Collision-Code

1.0

Generated by Doxygen 1.8.11

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Atom Class Reference

```
#include <Atom.h>
```

Inherited by StdAtom.

Public Member Functions

- virtual ∼Atom ()
- virtual Vector3D ∗ getPosition () const =0
- virtual Vector3D ∗ getInitialPosition () const =0
- virtual std::string getSymbol () const =0
- virtual double getCharge () const =0
- virtual void setPosition (Vector3D ∗c)=0
- virtual void setSymbol (std::string s)=0
- virtual void setCharge (double c)=0

4.1.1 Constructor & Destructor Documentation

4.1.1.1 virtual Atom::∼Atom ( ) `[inline], [virtual]`

Releases allocated resources.

4.1.2 Member Function Documentation

4.1.2.1 virtual double Atom::getCharge ( ) const `[pure virtual]`

Returns charge value of atom.

Implemented in StdAtom.

4.1.2.2 virtual Vector3D* Atom::getInitialPosition ( ) const [pure virtual]

Returns the initial position of atom.

Implemented in StdAtom.

4.1.2.3 virtual Vector3D* Atom::getPosition ( ) const [pure virtual]

Returns position of atom.

Implemented in StdAtom.

4.1.2.4 virtual std::string Atom::getSymbol ( ) const [pure virtual]

Returns symbol of atom.

Implemented in StdAtom.

4.1.2.5 virtual void Atom::setCharge (double *c*) [pure virtual]

Sets a new charge value for atom.

Parameters

c	One double.

Implemented in StdAtom.

4.1.2.6 virtual void Atom::setPosition (Vector3D * *c*) [pure virtual]

Sets a new position for the atom.

Parameters

c	One coordinate.

Implemented in StdAtom.

4.1.2.7 virtual void Atom::setSymbol (std::string *s*) [pure virtual]

Sets a new symbol value for atom.

Parameters

s	A string value.

Implemented in StdAtom.

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

- Collision-Code/molecule/Atom.h

4.2 AtomInformations Class Reference

```
#include <AtomInformations.h>
```

Public Member Functions

- virtual ∼AtomInformations ()
- void loadFile (std::string fileName)
- bool isExistingSymbol (std::string symb) const
- std::string getSymbol (int atomicMass)
- int getAtomicNumber (std::string symb) const
- double getAtomicMass (std::string symb)
- double getEOLJHe (std::string symb)
- double getROLJHe (std::string symb)
- double getHSRadius (std::string symb)

Static Public Member Functions

- static AtomInformations ∗const getInstance ()

4.2.1 Constructor & Destructor Documentation

4.2.1.1 AtomInformations::∼AtomInformations ( ) `[virtual]`

Destructor.

4.2.2 Member Function Documentation

4.2.2.1 double AtomInformations::getAtomicMass (std::string *symb*)

Parameters

symb	the symbol of the atom to search mass for.

Returns

the mass of the atom of symbol symb.

4.2.2.2 int AtomInformations::getAtomicNumber (std::string *symb*) const

Parameters

symb	the symbol of the atom to search atomic number for.

Returns

the atomic number of the atom of symbol symb.

4.2.2.3 double AtomInformations::getEOLJHe (std::string *symb*)

Parameters

symb	the symbol of the atom to search mass for.

Returns

EOLJ for Helium of the atom of symbol symb.

4.2.2.4 double AtomInformations::getHSRadius (std::string *symb*)

Parameters

symb	the symbol of the atom to search hard sphere radius for.

Returns

the hard sphere radius of the atom of symbol symb in meter.

4.2.2.5 static AtomInformations* const AtomInformations::getInstance ( ) [inline], [static]

Returns

an instance of AtomInformations to work with.

4.2.2.6 double AtomInformations::getROLJHe (std::string *symb*)

Parameters

symb	the symbol of the atom to search mass for.

Returns

ROLJ for Helium of the atom of symbol symb.

4.2.2.7 std::string AtomInformations::getSymbol (int *atomicMass*)

Search for the symbol having the integer part of its mass equals to atomicMass.

Parameters

atomicMass	the mass to search for.

Returns

the symbol corresponding to atomicMass.

4.2.2.8 bool AtomInformations::isExistingSymbol (std::string *symb*) const `[inline]`

Tests if a symbol exists.

Parameters

symb	the symbol to test.

Returns

true if symbol exists in data.

4.2.2.9 void AtomInformations::loadFile (std::string *fileName*)

Load data from file.

Parameters

fileName	the name of the file to load.

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

- Collision-Code/general/AtomInformations.h
- Collision-Code/general/AtomInformations.cpp

4.3 CalculationOperator Class Reference

`#include <CalculationOperator.h>`

Inherited by StdCalculationOperator.

Public Member Functions

- virtual ∼CalculationOperator ()
- virtual Result ∗ getResults ()=0
- virtual CalculationState ∗ getCalculationState () const =0
- virtual void runEHSSAndPA ()=0
- virtual void runTM ()=0

4.3.1 Constructor & Destructor Documentation

4.3.1.1 virtual CalculationOperator::∼CalculationOperator ( ) [inline], [virtual]

4.3.2 Member Function Documentation

4.3.2.1 virtual CalculationState∗ CalculationOperator::getCalculationState ( ) const [pure virtual]

Returns

the calculation state associated with this calculation operator.

Implemented in StdCalculationOperator.

4.3.2.2 virtual Result∗ CalculationOperator::getResults ( ) [pure virtual]

Returns the results.

Returns

a pointer to the results of calculations.

Implemented in StdCalculationOperator.

4.3.2.3 virtual void CalculationOperator::runEHSSAndPA ( ) [pure virtual]

Launches the calculation of EHSS and PA.

Implemented in StdCalculationOperator.

4.3.2.4 virtual void CalculationOperator::runTM ( ) [pure virtual]

Launches the calculation of TM.

Implemented in StdCalculationOperator.

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

- Collision-Code/math/CalculationOperator.h

4.4 CalculationState Class Reference

```
#include <CalculationState.h>
```

Inherits Observable.

Public Member Functions

- CalculationState (Molecule ∗molecule, int totalTrajectories)
- virtual ∼CalculationState ()
- Molecule ∗ getMolecule () const
- double getPercentageFinishedTrajectories () const
- int getNumberFinishedTractories () const
- int getNumberTotalTractories () const
- bool hasEHSSStarted () const
- bool hasEHSSEnded () const
- bool hasPAStarted () const
- bool hasPAEnded () const
- bool hasTMStarted () const
- bool hasTMEnded () const
- void setFinishedTrajectories (int n)
- void setEHSSStarted ()
- void setEHSSEnded ()
- void setPAStarted ()
- void setPAEnded ()
- void setTMStarted ()
- void setTMEnded ()
- void setEHSSResult (double r)
- double getEHSSResult () const
- void setPAResult (double r)
- double getPAResult () const
- void setTMResult (double r)
- double getTMResult () const
- void oneCalculationFinished ()

Additional Inherited Members

4.4.1 Constructor & Destructor Documentation

4.4.1.1 CalculationState::CalculationState (**Molecule** ∗ *molecule,* int *totalTrajectories*)

Constructor.

