed.
- int m_specialAgentCount

  The current number of special Agents registered with the Grid.
- int m_xSize

  The number of sites along the x-direction.
- int m_ySize

  The number of sites along the y-direction.
- int m_zSize

  The number of sites along the z-direction.
- int m_xyPlaneArea

  The number of sites in the xy-plane.
- int m_yzPlaneArea

The number of sites in the yz-plane.

- int m_xzPlaneArea

  The number of sites in the xz-plane.

- int m_volume

  The total number of sites.

7.14.1 Detailed Description

A class to hold Agents, calculate their positions, and store the background potential.

The x-direction

- perpendicular to the electrodes
- runs from left to right
- corresponds to the dimension called **length**.

The y-direction

- parallel to the electrodes
- runs from bottom to top
- corresponds to the dimension called **width**.

The z-direction

- parallel to the electrodes
- runs from back to front
- corresponds to the dimension called **height**.

7.14.2 Member Enumeration Documentation

7.14.2.1 enum **Langmuir::Grid::CubeFace**

A way to indicate the faces of a cube.

Enumerator:

Left x = 0, yz plane
Right x = lx, yz plane
Top z = 0, xy plane
Bottom z = lz, xy plane
Front y = 0, xz plane
Back y = ly, xz plane
NoFace undefined face

7.14.3 Constructor & Destructor Documentation

7.14.3.1 Langmuir::Grid::Grid (**World** & *world,* QObject ∗ *parent* = 0)

Create a grid.

Parameters

world	reference to the world object
parent	QObject this belongs to

7.14.3.2 Langmuir::Grid::∼Grid ( )

Destroy the grid.

7.14.4 Member Function Documentation

7.14.4.1 void Langmuir::Grid::addToPotential (int *site*, double *potential*)

Add some value to the background potential at a site.

Parameters

site	the "s-site ID"
potential	the value to add

7.14.4.2 Agent ∗ Langmuir::Grid::agentAddress (int *site*)

Get a pointer to the Agent at a site.

Parameters

site	the "s-site ID"

Warning

may be NULL if there is no Agent

7.14.4.3 Agent::Type Langmuir::Grid::agentType (int *site*)

Get the type of Agent at a site.

Parameters

site	the "s-site ID"

Warning

if there is no Agent, it should be Agent::Empty

7.14.4.4 int Langmuir::Grid::getIndexS (int *xIndex*, int *yIndex*, int *zIndex* = 0)

Get the serial site ID.

Parameters

xIndex	x site ID
yIndex	y site ID
zIndex	z site ID

The position of a particle in the Grid can be thought of as a 3-tuple of (x, y, z) site IDs. However, this 3-tuple can be mapped/hashed into a single number using the dimension of the grid, called the "serial site ID", the "s-site ID", or just the "site".

7.14.4.5 int Langmuir::Grid::getIndexX (int *site*)

Get the "x-site ID" from the "s-site ID".

Parameters

site	the "s-site ID"

The y-site ID can be thought of as the x-value of the cornor of a Grid site.

See Also

getIndexS

7.14.4.6 int Langmuir::Grid::getIndexY (int *site*)

Get the "y-site ID" from the "s-site ID".

Parameters

site	the "s-site ID"

The y-site ID can be thought of as the y-value of the cornor of a Grid site.

See Also

getIndexS

7.14.4.7 int Langmuir::Grid::getIndexZ (int *site*)

Get the "z-site ID" from the "s-site ID".

Parameters

site	the "s-site ID"

The y-site ID can be thought of as the z-value of the cornor of a Grid site.

See Also

getIndexS

7.14.4.8 double Langmuir::Grid::getPositionX (int *site*)

Get the x-position from the "s-site ID".

Parameters

site	the "s-site ID"

Particles are considered to reside in the "center" of Grid sites. The x-position is therefore the "x-site ID" plus 0.5 in reduced units.

7.14.4.9 double Langmuir::Grid::getPositionY (int *site*)

Get the y-position from the "s-site ID".

Parameters

site	the "s-site ID"

Particles are considered to reside in the "center" of Grid sites. The y-position is therefore the "y-site ID" plus 0.5 in reduced units.

7.14.4.10 double Langmuir::Grid::getPositionZ (int *site*)

Get the z-position from the "s-site ID".

Parameters

site	the "s-site ID"

Particles are considered to reside in the "center" of Grid sites. The z-position is therefore the "z-site ID" plus 0.5 in reduced units.

7.14.4.11 QList< Agent ∗ > & Langmuir::Grid::getSpecialAgentList (Grid::CubeFace *cubeFace*)

Get a list of special Agents assigned to a specific Grid::CubeFace.

Parameters

cubeFace	the face of the Grid

7.14.4.12 QVector< int > Langmuir::Grid::neighborsFace (Grid::CubeFace *cubeFace*)

Calculate the neighboring sites of a given face of the Grid.

Parameters

cubeFace	the face of the Grid to consider

7.14.4.13 QVector< int > Langmuir::Grid::neighborsSite (int *site*, int *hoppingRange* = 1)

Calculate the neighboring sites of a given site.

Parameters

site	the "s-site ID"
hoppingRange	the number of adjacent sites to consider in the calculation

7.14.4.14 double Langmuir::Grid::potential (int *site*)

Get the background potential at some site.

Parameters

site	the "s-site ID"

7.14.4.15 void Langmuir::Grid::registerAgent (**Agent** $*$ *agent*)

Assign an Agent to a site in the Grid.

Parameters

agent	a pointer to the Agent

Warning

uses Agent::getCurrentSite()
site must be Agent::Empty

Makes sure the site is empty first. After assigning the Agent to the site, calculates and assigns the neighbors to the Agent.

7.14.4.16 void Langmuir::Grid::registerDefect (int *site*)

Assign a site to be Agent::Defect.

Parameters

site	

7.14.4.17 void Langmuir::Grid::registerSpecialAgent (**Agent** $*$ *agent,* **Grid::CubeFace** *cubeFace*)

Assign an Agent to a special location.

Parameters

agent	a pointer to the Agent
cubeFace	the face of the Grid

Agents such as Sources and Drains do not occupy a site in the Grid, and so must be stored in a special location.

7.14.4.18 void Langmuir::Grid::setPotential (int *site,* double *potential*)

Set the background potential at a site to some value.

Parameters

site	the "s-site ID"
potential	the value to set

7.14.4.19 QVector< int > Langmuir::Grid::sliceIndex (int *xi,* int *xf,* int *yi,* int *yf,* int *zi,* int *zf*)

Calculate the list of sites occupying a given range.

Parameters

xi	starting x-site ID
xf	stopping x-site ID
yi	starting y-site ID
yf	stopping y-site ID
zi	starting z-site ID
zf	stopping z-site ID

7.14.4.20 int Langmuir::Grid::specialAgentCount ( )

The total number of special Agents.

7.14.4.21 QString Langmuir::Grid::toQString (const Grid::CubeFace *e*) [static]

7.14.4.22 double Langmuir::Grid::totalDistance (int *site1*, int *site2*)

Get the distance between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.23 void Langmuir::Grid::unregisterAgent (Agent * *agent*)

Remove an Agent from the Grid.

Parameters

agent	a pointer to the Agent

7.14.4.24 void Langmuir::Grid::unregisterDefect (int *site*)

Remove a defect from the Grid.

Parameters

site	the "s-site ID"

7.14.4.25 void Langmuir::Grid::unregisterSpecialAgent (Agent * *agent*, Grid::CubeFace *cubeFace*)

Remove an Agent from the special list of Agents in the Grid.

Parameters

agent	a pointer to the Agent
cubeFace	the face of the Grid

7.14.4.26 int Langmuir::Grid::volume ( )

Get the total number of sites.

7.14.4.27 double Langmuir::Grid::xDistance (int *site1*, int *site2*)

Get the distance along the x-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.28 int Langmuir::Grid::xDistancei (int *site1*, int *site2*)

Get the **integer** distance along the x-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.29 double Langmuir::Grid::xImageDistance (int *site1*, int *site2*)

Get the image distance along the x-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's x-position is taken to be the negative of its x-value (i.e., the particle is reflected through the yz-plane).

7.14.4.30 int Langmuir::Grid::xImageDistancei (int *site1*, int *site2*)

Get the **integer** image distance along the x-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's x-position is taken to be the negative of its x-value (i.e., the particle is reflected through the yz-plane).

7.14.4.31 int Langmuir::Grid::xSize ( )

Get the number of sites along the x-direction.

7.14.4.32 int Langmuir::Grid::xyPlaneArea ( )

Get the number of sites in the xy-plane.

7.14.4.33 double Langmuir::Grid::yDistance (int *site1*, int *site2*)

Get the distance along the y-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.34 int Langmuir::Grid::yDistancei (int *site1*, int *site2*)

Get the **integer** distance along the y-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.35 double Langmuir::Grid::yImageDistance (int *site1*, int *site2*)

Get the image distance along the y-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's y-position is taken to be the negative of its y-value (i.e., the particle is reflected through the xz-plane).

7.14.4.36 int Langmuir::Grid::yImageDistancei (int *site1*, int *site2*)

Get the **integer** image distance along the y-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's y-position is taken to be the negative of its y-value (i.e., the particle is reflected through the xz-plane).

7.14.4.37 int Langmuir::Grid::ySize ( )

Get the number of sites along the y-direction.

7.14.4.38 double Langmuir::Grid::zDistance (int *site1*, int *site2*)

Get the distance along the z-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.39 int Langmuir::Grid::zDistancei (int *site1*, int *site2*)

Get the **integer** distance along the z-direction between two sites.

Parameters

site1	the first site
site2	the second site

7.14.4.40 double Langmuir::Grid::zImageDistance (int *site1*, int *site2*)

Get the image distance along the z-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's z-position is taken to be the negative of its z-value (i.e., the particle is reflected through the xy-plane).

7.14.4.41 int Langmuir::Grid::zImageDistancei (int *site1*, int *site2*)

Get the **integer** image distance along the z-direction between two sites.

Parameters

site1	the first site
site2	the second site (reflected)

The second site's z-position is taken to be the negative of its z-value (i.e., the particle is reflected through the xy-plane).

7.14.4.42 int Langmuir::Grid::zSize ( )

Get the number of sites along the z-direction.

7.14.5 Member Data Documentation

7.14.5.1 QVector<Agent *> Langmuir::Grid::m_agents [protected]

1D list of Agent pointers, the size of which is the volume of the Grid + the max number of special Agents.

Warning

some of these may be NULL

Each position in the list is mapped to a position in the Grid. Use getIndexS() to calculate the serial site ID needed to index this list.

7.14.5.2 QVector<Agent::Type> Langmuir::Grid::m_agentType [protected]

1D list of Agent types, the size of which is the volume of the Grid + the max number of special Agents.

Warning

some of these may be Agent::Empty

Each position in the list is mapped to a position in the Grid. Use getIndexS() to calculate the serial site ID needed to index this list.

7.14.5.3 QVector<double> Langmuir::Grid::m_potentials [protected]

1D list of site potentials, the size of which is the volume of the Grid + the max number of special Agents.

Each position in the list is mapped to a position in the Grid. Use getIndexS() to calculate the serial site ID needed to index this list.

7.14.5.4 int Langmuir::Grid::m_specialAgentCount [protected]

The current number of special Agents registered with the Grid.

7.14.5.5 int Langmuir::Grid::m_specialAgentReserve [protected]

The max number of special Agents allowed.

7.14.5.6 QList< QList<Agent *> > Langmuir::Grid::m_specialAgents [protected]

A list of lists of special agents, where each sub-list is for a different Grid::CubeFace.

7.14.5.7 int Langmuir::Grid::m_volume [protected]

The total number of sites.

7.14.5.8 World& Langmuir::Grid::m_world [protected]

Reference to the World object.

7.14.5.9 int Langmuir::Grid::m_xSize [protected]

The number of sites along the x-direction.

7.14.5.10 int Langmuir::Grid::m_xyPlaneArea [protected]

The number of sites in the xy-plane.

7.14.5.11 int Langmuir::Grid::m_xzPlaneArea [protected]

The number of sites in the xz-plane.

7.14.5.12 int Langmuir::Grid::m_ySize [protected]

The number of sites along the y-direction.

7.14.5.13 int Langmuir::Grid::m_yzPlaneArea [protected]

The number of sites in the yz-plane.

7.14.5.14 int Langmuir::Grid::m_zSize [protected]

The number of sites along the z-direction.

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

- cubicgrid.h
- cubicgrid.cpp

7.15 Langmuir::GridImage Class Reference

A class to draw images of the grid.

`#include <writer.h>`

Public Member Functions

- GridImage (World &world, QColor bg=Qt::black, QObject ∗parent=0)

  create the image and painter, setting the background and size
- void drawSites (QList< int > &sites, QColor color, int layer)

  draw some sites
- void drawCharges (QList< ChargeAgent ∗ > &charges, QColor color, int layer)

  draw some sites
- void save (QString name, int scale=3)

  save the image to a file

Private Attributes

- QPainter m_painter

  the painter that paints the image
- QImage m_image

  the image we draw onto
- World & m_world

  reference to the world object

7.15.1 Detailed Description

A class to draw images of the grid.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Langmuir::GridImage::GridImage (World & *world*, QColor *bg* = Qt::black, QObject ∗ *parent* = 0)

create the image and painter, setting the background and size

Parameters

world	Reference to the world object
bg	Background color
parent	parent QObject

7.15.3 Member Function Documentation

7.15.3.1 void Langmuir::GridImage::drawCharges (QList< ChargeAgent * > & *charges,* QColor *color,* int *layer*)

draw some sites

Parameters

charges	A list of ChargeAgents, which have site ids • could be the list of electrons • could be the list of holes
color	The color of the points
layer	Which layer are we drawing? its a 2D image

7.15.3.2 void Langmuir::GridImage::drawSites (QList< int > & *sites,* QColor *color,* int *layer*)

draw some sites

Parameters

sites	A list of integers that are site ids • could be the list of trap ids • could be the list of defect ids
color	The color of the points
layer	Which layer are we drawing? its a 2D image

7.15.3.3 void Langmuir::GridImage::save (QString *name,* int *scale =* 3)

save the image to a file

Parameters

name	A file name that is passed to a OutputInfo object, the output is assummed png
scale	Multiply the image by some scale, increasing the resolution

7.15.4 Member Data Documentation

7.15.4.1 QImage Langmuir::GridImage::m_image [private]

the image we draw onto

7.15.4.2 QPainter Langmuir::GridImage::m_painter [private]

the painter that paints the image

7.15.4.3 World& Langmuir::GridImage::m_world [private]

reference to the world object

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

- writer.h
- writer.cpp

7.16 Langmuir::HoleAgent Class Reference

A class to represent moving positive charges.

```
#include <chargeagent.h>
```

Public Member Functions

- HoleAgent (World &world, int site, QObject ∗parent=0)

  Construct HoleAgent.

Protected Member Functions

- virtual double bindingPotential (int site)

  Calculate Exciton Binding Energy.
- virtual Agent::Type otherType ()

  Return other Agent::Type.
- virtual Grid & otherGrid ()

  Return other Grid.

Additional Inherited Members

7.16.1 Detailed Description

A class to represent moving positive charges.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 Langmuir::HoleAgent::HoleAgent (World & *world,* int *site,* QObject ∗ *parent* = 0)

Construct HoleAgent.

7.16.3 Member Function Documentation

7.16.3.1 double Langmuir::HoleAgent::bindingPotential (int *site*) [protected], [virtual]

Calculate Exciton Binding Energy.

Parameters

site	the site to check in other Grid

Returns

- -0.5 eV if exciton
- 0 otherwise

Implements Langmuir::ChargeAgent.

7.16.3.2 Grid & Langmuir::HoleAgent::otherGrid ( ) [protected], [virtual]

Return other Grid.

Returns

World::electronGrid

Implements Langmuir::ChargeAgent.

7.16.3.3 Agent::Type Langmuir::HoleAgent::otherType ( ) [protected], [virtual]

Return other Agent::Type.

Returns

Agent::Electron

Implements Langmuir::ChargeAgent.

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

- chargeagent.h
- chargeagent.cpp

7.17 Langmuir::HoleDrainAgent Class Reference

A class to remove HoleAgents.

`#include <drainagent.h>`

Public Member Functions

- HoleDrainAgent (World &world, int site, QObject ∗parent=0)

  create an HoleDrainAgent at a specific site
- HoleDrainAgent (World &world, Grid::CubeFace cubeFace, QObject ∗parent=0)

  create a HoleDrainAgent at a specific Grid::CubeFace

Protected Member Functions

- virtual double energyChange (int fSite)

  same as FluxAgent::energyChange(), but specialized for HoleAgents.

Additional Inherited Members

7.17.1 Detailed Description

A class to remove HoleAgents.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Langmuir::HoleDrainAgent::HoleDrainAgent (World & *world,* int *site,* QObject ∗ *parent* = 0)

create an HoleDrainAgent at a specific site

7.17.2.2 Langmuir::HoleDrainAgent::HoleDrainAgent (World & *world,* Grid::CubeFace *cubeFace,* QObject ∗ *parent =* 0)

create a HoleDrainAgent at a specific Grid::CubeFace

7.17.3 Member Function Documentation

7.17.3.1 double Langmuir::HoleDrainAgent::energyChange (int *fSite*) [protected], [virtual]

same as FluxAgent::energyChange(), but specialized for HoleAgents.

Note really used because the default FluxAgent::shouldTransport() behavoir, which is to use a simple constant probability, has not been reimplemented for DrainAgents.

Reimplemented from Langmuir::FluxAgent.

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

- drainagent.h
- drainagent.cpp

7.18 Langmuir::HoleSourceAgent Class Reference

A class to inject HoleAgents.

`#include <sourceagent.h>`

Public Member Functions

- HoleSourceAgent (World &world, int site, QObject ∗parent=0)

  create a HoleSourceAgent at a specific site
- HoleSourceAgent (World &world, Grid::CubeFace cubeFace, QObject ∗parent=0)

  create a HoleSourceAgent at a specific Grid::CubeFace

Protected Member Functions

- virtual bool validToInject (int site)

  same as SourceAgent::validToInject(), but specialized for HoleAgents.
- virtual double energyChange (int site)

  same as FluxAgent::energyChange(), but specialized for HoleAgents.
- virtual void inject (int site)

  same as SourceAgent::inject(), but specialized for HoleAgents.

Additional Inherited Members

7.18.1 Detailed Description

A class to inject HoleAgents.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Langmuir::HoleSourceAgent::HoleSourceAgent (World & *world,* int *site,* QObject ∗ *parent =* 0)

create a HoleSourceAgent at a specific site

7.18.2.2 Langmuir::HoleSourceAgent::HoleSourceAgent (World & *world,* Grid::CubeFace *cubeFace,* QObject ∗ *parent* = 0)

create a HoleSourceAgent at a specific Grid::CubeFace

7.18.3 Member Function Documentation

7.18.3.1 double Langmuir::HoleSourceAgent::energyChange (int *site*) `[protected]`, `[virtual]`

same as FluxAgent::energyChange(), but specialized for HoleAgents.

Reimplemented from Langmuir::FluxAgent.

7.18.3.2 void Langmuir::HoleSourceAgent::inject (int *site*) `[protected]`, `[virtual]`

same as SourceAgent::inject(), but specialized for HoleAgents.

Implements Langmuir::SourceAgent.

7.18.3.3 bool Langmuir::HoleSourceAgent::validToInject (int *site*) `[protected]`, `[virtual]`

same as SourceAgent::validToInject(), but specialized for HoleAgents.

Implements Langmuir::SourceAgent.

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

- sourceagent.h
- sourceagent.cpp

7.19 Langmuir::KeyValueParser Class Reference

A class to read the parameters and store them in the correct place.

```
#include <keyvalueparser.h>
```

Public Member Functions

- KeyValueParser (World &world, QObject ∗parent=0)

  Create a KeyValueParser.
- ∼KeyValueParser ()

  Destroy the KeyValueParser.
- SimulationParameters & parameters ()

  Get the SimulationParameters.
- void parse (const QString &line)

  Parse a string and assign a value to the correct parameter.
- void save (const QString &fileName="%stub.parm")

  Write the parameters to a file in a "key=value" fashion.
- Variable & getVariable (const QString &key)

  Get a reference to a variable by name.

Private Member Functions

- template<typename T >
  void registerVariable (const QString &key, T &value, Variable::VariableMode mode=0)

  Register an allowed variable with the parser.

Private Attributes

- QMap< QString, Variable ∗ > m_variableMap

  A map between variable names and variables.
- QStringList m_orderedNames

  A list of variable names in a specific order.
- SimulationParameters m_parameters

  The simulation parameters.
- World & m_world

  Reference to World object.

Friends

- std::ostream & operator<< (std::ostream &stream, const KeyValueParser &keyValueParser)

  Write the parameters to a std::ostream in a "key=value" fashion.

7.19.1 Detailed Description

A class to read the parameters and store them in the correct place.

The location of the SimulationParameters object for the entire simulation is a private variable of this class.

To add new variables, follow these steps:

- declare the new variable in the SimulationParameters struct (parameters.h)
- assign the default value of the new variable in the SimulationParameters constructor (parameters.h)
- implement validity checking for the variable in the checkSimulationParameters() function (parameters.h)
- register the variable in the KeyValueParser constructor using the registerVariable() function (keyvalueparser.-h)
- to use non-standard types, you must overload certain template functions in variable.h. See, for example, overloads for QDateTime in variable.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Langmuir::KeyValueParser::KeyValueParser (World & *world,* QObject ∗ *parent* = 0)

Create a KeyValueParser.

Parameters

world	reference to World Object
parent	QObject this belongs to

Add calls to registerVariable() to add new variables to the simulation.

7.19.2.2 Langmuir::KeyValueParser::∼KeyValueParser ( )

Destroy the KeyValueParser.

7.19.3 Member Function Documentation

7.19.3.1 Variable & Langmuir::KeyValueParser::getVariable (const QString & *key*)

Get a reference to a variable by name.

Parameters

name	the name of the variable

7.19.3.2 SimulationParameters & Langmuir::KeyValueParser::parameters ( )

Get the SimulationParameters.

7.19.3.3 void Langmuir::KeyValueParser::parse (const QString & *line*)

Parse a string and assign a value to the correct parameter.

7.19.3.4 template<typename T > void Langmuir::KeyValueParser::registerVariable (const QString & *key*, T & *value*, Variable::VariableMode *mode* = 0) [private]

Register an allowed variable with the parser.

7.19.3.5 void Langmuir::KeyValueParser::save (const QString & *fileName* = "%stub.parm")

Write the parameters to a file in a "key=value" fashion.

7.19.4 Friends And Related Function Documentation

7.19.4.1 std::ostream& operator<< (std::ostream & *stream*, const KeyValueParser & *keyValueParser*) [friend]

Write the parameters to a std::ostream in a "key=value" fashion.

7.19.5 Member Data Documentation

7.19.5.1 QStringList Langmuir::KeyValueParser::m_orderedNames [private]

A list of variable names in a specific order.

7.19.5.2 SimulationParameters Langmuir::KeyValueParser::m_parameters [private]

The simulation parameters.

7.19.5.3 QMap<QString,Variable∗> Langmuir::KeyValueParser::m_variableMap [private]

A map between variable names and variables.

7.19.5.4 World& Langmuir::KeyValueParser::m_world [private]

Reference to World object.

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

- keyvalueparser.h
- keyvalueparser.cpp

7.20 Langmuir::Logger Class Reference

A class that organizes output.

`#include <writer.h>`

Public Member Functions

- Logger (World &world, QObject ∗parent=0)

  create Logger
- virtual void saveTrapImage (const QString &name="%stub-traps.png")

  save an image of trap sites as png
- virtual void saveHoleImage (const QString &name="%stub-%step-holes.png")

  save an image of holes (at the current step) as png
- virtual void saveElectronImage (const QString &name="%stub-%step-electrons.png")

  save an image of electrons (at the current step) as png
- virtual void saveCarriersImage (const QString &name="%stub-%step-carriers.png")

  save an image of holes ***and*** *electrons (at the current step) as png*
- virtual void saveDefectImage (const QString &name="%stub-defects.png")

  save an image of defects as png
- virtual void saveImage (const QString &name="%stub-%step-all.png")

  save an image of electrons, holes, defects, and traps (at current step) as png
- virtual void saveGridPotential (const QString &name="%stub.grid")

  output the grid potential as (x, y, z, v) to a file
- virtual void saveCoulombEnergy (const QString &name="%stub-%step.coulomb")

  output the Coulomb potential as (x, y, z, v) to a file; ***requires*** *the use of the* ***GPU***
- virtual void reportFluxStream ()

  output information about Sources and Drains (at the current step) to the main output file
- virtual void reportXYZStream ()

  output xyz information (at the current step) to the xyz file
- virtual void reportCarrier (ChargeAgent &charge)

  output carrier information (for example pathlength) to the carrier file
- virtual void reportExciton (ChargeAgent &charge1, ChargeAgent &charge2, bool recombined=false)

  output carrier information (for example pathlength) on two carriers at once to the exciton file
- virtual void initialize ()

  open the various output streams if they are turned on

Protected Attributes

- World & m_world

    reference to world
- XYZWriter ∗ m_xyzWriter

    writer in charge of writing xyz files
- FluxWriter ∗ m_fluxWriter

    writer in charge of writing source & drain information
- CarrierWriter ∗ m_carrierWriter

    writer in charge of writing carrier information
- ExcitonWriter ∗ m_excitonWriter

    writer in charge of writing multiple carrier's information (excitons)

7.20.1 Detailed Description

A class that organizes output.

Warning

You must manually call initialize() to open output streams

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Langmuir::Logger::Logger (World & *world*, QObject ∗ *parent* = 0)

create Logger

7.20.3 Member Function Documentation

7.20.3.1 void Langmuir::Logger::initialize ( ) [virtual]

open the various output streams if they are turned on

7.20.3.2 void Langmuir::Logger::reportCarrier (ChargeAgent & *charge*) [virtual]

output carrier information (for example pathlength) to the carrier file

7.20.3.3 void Langmuir::Logger::reportExciton (ChargeAgent & *charge1*, ChargeAgent & *charge2*, bool *recombined* = false) [virtual]

output carrier information (for example pathlength) on two carriers at once to the exciton file

7.20.3.4 void Langmuir::Logger::reportFluxStream ( ) [virtual]

output information about Sources and Drains (at the current step) to the main output file

7.20.3.5 void Langmuir::Logger::reportXYZStream ( ) [virtual]

output xyz information (at the current step) to the xyz file

7.20.3.6 void Langmuir::Logger::saveCarriersImage (const QString & *name* = "%stub-%step-carriers.png") [virtual]

save an image of holes **and** electrons (at the current step) as png

7.20.3.7 void Langmuir::Logger::saveCoulombEnergy (const QString & *name* = "%stub-%step.coulomb") [virtual]

output the Coulomb potential as (x, y, z, v) to a file; **requires** the use of the **GPU**

7.20.3.8 void Langmuir::Logger::saveDefectImage (const QString & *name* = "%stub-defects.png") [virtual]

save an image of defects as png

7.20.3.9 void Langmuir::Logger::saveElectronImage (const QString & *name* = "%stub-%step-electrons.png") [virtual]

save an image of electrons (at the current step) as png

7.20.3.10 void Langmuir::Logger::saveGridPotential (const QString & *name* = "%stub.grid") [virtual]

output the grid potential as (x, y, z, v) to a file

7.20.3.11 void Langmuir::Logger::saveHoleImage (const QString & *name* = "%stub-%step-holes.png") [virtual]

save an image of holes (at the current step) as png

7.20.3.12 void Langmuir::Logger::saveImage (const QString & *name* = "%stub-%step-all.png") [virtual]

save an image of electrons, holes, defects, and traps (at current step) as png

7.20.3.13 void Langmuir::Logger::saveTrapImage (const QString & *name* = "%stub-traps.png") [virtual]

save an image of trap sites as png

7.20.4 Member Data Documentation

7.20.4.1 CarrierWriter* Langmuir::Logger::m_carrierWriter [protected]

writer in charge of writing carrier information

7.20.4.2 ExcitonWriter* Langmuir::Logger::m_excitonWriter [protected]

writer in charge of writing multiple carrier's information (excitons)

7.20.4.3 FluxWriter* Langmuir::Logger::m_fluxWriter [protected]

writer in charge of writing source & drain information

7.20.4.4 World& Langmuir::Logger::m_world [protected]

reference to world

7.20.4.5 XYZWriter∗ Langmuir::Logger::m_xyzWriter [protected]

writer in charge of writing xyz files

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

- writer.h
- writer.cpp

7.21 Langmuir::OpenClHelper Class Reference

A Class to run OpenCL calculations.

```
#include <openclhelper.h>
```

Public Member Functions

- OpenClHelper (World &world, QObject ∗parent=0)

  *Create **THE** OpenClHelper; don't make more than one.*
- void initializeOpenCL ()

  Perform the tedious boilerplate code to initialize OpenCL.
- void launchCoulombKernel1 ()

  *Kernel1 calculates the coulomb potential at **every** site.*
- void launchCoulombKernel2 ()

  Kernel2 calculates the coulomb potential at current and future sites only.
- void launchGaussKernel1 ()

  *Kernel1 calculates the coulomb potential with erf at **every** site.*
- void launchGaussKernel2 ()

  Kernel2 calculates the coulomb potential with erf at current and future sites only.
- void copySiteAndChargeToHostVector (int index, int site, int charge=-1)

  Does exactly what it says (host means the memory on the CPU)
- double getOutputHost (int index) const

  Get the result stored in host memory (for current site)
- double getOutputHostFuture (int index) const

  Get the result stored in host memory (for future site)
- void compareHostAndDeviceForAllCarriers ()

  Compare GPU and CPU results.
- bool toggleOpenCL (bool on)

  Turn on/off OpenCL in a smart-way.

Private Attributes

- World & m_world

  Reference to World object.

7.21.1 Detailed Description

A Class to run OpenCL calculations.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Langmuir::OpenClHelper::OpenClHelper (World & *world*, QObject ∗ *parent* = 0)

Create **THE** OpenClHelper; don't make more than one.

Parameters

world	reference to World Object
parent	QObject this belongs to

Warning

initializeOpenCL() must be called seperately

7.21.3 Member Function Documentation

7.21.3.1 void Langmuir::OpenClHelper::compareHostAndDeviceForAllCarriers ( )

Compare GPU and CPU results.

7.21.3.2 void Langmuir::OpenClHelper::copySiteAndChargeToHostVector (int *index*, int *site*, int *charge* = -1)

Does exactly what it says (host means the memory on the CPU)

Parameters

index	position in host vectors
site	serial site-id
charge	charge of carrier

7.21.3.3 double Langmuir::OpenClHelper::getOutputHost (int *index*) const

Get the result stored in host memory (for current site)

Parameters

index	position in host vectors

7.21.3.4 double Langmuir::OpenClHelper::getOutputHostFuture (int *index*) const

Get the result stored in host memory (for future site)

Parameters

index	position in host vectors

There is a fixed offset in the host memory between the current and future site results

7.21.3.5 void Langmuir::OpenClHelper::initializeOpenCL ( )

Perform the tedious boilerplate code to initialize OpenCL.

7.21.3.6 void Langmuir::OpenClHelper::launchCoulombKernel1 ( )

Kernel1 calculates the coulomb potential at **every** site.

This is extremely expensive on the CPU, it would take forever. The GPU will do it in a few seconds. Luckily, the only reason to ever call this kernel is if we want to save a snapshot of the coulomb potential - something that is not needed during a normal simulation.

7.21.3.7 void Langmuir::OpenClHelper::launchCoulombKernel2 ( )

Kernel2 calculates the coulomb potential at current and future sites only.

7.21.3.8 void Langmuir::OpenClHelper::launchGaussKernel1 ( )

Kernel1 calculates the coulomb potential with erf at **every** site.

7.21.3.9 void Langmuir::OpenClHelper::launchGaussKernel2 ( )

Kernel2 calculates the coulomb potential with erf at current and future sites only.

7.21.3.10 bool Langmuir::OpenClHelper::toggleOpenCL (bool *on*)

Turn on/off OpenCL in a smart-way.

Parameters

on	True if on

Returns

The on/off status

For example, don't allow one to turn OpenCL on if OpenCL can't be used on this platform.

7.21.4 Member Data Documentation

7.21.4.1 World& Langmuir::OpenClHelper::m_world [private]

Reference to World object.

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

- openclhelper.h
- openclhelper.cpp

7.22 Langmuir::OutputInfo Class Reference

A class to generate file names using the SimulationParameters.

`#include <output.h>`

Public Member Functions

- OutputInfo (const QString &name, const SimulationParameters ∗par=0)

  Generate file name according to SimulationParameters.