Parameters

molecule	the molecule on which the calculations are proceeded.
totalTrajectories	the total number of trajectories in TM calculation.

4.4.1.2 CalculationState::∼CalculationState ( ) [virtual]

Destructor.

4.4.2 Member Function Documentation

4.4.2.1 double CalculationState::getEHSSResult ( ) const [inline]

Returns

the EHSS result.

4.4.2.2 Molecule∗ CalculationState::getMolecule ( ) const [inline]

Returns

the molecule on which the calculations are proceeded.

4.4.2.3 int CalculationState::getNumberFinishedTractories ( ) const [inline]

Returns

the number of finished trajectories.

4.4.2.4 int CalculationState::getNumberTotalTractories ( ) const [inline]

Returns

the total number of trajectories to calculate.

4.4.2.5 double CalculationState::getPAResult ( ) const [inline]

Returns

the PA result.

4.4.2.6 double CalculationState::getPercentageFinishedTrajectories ( ) const [inline]

Returns

the percentage of trajectories finished.

4.4.2.7 double CalculationState::getTMResult ( ) const [inline]

Returns

the TM result.

4.4.2.8 bool CalculationState::hasEHSSEnded ( ) const [inline]

Returns

true if EHSS calculations have ended, false otherwise.

4.4.2.9 bool CalculationState::hasEHSSStarted ( ) const [inline]

Returns

true if EHSS calculations have started, false otherwise.

4.4.2.10 bool CalculationState::hasPAEnded ( ) const [inline]

Returns

true if PA calculations have ended, false otherwise.

4.4.2.11 bool CalculationState::hasPAStarted ( ) const [inline]

Returns

true if PA calculations have started, false otherwise.

4.4.2.12 bool CalculationState::hasTMEnded ( ) const [inline]

Returns

true if TM calculations have ended, false otherwise.

4.4.2.13 bool CalculationState::hasTMStarted ( ) const [inline]

Returns

true if TM calculations have started, false otherwise.

4.4.2.14 void CalculationState::oneCalculationFinished ( ) [inline]

4.4.2.15 void CalculationState::setEHSSEnded ( ) [inline]

Indicates that EHSS calculations have ended.

4.4.2.16 void CalculationState::setEHSSResult (double *r*) [inline]

Sets the EHSS result.

Parameters

r	the EHSS result.

4.4.2.17 void CalculationState::setEHSSStarted ( ) [inline]

Indicates that EHSS calculations have started.

4.4.2.18 void CalculationState::setFinishedTrajectories (int *n*)

Sets the number of trajectories finished to n.

Parameters

n	the number of trajectories finished by the TM calculations.

4.4.2.19 void CalculationState::setPAEnded ( ) [inline]

Indicates that EHSS calculations have ended.

4.4.2.20 void CalculationState::setPAResult (double *r*) [inline]

Sets the PA result.

Parameters

r	the PA result.

4.4.2.21 void CalculationState::setPAStarted ( ) [inline]

Indicates that EHSS calculations have started.

4.4.2.22 void CalculationState::setTMEnded ( ) [inline]

Indicates that EHSS calculations have ended.

4.4.2.23 void CalculationState::setTMResult (double *r*) [inline]

Sets the TM result.

Parameters

r	the TM result.

4.4.2.24 void CalculationState::setTMStarted ( ) [inline]

Indicates that EHSS calculations have started.

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

- Collision-Code/observer/state/CalculationState.h
- Collision-Code/observer/state/CalculationState.cpp

4.5 GeometryCalculator::CalculationValues Struct Reference

```
#include <GeometryCalculator.h>
```

Public Attributes

- double temperature
- double potentialEnergyStart
- double timeStepStart
- double potentialEnergyCloseCollision
- double timeStepCloseCollision
- double numberCyclesTM
- double numberPointsVelocity
- double numberPointsMCIntegrationTM
- double energyConservationThreshold
- double numberPointsMCIntegrationEHSSPA

4.5.1 Detailed Description

Structure containing values for calculation.

4.5.2 Member Data Documentation

4.5.2.1 double GeometryCalculator::CalculationValues::energyConservationThreshold

4.5.2.2 double GeometryCalculator::CalculationValues::numberCyclesTM

4.5.2.3 double GeometryCalculator::CalculationValues::numberPointsMCIntegrationEHSSPA

4.5.2.4 double GeometryCalculator::CalculationValues::numberPointsMCIntegrationTM

4.5.2.5 double GeometryCalculator::CalculationValues::numberPointsVelocity

4.5.2.6 double GeometryCalculator::CalculationValues::potentialEnergyCloseCollision

4.5.2.7 double GeometryCalculator::CalculationValues::potentialEnergyStart

4.5.2.8 double GeometryCalculator::CalculationValues::temperature

4.5.2.9 double GeometryCalculator::CalculationValues::timeStepCloseCollision

4.5.2.10 double GeometryCalculator::CalculationValues::timeStepStart

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

- Collision-Code/general/GeometryCalculator.h

4.6 CCFrame Class Reference

```
#include <CCFrame.h>
```

Inherits QMainWindow, and Observer.

Public Slots

- void openChemicalFile ()
- void openChargeFile ()
- void openAtomInfosFile ()
- void saveResults ()
- void about ()
- void updateModelShouldPABeCalculated (bool value)
- void updateModelShouldEHSSBeCalculated (bool value)
- void updateModelShouldTMBeCalculated (bool value)
- void updateModelLaunchCalculation ()
- void updateModelMaxNumberThreads (int value)
- void updateModelNbPointsMCIntegrationEHSSPA (int value)
- void updateModelTemperature (double value)
- void updateModelEnergyConservationThreshold (double value)

- void updateModelNbCompleteCycles (int value)
- void updateModelNbVelocityPoints (int value)
- void updateModelNbPointsMCIntegrationTM (int value)
- void updateModelPotentialEnergyStart (double value)
- void updateModelPotentialEnergyCloseCollision (double value)
- void updateModelTimeStepStart (double value)
- void updateModelTimeStepCloseCollision (double value)
- void expandAllNodes (bool value)
- void updateResultList (QString method, int index, double value)
- void printResults (QString str)
- void resultsAreReady ()
- void killThreadAndExit ()

Signals

- void totalPoints (int value)
- void changeProgressBarValue (int value)
- void changeProgressBarVisibility (bool value)
- void disableWidgets (bool value)
- void changeResults (QString str)
- void resultHasChanged (QString method, int index, double value)
- void callWorkerThread (StdCmdView ∗cmd)

Public Member Functions

- CCFrame ()
- virtual ∼CCFrame ()
- void update (ObservableEvent cond, Observable ∗obs)

4.6.1 Detailed Description

A class describing the graphical user interface.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 CCFrame::CCFrame ( )

Constructor.