7.22.1 Detailed Description

A class to generate file names using the SimulationParameters.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 Langmuir::OutputInfo::OutputInfo (const QString & *name*, const SimulationParameters ∗ *par* = 0)

Generate file name according to SimulationParameters.

The constructor makes useful substitutions into the passed name (deatiled below) as well as making sure the name generated is valid (according to the passed SimulationParameters). If the directory of the passed name doesn't exist, it will be created.

Parameters

name	the file name desired. The following substitutions can be made: • **"%stub"**, substitutes in SimulationParameters::outputStub • **"%step"**, substitutes in SimulationParameters::currentStep
par	pointer to a SimulationParameters object • if 0 or NULL, then all substitutions become empty strings

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

- output.h
- output.cpp

7.23 Langmuir::OutputStream Class Reference

A class to combine QFile, QTextStream and OutputInfo (QFileInfo).

```
#include <output.h>
```

Public Member Functions

- OutputStream (const QString &name, const SimulationParameters ∗par=0, QObject ∗parent=0)

  Setup the QTextStream, QFile, and OutputInfo.
- ∼OutputStream ()

  Flush the stream and close the file.
- const OutputInfo & info ()

  Get the info object to get things like file name and path.
- const QFile & file ()

  Get the file object, though you probably have no need for it.

Private Attributes

- OutputInfo m_info

    OutputInfo object that generated file name.
- QFile m_file

    QFile object, the device of this QTextStream.

7.23.1 Detailed Description

A class to combine QFile, QTextStream and OutputInfo (QFileInfo).

Only for used for output. Derived from QObject so destruction ensures streams are flushed and files are closed.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Langmuir::OutputStream::OutputStream (const QString & *name*, const SimulationParameters ∗ *par* = 0, QObject ∗ *parent* = 0)

Setup the QTextStream, QFile, and OutputInfo.

The parameters are the same as OutputInfo. Opens the file as QIODevice::Text|QIODevice::WriteOnly. Will open with QIODevice::Append if Outout::Options::AppendMode is given.

See Also

OutputInfo::OutputInfo

7.23.2.2 Langmuir::OutputStream::∼OutputStream ( )

Flush the stream and close the file.

7.23.3 Member Function Documentation

7.23.3.1 const QFile & Langmuir::OutputStream::file ( )

Get the file object, though you probably have no need for it.

7.23.3.2 const OutputInfo & Langmuir::OutputStream::info ( )

Get the info object to get things like file name and path.

7.23.4 Member Data Documentation

7.23.4.1 QFile Langmuir::OutputStream::m_file [private]

QFile object, the device of this QTextStream.

7.23.4.2 OutputInfo Langmuir::OutputStream::m_info [private]

OutputInfo object that generated file name.

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

- output.h
- output.cpp

7.24 Langmuir::PBSGPUParser Class Reference

```
#include <pbsgpuparser.h>
```

Public Member Functions

- PBSGPUParser (QObject ∗parent=0)

  create the PBSGPUParser
- void setPath (const QString &path="")

  set the path of the PBS_GPUFILE
- void aquirePath ()

  get the path of the PBS_GPUFILE from the ENVIRONMENT variables
- void setDefault ()

  reset to default GPU(0)
- void parse ()

  parse the PBS_GPUFILE
- QList< int > nodes ()

  get the list of nodes ids found
- QList< int > gpus ()

  get the list of GPU ids found
- int size ()

  get the number of GPUs found
- int node ()

  get the first node found
- int gpu ()

  get the first GPU found

Private Attributes

- QList< int > m_nodes

  list of nodes
- QList< int > m_gpus

  list of GPU ids
- QString m_path

  path to PBS_GPUFILE

7.24.1 Detailed Description

A class to parse the PBS_GPUFILE - needed for selecting the correct GPU on a cluster using PBS

7.24.2 Constructor & Destructor Documentation

7.24.2.1 Langmuir::PBSGPUParser::PBSGPUParser (QObject ∗ *parent* = 0) [explicit]

create the PBSGPUParser

7.24.3 Member Function Documentation

7.24.3.1 void Langmuir::PBSGPUParser::aquirePath ( )

get the path of the PBS_GPUFILE from the ENVIRONMENT variables

7.24.3.2 int Langmuir::PBSGPUParser::gpu ( )

get the first GPU found

7.24.3.3 QList< int > Langmuir::PBSGPUParser::gpus ( )

get the list of GPU ids found

7.24.3.4 int Langmuir::PBSGPUParser::node ( )

get the first node found

7.24.3.5 QList< int > Langmuir::PBSGPUParser::nodes ( )

get the list of nodes ids found

7.24.3.6 void Langmuir::PBSGPUParser::parse ( )

parse the PBS_GPUFILE

7.24.3.7 void Langmuir::PBSGPUParser::setDefault ( )

reset to default GPU(0)

7.24.3.8 void Langmuir::PBSGPUParser::setPath (const QString & *path* = "")

set the path of the PBS_GPUFILE

7.24.3.9 int Langmuir::PBSGPUParser::size ( )

get the number of GPUs found

7.24.4 Member Data Documentation

7.24.4.1 QList<int> Langmuir::PBSGPUParser::m_gpus `[private]`

list of GPU ids

7.24.4.2 QList<int> Langmuir::PBSGPUParser::m_nodes `[private]`

list of nodes

7.24.4.3 QString Langmuir::PBSGPUParser::m_path [private]

path to PBS_GPUFILE

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

- pbsgpuparser.h
- pbsgpuparser.cpp

7.25 Langmuir::Potential Class Reference

A class to calculate the potential.

`#include <potential.h>`

Public Member Functions

- Potential (World &world, QObject ∗parent=0)

  Potential Create the potential.
- void setPotentialZero ()

  sets the value of the potential to zero at every grid site
- void setPotentialLinear ()

  Adds a linear potential calcualted from voltage.left and voltage.right along the x-direction.
- void setPotentialGate ()

  Adds a linear potential calculated from slope.z along the z-direction.
- void setPotentialTraps (const QList< int > &trapIDs=QList< int >(), const QList< double > &trapPotentials=QList< double >())

  Adds shifts to the potential at the various sites.
- void precalculateArrays ()

  pre-calculates r2, r, and 1/r
- void updateCouplingConstants ()

  pre-calculates coupling constants
- double coulombE (int site_i)

  calculates Coulomb potential from electrons at specific grid site
- double coulombImageE (int site_i)

  calculates Coulomb image-potential from electrons at specific grid site
- double gaussE (int site_i)

  calculates Coulomb potential from electrons at specific grid site, assuming gaussians
- double gaussImageE (int site_i)

  calculates Coulomb image-potential from electrons at specific grid site, assuming gaussians
- double coulombH (int site_i)

  calculates Coulomb potential from holes at specific grid site
- double coulombImageH (int site_i)

  calculates Coulomb image-potential from holes at specific grid site
- double gaussH (int site)

  calculates Coulomb potential from holes at specific grid site, assuming gaussians
- double gaussImageH (int site)

  calculates Coulomb image-potential from holes at specific grid site, assuming gaussians
- double coulombD (int site_i)

  calculates Coulomb potential from charged defects at specific grid site
- double coulombImageD (int site_i)

calculates Coulomb image-potential from charged defects at specific grid site

- double gaussD (int site_i)

  calculates Coulomb potential from charged defects at specific grid site, assuming gaussians

- double gaussImageD (int site_i)

  calculates Coulomb image-potential from charged defects at specific grid site, assuming gaussians

Private Attributes

- World & m_world

  reference to the World

7.25.1 Detailed Description

A class to calculate the potential.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 Langmuir::Potential::Potential (World & *world,* QObject ∗ *parent* = 0)

Potential Create the potential.

Parameters

world	reference to the World
parent	QObject this belongs to

7.25.3 Member Function Documentation

7.25.3.1 double Langmuir::Potential::coulombD (int *site_i*)

calculates Coulomb potential from charged defects at specific grid site

Parameters

site	the site of interest

7.25.3.2 double Langmuir::Potential::coulombE (int *site_i*)

calculates Coulomb potential from electrons at specific grid site

Parameters

site	the site of interest

7.25.3.3 double Langmuir::Potential::coulombH (int *site_i*)

calculates Coulomb potential from holes at specific grid site

Parameters

site	the site of interest

7.25.3.4 double Langmuir::Potential::coulombImageD (int *site_i*)

calculates Coulomb image-potential from charged defects at specific grid site

Parameters

site	the site of interest

7.25.3.5 double Langmuir::Potential::coulombImageE (int *site_i*)

calculates Coulomb image-potential from electrons at specific grid site

Parameters

site	the site of interest

7.25.3.6 double Langmuir::Potential::coulombImageH (int *site_i*)

calculates Coulomb image-potential from holes at specific grid site

Parameters

site	the site of interest

7.25.3.7 double Langmuir::Potential::gaussD (int *site_i*)

calculates Coulomb potential from charged defects at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.8 double Langmuir::Potential::gaussE (int *site_i*)

calculates Coulomb potential from electrons at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.9 double Langmuir::Potential::gaussH (int *site*)

calculates Coulomb potential from holes at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.10 double Langmuir::Potential::gaussImageD (int *site_i*)

calculates Coulomb image-potential from charged defects at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.11 double Langmuir::Potential::gaussImageE (int *site_i*)

calculates Coulomb image-potential from electrons at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.12 double Langmuir::Potential::gaussImageH (int *site*)

calculates Coulomb image-potential from holes at specific grid site, assuming gaussians

Parameters

site	the site of interest

7.25.3.13 void Langmuir::Potential::precalculateArrays ( )

pre-calculates r2, r, and 1/r

7.25.3.14 void Langmuir::Potential::setPotentialGate ( )

Adds a linear potential calculated from slope.z along the z-direction.

7.25.3.15 void Langmuir::Potential::setPotentialLinear ( )

Adds a linear potential calcualted from voltage.left and voltage.right along the x-direction.

7.25.3.16 void Langmuir::Potential::setPotentialTraps (const QList< int > & *trapIDs* = `QList<int>()`, const QList< double > & *trapPotentials* = `QList<double>()`)

Adds shifts to the potential at the various sites.

Parameters

trapIDs	list of site ids
trapPotentials	list of shifts

7.25.3.17 void Langmuir::Potential::setPotentialZero ( )

sets the value of the potential to zero at every grid site

7.25.3.18 void Langmuir::Potential::updateCouplingConstants ( )

pre-calculates coupling constants

7.25.4 Member Data Documentation

7.25.4.1 World& Langmuir::Potential::m_world [private]

reference to the World

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

- potential.h
- potential.cpp

7.26 Langmuir::Random Class Reference

A class to generate random numbers.

`#include <rand.h>`

Public Member Functions

- Random (quint64 seed=0, QObject ∗parent=0)

  Random.
- ∼Random ()

  Destroy objects.
- quint64 seed ()

  Get the seed that was used.
- void seed (quint64 seed)

  seed the generator (again)
- double random ()

  Generate a random double from the uniform distribution [0, 1].
- double range (const double low=0.0, const double high=1.0)

  Generate a random double from the uniform distribution [low, high].
- double normal (const double mean, const double sigma)

  Generate a random double from the normal distribution.
- int integer (const int low=0, const int high=1)

  Generate a random int from the uniform distribution [low, high].
- bool metropolis (double energyChange, double inversekT)

  Randomly choose yes using a Boltzmann factor.
- bool metropolisWithCoupling (double energyChange, double inversekT, double coupling)

  Randomly choose yes using a Boltzmann factor and coupling constant.
- bool chooseYes (double percent)

  Randomly choose yes a percent of the time.
- bool chooseNo (double percent)

  Randomly choose no a percent of the time.

Private Attributes

- boost::mt19937 ∗ twister

  The underlying random number generator.
- boost::variate_generator
  < boost::mt19937
  &, boost::uniform_01< double > > ∗ generator01

  The underlying generator coupled to the uniform distribution on [0,1].
- quint64 m_seed

  The seed used to start the generator.

Friends

- QDataStream & operator<< (QDataStream &stream, Random &random)

  *Output the random state to a QDataStream **Possibly Broken**.*
- QDataStream & operator>> (QDataStream &stream, Random &random)

  *Load the random state from a QDataStream **Possibly Broken**.*
- QTextStream & operator<< (QTextStream &stream, Random &random)

  *Output the random state to a QTextStream **Possibly Broken**.*
- QTextStream & operator>> (QTextStream &stream, Random &random)

  *Load the random state from a QTextStream **Possibly Broken**.*
- std::ostream & operator<< (std::ostream &stream, Random &random)

  Output the random state to a std::ostream.
- std::istream & operator>> (std::istream &stream, Random &random)

  Load the random state from a std::istream.

7.26.1 Detailed Description

A class to generate random numbers.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Langmuir::Random::Random (quint64 *seed* = 0, QObject ∗ *parent* = 0)

Random.

Parameters

seed	makes the generator deterministic • seed == 0 uses the current clock time
parent	object this belongs to

7.26.2.2 Langmuir::Random::∼Random ( )

Destroy objects.

7.26.3 Member Function Documentation

7.26.3.1 bool Langmuir::Random::chooseNo (double *percent*)

Randomly choose no a percent of the time.

7.26.3.2 bool Langmuir::Random::chooseYes (double *percent*)

Randomly choose yes a percent of the time.

7.26.3.3 int Langmuir::Random::integer (const int *low* = 0, const int *high* = 1)

Generate a random int from the uniform distribution [low, high].

7.26.3.4 bool Langmuir::Random::metropolis (double *energyChange,* double *inversekT*)

Randomly choose yes using a Boltzmann factor.

Parameters

energyChange	change in energy when going from initial to final state
inversekT	decay constant in exponential

7.26.3.5 bool Langmuir::Random::metropolisWithCoupling (double *energyChange,* double *inversekT,* double *coupling*)

Randomly choose yes using a Boltzmann factor and coupling constant.

Parameters

energyChange	change in energy when going from initial to final state
inversekT	decay constant in exponential
coupling	alters acceptance probability • if energy $<$ 0 : chooses yes coupling $*$ Boltzmann factor percent of the time • if energy $>$ 0 : chooses yes 1 - coupling percent of the time

7.26.3.6 double Langmuir::Random::normal (const double *mean,* const double *sigma*)

Generate a random double from the normal distribution.

Parameters

mean	average value of the normal distribution sampled
sigma	standard deviation of the normal distribution sampled

7.26.3.7 double Langmuir::Random::random ( )

Generate a random double from the uniform distribution [0, 1].

7.26.3.8 double Langmuir::Random::range (const double *low* = `0.0`, const double *high* = `1.0`)

Generate a random double from the uniform distribution [low, high].

7.26.3.9 quint64 Langmuir::Random::seed ( )

Get the seed that was used.

7.26.3.10 void Langmuir::Random::seed (quint64 *seed*)

seed the generator (again)

If seed == 0 is used, then the generator **does not** use the current time. This is different than the constructor Random.

7.26.4 Friends And Related Function Documentation

7.26.4.1 QDataStream& operator<< (QDataStream & *stream*, Random & *random*) [friend]

Output the random state to a QDataStream **Possibly Broken**.

Warning

This may not quite be working correctly

7.26.4.2 QTextStream& operator<< (QTextStream & *stream*, Random & *random*) [friend]

Output the random state to a QTextStream **Possibly Broken**.

Warning

This may not quite be working correctly

7.26.4.3 std::ostream& operator<< (std::ostream & *stream*, Random & *random*) [friend]

Output the random state to a std::ostream.

7.26.4.4 QDataStream& operator>> (QDataStream & *stream*, Random & *random*) [friend]

Load the random state from a QDataStream **Possibly Broken**.

Warning

This may not quite be working correctly

7.26.4.5 QTextStream& operator>> (QTextStream & *stream*, Random & *random*) [friend]

Load the random state from a QTextStream **Possibly Broken**.

Warning

This may not quite be working correctly

7.26.4.6 std::istream& operator>> (std::istream & *stream*, Random & *random*) [friend]

Load the random state from a std::istream.

7.26.5 Member Data Documentation

7.26.5.1 boost::variate_generator<boost::mt19937&, boost::uniform_01<double> >* Langmuir::Random::generator01 [private]

The underlying generator coupled to the uniform distribution on [0,1].

7.26.5.2 quint64 Langmuir::Random::m_seed [private]

The seed used to start the generator.

7.26.5.3 boost::mt19937∗ Langmuir::Random::twister [private]

The underlying random number generator.

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

- rand.h
- rand.cpp

7.27 Langmuir::RecombinationAgent Class Reference

A class to remove Excitons.

`#include <drainagent.h>`

Public Member Functions

- RecombinationAgent (World &world, QObject ∗parent=0)

  create a RecombinationAgent
- virtual bool tryToAccept (ChargeAgent ∗charge)

  accept charge with constant probability
- void guessProbability ()

  calculate an acceptance probability based upon the desired rate and encounter frequency

Private Member Functions

- virtual double energyChange (int fSite)

  currently implemented as zero and not really used

Additional Inherited Members

7.27.1 Detailed Description

A class to remove Excitons.

Currently, the RecombinationAgent is not really doing much of anything outside of keeping track of recombination statistics. The meat of the recombination resides in Simulation::performRecombinations(). This should probably be changes in the future.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 Langmuir::RecombinationAgent::RecombinationAgent (World & *world*, QObject ∗ *parent* = 0)

create a RecombinationAgent

7.27.3 Member Function Documentation

7.27.3.1 double Langmuir::RecombinationAgent::energyChange (int *fSite*) [private], [virtual]

currently implemented as zero and not really used

7.27.3.2 void Langmuir::RecombinationAgent::guessProbability ( )

calculate an acceptance probability based upon the desired rate and encounter frequency

7.27.3.3 bool Langmuir::RecombinationAgent::tryToAccept (ChargeAgent ∗ *charge*) [virtual]

accept charge with constant probability

Reimplemented from Langmuir::DrainAgent.

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

- drainagent.h
- drainagent.cpp

7.28 Langmuir::Simulation Class Reference

A class to orchestrate the calculation.

`#include <simulation.h>`

Public Member Functions

- Simulation (World &world, QObject ∗parent=0)

  Create a Simulation.
- virtual ∼Simulation ()

  Destroy the Simulation.
- virtual void performIterations (int nIterations)

  simulate for a set number of steps

Protected Member Functions

- void performRecombinations ()

  Recombine holes and electrons (in solarcell simulations only)
- void performInjections ()

  Tell sources to inject charges.
- void balanceCharges ()

  ***Try** to use the sources to keep the number of ChargeAgents balanced*
- void nextTick ()

  Remove charges from the simulation.

Static Protected Member Functions

- static void chargeAgentCoulombInteractionQtConcurrentCPU (ChargeAgent ∗chargeAgent)

  A method needed to call ChargeAgent::coulombCPU() in parallel.
- static void chargeAgentCoulombInteractionQtConcurrentGPU (ChargeAgent ∗chargeAgent)

  A method needed to call ChargeAgent::coulombGPU() in parallel.

Protected Attributes

- World & m_world

  Reference to World object.

7.28.1 Detailed Description

A class to orchestrate the calculation.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Langmuir::Simulation::Simulation (World & *world*, QObject ∗ *parent* = 0)

Create a Simulation.

Parameters

world	reference to World Object
parent	QObject this belongs to

7.28.2.2 Langmuir::Simulation::∼Simulation ( ) [virtual]

Destroy the Simulation.

7.28.3 Member Function Documentation

7.28.3.1 void Langmuir::Simulation::balanceCharges ( ) [protected]

Try to use the sources to keep the number of ChargeAgents balanced

7.28.3.2 void Langmuir::Simulation::chargeAgentCoulombInteractionQtConcurrentCPU (ChargeAgent ∗ *chargeAgent*) [inline], [static], [protected]

A method needed to call ChargeAgent::coulombCPU() in parallel.

7.28.3.3 void Langmuir::Simulation::chargeAgentCoulombInteractionQtConcurrentGPU (ChargeAgent ∗ *chargeAgent*) [inline], [static], [protected]

A method needed to call ChargeAgent::coulombGPU() in parallel.

Does not perform GPU calcuations. The coulomb kernel in OpenCLHelper is used to do that. This function copies the GPU results from OpenCLHelper to each ChargeAgent. It is assumed that the coulomb kernel was launched beforehand.

7.28.3.4 void Langmuir::Simulation::nextTick ( ) [protected]

Remove charges from the simulation.

Charges are removed only if a DrainAgent sets their removed status to True. This function will also output carrier statistics if output.id.on.delete is set.

7.28.3.5 void Langmuir::Simulation::performInjections ( ) [protected]

Tell sources to inject charges.

7.28.3.6 void Langmuir::Simulation::performIterations (int *nIterations*) [virtual]

simulate for a set number of steps

Parameters

nIterations	the number of steps to simulate

7.28.3.7 void Langmuir::Simulation::performRecombinations ( ) `[protected]`

Recombine holes and electrons (in solarcell simulations only)

7.28.4 Member Data Documentation

7.28.4.1 World& Langmuir::Simulation::m_world `[protected]`

Reference to World object.

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

- simulation.h
- simulation.cpp

7.29 Langmuir::SimulationParameters Struct Reference

A struct to store all simulation options To add new variables, follow these steps:

`#include <parameters.h>`

Public Member Functions

- SimulationParameters ()

Public Attributes

- QString simulationType

  tells Langmuir how to set up the Sources and Drains: (`"transistor", "solarcell"`*)*
- quint64 randomSeed

  seed the random number generator, if negative, uses the current time (making seperate runs random)
- qint32 gridZ

  the number of sites per layer, at least one
- qint32 gridY

  the number of sites along the device width, at least one
- qint32 gridX

  the number of sites along the device length, at least one
- bool coulombCarriers

  turn on Coulomb interactions between ChargeAgents
- qreal coulombGaussianSigma

  multiply Coulomb terms by erf[r/(sigma sqrt[2])]; nothing happens if its zero
- qint32 defectsCharge

  the charge of defect sites
- qint32 outputXyz

  output trajectory file (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)
- bool outputXyzE

  output electrons in trajectory file (ignored if outputXyz is off)

- bool outputXyzH

  output holes in trajectory file (ignored if outputXyz is off)
- bool outputXyzD

  output defects in trajectory file (ignored if outputXyz is off)
- bool outputXyzT

  output traps in trajectory file (ignored if outputXyz is off)
- qint32 outputXyzMode

  output mode for xyz file (if 0, particle count varies; if 1, particle count is constant using "phantom particles")
- bool outputIdsOnDelete

  output carrier lifetime and pathlength when they are deleted
- qint32 outputCoulomb

  output coulomb energy for the entire grid (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)
- qint32 outputStepChk

  output a checkpoint file every this $*$ iterationsPrint
- bool outputChkTrapPotential

  output trap potentials in checkpoint files
- bool outputPotential

  output grid potential at the start of the simulation, includes the trap potential
- bool outputIsOn

  if false, produce no output (useful for LangmuirView)
- bool imageDefects

  output images of defects
- bool imageTraps

  output images of defects
- qint32 imageCarriers

  output images of carriers (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)
- qint32 iterationsPrint

  if Langmuir, how often to output; if LangmuirView, how many steps between rendering
- qint32 iterationsReal

  number of simulation steps after equilibration
- qint32 outputPrecision

  number of significant figures used for doubles in output
- qint32 outputWidth

  width of columns in output, ignored in certain files, like trajectory files
- QString outputStub

  the stub to use when naming output files
- qreal electronPercentage

  the percent of the grid that is reserved for electrons, between 0 and 1
- qreal holePercentage

  the percent of the grid that is reserved for holes, between 0 and 1
- qreal seedCharges

  if true, place charges randomly before the simulation starts
- qreal defectPercentage

  the percent of the grid that is reserved for electrons, between 0 and 1
- qreal trapPercentage

  the percent of the grid that is reserved for traps, between 0 and 1
- qreal trapPotential

  the potential of traps
- qreal gaussianStdev

  the standard deviation of trap sites

- qreal seedPercentage

    the percent of the traps to be placed and grown upon to form islands
- qreal voltageRight

    the potential on the right side of the grid, used in setting up an electric field
- qreal voltageLeft

    the potential on the left side of the grid, used in setting up an electric field
- qreal excitonBinding

    the energy change (eV) when a hole/electron pair goes from the same site to adjacent sites
- qreal temperatureKelvin

    the temperature used in the boltzmann factor
- qreal sourceRate

    the rate at which all sources inject charges
- qreal eSourceLRate

    the rate at which the left electron source injects charges (overrides default)
- qreal eSourceRRate

    the rate at which the right electron source injects charges (overrides default)
- qreal hSourceLRate

    the rate at which the left hole source injects charges (overrides default)
- qreal hSourceRRate

    the rate at which the right hole source injects charges (overrides default)
- qreal generationRate

    the rate at which the exciton source injects charges (overrides default)
- bool balanceCharges

    if true, try to keep the number of charges in the simulation balanced
- qreal drainRate

    the rate at which all drains accept charges (default, used when eDrainL, etc. are $<$ 0)
- qreal eDrainLRate

    the rate at which the left electron drain accepts charges (overrides default)
- qreal eDrainRRate

    the rate at which the right electron drain accepts charges (overrides default)
- qreal hDrainLRate

    the rate at which the left hole drain accepts charges (overrides default)
- qreal hDrainRRate

    the rate at which the right hole drain accepts charges (overrides default)
- bool useOpenCL

    if true, try to use OpenCL to speed up Coulomb interaction calculations
- qint32 workX

    the x size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb
- qint32 workY

    the y size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb
- qint32 workZ

    the z size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb
- qint32 workSize

    the size of OpenCL 1DRange kernel work groups
- qint32 openclThreshold

    the minimum number of charges that must be present to use OpenCL
- qint32 openclDeviceID

    the device to choose if there are multiple
- qreal boltzmannConstant

    physical constant, the boltzmann constant
- qreal dielectricConstant

physical constant, the dielectic constant

- qreal elementaryCharge

    physical constant, the elementary charge
- qreal permittivitySpace

    physical constant, the permittivity of free spece
- qreal gridFactor

    size constant, the size associated with grid sites (∼1nm)
- qint32 electrostaticCutoff

    the cut off for Coulomb interations
- qreal electrostaticPrefactor

    a compilation of physical constants
- qreal inverseKT

    a compilation of physical constants
- bool okCL

    if true, OpenCL can be used on this platform
- quint32 currentStep

    the current step of the simulation
- QDateTime simulationStart

    the time this simulation started
- qint32 hoppingRange

    the number of sites away from a given site used when calculating neighboring sites
- qreal slopeZ

    slope of potential along z direction when there are multiple layers (as if there were a gate electrode)
- bool sourceMetropolis

    if true, use an energy change (source potential and site) and metropolis criterion to calculate injection probability for hole and electron sources (not exciton sources)
- bool sourceCoulomb

    if true, use the Coulomb + Image interaction when calcualting energy change for injection
- qreal recombinationRate

    the rate at which holes and electrons can combine when they sit upon one another
- qint32 recombinationRange

    the number of sites to consider when performing recombinations (0 means same-site, 1 means one-site away and same-site)
- bool outputIdsOnEncounter

    output carrier lifetime and pathlength when holes and electrons encounter one another
- qreal sourceScaleArea

    for SimulationParameters::simulationType == "solarcell", multiply SimulationParameters::sourceRate by (Grid::xyPlaneArea)/(SimulationParameters::sourceScaleArea); if <= 0, does not scale rate
- qint32 maxThreads

    max threads allowed for QThreadPool - if its <= 0 then the QThread::idealThreadCount is used; note that Qt ignores PBS and SGE so when this isn't set Qt will use all the cores on a node

7.29.1 Detailed Description

A struct to store all simulation options To add new variables, follow these steps:

- declare the new variable in the SimulationParameters struct (parameters.h)
- assign the default value of the new variable in the SimulationParameters constructor (parameters.h)
- implement validity checking for the variable in the checkSimulationParameters() function (parameters.h)
- register the variable in the KeyValueParser constructor using the registerVariable() function (keyvalueparser.h)
- to use non-standard types, you must overload certain template functions in variable.h. See, for example, overloads for QDateTime in variable.h.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 Langmuir::SimulationParameters::SimulationParameters ( ) [inline]

7.29.3 Member Data Documentation

7.29.3.1 bool Langmuir::SimulationParameters::balanceCharges

if true, try to keep the number of charges in the simulation balanced

7.29.3.2 qreal Langmuir::SimulationParameters::boltzmannConstant

physical constant, the boltzmann constant

7.29.3.3 bool Langmuir::SimulationParameters::coulombCarriers

turn on Coulomb interactions between ChargeAgents

7.29.3.4 qreal Langmuir::SimulationParameters::coulombGaussianSigma

multiply Coulomb terms by erf[r/(sigma sqrt[2])]; nothing happens if its zero

7.29.3.5 quint32 Langmuir::SimulationParameters::currentStep

the current step of the simulation

7.29.3.6 qreal Langmuir::SimulationParameters::defectPercentage

the percent of the grid that is reserved for electrons, between 0 and 1

7.29.3.7 qint32 Langmuir::SimulationParameters::defectsCharge

the charge of defect sites

7.29.3.8 qreal Langmuir::SimulationParameters::dielectricConstant

physical constant, the dielectic constant

7.29.3.9 qreal Langmuir::SimulationParameters::drainRate

the rate at which all drains accept charges (default, used when eDrainL, etc. are $<$ 0)

7.29.3.10 qreal Langmuir::SimulationParameters::eDrainLRate

the rate at which the left electron drain accepts charges (overrides default)

7.29.3.11 qreal Langmuir::SimulationParameters::eDrainRRate

the rate at which the right electron drain accepts charges (overrides default)

7.29.3.12 qreal Langmuir::SimulationParameters::electronPercentage

the percent of the grid that is reserved for electrons, between 0 and 1

7.29.3.13 qint32 Langmuir::SimulationParameters::electrostaticCutoff

the cut off for Coulomb interations

7.29.3.14 qreal Langmuir::SimulationParameters::electrostaticPrefactor

a compilation of physical constants

7.29.3.15 qreal Langmuir::SimulationParameters::elementaryCharge

physical constant, the elementary charge

7.29.3.16 qreal Langmuir::SimulationParameters::eSourceLRate

the rate at which the left electron source injects charges (overrides default)

7.29.3.17 qreal Langmuir::SimulationParameters::eSourceRRate

the rate at which the right electron source injects charges (overrides default)

7.29.3.18 qreal Langmuir::SimulationParameters::excitonBinding

the energy change (eV) when a hole/electron pair goes from the same site to adjacent sites

7.29.3.19 qreal Langmuir::SimulationParameters::gaussianStdev

the standard deviation of trap sites

7.29.3.20 qreal Langmuir::SimulationParameters::generationRate

the rate at which the exciton source injects charges (overrides default)

7.29.3.21 qreal Langmuir::SimulationParameters::gridFactor

size constant, the size associated with grid sites ($\sim$1nm)

7.29.3.22 qint32 Langmuir::SimulationParameters::gridX

the number of sites along the device length, at least one

7.29.3.23 qint32 Langmuir::SimulationParameters::gridY

the number of sites along the device width, at least one

7.29.3.24 qint32 Langmuir::SimulationParameters::gridZ

the number of sites per layer, at least one

7.29.3.25 qreal Langmuir::SimulationParameters::hDrainLRate

the rate at which the left hole drain accepts charges (overrides default)

7.29.3.26 qreal Langmuir::SimulationParameters::hDrainRRate

the rate at which the right hole drain accepts charges (overrides default)

7.29.3.27 qreal Langmuir::SimulationParameters::holePercentage

the percent of the grid that is reserved for holes, between 0 and 1

7.29.3.28 qint32 Langmuir::SimulationParameters::hoppingRange

the number of sites away from a given site used when calculating neighboring sites

7.29.3.29 qreal Langmuir::SimulationParameters::hSourceLRate

the rate at which the left hole source injects charges (overrides default)

7.29.3.30 qreal Langmuir::SimulationParameters::hSourceRRate

the rate at which the right hole source injects charges (overrides default)

7.29.3.31 qint32 Langmuir::SimulationParameters::imageCarriers

output images of carriers (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)

7.29.3.32 bool Langmuir::SimulationParameters::imageDefects

output images of defects

7.29.3.33 bool Langmuir::SimulationParameters::imageTraps

output images of defects

7.29.3.34 qreal Langmuir::SimulationParameters::inverseKT

a compilation of physical constants

7.29.3.35 qint32 Langmuir::SimulationParameters::iterationsPrint

if Langmuir, how often to output; if LangmuirView, how many steps between rendering

7.29.3.36 qint32 Langmuir::SimulationParameters::iterationsReal

number of simulation steps after equilibration

7.29.3.37 qint32 Langmuir::SimulationParameters::maxThreads

max threads allowed for QThreadPool - if its $<=$ 0 then the QThread::idealThreadCount is used; note that Qt ignores PBS and SGE so when this isn't set Qt will use all the cores on a node