4.6.2.2 CCFrame::∼CCFrame ( ) [virtual]

Destructor.

4.6.3 Member Function Documentation

4.6.3.1 void CCFrame::about ( ) [slot]

Opens a window displaying the "about" text.

4.6.3.2 void CCFrame::callWorkerThread (StdCmdView ∗ *cmd*) [signal]

4.6.3.3 void CCFrame::changeProgressBarValue (int *value*) [signal]

4.6.3.4 void CCFrame::changeProgressBarVisibility (bool *value*) [signal]

4.6.3.5 void CCFrame::changeResults (QString *str*) [signal]

4.6.3.6 void CCFrame::disableWidgets (bool *value*) [signal]

4.6.3.7 void CCFrame::expandAllNodes (bool *value*) [slot]

Expands or collapses all nodes of the geometries list.

Parameters

value	true if we need to expand all nodes, false to collapse them.

4.6.3.8 void CCFrame::killThreadAndExit ( ) [slot]

Stop calculation thread and exit the application.

4.6.3.9 void CCFrame::openAtomInfosFile ( ) [slot]

Opens a window to choose a modelling file.

4.6.3.10 void CCFrame::openChargeFile ( ) [slot]

Opens a window to choose a charge file.

4.6.3.11 void CCFrame::openChemicalFile ( ) [slot]

Opens a window to choose a geometries file.

4.6.3.12 void CCFrame::printResults (QString *str*) [slot]

Prints results in the correct location.

4.6.3.13 void CCFrame::resultHasChanged (QString *method,* int *index,* double *value*) [signal]

4.6.3.14 void CCFrame::resultsAreReady ( ) [slot]

Indicates to the graphical user interface that all calculations are finished.

4.6.3.15 void CCFrame::saveResults ( ) [slot]

Opens a window to choose a file to save results.

4.6.3.16 void CCFrame::totalPoints (int *value*) [signal]

4.6.3.17 void CCFrame::update (ObservableEvent *cond*, Observable ∗ *obs*) [virtual]

Updates the observer.

Parameters

cond	the condition that triggered the notification.
obs	the Observable which triggered the call. May be null.

Implements Observer.

4.6.3.18 void CCFrame::updateModelEnergyConservationThreshold (double *value*) [slot]

Indicates to the model the energy conservation threshold.

Parameters

value	the energy conservation threshold.

4.6.3.19 void CCFrame::updateModelLaunchCalculation ( ) [slot]

Indicates to the model to launch the calculations.

4.6.3.20 void CCFrame::updateModelMaxNumberThreads (int *value*) [slot]

Indicates to the model the maximum number of thread for calculations by TM method.

Parameters

value	the maximum number of threads for calculations by TM method.

4.6.3.21 void CCFrame::updateModelNbCompleteCycles (int *value*) [slot]

Indicates to the model the number of complete cycles in TM method.

Parameters

value	the number of complete cycles in TM method.

4.6.3.22 void CCFrame::updateModelNbPointsMCIntegrationEHSSPA (int *value*) [slot]

Indicates to the model the number of points in Monte-Carlo integrations for EHSS and PA methods.

Parameters

value	the number of points in Monte-Carlo integrations for EHSS and PA methods.

4.6.3.23 void CCFrame::updateModelNbPointsMCIntegrationTM (int *value*) [slot]

Indicates to the model the number of points for Monte-Carlo integrations in TM method.

Parameters

value	the number of points for Monte-Carlo integrations in TM method.

4.6.3.24 void CCFrame::updateModelNbVelocityPoints (int *value*) [slot]

Indicates to the model the number of velocity points in TM method.

Parameters

value	the number of velocity points in TM method.

4.6.3.25 void CCFrame::updateModelPotentialEnergyCloseCollision (double *value*) [slot]

Indicates to the model the potential energy when close to a collision for TM method.

Parameters

value	the potential energy when close to a collision for TM method.

4.6.3.26 void CCFrame::updateModelPotentialEnergyStart (double *value*) [slot]

Indicates to the model the potential energy at the start of a trajectory for TM method.

Parameters

value	the potential energy at the start of a trajectory for TM method.

4.6.3.27 void CCFrame::updateModelShouldEHSSBeCalculated (bool *value*) `[slot]`

Indicates to the model if EHSS should be calculated.

Parameters

value	true if EHSS should be calculated, false otherwise.

4.6.3.28 void CCFrame::updateModelShouldPABeCalculated (bool *value*) `[slot]`

Indicates to the model if PA should be calculated.

Parameters

value	true if PA should be calculated, false otherwise.

4.6.3.29 void CCFrame::updateModelShouldTMBeCalculated (bool *value*) `[slot]`

Indicates to the model if TM should be calculated.

Parameters

value	true if TM should be calculated, false otherwise.

4.6.3.30 void CCFrame::updateModelTemperature (double *value*) `[slot]`

Indicates to the model the temperature for calculations.

Parameters

value	the temperature for calculations.

4.6.3.31 void CCFrame::updateModelTimeStepCloseCollision (double *value*) `[slot]`

Indicates to the model the time step between two points when close to a collision for TM method.

Parameters

value	the time step between two points when close to a collision for TM method.

4.6.3.32 void CCFrame::updateModelTimeStepStart (double *value*) [slot]

Indicates to the model the time step between two points at the start of a trajectory for TM method.

Parameters

value	the time step between two points at the start of a trajectory for TM method.

4.6.3.33 void CCFrame::updateResultList (QString *method,* int *index,* double *value*) [slot]

Updates a result in the geometries list.

Parameters

method	the method between EHSS, PA and TM.
index	the geometry index.
value	the value of the result.

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

- Collision-Code/gui/CCFrame.h
- Collision-Code/gui/CCFrame.cpp
- Collision-Code/gui/moc_CCFrame.cpp

4.7 ChargesReader Class Reference

```
#include <ChargesReader.h>
```

Inherited by ChgChargesReader.

Public Member Functions

- virtual ∼ChargesReader ()
- virtual std::string getFileName () const =0
- virtual void setFileName (std::string f)=0
- virtual std::vector< Molecule ∗ > ∗ loadResources (std::vector< Molecule ∗ > ∗molGeometries)=0

4.7.1 Constructor & Destructor Documentation

4.7.1.1 virtual ChargesReader::∼ChargesReader ( ) [inline], [virtual]

Destructor.

4.7.2 Member Function Documentation

4.7.2.1 virtual std::string ChargesReader::getFileName ( ) const `[pure virtual]`

Return name of file onload.

Returns

a string value giving the complete file name.

Implemented in ChgChargesReader.

4.7.2.2 virtual std::vector< **Molecule**∗>∗ ChargesReader::loadResources (std::vector< **Molecule** ∗ > ∗ *molGeometries*) `[pure virtual]`

Return all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implemented in ChgChargesReader.

4.7.2.3 virtual void ChargesReader::setFileName (std::string *f*) `[pure virtual]`

Change the actual file by a new one.