7.29.3.38 bool Langmuir::SimulationParameters::okCL

if true, OpenCL can be used on this platform

7.29.3.39 qint32 Langmuir::SimulationParameters::openclDeviceID

the device to choose if there are multiple

7.29.3.40 qint32 Langmuir::SimulationParameters::openclThreshold

the minimum number of charges that must be present to use OpenCL

7.29.3.41 bool Langmuir::SimulationParameters::outputChkTrapPotential

output trap potentials in checkpoint files

7.29.3.42 qint32 Langmuir::SimulationParameters::outputCoulomb

output coulomb energy for the entire grid (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)

7.29.3.43 bool Langmuir::SimulationParameters::outputIdsOnDelete

output carrier lifetime and pathlength when they are deleted

7.29.3.44 bool Langmuir::SimulationParameters::outputIdsOnEncounter

output carrier lifetime and pathlength when holes and electrons encounter one another

7.29.3.45 bool Langmuir::SimulationParameters::outputIsOn

if false, produce no output (useful for LangmuirView)

7.29.3.46 bool Langmuir::SimulationParameters::outputPotential

output grid potential at the start of the simulation, includes the trap potential

7.29.3.47 qint32 Langmuir::SimulationParameters::outputPrecision

number of significant figures used for doubles in output

7.29.3.48 qint32 Langmuir::SimulationParameters::outputStepChk

output a checkpoint file every this $*$ iterationsPrint

7.29.3.49 QString Langmuir::SimulationParameters::outputStub

the stub to use when naming output files

7.29.3.50 qint32 Langmuir::SimulationParameters::outputWidth

width of columns in output, ignored in certain files, like trajectory files

7.29.3.51 qint32 Langmuir::SimulationParameters::outputXyz

output trajectory file (if n $<$ 0, only at the end; if n == 0, never; if n $>$ 0, every n $*$ iterations.print steps)

7.29.3.52 bool Langmuir::SimulationParameters::outputXyzD

output defects in trajectory file (ignored if outputXyz is off)

7.29.3.53 bool Langmuir::SimulationParameters::outputXyzE

output electrons in trajectory file (ignored if outputXyz is off)

7.29.3.54 bool Langmuir::SimulationParameters::outputXyzH

output holes in trajectory file (ignored if outputXyz is off)

7.29.3.55 qint32 Langmuir::SimulationParameters::outputXyzMode

output mode for xyz file (if 0, particle count varies; if 1, particle count is constant using "phantom particles")

7.29.3.56 bool Langmuir::SimulationParameters::outputXyzT

output traps in trajectory file (ignored if outputXyz is off)

7.29.3.57 qreal Langmuir::SimulationParameters::permittivitySpace

physical constant, the permittivity of free spece

7.29.3.58 quint64 Langmuir::SimulationParameters::randomSeed

seed the random number generator, if negative, uses the current time (making seperate runs random)

7.29.3.59 qint32 Langmuir::SimulationParameters::recombinationRange

the number of sites to consider when performing recombinations (0 means same-site, 1 means one-site away and same-site)

7.29.3.60 qreal Langmuir::SimulationParameters::recombinationRate

the rate at which holes and electrons can combine when they sit upon one another

7.29.3.61 qreal Langmuir::SimulationParameters::seedCharges

if true, place charges randomly before the simulation starts

7.29.3.62 qreal Langmuir::SimulationParameters::seedPercentage

the percent of the traps to be placed and grown upon to form islands

7.29.3.63 QDateTime Langmuir::SimulationParameters::simulationStart

the time this simulation started

7.29.3.64 QString Langmuir::SimulationParameters::simulationType

tells Langmuir how to set up the Sources and Drains: (`"transistor"`, `"solarcell"`)

7.29.3.65 qreal Langmuir::SimulationParameters::slopeZ

slope of potential along z direction when there are multiple layers (as if there were a gate electrode)

7.29.3.66 bool Langmuir::SimulationParameters::sourceCoulomb

if true, use the Coulomb + Image interaction when calcualting energy change for injection

7.29.3.67 bool Langmuir::SimulationParameters::sourceMetropolis

if true, use an energy change (source potential and site) and metropolis criterion to calculate injection probability for hole and electron sources (not exciton sources)

7.29.3.68 qreal Langmuir::SimulationParameters::sourceRate

the rate at which all sources inject charges

7.29.3.69 qreal Langmuir::SimulationParameters::sourceScaleArea

for SimulationParameters::simulationType == "solarcell", multiply SimulationParameters::sourceRate by (Grid::xyPlaneArea)/(SimulationParameters::sourceScaleArea); if <= 0, does not scale rate

7.29.3.70 qreal Langmuir::SimulationParameters::temperatureKelvin

the temperature used in the boltzmann factor

7.29.3.71 qreal Langmuir::SimulationParameters::trapPercentage

the percent of the grid that is reserved for traps, between 0 and 1

7.29.3.72 qreal Langmuir::SimulationParameters::trapPotential

the potential of traps

7.29.3.73 bool Langmuir::SimulationParameters::useOpenCL

if true, try to use OpenCL to speed up Coulomb interaction calculations

7.29.3.74 qreal Langmuir::SimulationParameters::voltageLeft

the potential on the left side of the grid, used in setting up an electric field

7.29.3.75 qreal Langmuir::SimulationParameters::voltageRight

the potential on the right side of the grid, used in setting up an electric field

7.29.3.76 qint32 Langmuir::SimulationParameters::workSize

the size of OpenCL 1DRange kernel work groups

7.29.3.77 qint32 Langmuir::SimulationParameters::workX

the x size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb

7.29.3.78 qint32 Langmuir::SimulationParameters::workY

the y size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb

7.29.3.79 qint32 Langmuir::SimulationParameters::workZ

the z size of OpenCL 3DRange kernel work groups - only needed if using SimulationParameters::outputCoulomb

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

- parameters.h

7.30 Langmuir::SourceAgent Class Reference

A class to inject charges.

```
#include <sourceagent.h>
```

Public Member Functions

- SourceAgent (World &world, Grid &grid, QObject ∗parent=0)

  create a SourceAgent
- bool tryToSeed ()

  seed a charge at a random site
- bool tryToSeed (int site)

  *seed a charge at a **specific** site*

- bool tryToInject ()

  attempt to inject a carrier

Protected Member Functions

- virtual int chooseSite ()

  choose a site to inject to
- virtual bool validToInject (int site)=0

  checks to see if a carrier can actually be injected at the requested site
- virtual void inject (int site)=0

  actually injects carrier.
- virtual bool shouldTransport (int site)

  decides if charge should be injected using a constant probability
- int randomSiteID ()

  choose a random site ID
- int randomNeighborSiteID ()

  choose a random site ID from the neighborlist.

Additional Inherited Members

7.30.1 Detailed Description

A class to inject charges.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 Langmuir::SourceAgent::SourceAgent (World & *world*, Grid & *grid*, QObject ∗ *parent* = `0`)

create a SourceAgent

7.30.3 Member Function Documentation

7.30.3.1 int Langmuir::SourceAgent::chooseSite ( ) `[protected]`, `[virtual]`

choose a site to inject to

By default, choose a site from the SourceAgent's neighborlist. The Grid::CubeFace used to construct the SourceAgent determines the neighborlist.

Reimplemented in Langmuir::ExcitonSourceAgent.

7.30.3.2 virtual void Langmuir::SourceAgent::inject (int *site*) `[protected]`, `[pure virtual]`

actually injects carrier.

Warning

> this function assumes that injecting a charge at the requested site is allowed

Creates a new carrier. Does not perform checks. Forcefully injects charge. Don't call this function unless you know what you are doing.

Implemented in Langmuir::ExcitonSourceAgent, Langmuir::HoleSourceAgent, and Langmuir::ElectronSourceAgent.

7.30.3.3 int Langmuir::SourceAgent::randomNeighborSiteID ( ) [protected]

choose a random site ID from the neighborlist.

7.30.3.4 int Langmuir::SourceAgent::randomSiteID ( ) [protected]

choose a random site ID

It can be any possible site in the grid.

7.30.3.5 bool Langmuir::SourceAgent::shouldTransport (int *site*) [protected], [virtual]

decides if charge should be injected using a constant probability

Parameters

site	the site involved

If SimulationParameters::sourceMetropolis is true, then use the metropolis criterion with an energy change to decide if charge should be injected.

Reimplemented from Langmuir::FluxAgent.

Reimplemented in Langmuir::ExcitonSourceAgent.

7.30.3.6 bool Langmuir::SourceAgent::tryToInject ( )

attempt to inject a carrier

This is the main transport method of a SourceAgent. This function uses chooseSite(), shouldTransport() and validToInject() to inject the charge. It is not garunteed that a charge will be injected.

7.30.3.7 bool Langmuir::SourceAgent::tryToSeed ( )

seed a charge at a random site

Warning

does not call shouldTransport()

Attempts to seed a charge at a random site, without calling shouldTransport(). However, validToInject() is still called. This function is used when randomly placing charges in the system.

7.30.3.8 bool Langmuir::SourceAgent::tryToSeed (int *site*)

seed a charge at a **specific** site

Warning

does not call shouldTransport()

Attempts to seed a charge at a specific site, without calling shouldTransport(). However, validToInject() is still called. This function is used when placing charges at specific places. For example, when sometimes the checkpoint file has information on where charges are/were, and these need to be placed.

7.30.3.9 virtual bool Langmuir::SourceAgent::validToInject (int *site*) [protected], [pure virtual]

checks to see if a carrier can actually be injected at the requested site

For example, if the site contains a defect, or a carrier is already present at the site, then it is not valid to inject the carrier at this site.

Implemented in Langmuir::ExcitonSourceAgent, Langmuir::HoleSourceAgent, and Langmuir::ElectronSourceAgent.

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

- sourceagent.h
- sourceagent.cpp

7.31 Langmuir::Tolerance Class Reference

A class to check if the simulation is converging.

```
#include <tolerance.h>
```

Public Member Functions

- Tolerance (World &world, FluxAgent ∗flux, double criteria=0.01, QObject ∗parent=0)

  Create the tolerance checking object.
- void check ()

  check if the convergence criteria has been met
- bool converged ()

  true if the convergence criteria has been met the appropriate number of times

Private Member Functions

- void checkCritera ()

  make sure the criteria is a valid percent
- void setPrevious ()

  set the previous success/step ratio to be whatever flux currently reports
- void setCurrent ()

  set the current success/step ratio to whatever flux currently reports
- double percentChange ()

  calculate the percent change between previous and current flux success/step ratios

Private Attributes

- double m_current

  current flux success/step ratio
- double m_previous

  previous flux success/step ratio
- double m_criteria

  the maximum percent difference between previous and current for the criteria to be met
- int m_converged

  the number of consecutive times the flux agent has met the convergence criteria
- FluxAgent ∗ m_flux

  the flux agent being tracked

- World & m_world

  reference to the world object

7.31.1 Detailed Description

A class to check if the simulation is converging.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 Langmuir::Tolerance::Tolerance (World & *world,* FluxAgent ∗ *flux,* double *criteria =* `0.01`, QObject ∗ *parent =* `0`) `[explicit]`

Create the tolerance checking object.

Parameters

reference	to the world object
a	flux agent to track
the	criterion for convergence
parent	the parent of this QObject

7.31.3 Member Function Documentation

7.31.3.1 void Langmuir::Tolerance::check ( )

check if the convergence criteria has been met

7.31.3.2 void Langmuir::Tolerance::checkCritera ( ) `[private]`

make sure the criteria is a valid percent

7.31.3.3 bool Langmuir::Tolerance::converged ( )

true if the convergence criteria has been met the appropriate number of times

7.31.3.4 double Langmuir::Tolerance::percentChange ( ) `[private]`

calculate the percent change between previous and current flux success/step ratios

7.31.3.5 void Langmuir::Tolerance::setCurrent ( ) `[private]`

set the current success/step ratio to whatever flux currently reports

7.31.3.6 void Langmuir::Tolerance::setPrevious ( ) `[private]`

set the previous success/step ratio to be whatever flux currently reports

7.31.4 Member Data Documentation

7.31.4.1 int Langmuir::Tolerance::m_converged [private]

the number of consecutive times the flux agent has met the convergence criteria

7.31.4.2 double Langmuir::Tolerance::m_criteria [private]

the maximum percent difference between previous and current for the criteria to be met

7.31.4.3 double Langmuir::Tolerance::m_current [private]

current flux success/step ratio

7.31.4.4 FluxAgent* Langmuir::Tolerance::m_flux [private]

the flux agent being tracked

7.31.4.5 double Langmuir::Tolerance::m_previous [private]

previous flux success/step ratio

7.31.4.6 World& Langmuir::Tolerance::m_world [private]

reference to the world object

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

- tolerance.h
- tolerance.cpp

7.32 Langmuir::TypedVariable< T > Class Template Reference

A template class to map between variable names (keys) and locations (references)

`#include <variable.h>`

Public Member Functions

- TypedVariable (const QString &key, T &value, VariableMode mode=0, QObject *parent=0)

  Create a new variable.
- virtual void read (const QString &token)

  Cast the value stored in string to the correct type and store it in the correct location.
- virtual QString key () const

  Get this variable's key (name)
- virtual QString value () const

  Get this variable's value as a QString.
- virtual QString keyValue () const

  Get this variable's key and value in the form 'key = value'.
- template<>
  QString value () const

  Get this variable's value as a QString.

- template<>
  QString value () const

  Get this variable's value as a QString.
- template<>
  QString value () const

  Get this variable's value as a QString.
- template<>
  QString keyValue () const

  Get this variable's key and value in the form 'key = value'.
- template<>
  void convert (const QString &token, QString &result)
- template<>
  void convert (const QString &token, qreal &result)
- template<>
  void convert (const QString &token, float &result)
- template<>
  void convert (const QString &token, bool &result)
- template<>
  void convert (const QString &token, qint32 &result)
- template<>
  void convert (const QString &token, quint32 &result)
- template<>
  void convert (const QString &token, qint64 &result)
- template<>
  void convert (const QString &token, quint64 &result)
- template<>
  void convert (const QString &token, QDateTime &result)

Static Public Member Functions

- static void convert (const QString &token, T &result)

  A template function for converting a QString to some type T.

Protected Member Functions

- virtual void write (QTextStream &stream) const

  Write 'key = value' to a stream.

Protected Attributes

- T & m_value

  Reference to the object being tracked.

Additional Inherited Members

7.32.1 Detailed Description

template<class T>class Langmuir::TypedVariable< T >

A template class to map between variable names (keys) and locations (references)

7.32.2 Constructor & Destructor Documentation

7.32.2.1 template<class T > Langmuir::TypedVariable< T >::TypedVariable (const QString & *key*, T & *value*, VariableMode *mode* = 0, QObject ∗ *parent* = 0) [inline]

Create a new variable.

initialize Variable with a key and a location

Parameters

key	the name of the variable
value	the location where the value of the variable is stored
mode	flags to alter variable's behavoir, see AbstractVariable::VariableModeFlag
parent	QObject this belongs to

7.32.3 Member Function Documentation

7.32.3.1 template<class T > static void Langmuir::TypedVariable< T >::convert (const QString & *token*, T & *result*) [static]

A template function for converting a QString to some type T.

To implement this function for a new data type, use declarations of the form:

- template <> inline void Variable<T>::convert(const QString& token, T& result)
- replace the **'T'** with the data type you want to implement (for example, double).

7.32.3.2 template<> void Langmuir::TypedVariable< QString >::convert (const QString & *token*, QString & *result*) [inline]

7.32.3.3 template<> void Langmuir::TypedVariable< qreal >::convert (const QString & *token*, qreal & *result*) [inline]

7.32.3.4 template<> void Langmuir::TypedVariable< float >::convert (const QString & *token*, float & *result*) [inline]

7.32.3.5 template<> void Langmuir::TypedVariable< bool >::convert (const QString & *token*, bool & *result*) [inline]

7.32.3.6 template<> void Langmuir::TypedVariable< qint32 >::convert (const QString & *token*, qint32 & *result*) [inline]

7.32.3.7 template<> void Langmuir::TypedVariable< quint32 >::convert (const QString & *token*, quint32 & *result*) [inline]

7.32.3.8 template<> void Langmuir::TypedVariable< qint64 >::convert (const QString & *token*, qint64 & *result*) [inline]

7.32.3.9 template<> void Langmuir::TypedVariable< quint64 >::convert (const QString & *token*, quint64 & *result*) [inline]

7.32.3.10 template<> void Langmuir::TypedVariable< QDateTime >::convert (const QString & *token*, QDateTime & *result*) [inline]

7.32.3.11 template< class T > QString Langmuir::TypedVariable< T >::key ( ) const [inline], [virtual]

Get this variable's key (name)

get the variable's key

Implements Langmuir::Variable.