Implemented in ChgChargesReader.

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

- Collision-Code/reader/ChargesReader.h

4.8 ChgChargesReader Class Reference

```
#include <ChgChargesReader.h>
```

Inherits ChargesReader.

Public Member Functions

- ChgChargesReader (std::string filename)
- virtual ∼ChgChargesReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources (std::vector< Molecule ∗ > ∗molGeometries)

4.8.1 Constructor & Destructor Documentation

4.8.1.1 ChgChargesReader::ChgChargesReader (std::string *filename*)

Constructor.

Parameters

filename	the name of file to work with.

4.8.1.2 ChgChargesReader::∼ChgChargesReader ( ) `[virtual]`

Destructor.

4.8.2 Member Function Documentation

4.8.2.1 std::string ChgChargesReader::getFileName ( ) const `[inline]`, `[virtual]`

Return name of file onload.

Returns

a string value giving the complete file name.

Implements ChargesReader.

4.8.2.2 std::vector< **Molecule** ∗ > ∗ ChgChargesReader::loadResources (std::vector< **Molecule** ∗ > ∗ *molGeometries*) `[virtual]`

Return all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implements ChargesReader.

4.8.2.3 void ChgChargesReader::setFileName (std::string *filename*) `[virtual]`

Change the actual file by a new one.

Implements ChargesReader.

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

- Collision-Code/reader/ChgChargesReader.h
- Collision-Code/reader/ChgChargesReader.cpp

4.9 CmdView Class Reference

```
#include <CmdView.h>
```

Inherits Observable.

Inherited by StdCmdView.

Public Member Functions

- virtual ∼CmdView ()
- virtual std::vector< std::string > getInputFiles () const =0
- virtual std::vector< Molecule ∗ > getLoadedGeometries () const =0
- virtual bool willEHSSBeCalculated () const =0
- virtual bool willPABeCalculated () const =0
- virtual bool willTMBeCalculated () const =0
- virtual void shouldEHSSBeCalculated (bool b)=0
- virtual void shouldPABeCalculated (bool b)=0
- virtual void shouldTMBeCalculated (bool b)=0
- virtual void addInputFile (std::string fileName)=0
- virtual void removeInputFile (std::string fileName)=0
- virtual int loadInputFiles ()=0
- virtual std::string getChargeFile () const =0
- virtual void setChargeFile (std::string chargeFileName)=0
- virtual std::string getResultFormat () const =0
- virtual void setOutputFile (std::string outputFileName)=0
- virtual std::string getOutputFile () const =0
- virtual void saveResults ()=0
- virtual void launch ()=0

Additional Inherited Members

4.9.1 Constructor & Destructor Documentation

4.9.1.1 virtual CmdView::∼CmdView ( ) `[inline], [virtual]`

Releases all allocated resources.

4.9.2 Member Function Documentation

4.9.2.1 virtual void CmdView::addInputFile (std::string *fileName*) `[pure virtual]`

Indicates a new file to load.

Parameters

fileName	the name of the file to load.

Implemented in StdCmdView.

4.9.2.2 virtual std::string CmdView::getChargeFile ( ) const [pure virtual]

Returns

the name of the charge file.

Implemented in StdCmdView.

4.9.2.3 virtual std::vector<std::string> CmdView::getInputFiles ( ) const [pure virtual]

Returns

the list of input files.

Implemented in StdCmdView.

4.9.2.4 virtual std::vector<Molecule*> CmdView::getLoadedGeometries ( ) const [pure virtual]

Returns

all loaded geometries.

Implemented in StdCmdView.

4.9.2.5 virtual std::string CmdView::getOutputFile ( ) const [pure virtual]

Returns

the file name of the output file.

Implemented in StdCmdView.

4.9.2.6 virtual std::string CmdView::getResultFormat ( ) const [pure virtual]

Returns

a string representing the content of the calculations save.

Implemented in StdCmdView.

4.9.2.7 virtual void CmdView::launch ( ) [pure virtual]

Launches all the calculations, on all input files. Write the results in the output file.

Implemented in StdCmdView.

4.9.2.8 virtual int CmdView::loadInputFiles ( ) [pure virtual]

Loads all saved input files with charge file if present.

Returns

the number of geometries loaded.

Implemented in StdCmdView.

4.9.2.9 virtual void CmdView::removeInputFile (std::string *fileName*) [pure virtual]

Remove a file from the vector of the file to load.

Parameters

fileName	the name of the file to remove of the vector of the file to load.

Implemented in StdCmdView.

4.9.2.10 virtual void CmdView::saveResults ( ) [pure virtual]

Save the results in the output file. Delete previous content of the file.

Implemented in StdCmdView.

4.9.2.11 virtual void CmdView::setChargeFile (std::string *chargeFileName*) [pure virtual]

Indicates the charge file name.

Parameters

chargeFileName	the name of the charge file.

Implemented in StdCmdView.

4.9.2.12 virtual void CmdView::setOutputFile (std::string *outputFileName*) [pure virtual]

Sets the output file.

Parameters

outputFileName	the file name of the output file.

Implemented in StdCmdView.

4.9.2.13 virtual void CmdView::shouldEHSSBeCalculated (bool *b*) [pure virtual]

Indicates if yes or no, EHSS should be calculated.

Parameters

b	true if EHSS should be calculated, else otherwise.

Implemented in StdCmdView.

4.9.2.14 virtual void CmdView::shouldPABeCalculated (bool *b*) [pure virtual]

Indicates if yes or no, PA should be calculated.

Parameters

b	true if PA should be calculated, else otherwise.

Implemented in StdCmdView.

4.9.2.15 virtual void CmdView::shouldTMBeCalculated (bool *b*) `[pure virtual]`

Indicates if yes or no, TM should be calculated.

Parameters

b	true if TM should be calculated, else otherwise.

Implemented in StdCmdView.

4.9.2.16 virtual bool CmdView::willEHSSBeCalculated ( ) const `[pure virtual]`

Returns

true if EHSS should be calculated, else otherwise.

Implemented in StdCmdView.

4.9.2.17 virtual bool CmdView::willPABeCalculated ( ) const `[pure virtual]`

Returns

true if PA should be calculated, else otherwise.

Implemented in StdCmdView.

4.9.2.18 virtual bool CmdView::willTMBeCalculated ( ) const `[pure virtual]`

Returns

true if TM should be calculated, else otherwise.

Implemented in StdCmdView.

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

- Collision-Code/general/CmdView.h

4.10 ConsoleView Class Reference

```
#include <ConsoleView.h>
```

Inherits Observer.

Public Member Functions

- ConsoleView (int argc, char ∗const argv[ ])
- virtual ∼ConsoleView ()
- bool isThereAnError () const
- void update (ObservableEvent cond, Observable ∗obs)
- void launch ()

4.10.1 Constructor & Destructor Documentation

4.10.1.1 ConsoleView::ConsoleView (int *argc,* char ∗const *argv[ ]*)

Constructor. Take the main command line in parameter.