7.32.3.12 template< class T > QString Langmuir::TypedVariable< T >::keyValue ( ) const [inline], [virtual]

Get this variable's key and value in the form 'key = value'.

get the variable's key

Implements Langmuir::Variable.

7.32.3.13 template<> QString Langmuir::TypedVariable< QString >::keyValue ( ) const [inline], [virtual]

Get this variable's key and value in the form 'key = value'.

Implements Langmuir::Variable.

7.32.3.14 template< class T > void Langmuir::TypedVariable< T >::read (const QString & *token*) [inline], [virtual]

Cast the value stored in string to the correct type and store it in the correct location.

convert a QString token to its correct type

Implements Langmuir::Variable.

7.32.3.15 template< class T > QString Langmuir::TypedVariable< T >::value ( ) const [inline], [virtual]

Get this variable's value as a QString.

get the variable's value (converted to string)

Implements Langmuir::Variable.

7.32.3.16 template<> QString Langmuir::TypedVariable< float >::value ( ) const [inline], [virtual]

Get this variable's value as a QString.

Implements Langmuir::Variable.

7.32.3.17 template<> QString Langmuir::TypedVariable< qreal >::value ( ) const [inline], [virtual]

Get this variable's value as a QString.

Implements Langmuir::Variable.

7.32.3.18 template<> QString Langmuir::TypedVariable< bool >::value ( ) const [inline], [virtual]

Get this variable's value as a QString.

Implements Langmuir::Variable.

7.32.3.19 template<class T > void Langmuir::TypedVariable< T >::write (QTextStream & *stream*) const [inline], [protected], [virtual]

Write 'key = value' to a stream.

write keyValue() to a stream

Implements Langmuir::Variable.

7.32.4 Member Data Documentation

7.32.4.1 template<class T > T& Langmuir::TypedVariable< T >::m_value [protected]

Reference to the object being tracked.

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

- variable.h

7.33 Langmuir::Variable Class Reference

A class to map between variable names (keys) and locations (references)

```
#include <variable.h>
```

Public Types

- enum VariableModeFlag { Constant = 1 }

  A Flag to alter the behavoir of certain variable member functions.

Public Member Functions

- Variable (const QString &key, VariableMode mode=0, QObject ∗parent=0)

  Create a Variable, see Variable::Variable for description.
- virtual void read (const QString &token)=0

  Cast the value stored in string to the correct type and store it in the correct location.
- virtual QString key () const =0

  Get this variable's key (name)
- virtual QString value () const =0

  Get this variable's value as a QString.
- virtual QString keyValue () const =0

  Get this variable's key and value in the form 'key = value'.
- bool isConstant () const

  True if the Variable::Constant mode flag was set.
- const VariableMode & mode () const

  Get this variable's mode flags.

Protected Member Functions

- virtual void write (QTextStream &stream) const =0

  Write 'key = value' to a stream.

Protected Attributes

- QString m_key

  The name of this variable.
- VariableMode m_mode

  The mode flags for this variable.

Friends

- QTextStream & operator<< (QTextStream &stream, const Variable &variable)

  Operator overload to output 'key = value' to QTextStream.
- QDebug operator<< (QDebug dbg, const Variable &variable)

  Operator overload to output 'key = value' to QDebug.
- std::ostream & operator<< (std::ostream &stream, Variable &variable)

  Operator overload to output to output 'key = value' to std::ofstream.

7.33.1 Detailed Description

A class to map between variable names (keys) and locations (references)

7.33.2 Member Enumeration Documentation

7.33.2.1 enum **Langmuir::Variable::VariableModeFlag**

A Flag to alter the behavoir of certain variable member functions.

Enumerator:

Constant When constant, a variable's read / convert function does nothing.

7.33.3 Constructor & Destructor Documentation

7.33.3.1 Langmuir::Variable::Variable (const QString & *key*, VariableMode *mode* = 0, QObject * *parent* = 0) [inline]

Create a Variable, see Variable::Variable for description.

initialize a Variable with a key

7.33.4 Member Function Documentation

7.33.4.1 bool Langmuir::Variable::isConstant ( ) const [inline]

True if the Variable::Constant mode flag was set.

see if the Variable is really a Constant

7.33.4.2 virtual QString Langmuir::Variable::key ( ) const [pure virtual]

Get this variable's key (name)

Implemented in Langmuir::TypedVariable< T >.

7.33.4.3 virtual QString Langmuir::Variable::keyValue ( ) const [pure virtual]

Get this variable's key and value in the form 'key = value'.

Implemented in Langmuir::TypedVariable< T >, and Langmuir::TypedVariable< T >.

7.33.4.4 const Variable::VariableMode & Langmuir::Variable::mode ( ) const [inline]

Get this variable's mode flags.

get the mode flags of this Variable

7.33.4.5 virtual void Langmuir::Variable::read (const QString & *token*) [pure virtual]

Cast the value stored in string to the correct type and store it in the correct location.

This function assumes **'QTextStream&** operator<<' has been implemented for that data type T. Keep this in mind if adding a new data type.

Implemented in Langmuir::TypedVariable< T >.

7.33.4.6 virtual QString Langmuir::Variable::value ( ) const [pure virtual]

Get this variable's value as a QString.

Implemented in Langmuir::TypedVariable< T >, Langmuir::TypedVariable< T >, Langmuir::TypedVariable< T >, and Langmuir::TypedVariable< T >.

7.33.4.7 virtual void Langmuir::Variable::write (QTextStream & *stream*) const [protected], [pure virtual]

Write 'key = value' to a stream.

Implemented in Langmuir::TypedVariable< T >.

7.33.5 Friends And Related Function Documentation

7.33.5.1 QTextStream& operator<< (QTextStream & *stream,* const Variable & *variable*) [friend]

Operator overload to output 'key = value' to QTextStream.

7.33.5.2 QDebug operator<< (QDebug *dbg,* const Variable & *variable*) [friend]

Operator overload to output 'key = value' to QDebug.

7.33.5.3 std::ostream& operator<< (std::ostream & *stream,* Variable & *variable*) [friend]

Operator overload to output to output 'key = value' to std::ofstream.

7.33.6 Member Data Documentation

7.33.6.1 QString Langmuir::Variable::m_key [protected]

The name of this variable.

7.33.6.2 VariableMode Langmuir::Variable::m_mode [protected]

The mode flags for this variable.

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

- variable.h

7.34 Langmuir::World Class Reference

A class to hold all objects in a simulation.

`#include <world.h>`

Public Member Functions

- World (const QString &fileName, QObject ∗parent=0)

  create a world to simulate in
- ∼World ()

  destroys the entire World, and everything in it...including you.
- KeyValueParser & keyValueParser ()

  get the KeyValueParser, used for parsing input files.
- CheckPointer & checkPointer ()

  get the CheckPointer, used for reading and writing input files.
- Grid & electronGrid ()

  get the Grid used, used for holding ElectronAgents
- Grid & holeGrid ()

  get the hole Grid, used for holding HoleAgents
- Potential & potential ()

  get the Potential, a calculator used for...calculating the potential.
- SimulationParameters & parameters ()

  get the SimulationParameters, a struct used for holding simulation parameters.
- Random & randomNumberGenerator ()

  get the Random, used for creating random numbers
- Logger & logger ()

  get the Logger, used for writing output
- OpenClHelper & opencl ()

  get the OpenClHelper, used for calculating Coulomb interactions with a Graphics Card
- QList< SourceAgent ∗ > & sources ()

  get a list of all SourceAgents
- QList< SourceAgent ∗ > & eSources ()

  get a list of all ElectronSourceAgents
- QList< SourceAgent ∗ > & hSources ()

  get a list of all ElectronSourceAgents
- QList< SourceAgent ∗ > & xSources ()

  get a list of all ElectronSourceAgents
- QList< DrainAgent ∗ > & drains ()

  get a list of all DrainAgents
- QList< DrainAgent ∗ > & eDrains ()

  get a list of all ElectronSourceAgents
- QList< DrainAgent ∗ > & hDrains ()

get a list of all ElectronSourceAgents

- QList< DrainAgent ∗ > & xDrains ()

  get a list of all ElectronSourceAgents
- QList< FluxAgent ∗ > & fluxes ()

  get a list of all FluxAgents
- ElectronSourceAgent & electronSourceAgentRight ()

  get the right ElectronSourceAgent
- ElectronSourceAgent & electronSourceAgentLeft ()

  get the left ElectronSourceAgent
- HoleSourceAgent & holeSourceAgentRight ()

  get the right HoleSourceAgent
- HoleSourceAgent & holeSourceAgentLeft ()

  get the left HoleSourceAgent
- ExcitonSourceAgent & excitonSourceAgent ()

  get the RecombinationAgent
- ElectronDrainAgent & electronDrainAgentRight ()

  get the right ElectronDrainAgent
- ElectronDrainAgent & electronDrainAgentLeft ()

  get the left ElectronDrainAgent
- HoleDrainAgent & holeDrainAgentRight ()

  get the right HoleDrainAgent
- HoleDrainAgent & holeDrainAgentLeft ()

  get the left HoleDrainAgent
- RecombinationAgent & recombinationAgent ()

  get the RecombinationAgent
- QList< ChargeAgent ∗ > & electrons ()

  get a list of all ElectronAgents
- QList< ChargeAgent ∗ > & holes ()

  get a list of all HoleAgents
- QList< int > & defectSiteIDs ()

  get a list of all defect sites
- QList< int > & trapSiteIDs ()

  get a list of all trap sites
- QList< double > & trapSitePotentials ()

  get a list of all trap potentials
- boost::multi_array< double, 3 > & R1 ()

  get the array of precomputed r-squared values
- boost::multi_array< double, 3 > & R2 ()

  get the array of precomputed r values
- boost::multi_array< double, 3 > & iR ()

  get the array of precomputed inverse-r values
- boost::multi_array< double, 3 > & eR ()

  get the array of precomputed erf(r/(sqrt(2)∗sigma))
- boost::multi_array< double, 3 > & sI ()

  get the self interactions
- boost::multi_array< double, 3 > & couplingConstants ()

  get the coupling constants
- int maxElectronAgents ()

  get the max number of ElectronAgents allowed
- int maxHoleAgents ()

  get the max number of HoleAgents allowed

- int maxChargeAgents ()

  get the max number of ChargeAgents allowed
- int maxChargeAgentsAndChargedDefects ()

  get the max number of ChargeAgents & charged defects
- int maxDefects ()

  get the max number of Defects allowed
- int maxTraps ()

  get the max number of Traps allowed
- int numElectronAgents ()

  get the current number of ElectronAgents
- int numHoleAgents ()

  get the current number of HoleAgents
- int numChargeAgents ()

  get the current number of ChargeAgents
- int electronsMinusHoles ()

  The number of electrons - holes.
- int holesMinusElectrons ()

  The number of holes - electrons.
- bool chargesAreBalanced ()

  true when electrons and holes are balanced
- int numChargeAgentsAndChargedDefects ()

  get the current number of ChargeAgents & charged defects
- int numDefects ()

  get the current number of Defects
- int numTraps ()

  get the current number of Traps
- double reachedChargeAgents ()

  get the percent of ChargeAgents reached, of the max allowed
- double reachedElectronAgents ()

  get the percent of ElectronAgents reached, of the max allowed
- double reachedHoleAgents ()

  get the percent of HoleAgents reached, of the max allowed
- double percentHoleAgents ()

  get the percent of HoleAgents reached, of the total grid volume
- double percentElectronAgents ()

  get the percent of ElectronAgents reached, of the total grid volume
- bool atMaxElectrons ()

  check if the maximum number of electrons has been reached
- bool atMaxHoles ()

  check if the maximum number of holes has been reached
- bool atMaxCharges ()

  check if the maximum number of charges has been reached

Private Member Functions

- void placeDefects (const QList< int > &siteIDs=QList< int >())

  places defects
- void placeElectrons (const QList< int > &siteIDs=QList< int >())

  places electrons
- void placeHoles (const QList< int > &siteIDs=QList< int >())

places holes

- void createSources ()

  create SourceAgents
- void createDrains ()

  create DrainAgents
- void setFluxInfo (const QList< quint64 > &fluxInfo)

  set attempts / successes for sources / drains
- void initialize (const QString &fileName="")

  initialize all objects

Private Attributes

- KeyValueParser ∗ m_keyValueParser

  pointer to KeyValueParser, used for parsing key=value pairs
- CheckPointer ∗ m_checkPointer

  pointer to CheckPointer, used for reading/writing input(checkpoint) files
- QList< SourceAgent ∗ > m_sources

  list of SourceAgents
- QList< SourceAgent ∗ > m_eSources

  list of ElectronSourceAgents
- QList< SourceAgent ∗ > m_hSources

  list of HoleSourceAgents
- QList< SourceAgent ∗ > m_xSources

  list of ExcitonSourceAgents
- QList< DrainAgent ∗ > m_drains

  list of DrainAgents
- QList< DrainAgent ∗ > m_eDrains

  list of ElectronSourceAgents
- QList< DrainAgent ∗ > m_hDrains

  list of HoleSourceAgents
- QList< DrainAgent ∗ > m_xDrains

  list of ExcitonSourceAgents
- QList< FluxAgent ∗ > m_fluxAgents

  list of all FluxAgents, such as SoureAgents, DrainAgents, etc.
- ElectronSourceAgent ∗ m_electronSourceAgentRight

  pointer to right ElectronDrainAgent
- ElectronSourceAgent ∗ m_electronSourceAgentLeft

  pointer to left ElectronDrainAgent
- HoleSourceAgent ∗ m_holeSourceAgentRight

  pointer to right HoleDrainAgent
- HoleSourceAgent ∗ m_holeSourceAgentLeft

  pointer to left HoleDrainAgent
- ExcitonSourceAgent ∗ m_excitonSourceAgent

  pointer to ExcitonSourceAgent, used for injecting Excitons
- ElectronDrainAgent ∗ m_electronDrainAgentRight

  pointer to right ElectronDrainAgent
- ElectronDrainAgent ∗ m_electronDrainAgentLeft

  pointer to left ElectronDrainAgent
- HoleDrainAgent ∗ m_holeDrainAgentRight

  pointer to right HoleDrainAgent

- HoleDrainAgent ∗ m_holeDrainAgentLeft

    pointer to left HoleDrainAgent

- RecombinationAgent ∗ m_recombinationAgent

    pointer to electron/hole RecombinationAgent, used for removing Excitons

- Grid ∗ m_electronGrid

    pointer to electron Grid, used for keeping track of ElectronAgents

- Grid ∗ m_holeGrid

    pointer to hole Grid, used for keeping track of HoleAgents

- Random ∗ m_rand

    pointer to Random, used for generating random numbers

- Potential ∗ m_potential

    pointer to Potential, used for calculating the potential

- SimulationParameters ∗ m_parameters

    pointer to SimulationParameters

- Logger ∗ m_logger

    pointer to Logger, used for output

- OpenClHelper ∗ m_ocl

    pointer to OpenClHelper, used for Graphics Card calculations

- QList< ChargeAgent ∗ > m_electrons

    list of electrons

- QList< ChargeAgent ∗ > m_holes

    list of holes

- QList< int > m_defectSiteIDs

    list of defect sites

- QList< int > m_trapSiteIDs

    list of trap sites

- QList< double > m_trapSitePotentials

    list of trap potentials

- boost::multi_array< double, 3 > m_R2

    array of precomputed r-squared values

- boost::multi_array< double, 3 > m_R1

    array of precomputed r values

- boost::multi_array< double, 3 > m_iR

    array of precomputed inverse-r values

- boost::multi_array< double, 3 > m_eR

    array of precomputed erf(r/(s∗sqrt(2)) values

- boost::multi_array< double, 3 > m_sI

    self interaction, which is 1/(4 pi e e0 r), with r=1 grid unit When a charge at it's future site interacts with other charges at their current site, the charge will interact with it's own current site. So, this value needs to be subtracted off.