4.10.1.2 ConsoleView::∼ConsoleView ( ) `[virtual]`

Destructor.

4.10.2 Member Function Documentation

4.10.2.1 bool ConsoleView::isThereAnError ( ) const `[inline]`

Returns

true if there is an error with the command line, false otherwise.

4.10.2.2 void ConsoleView::launch ( )

Launches calculations.

4.10.2.3 void ConsoleView::update (ObservableEvent *cond,* Observable ∗ *obs*) `[virtual]`

Updates the observer.

Parameters

cond	the condition that triggered the notification.
obs	the Observable which triggered the call. May be null.

Implements Observer.

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

- Collision-Code/console/ConsoleView.h
- Collision-Code/console/ConsoleView.cpp

4.11 ExtractFactory Class Reference

```
#include <ExtractFactory.h>
```

Inherited by StdExtractFactory.

Public Member Functions

- virtual ∼ExtractFactory ()
- virtual FileReader ∗ getReader (std::string fileName)=0

4.11.1 Constructor & Destructor Documentation

4.11.1.1 virtual ExtractFactory::∼ExtractFactory ( ) `[inline], [virtual]`

Releases allocated resources.

4.11.2 Member Function Documentation

4.11.2.1 virtual **FileReader**∗ ExtractFactory::getReader (std::string *fileName*) `[pure virtual]`

Returns the FileReader necessary to read the file.

Parameters

fileName	the file name.

Returns

a pointer to a FileReader which can read the file, or null if the file can't be read.

Implemented in StdExtractFactory.

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

- Collision-Code/reader/ExtractFactory.h

4.12 ExtractResources Class Reference

```
#include <ExtractResources.h>
```

Inherited by StdExtractResources.

Public Member Functions

- virtual ∼ExtractResources ()
- virtual std::vector< Molecule ∗ > ∗ getGeometriesFromFile (std::string fileName)=0

4.12.1 Constructor & Destructor Documentation

4.12.1.1 virtual ExtractResources::∼ExtractResources ( ) [inline], [virtual]

Releases allocated resources.

4.12.2 Member Function Documentation

4.12.2.1 virtual std::vector<Molecule∗>∗ ExtractResources::getGeometriesFromFile (std::string *fileName*) [pure virtual]

Returns a vector of molecules loaded from the file.

Parameters

fileName	the name of the file in which are the molecules.

Returns

a pointer to a vector containing the loaded molecules, or null if the file can't be loaded.

Implemented in StdExtractResources.

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

- Collision-Code/reader/ExtractResources.h

4.13 FileReader Class Reference

```
#include <FileReader.h>
```

Inherited by LogFileReader, MfjFileReader, MolFileReader, PdbFileReader, and XyzFileReader.

Public Member Functions

- virtual ∼FileReader ()
- virtual std::string getFileName () const =0
- virtual void setFileName (std::string f)=0
- virtual std::vector< Molecule ∗ > ∗ loadResources ()=0

4.13.1 Constructor & Destructor Documentation

4.13.1.1 virtual FileReader::∼FileReader ( ) [inline], [virtual]

Destructor.

4.13.2 Member Function Documentation

4.13.2.1 virtual std::string FileReader::getFileName ( ) const [pure virtual]

Returns name of file on load.

Returns

a string value giving the complete file name.

Implemented in LogFileReader, MfjFileReader, MolFileReader, XyzFileReader, and PdbFileReader.

4.13.2.2 virtual std::vector<Molecule∗>∗ FileReader::loadResources ( ) [pure virtual]

Returns all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implemented in LogFileReader, MfjFileReader, MolFileReader, PdbFileReader, and XyzFileReader.

4.13.2.3 virtual void FileReader::setFileName (std::string *f*) [pure virtual]

Changes the actual file by a new one.

Implemented in LogFileReader, MfjFileReader, MolFileReader, PdbFileReader, and XyzFileReader.

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

- Collision-Code/reader/FileReader.h

4.14 FileWriter Class Reference

```
#include <FileWriter.h>
```

Inherited by StdFileWriter.

Public Member Functions

- virtual ∼FileWriter ()
- virtual void visitResult (Result ∗result)=0
- virtual void visitMean (Mean ∗mean)=0

4.14.1 Constructor & Destructor Documentation

4.14.1.1 virtual FileWriter::∼FileWriter ( ) `[inline], [virtual]`

Releases all allocated resources.

4.14.2 Member Function Documentation

4.14.2.1 virtual void FileWriter::visitMean (**Mean** ∗ ***mean***) `[pure virtual]`

Writes a mean of results in a file.

Implemented in StdFileWriter.

4.14.2.2 virtual void FileWriter::visitResult (**Result** ∗ ***result***) `[pure virtual]`

Writes a result in a file.

Implemented in StdFileWriter.

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

- Collision-Code/writer/FileWriter.h

4.15 GeometryCalculator Class Reference

```
#include <GeometryCalculator.h>
```

Inherited by StdGeometryCalculator.

Classes

- struct CalculationValues

Public Member Functions

- virtual ∼GeometryCalculator ()
- virtual bool willEHSSBeCalculated () const =0
- virtual bool willPABeCalculated () const =0
- virtual bool willTMBeCalculated () const =0
- virtual bool areCalculationsFinished (Molecule ∗mol) const =0
- virtual Result ∗ getResults (Molecule ∗mol) const =0
- virtual CalculationValues getCalculationValues () const =0
- virtual void saveCalculationValues ()=0
- virtual void shouldEHSSBeCalculated (bool b)=0
- virtual void shouldPABeCalculated (bool b)=0
- virtual void shouldTMBeCalculated (bool b)=0
- virtual void setGeometries (std::vector< Molecule ∗ > ∗geometries)=0
- virtual void takeObservers (std::vector< Observer ∗ > obs)=0
- virtual void launchCalculations ()=0

4.15.1 Constructor & Destructor Documentation

4.15.1.1 virtual GeometryCalculator::∼GeometryCalculator ( ) [inline], [virtual]

Releases all allocated resources.

4.15.2 Member Function Documentation

4.15.2.1 virtual bool GeometryCalculator::areCalculationsFinished (Molecule ∗ *mol*) const [pure virtual]

Indicates if calculations are finished for the molecule.

Parameters

mol	the molecule.

Returns

true if calculations are finished for the molecule, false otherwise.

Implemented in StdGeometryCalculator.

4.15.2.2 virtual CalculationValues GeometryCalculator::getCalculationValues ( ) const [pure virtual]

Returns the values used for the calculations.

Returns

the values used for the calculations.

Implemented in StdGeometryCalculator.

4.15.2.3 virtual Result∗ GeometryCalculator::getResults (Molecule ∗ *mol*) const [pure virtual]

Returns the results of CCS calculation of a molecule.

Parameters

mol	the molecule.

Returns

the results for the molecule.

Implemented in StdGeometryCalculator.

4.15.2.4 virtual void GeometryCalculator::launchCalculations ( ) `[pure virtual]`

Launches all the calculations, on all geometries.

Implemented in StdGeometryCalculator.

4.15.2.5 virtual void GeometryCalculator::saveCalculationValues ( ) `[pure virtual]`

Forces the GeometryCalculator to save the calculation values from GlobalParameters at this moment.

Implemented in StdGeometryCalculator.

4.15.2.6 virtual void GeometryCalculator::setGeometries (std::vector< Molecule * > * *geometries*) `[pure virtual]`

Sets a vector of molecules (geometries) for CCS calculation.

Parameters

geometries	a vector of geometries.

Implemented in StdGeometryCalculator.

4.15.2.7 virtual void GeometryCalculator::shouldEHSSBeCalculated (bool *b*) `[pure virtual]`

Indicates if yes or no, EHSS should be calculated.

Parameters

b	true if EHSS should be calculated, else otherwise.

Implemented in StdGeometryCalculator.

4.15.2.8 virtual void GeometryCalculator::shouldPABeCalculated (bool *b*) `[pure virtual]`

Indicates if yes or no, PA should be calculated.

Parameters

b	true if PA should be calculated, else otherwise.

Implemented in StdGeometryCalculator.

4.15.2.9 virtual void GeometryCalculator::shouldTMBeCalculated (bool *b*) `[pure virtual]`

Indicates if yes or no, TM should be calculated.

Parameters

b	true if TM should be calculated, else otherwise.

Implemented in StdGeometryCalculator.

4.15.2.10 virtual void GeometryCalculator::takeObservers (std::vector< Observer * > *obs*) `[pure virtual]`

Indicates that these Observers want to be notified about the calculations.

Parameters

obs	the observers list.

Implemented in StdGeometryCalculator.

4.15.2.11 virtual bool GeometryCalculator::willEHSSBeCalculated ( ) const `[pure virtual]`

Returns

true if EHSS will be calculated, false otherwise.

Implemented in StdGeometryCalculator.

4.15.2.12 virtual bool GeometryCalculator::willPABeCalculated ( ) const `[pure virtual]`

Returns

true if PA will be calculated, false otherwise.

Implemented in StdGeometryCalculator.

4.15.2.13 virtual bool GeometryCalculator::willTMBeCalculated ( ) const `[pure virtual]`

Returns

true if TM will be calculated, false otherwise.

Implemented in StdGeometryCalculator.

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

- Collision-Code/general/GeometryCalculator.h

4.16 GlobalParameters Class Reference

`#include <GlobalParameters.h>`

Public Member Functions

- virtual ∼GlobalParameters ()
- double getTemperature () const
- double getPotentialEnergyStart () const
- double getTimeStepStart () const
- double getPotentialEnergyCloseCollision () const
- double getTimeStepCloseCollision () const
- int getNumberCompleteCycles () const
- int getNumberVelocityPoints () const
- int getNbPointsMCIntegrationTM () const
- int getNbPointsMCIntegrationEHSSPA () const
- double getEnergyConservationThreshold () const
- void setTemperature (double t)
- void setPotentialEnergyStart (double pES)
- void setTimeStepStart (double dt)
- void setPotentialEnergyCloseCollision (double pECC)
- void setTimeStepCloseCollision (double dt)
- void setNumberCompleteCycles (int n)
- void setNumberVelocityPoints (int n)
- void setNbPointsMCIntegrationTM (int n)
- void setNbPointsMCIntegrationEHSSPA (int n)
- void setEnergyConservationThreshold (double eCT)

Static Public Member Functions

- static GlobalParameters ∗ getInstance ()

4.16.1 Constructor & Destructor Documentation

4.16.1.1 GlobalParameters::∼GlobalParameters ( ) `[virtual]`

Destroys all allocated resources.

4.16.2 Member Function Documentation

4.16.2.1 double GlobalParameters::getEnergyConservationThreshold ( ) const [inline]

Returns the energy conservation threshold, in percent.

Returns

the energy conservation threshold, in percent.

4.16.2.2 static GlobalParameters∗ GlobalParameters::getInstance ( ) [inline], [static]

Returns

an instance of GlobalParameters to work with.

4.16.2.3 int GlobalParameters::getNbPointsMCIntegrationEHSSPA ( ) const [inline]

Returns the number of points in Monte-Carlo integrations for EHSS and PA methods.

Returns

the number of points in Monte-Carlo integrations for EHSS and PA methods.

4.16.2.4 int GlobalParameters::getNbPointsMCIntegrationTM ( ) const [inline]

Returns the number of points in Monte-Carlo integrations of impact parameter and orientation for TM method.

Returns

the number of points in Monte-Carlo integrations of impact parameter and orientation for TM method.

4.16.2.5 int GlobalParameters::getNumberCompleteCycles ( ) const [inline]

Returns the number of complete cycles for TM method.

Returns

the number of complete cycles for TM method.

4.16.2.6 int GlobalParameters::getNumberVelocityPoints ( ) const [inline]

Returns the number of points in velocity integration.

Returns

the number of points in velocity integration.

4.16.2.7 double GlobalParameters::getPotentialEnergyCloseCollision ( ) const [inline]

Returns the potential energy when close to a collision.

Returns

the potential energy when close to a collision.

4.16.2.8 double GlobalParameters::getPotentialEnergyStart ( ) const [inline]

Returns the potential energy at the start of a trajectory.

Returns

the potential energy at the start of a trajectory.

4.16.2.9 double GlobalParameters::getTemperature ( ) const [inline]

Returns the temperature.

Returns

the temperature.

4.16.2.10 double GlobalParameters::getTimeStepCloseCollision ( ) const [inline]

Returns the time step when close to a collision.

Returns

the time step when close to a collision.

4.16.2.11 double GlobalParameters::getTimeStepStart ( ) const [inline]

Returns the time step at the start of a trajectory.

Returns

the time step at the start of a trajectory.

4.16.2.12 void GlobalParameters::setEnergyConservationThreshold (double *eCT*) [inline]

Sets the energy conservation threshold, in percent, to eCT.

Parameters

eCT	the new energy conservation threshold, in percent.

4.16.2.13 void GlobalParameters::setNbPointsMCIntegrationEHSSPA (int *n*) [inline]

Sets the number of points in Monte-Carlo integrations for EHSS and PA methods to n.

Parameters

n	the new number of points in Monte-Carlo integrations for EHSS and PA methods.

4.16.2.14 void GlobalParameters::setNbPointsMCIntegrationTM (int *n*) [inline]

Sets the number of points in Monte-Carlo integrations of impact parameter and orientation for TM method to n.

Parameters

n	the new number of points in Monte-Carlo integrations of impact parameter and orientation for TM method.

4.16.2.15 void GlobalParameters::setNumberCompleteCycles (int *n*) [inline]

Sets the number of complete cycles for TM method to n.

Parameters

n	the new number of complete cycles for TM method.

4.16.2.16 void GlobalParameters::setNumberVelocityPoints (int *n*) [inline]

Sets the number of points in velocity integration to n.