- boost::multi_array< double, 3 > m_couplingConstants

    array of coupling constants

- int m_maxElectrons

    max number of electrons

- int m_maxHoles

    max number of holes

- int m_maxDefects

    max number of defects

- int m_maxTraps

    max number of traps

7.34.1 Detailed Description

A class to hold all objects in a simulation.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 Langmuir::World::World (const QString & *fileName*, QObject ∗ *parent* = 0)

create a world to simulate in

Parameters

fileName	the input file name
parent	QObject this belongs to

Calls the initialize() function.

7.34.2.2 Langmuir::World::∼World ( )

destroys the entire World, and everything in it...including you.

7.34.3 Member Function Documentation

7.34.3.1 bool Langmuir::World::atMaxCharges ( )

check if the maximum number of charges has been reached

7.34.3.2 bool Langmuir::World::atMaxElectrons ( )

check if the maximum number of electrons has been reached

7.34.3.3 bool Langmuir::World::atMaxHoles ( )

check if the maximum number of holes has been reached

7.34.3.4 bool Langmuir::World::chargesAreBalanced ( )

true when electrons and holes are balanced

7.34.3.5 CheckPointer & Langmuir::World::checkPointer ( )

get the CheckPointer, used for reading and writing input files.

7.34.3.6 boost::multi_array< double, 3 > & Langmuir::World::couplingConstants ( )

get the coupling constants

7.34.3.7 void Langmuir::World::createDrains ( ) [private]

create DrainAgents

7.34.3.8 void Langmuir::World::createSources ( ) [private]

create SourceAgents

7.34.3.9 QList< int > & Langmuir::World::defectSiteIDs ( )

get a list of all defect sites

7.34.3.10 QList< DrainAgent * > & Langmuir::World::drains ( )

get a list of all DrainAgents

7.34.3.11 QList< DrainAgent * > & Langmuir::World::eDrains ( )

get a list of all ElectronSourceAgents

7.34.3.12 ElectronDrainAgent & Langmuir::World::electronDrainAgentLeft ( )

get the left ElectronDrainAgent

7.34.3.13 ElectronDrainAgent & Langmuir::World::electronDrainAgentRight ( )

get the right ElectronDrainAgent

7.34.3.14 Grid & Langmuir::World::electronGrid ( )

get the Grid used, used for holding ElectronAgents

7.34.3.15 QList< ChargeAgent * > & Langmuir::World::electrons ( )

get a list of all ElectronAgents

7.34.3.16 int Langmuir::World::electronsMinusHoles ( )

The number of electrons - holes.

7.34.3.17 ElectronSourceAgent & Langmuir::World::electronSourceAgentLeft ( )

get the left ElectronSourceAgent

7.34.3.18 ElectronSourceAgent & Langmuir::World::electronSourceAgentRight ( )

get the right ElectronSourceAgent

7.34.3.19 boost::multi_array< double, 3 > & Langmuir::World::eR ( )

get the array of precomputed erf(r/(sqrt(2)*sigma))

7.34.3.20 QList< **SourceAgent** ∗ > & Langmuir::World::eSources ( )

get a list of all ElectronSourceAgents

7.34.3.21 **ExcitonSourceAgent** & Langmuir::World::excitonSourceAgent ( )

get the RecombinationAgent

7.34.3.22 QList< **FluxAgent** ∗ > & Langmuir::World::fluxes ( )

get a list of all FluxAgents

7.34.3.23 QList< **DrainAgent** ∗ > & Langmuir::World::hDrains ( )

get a list of all ElectronSourceAgents

7.34.3.24 **HoleDrainAgent** & Langmuir::World::holeDrainAgentLeft ( )

get the left HoleDrainAgent

7.34.3.25 **HoleDrainAgent** & Langmuir::World::holeDrainAgentRight ( )

get the right HoleDrainAgent

7.34.3.26 **Grid** & Langmuir::World::holeGrid ( )

get the hole Grid, used for holding HoleAgents

7.34.3.27 QList< **ChargeAgent** ∗ > & Langmuir::World::holes ( )

get a list of all HoleAgents

7.34.3.28 int Langmuir::World::holesMinusElectrons ( )

The number of holes - electrons.

7.34.3.29 **HoleSourceAgent** & Langmuir::World::holeSourceAgentLeft ( )

get the left HoleSourceAgent

7.34.3.30 **HoleSourceAgent** & Langmuir::World::holeSourceAgentRight ( )

get the right HoleSourceAgent

7.34.3.31 QList< **SourceAgent** ∗ > & Langmuir::World::hSources ( )

get a list of all ElectronSourceAgents

7.34.3.32 void Langmuir::World::initialize (const QString & *fileName* = "") [private]

initialize all objects

Parameters

fileName	input file name

A very long, though not all that complicated function that creates all the simulation objects. Best to read through it in the source code.

7.34.3.33 boost::multi_array< double, 3 > & Langmuir::World::iR ( )

get the array of precomputed inverse-r values

7.34.3.34 KeyValueParser & Langmuir::World::keyValueParser ( )

get the KeyValueParser, used for parsing input files.

7.34.3.35 Logger & Langmuir::World::logger ( )

get the Logger, used for writing output

7.34.3.36 int Langmuir::World::maxChargeAgents ( )

get the max number of ChargeAgents allowed

7.34.3.37 int Langmuir::World::maxChargeAgentsAndChargedDefects ( )

get the max number of ChargeAgents & charged defects

7.34.3.38 int Langmuir::World::maxDefects ( )

get the max number of Defects allowed

7.34.3.39 int Langmuir::World::maxElectronAgents ( )

get the max number of ElectronAgents allowed

7.34.3.40 int Langmuir::World::maxHoleAgents ( )

get the max number of HoleAgents allowed

7.34.3.41 int Langmuir::World::maxTraps ( )

get the max number of Traps allowed

7.34.3.42 int Langmuir::World::numChargeAgents ( )

get the current number of ChargeAgents

7.34.3.43 int Langmuir::World::numChargeAgentsAndChargedDefects ( )

get the current number of ChargeAgents & charged defects

7.34.3.44 int Langmuir::World::numDefects ( )

get the current number of Defects

7.34.3.45 int Langmuir::World::numElectronAgents ( )

get the current number of ElectronAgents

7.34.3.46 int Langmuir::World::numHoleAgents ( )

get the current number of HoleAgents

7.34.3.47 int Langmuir::World::numTraps ( )

get the current number of Traps

7.34.3.48 OpenClHelper & Langmuir::World::opencl ( )

get the OpenClHelper, used for calculating Coulomb interactions with a Graphics Card

7.34.3.49 SimulationParameters & Langmuir::World::parameters ( )

get the SimulationParameters, a struct used for holding simulation parameters.

7.34.3.50 double Langmuir::World::percentElectronAgents ( )

get the percent of ElectronAgents reached, of the total grid volume

7.34.3.51 double Langmuir::World::percentHoleAgents ( )

get the percent of HoleAgents reached, of the total grid volume

7.34.3.52 void Langmuir::World::placeDefects (const QList< int > & *siteIDs* = `QList<int>()`) `[private]`

places defects

Parameters

siteIDs	a list of defect site ids

Places carriers according to the site ids passed. If more need placing (according to SimulationParameters::seedCharges), then they are placed randomly.

7.34.3.53 void Langmuir::World::placeElectrons (const QList< int > & *siteIDs* = `QList<int>()`) `[private]`

places electrons

Parameters

siteIDs	a list of electron site ids

Places carriers according to the site ids passed. If more need placing (according to SimulationParameters::seedCharges), then they are placed randomly.

7.34.3.54 void Langmuir::World::placeHoles (const QList< int > & *siteIDs* = `QList<int>()`) `[private]`

places holes

Parameters

siteIDs	a list of hole site ids

Places carriers according to the site ids passed. If more need placing (according to SimulationParameters::seedCharges), then they are placed randomly.

7.34.3.55 Potential & Langmuir::World::potential ( )

get the Potential, a calculator used for...calculating the potential.

7.34.3.56 boost::multi_array< double, 3 > & Langmuir::World::R1 ( )

get the array of precomputed r-squared values

7.34.3.57 boost::multi_array< double, 3 > & Langmuir::World::R2 ( )

get the array of precomputed r values

7.34.3.58 Random & Langmuir::World::randomNumberGenerator ( )

get the Random, used for creating random numbers

7.34.3.59 double Langmuir::World::reachedChargeAgents ( )

get the percent of ChargeAgents reached, of the max allowed

7.34.3.60 double Langmuir::World::reachedElectronAgents ( )

get the percent of ElectronAgents reached, of the max allowed

7.34.3.61 double Langmuir::World::reachedHoleAgents ( )

get the percent of HoleAgents reached, of the max allowed

7.34.3.62 RecombinationAgent & Langmuir::World::recombinationAgent ( )

get the RecombinationAgent

7.34.3.63 void Langmuir::World::setFluxInfo (const QList< quint64 > & *fluxInfo*) [private]

set attempts / successes for sources / drains

7.34.3.64 boost::multi_array< double, 3 > & Langmuir::World::sI ( )

get the self interactions

7.34.3.65 QList< SourceAgent ∗ > & Langmuir::World::sources ( )

get a list of all SourceAgents

7.34.3.66 QList< int > & Langmuir::World::trapSiteIDs ( )

get a list of all trap sites

7.34.3.67 QList< double > & Langmuir::World::trapSitePotentials ( )

get a list of all trap potentials

7.34.3.68 QList< DrainAgent ∗ > & Langmuir::World::xDrains ( )

get a list of all ElectronSourceAgents

7.34.3.69 QList< SourceAgent ∗ > & Langmuir::World::xSources ( )

get a list of all ElectronSourceAgents

7.34.4 Member Data Documentation

7.34.4.1 CheckPointer∗ Langmuir::World::m_checkPointer [private]

pointer to CheckPointer, used for reading/writing input(checkpoint) files

7.34.4.2 boost::multi_array<double,3> Langmuir::World::m_couplingConstants [private]

array of coupling constants

This array is indexed by dx, dy, dz values.

7.34.4.3 QList<int> Langmuir::World::m_defectSiteIDs [private]

list of defect sites

7.34.4.4 QList<DrainAgent∗> Langmuir::World::m_drains [private]

list of DrainAgents

7.34.4.5 QList< DrainAgent ∗ > Langmuir::World::m_eDrains [private]

list of ElectronSourceAgents

7.34.4.6 ElectronDrainAgent∗ Langmuir::World::m_electronDrainAgentLeft [private]

pointer to left ElectronDrainAgent

7.34.4.7 ElectronDrainAgent∗ Langmuir::World::m_electronDrainAgentRight [private]

pointer to right ElectronDrainAgent

7.34.4.8 Grid∗ Langmuir::World::m_electronGrid [private]

pointer to electron Grid, used for keeping track of ElectronAgents

7.34.4.9 QList< ChargeAgent ∗ > Langmuir::World::m_electrons [private]

list of electrons

7.34.4.10 ElectronSourceAgent∗ Langmuir::World::m_electronSourceAgentLeft [private]

pointer to left ElectronDrainAgent

7.34.4.11 ElectronSourceAgent∗ Langmuir::World::m_electronSourceAgentRight [private]

pointer to right ElectronDrainAgent

7.34.4.12 boost::multi_array< double,3 > Langmuir::World::m_eR [private]

array of precomputed erf(r/(s∗sqrt(2)) values

This array is indexed by dx, dy, dz values, and r is in grid-units

7.34.4.13 QList< SourceAgent ∗ > Langmuir::World::m_eSources [private]

list of ElectronSourceAgents

7.34.4.14 ExcitonSourceAgent∗ Langmuir::World::m_excitonSourceAgent [private]

pointer to ExcitonSourceAgent, used for injecting Excitons

7.34.4.15 QList< FluxAgent ∗ > Langmuir::World::m_fluxAgents [private]

list of all FluxAgents, such as SoureAgents, DrainAgents, etc.

7.34.4.16 QList< DrainAgent ∗ > Langmuir::World::m_hDrains [private]

list of HoleSourceAgents

7.34.4.17 HoleDrainAgent∗ Langmuir::World::m_holeDrainAgentLeft [private]

pointer to left HoleDrainAgent

7.34.4.18 HoleDrainAgent∗ Langmuir::World::m_holeDrainAgentRight [private]

pointer to right HoleDrainAgent

7.34.4.19 Grid∗ Langmuir::World::m_holeGrid [private]

pointer to hole Grid, used for keeping track of HoleAgents

7.34.4.20 QList<ChargeAgent∗> Langmuir::World::m_holes [private]

list of holes

7.34.4.21 HoleSourceAgent∗ Langmuir::World::m_holeSourceAgentLeft [private]

pointer to left HoleDrainAgent

7.34.4.22 HoleSourceAgent∗ Langmuir::World::m_holeSourceAgentRight [private]

pointer to right HoleDrainAgent

7.34.4.23 QList<SourceAgent∗> Langmuir::World::m_hSources [private]

list of HoleSourceAgents

7.34.4.24 boost::multi_array<double,3> Langmuir::World::m_iR [private]

array of precomputed inverse-r values

This array is indexed by dx, dy, dz values, and r is in grid-units

7.34.4.25 KeyValueParser∗ Langmuir::World::m_keyValueParser [private]

pointer to KeyValueParser, used for parsing key=value pairs

7.34.4.26 Logger∗ Langmuir::World::m_logger [private]

pointer to Logger, used for output

7.34.4.27 int Langmuir::World::m_maxDefects [private]

max number of defects

7.34.4.28 int Langmuir::World::m_maxElectrons [private]

max number of electrons

7.34.4.29 int Langmuir::World::m_maxHoles [private]

max number of holes

7.34.4.30 int Langmuir::World::m_maxTraps [private]

max number of traps

7.34.4.31 OpenClHelper* Langmuir::World::m_ocl [private]

pointer to OpenClHelper, used for Graphics Card calculations

7.34.4.32 SimulationParameters* Langmuir::World::m_parameters [private]

pointer to SimulationParameters

7.34.4.33 Potential* Langmuir::World::m_potential [private]

pointer to Potential, used for calculating the potential

7.34.4.34 boost::multi_array<double,3> Langmuir::World::m_R1 [private]

array of precomputed r values

This array is indexed by dx, dy, dz values, and r is in grid-units

7.34.4.35 boost::multi_array<double,3> Langmuir::World::m_R2 [private]

array of precomputed r-squared values

This array is indexed by dx, dy, dz values, and r is in grid-units

7.34.4.36 Random* Langmuir::World::m_rand [private]

pointer to Random, used for generating random numbers

7.34.4.37 RecombinationAgent* Langmuir::World::m_recombinationAgent [private]

pointer to electron/hole RecombinationAgent, used for removing Excitons

7.34.4.38 boost::multi_array<double, 3> Langmuir::World::m_sI [private]

self interaction, which is 1/(4 pi e e0 r), with r=1 grid unit When a charge at it's future site interacts with other charges at their current site, the charge will interact with it's own current site. So, this value needs to be subtracted off.

7.34.4.39 QList<SourceAgent*> Langmuir::World::m_sources [private]

list of SourceAgents

7.34.4.40 QList<int> Langmuir::World::m_trapSiteIDs [private]

list of trap sites

7.34.4.41 QList<double> Langmuir::World::m_trapSitePotentials [private]

list of trap potentials

7.34.4.42 QList<**DrainAgent**∗> Langmuir::World::m_xDrains [private]

list of ExcitonSourceAgents

7.34.4.43 QList<**SourceAgent**∗> Langmuir::World::m_xSources [private]

list of ExcitonSourceAgents

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

- world.h
- world.cpp

7.35 Langmuir::XYZWriter Class Reference

A class to output xyz files.

`#include <writer.h>`

Public Member Functions

- XYZWriter (World &world, const QString &name, QObject ∗parent=0)

  constructs the writer, has the same parameters as OutputInfo
- void write ()

  Write XYZ of the current step to the stream.

Protected Member Functions

- void writeVMDInitFile ()

  write a VMD script useful for opening the XYZ file

Protected Attributes

- World & m_world

  reference to the world object
- OutputStream m_stream

  output file stream

7.35.1 Detailed Description

A class to output xyz files.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 Langmuir::XYZWriter::XYZWriter (World & *world*, const QString & *name*, QObject ∗ *parent* = 0)

constructs the writer, has the same parameters as OutputInfo

7.35.3 Member Function Documentation

7.35.3.1 void Langmuir::XYZWriter::write ( )

Write XYZ of the current step to the stream.