Parameters

n	the new number of points in velocity integration.

4.16.2.17 void GlobalParameters::setPotentialEnergyCloseCollision (double *pECC*) [inline]

Sets the potential energy when close to a collision to pECC.

Parameters

pECC	the new potential energy when close to a collision.

4.16.2.18 void GlobalParameters::setPotentialEnergyStart (double *pES*) [inline]

Sets the potential energy at the start of a trajectory to pES.

Parameters

pES	the new potential energy at the start of a trajectory.

4.16.2.19 void GlobalParameters::setTemperature (double *t*) [inline]

Sets the temperature to t.

Parameters

t	the new temperature.

4.16.2.20 void GlobalParameters::setTimeStepCloseCollision (double *dt*) [inline]

Sets the time step when close to a collision to dt.

Parameters

dt	the new time step when close to a collision.

4.16.2.21 void GlobalParameters::setTimeStepStart (double *dt*) [inline]

Sets the time step at the start of a trajectory to dt.

Parameters

dt	the new time step at the start of a trajectory.

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

- Collision-Code/general/GlobalParameters.h
- Collision-Code/general/GlobalParameters.cpp

4.17 LogFileReader Class Reference

```
#include <LogFileReader.h>
```

Inherits FileReader.

Public Member Functions

- LogFileReader (std::string filename)
- virtual ∼LogFileReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources ()

4.17.1 Constructor & Destructor Documentation

4.17.1.1 LogFileReader::LogFileReader (std::string *filename*)

MolFileReader's cosntructor.

Parameters

filename	the name of file to work with.

4.17.1.2 LogFileReader::∼LogFileReader ( ) `[virtual]`

Destructor.

4.17.2 Member Function Documentation

4.17.2.1 std::string LogFileReader::getFileName ( ) const `[inline]`, `[virtual]`

Returns name of file on load.

Returns

a string value giving the complete file name.

Implements FileReader.

4.17.2.2 std::vector< Molecule ∗ > ∗ LogFileReader::loadResources ( ) `[virtual]`

Returns all molecule from the actual file.

Returns

a pointer to a molecule list extract from file.

Implements FileReader.

4.17.2.3 void LogFileReader::setFileName (std::string *filename*) [virtual]

Changes the actual file by a new one.

Implements FileReader.

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

- Collision-Code/reader/LogFileReader.h
- Collision-Code/reader/LogFileReader.cpp

4.18 MathLib Class Reference

```
#include <MathLib.h>
```

Inherited by StdMathLib.

Public Member Functions

- virtual ∼MathLib ()
- virtual void rotate (Molecule ∗mol, double angleX, double angleY, double angleZ)=0
- virtual void rotate (const std::vector< Vector3D > &initPos, std::vector< Vector3D > &pos, double angleX, double angleY, double angleZ)=0
- virtual void randomRotation (Molecule ∗mol)=0
- virtual void randomRotation (const std::vector< Vector3D > &initPos, std::vector< Vector3D > &pos)=0
- virtual Vector3D calculateMassCenter (const Molecule &mol)=0
- virtual Atom ∗ findFarthestAtom (const Molecule &mol)=0
- virtual double monteCarloIntegration (double(∗f)(double), double minLimit, double maxLimit, int n)=0

4.18.1 Constructor & Destructor Documentation

4.18.1.1 virtual MathLib::∼MathLib ( ) [inline], [virtual]

Destructor.

4.18.2 Member Function Documentation

4.18.2.1 virtual Vector3D MathLib::calculateMassCenter (const Molecule & *mol*) [pure virtual]

Calculates the center of mass of a molecule.

Parameters

mol	the molecule.

Returns

the coordinates of the center of mass of the molecule mol.

Implemented in StdMathLib.

4.18.2.2 virtual Atom∗ MathLib::findFarthestAtom (const Molecule & *mol*) [pure virtual]

Finds the atom the farthest of the center of mass.

Parameters

mol	the molecule.

Returns

a pointer to the atom which is the farthest of the center of mass of mol.

Implemented in StdMathLib.

4.18.2.3 virtual double MathLib::monteCarloIntegration (double(∗)(double) *f*, double *minLimit*, double *maxLimit*, int *n*) [pure virtual]

Implementation of the Monte-Carlo method for calculating integrals.

Parameters

f	the function to integrate.
minLimit	the lower limit of the integral.
maxLimit	the upper limit of the integral.
n	the number of points generate to calculate the integral. More points increase the result precision.

Returns

the result of the integration.

Implemented in StdMathLib.

4.18.2.4 virtual void MathLib::randomRotation (Molecule ∗ *mol*) [pure virtual]

Rotates the molecule by random angles on each axis.

Parameters

mol	the molecule to rotate.

Implemented in StdMathLib.

4.18.2.5 virtual void MathLib::randomRotation (const std::vector< Vector3D > & *initPos*, std::vector< Vector3D > & *pos*) [pure virtual]

Rotates the positions by random angles on each axis.

Parameters

pos	the positions to rotate.

Implemented in StdMathLib.

4.18.2.6 virtual void MathLib::rotate (Molecule ∗ *mol*, double *angleX*, double *angleY*, double *angleZ*) [pure virtual]

Rotates the molecule by angles specified on each axis.

Parameters

mol	the molecule to rotate.
angleX	the angle of rotation one the X axis.
angleY	the angle of rotation one the Y axis.
angleZ	the angle of rotation one the Z axis.

Implemented in StdMathLib.

4.18.2.7 virtual void MathLib::rotate (const std::vector< Vector3D > & *initPos*, std::vector< Vector3D > & *pos*, double *angleX*, double *angleY*, double *angleZ*) [pure virtual]

Rotates the position by angles specified on each axis.

Parameters

pos	the positions to rotate.
angleX	the angle of rotation one the X axis.
angleY	the angle of rotation one the Y axis.
angleZ	the angle of rotation one the Z axis.

Implemented in StdMathLib.

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

- Collision-Code/math/MathLib.h

4.19 Mean Class Reference

```
#include <Mean.h>
```

Inherited by StdMean.

Public Member Functions

- virtual ∼Mean ()
- virtual double getMeanEHSS ()=0
- virtual double getMeanPA ()=0
- virtual double getMeanTM ()=0
- virtual double getMeanStructAsymParam ()=0
- virtual double getMeanStandardDeviation ()=0
- virtual int getMeanNumberOfFailedTrajectories ()=0
- virtual bool isEHSSSaved ()=0
- virtual bool isPASaved ()=0
- virtual bool isTMSaved ()=0
- virtual bool isEHSSPrintable ()=0
- virtual bool isPAPrintable ()=0
- virtual bool isTMPrintable ()=0
- virtual void addResult (Result ∗r)=0
- virtual void accept (class FileWriter &fileWriter)=0

4.19.1 Constructor & Destructor Documentation

4.19.1.1 virtual Mean::∼Mean ( ) `[inline], [virtual]`

Destructor.