7.35.3.2 void Langmuir::XYZWriter::writeVMDInitFile ( ) [protected]

write a VMD script useful for opening the XYZ file

7.35.4 Member Data Documentation

7.35.4.1 OutputStream Langmuir::XYZWriter::m_stream [protected]

output file stream

7.35.4.2 World& Langmuir::XYZWriter::m_world [protected]

reference to the world object

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

- writer.h
- writer.cpp

Chapter 8

File Documentation

8.1 agent.h File Reference

```
#include <QTextStream>
#include <QMetaObject>
#include <QMetaEnum>
#include <QVector>
#include <QObject>
#include <QString>
#include <QDebug>
```

Classes

- class Langmuir::Agent

    A class that abstractly represents an object that can occupy grid sites.

Namespaces

- namespace Langmuir

Functions

- QTextStream & Langmuir::operator<< (QTextStream &stream, const Agent::Type e)

    Output Agent type enum to stream.

- QDebug Langmuir::operator<< (QDebug dbg, const Agent::Type e)

    Output Agent type enum to debug information.

8.2 chargeagent.cpp File Reference

```
#include "openclhelper.h"
```

```
#include "chargeagent.h"
#include "drainagent.h"
#include "parameters.h"
#include "simulation.h"
#include "potential.h"
#include "cubicgrid.h"
#include "world.h"
#include "rand.h"
```

Namespaces

- namespace Langmuir

8.3 chargeagent.h File Reference

```
#include "agent.h"
```

Classes

- class Langmuir::ChargeAgent

  A class to represent moving charged particles.
- class Langmuir::ElectronAgent

  A class to represent moving negative charges.
- class Langmuir::HoleAgent

  A class to represent moving positive charges.

Namespaces

- namespace Langmuir

8.4 checkpointer.cpp File Reference

```
#include "checkpointer.h"
#include "chargeagent.h"
#include "output.h"
#include "world.h"
#include "rand.h"
#include "keyvalueparser.h"
#include "fluxagent.h"
#include <fstream>
#include <limits>
#include <iomanip>
```

Namespaces

- namespace Langmuir

8.5 checkpointer.h File Reference

```
#include <QObject>
#include <QMap>
#include "parameters.h"
```

Classes

- class Langmuir::CheckPointer

  A class to read and write checkpoint files.

Namespaces

- namespace Langmuir

Functions

- static std::ostream & Langmuir::operator<< (std::ostream &stream, QString &string)
- static std::istream & Langmuir::operator>> (std::istream &stream, QString &string)

8.6 copy_to_frank.py File Reference

Namespaces

- namespace copy_to_frank

Variables

- tuple copy_to_frank.work = os.getcwd()
- list copy_to_frank.exclude
- list copy_to_frank.found = [ ]
- copy_to_frank.copy_me = True

8.7 cubicgrid.cpp File Reference

```
#include "cubicgrid.h"
#include <cmath>
#include "world.h"
#include "parameters.h"
#include "drainagent.h"
```

Namespaces

- namespace Langmuir

Functions

- QTextStream & Langmuir::operator<< (QTextStream &stream, const Grid::CubeFace e)

  Overload QTextStream for the Grid::CubeFace Enum.
- QDebug Langmuir::operator<< (QDebug dbg, const Grid::CubeFace e)

  Overload QDebug for the Grid::CubeFace Enum.

8.8 cubicgrid.h File Reference

```
#include "agent.h"
#include <QTextStream>
#include <QVector>
#include <QString>
#include <QObject>
#include <QDebug>
```

Classes

- class Langmuir::Grid

  A class to hold Agents, calculate their positions, and store the background potential.

Namespaces

- namespace Langmuir

Functions

- QTextStream & Langmuir::operator<< (QTextStream &stream, const Grid::CubeFace e)

  Overload QTextStream for the Grid::CubeFace Enum.
- QDebug Langmuir::operator<< (QDebug dbg, const Grid::CubeFace e)

  Overload QDebug for the Grid::CubeFace Enum.

8.9 drainagent.cpp File Reference

```
#include "drainagent.h"
#include "chargeagent.h"
#include "parameters.h"
#include "writer.h"
#include "world.h"
#include "rand.h"
```

Namespaces

- namespace Langmuir

8.10 drainagent.h File Reference

```
#include "fluxagent.h"
```

Classes

- class Langmuir::DrainAgent

  A class to remove charges.
- class Langmuir::ElectronDrainAgent

  A class to remove ElectronAgents.
- class Langmuir::HoleDrainAgent

  A class to remove HoleAgents.
- class Langmuir::RecombinationAgent

  A class to remove Excitons.

Namespaces

- namespace Langmuir

8.11 fluxagent.cpp File Reference

```
#include "fluxagent.h"
#include "parameters.h"
#include "world.h"
#include "rand.h"
```

Namespaces

- namespace Langmuir

8.12 fluxagent.h File Reference

```
#include "agent.h"
#include "cubicgrid.h"
```

Classes

- class Langmuir::FluxAgent

  A class to change the number of carriers in the system.

Namespaces

- namespace Langmuir

8.13 gridview.cpp File Reference

```
#include "openclhelper.h"
#include "drainagent.h"
#include "gridview.h"
#include "GL/glu.h"
#include "writer.h"
```

Namespaces

- namespace Langmuir

8.14 keyvalueparser.cpp File Reference

```
#include "keyvalueparser.h"
#include <QDebug>
#include <QRegExp>
#include <QStringList>
#include <QFile>
#include "output.h"
#include <ostream>
```

Namespaces

- namespace Langmuir

Functions

- std::ostream & Langmuir::operator<< (std::ostream &stream, const KeyValueParser &keyValueParser)

8.15 keyvalueparser.h File Reference

```
#include <QObject>
#include "variable.h"
#include "parameters.h"
#include "world.h"
```

Classes

- class Langmuir::KeyValueParser

  A class to read the parameters and store them in the correct place.

Namespaces

- namespace Langmuir

8.16 langmuir.cpp File Reference

```
#include "sourceagent.h"
#include "simulation.h"
#include "drainagent.h"
#include "cubicgrid.h"
#include "keyvalueparser.h"
#include "writer.h"
#include "world.h"
#include "openclhelper.h"
#include "checkpointer.h"
#include "parameters.h"
#include <QApplication>
#include <QPrinter>
#include <QPainter>
#include <QColor>
#include <QtCore/QStringList>
#include <QtCore/QFile>
#include <QtCore/QTextStream>
#include <QtCore/QDateTime>
#include <QtCore/QDebug>
#include <QThreadPool>
```

Functions

- QTextStream & progress (QTextStream &stream, SimulationParameters &par)
- void alterMaxThreads (SimulationParameters &par)
- int main (int argc, char ∗argv[])

8.16.1 Function Documentation

8.16.1.1 void alterMaxThreads (SimulationParameters & *par*)

8.16.1.2 int main (int *argc,* char ∗ *argv[]*)

8.16.1.3 QTextStream& progress (QTextStream & *stream,* SimulationParameters & *par*)

8.17 langmuirview.cpp File Reference

```
#include "gridview.h"
#include <QtGui>
```

Functions

- int main (int argc, char ∗∗argv)

8.17.1 Function Documentation

8.17.1.1 int main (int *argc,* char ∗∗ *argv*)

8.18 openclhelper.cpp File Reference

```
#include <QTextStream>
#include "pbsgpuparser.h"
#include "openclhelper.h"
#include "chargeagent.h"
#include "parameters.h"
#include "cubicgrid.h"
#include "potential.h"
#include "world.h"
```

Namespaces

- namespace Langmuir

8.19 openclhelper.h File Reference

```
#include <QObject>
#include <QVector>
```

Classes

- class Langmuir::OpenClHelper

  A Class to run OpenCL calculations.

Namespaces

- namespace Langmuir

Macros

- #define __CL_ENABLE_EXCEPTIONS

8.19.1 Macro Definition Documentation

8.19.1.1 #define __CL_ENABLE_EXCEPTIONS

8.20 output.cpp File Reference

```
#include "output.h"
#include <QDateTime>
#include <QFileInfo>
#include <QDir>
```

Namespaces

- namespace Langmuir

Functions

- void Langmuir::backupFile (const QString &name)

  Back up a file.
- QTextStream & newline (QTextStream &s)

  put a newline character in the stream that ignores the streams current FieldWidth
- QTextStream & space (QTextStream &s)

  put a space in the stream that ignores the streams current FieldWidth

8.20.1 Function Documentation

8.20.1.1 QTextStream& newline (QTextStream & *s*)

put a newline character in the stream that ignores the streams current FieldWidth

8.20.1.2 QTextStream& space (QTextStream & *s*)

put a space in the stream that ignores the streams current FieldWidth

8.21 output.h File Reference

```
#include "parameters.h"
#include <QTextStream>
#include <QObject>
#include <QFile>
```

Classes

- class Langmuir::OutputInfo

  A class to generate file names using the SimulationParameters.
- class Langmuir::OutputStream

  A class to combine QFile, QTextStream and OutputInfo (QFileInfo).

Namespaces

- namespace Langmuir

Functions

- QTextStream & newline (QTextStream &s)

  put a newline character in the stream that ignores the streams current FieldWidth
- QTextStream & space (QTextStream &s)

  put a space in the stream that ignores the streams current FieldWidth
- void Langmuir::backupFile (const QString &name)

  Back up a file.

8.21.1 Function Documentation

8.21.1.1 QTextStream& newline (QTextStream & *s*)

put a newline character in the stream that ignores the streams current FieldWidth

8.21.1.2 QTextStream& space (QTextStream & *s*)

put a space in the stream that ignores the streams current FieldWidth

8.22 parameters.h File Reference

```
#include <QDateTime>
#include <QFileInfo>
#include <QDebug>
#include <cmath>
#include <QDir>
```

Classes

- struct Langmuir::ConfigurationInfo

    A struct to temporarily store site IDs.
- struct Langmuir::SimulationParameters

    A struct to store all simulation options To add new variables, follow these steps:

Namespaces

- namespace Langmuir

Functions

- void Langmuir::setCalculatedValues (SimulationParameters &par)

    sets parameters that depend upon other parameters
- void Langmuir::checkSimulationParameters (SimulationParameters &par)

    check the parameters, making sure they are valid

8.23 pbsgpuparser.cpp File Reference

```
#include "pbsgpuparser.h"
#include <QRegExp>
#include <QDebug>
#include <QFile>
```

Namespaces

- namespace Langmuir

8.24 pbsgpuparser.h File Reference

```
#include <QObject>
#include <QList>
```

Classes

- class Langmuir::PBSGPUParser

Namespaces

- namespace Langmuir

8.25 potential.cpp File Reference

```
#include "potential.h"
#include "parameters.h"
#include "chargeagent.h"
#include "cubicgrid.h"
#include "world.h"
#include "rand.h"
#include <cmath>
```

Namespaces

- namespace Langmuir

8.26 potential.h File Reference

```
#include <QObject>
#include "boost/multi_array.hpp"
```

Classes

- class Langmuir::Potential

  A class to calculate the potential.

Namespaces

- namespace Langmuir

Macros

- #define BOOST_DISABLE_ASSERTS

8.26.1 Macro Definition Documentation

8.26.1.1 #define BOOST_DISABLE_ASSERTS

8.27 rand.cpp File Reference

```
#include "rand.h"
#include <fstream>
#include <sstream>
```

Namespaces

- namespace Langmuir

Functions

- QDataStream & Langmuir::operator<< (QDataStream &stream, Random &random)
- QDataStream & Langmuir::operator>> (QDataStream &stream, Random &random)
- QTextStream & Langmuir::operator<< (QTextStream &stream, Random &random)
- QTextStream & Langmuir::operator>> (QTextStream &stream, Random &random)
- std::ostream & Langmuir::operator<< (std::ostream &stream, Random &random)
- std::istream & Langmuir::operator>> (std::istream &stream, Random &random)

8.28 rand.h File Reference

```
#include <QObject>
#include <QDataStream>
#include <QTextStream>
#include <boost/random.hpp>
#include <ctime>
```

Classes

- class Langmuir::Random

  A class to generate random numbers.

Namespaces

- namespace Langmuir

8.29 README.md File Reference

8.30 simulation.cpp File Reference

```
#include "simulation.h"
#include "openclhelper.h"
#include "parameters.h"
#include "chargeagent.h"
#include "sourceagent.h"
#include "drainagent.h"
#include "potential.h"
#include "cubicgrid.h"
#include "checkpointer.h"
#include "writer.h"
#include "world.h"
#include "rand.h"
```

Namespaces

- namespace Langmuir

8.31 simulation.h File Reference

```
#include <QObject>
```

Classes

- class Langmuir::Simulation

  A class to orchestrate the calculation.

Namespaces

- namespace Langmuir

8.32 sourceagent.cpp File Reference

```
#include "sourceagent.h"
#include "chargeagent.h"
#include "parameters.h"
#include "potential.h"
#include "world.h"
#include "rand.h"
```

Namespaces

- namespace Langmuir

8.33 sourceagent.h File Reference

```
#include "fluxagent.h"
```

Classes

- class Langmuir::SourceAgent

  A class to inject charges.
- class Langmuir::ElectronSourceAgent

  A class to inject ElectronAgents.
- class Langmuir::HoleSourceAgent

  A class to inject HoleAgents.
- class Langmuir::ExcitonSourceAgent

  A class to inject Excitons.

Namespaces

- namespace Langmuir

8.34 test.cpp File Reference

```
#include <QtCore>
#include <iostream>
#include <fstream>
#include <boost/random.hpp>
#include <ctime>
```

Functions

- int main (int argc, char ∗argv[])

8.34.1 Function Documentation

8.34.1.1 int main (int *argc,* char ∗ *argv[]*)

8.35 tolerance.cpp File Reference

```
#include "tolerance.h"
#include "fluxagent.h"
#include "world.h"
#include "parameters.h"
```

Namespaces

- namespace Langmuir

8.36 tolerance.h File Reference

```
#include <QObject>
```

Classes

- class Langmuir::Tolerance

  A class to check if the simulation is converging.

Namespaces

- namespace Langmuir

8.37 variable.h File Reference

```
#include <QTextStream>
#include <QDateTime>
#include <QObject>
#include <QDebug>
#include <limits>
#include <ostream>
```

Classes

- class Langmuir::Variable

  A class to map between variable names (keys) and locations (references)
- class Langmuir::TypedVariable< T >

  A template class to map between variable names (keys) and locations (references)

Namespaces

- namespace Langmuir

Functions

- QTextStream & Langmuir::operator<< (QTextStream &stream, const QDateTime &datetime)

  output QDateTime as qint64 mSecsSinceEpoch
- QTextStream & Langmuir::operator<< (QTextStream &stream, const Variable &variable)

  overload operator to write keyValue() to a stream
- QDebug Langmuir::operator<< (QDebug dbg, const Variable &variable)

  overload operator to write keyValue() to a QDebug
- std::ostream & Langmuir::operator<< (std::ostream &stream, Variable &variable)

  Operator overload to output to output 'key = value' to std::ostream.

8.38 world.cpp File Reference

```
#include "parameters.h"
#include "openclhelper.h"
#include "chargeagent.h"
#include "sourceagent.h"
#include "drainagent.h"
#include "potential.h"
#include "cubicgrid.h"
#include "writer.h"
#include "world.h"
#include "rand.h"
#include "fluxagent.h"
#include "output.h"
#include "keyvalueparser.h"
#include "checkpointer.h"
```

Namespaces

- namespace Langmuir

8.39 world.h File Reference

```
#include <QtCore>
#include <QtGui>
#include "boost/multi_array.hpp"
```

Classes

- class Langmuir::World

  A class to hold all objects in a simulation.

Namespaces

- namespace Langmuir

Macros

- #define BOOST_DISABLE_ASSERTS

8.39.1 Macro Definition Documentation

8.39.1.1 #define BOOST_DISABLE_ASSERTS

8.40 writer.cpp File Reference

```
#include "writer.h"
#include "parameters.h"
#include "world.h"
#include "cubicgrid.h"
#include "chargeagent.h"
#include "fluxagent.h"
#include "openclhelper.h"
```

Namespaces

- namespace Langmuir

8.41 writer.h File Reference

```
#include <QObject>
#include <QPainter>
#include <QColor>
#include <QImage>
#include "output.h"
```

Classes

- class Langmuir::XYZWriter

  A class to output xyz files.
- class Langmuir::FluxWriter

  A class to output source and drain info.
- class Langmuir::CarrierWriter

  A class to output carrier stats (lifetime and pathlength)
- class Langmuir::ExcitonWriter

  A class to output exciton stats (lifetime and pathlength)
- class Langmuir::GridImage

  A class to draw images of the grid.
- class Langmuir::Logger

  A class that organizes output.

Namespaces

- namespace Langmuir

Index