4.19.2 Member Function Documentation

4.19.2.1 virtual void Mean::accept (class **FileWriter** & *fileWriter*) `[pure virtual]`

Write the mean object via the FileWriter.

Implemented in StdMean.

4.19.2.2 virtual void Mean::addResult (**Result** ∗ *r*) `[pure virtual]`

Add a result to the results used to calculate the means.

Parameters

r	the result to add to the list.

Implemented in StdMean.

4.19.2.3 virtual double Mean::getMeanEHSS ( ) `[pure virtual]`

Returns the mean of EHSS results.

Returns

the mean of EHSS results, or 0 is !isEHSSSaved().

Implemented in StdMean.

4.19.2.4 virtual int Mean::getMeanNumberOfFailedTrajectories ( ) [pure virtual]

Returns the mean of the numbers of failed trajectories.

Returns

the mean of the numbers of failed trajectories.

Implemented in StdMean.

4.19.2.5 virtual double Mean::getMeanPA ( ) [pure virtual]

Returns the mean of PA results.

Returns

the mean of PA results, or 0 is !isPASaved().

Implemented in StdMean.

4.19.2.6 virtual double Mean::getMeanStandardDeviation ( ) [pure virtual]

Returns the mean of the standard deviations.

Returns

the mean of the standard deviation.

Implemented in StdMean.

4.19.2.7 virtual double Mean::getMeanStructAsymParam ( ) [pure virtual]

Returns the mean of the structural asymmetry parameters.

Returns

the mean of the structural asymmetry parameters.

Implemented in StdMean.

4.19.2.8 virtual double Mean::getMeanTM ( ) [pure virtual]

Returns the mean of TM results.

Returns

the mean of TM results, or 0 is !isTMSaved().

Implemented in StdMean.

4.19.2.9 virtual bool Mean::isEHSSPrintable ( ) [pure virtual]

Indicates if EHSS needs to be printed.

Returns

true if EHSS needs to be printed, false otherwise.

Implemented in StdMean.

4.19.2.10 virtual bool Mean::isEHSSSaved ( ) [pure virtual]

Returns

true if EHSS was saved, false in the other case.

Implemented in StdMean.

4.19.2.11 virtual bool Mean::isPAPrintable ( ) [pure virtual]

Indicates if PA needs to be printed.

Returns

true if PA needs to be printed, false otherwise.

Implemented in StdMean.

4.19.2.12 virtual bool Mean::isPASaved ( ) [pure virtual]

Returns

true if PA was saved, false in the other case.

Implemented in StdMean.

4.19.2.13 virtual bool Mean::isTMPrintable ( ) `[pure virtual]`

Indicates if TM needs to be printed.

Returns

true if TM needs to be printed, false otherwise.

Implemented in StdMean.

4.19.2.14 virtual bool Mean::isTMSaved ( ) `[pure virtual]`

Returns

true if TM was saved, false in the other case.

Implemented in StdMean.

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

- Collision-Code/math/Mean.h

4.20 MfjFileReader Class Reference

`#include <MfjFileReader.h>`

Inherits FileReader.

Public Member Functions

- MfjFileReader (std::string filename)
- virtual ∼MfjFileReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources ()

4.20.1 Constructor & Destructor Documentation

4.20.1.1 MfjFileReader::MfjFileReader (std::string *filename*)

MfjFileReader's constructor.

Parameters

filename	the name of file to work with.

4.20.1.2 MfjFileReader::∼MfjFileReader ( ) [virtual]

Destructor.

4.20.2 Member Function Documentation

4.20.2.1 std::string MfjFileReader::getFileName ( ) const [inline], [virtual]

Returns name of file on load.

Returns

a string value giving the complete file name.

Implements FileReader.

4.20.2.2 std::vector< Molecule ∗ > ∗ MfjFileReader::loadResources ( ) [virtual]

Returns all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implements FileReader.

4.20.2.3 void MfjFileReader::setFileName (std::string *filename*) [virtual]

Change the actual file by a new one.

Implements FileReader.

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

- Collision-Code/reader/MfjFileReader.h
- Collision-Code/reader/MfjFileReader.cpp

4.21 Molecule Class Reference

```
#include <Molecule.h>
```

Inherited by StdMolecule.

Public Member Functions

- virtual ∼Molecule ()
- virtual std::string getName ()=0
- virtual unsigned int getAtomNumber () const =0
- virtual double getTotalMass () const =0
- virtual std::vector< Atom ∗ > ∗ getAllAtoms () const =0
- virtual Atom ∗ getAtom (const Vector3D &c) const =0
- virtual void toInitialPosition ()=0
- virtual void setName (std::string n)=0
- virtual void addAtom (Atom ∗a)=0
- virtual void deleteAtom (Atom ∗a)=0
- virtual void deleteAtom (const Vector3D &c)=0

4.21.1 Constructor & Destructor Documentation

4.21.1.1 virtual Molecule::∼Molecule ( ) `[inline], [virtual]`

Release allocates resources.

4.21.2 Member Function Documentation

4.21.2.1 virtual void Molecule::addAtom (Atom ∗ *a*) `[pure virtual]`

Adds an atom on the molecule.

Parameters

a	a pointer on an atom.

Implemented in StdMolecule.

4.21.2.2 virtual void Molecule::deleteAtom (Atom ∗ *a*) `[pure virtual]`

Deletes the specified atom.

Parameters

a	a pointer on an atom.

Implemented in StdMolecule.

4.21.2.3 virtual void Molecule::deleteAtom (const Vector3D & *c*) `[pure virtual]`

Deletes the atom at specified position.

Parameters

c	a coordinate.

Implemented in StdMolecule.

4.21.2.4 virtual std::vector<Atom*>* Molecule::getAllAtoms ( ) const [pure virtual]

Returns

a pointer on atom collection.

Implemented in StdMolecule.

4.21.2.5 virtual Atom* Molecule::getAtom (const Vector3D & *c*) const [pure virtual]

Parameters

c	a coordinate

Returns

the atom from the specified position.

Implemented in StdMolecule.

4.21.2.6 virtual unsigned int Molecule::getAtomNumber ( ) const [pure virtual]

Returns

the total number of atom forming molecule composition.

Implemented in StdMolecule.

4.21.2.7 virtual std::string Molecule::getName ( ) [pure virtual]

Returns

the name of molecule.

Implemented in StdMolecule.

4.21.2.8 virtual double Molecule::getTotalMass ( ) const [pure virtual]

Returns

the mass of the molecule.

Implemented in StdMolecule.

4.21.2.9 virtual void Molecule::setName (std::string *n*) [pure virtual]

Replaces the current name of molecule by a new one.

Parameters

n	a string value.

Implemented in StdMolecule.

4.21.2.10 virtual void Molecule::toInitialPosition ( ) `[pure virtual]`

Replaces the molecule at its initial position.

Implemented in StdMolecule.

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

- Collision-Code/molecule/Molecule.h

4.22 MolFileReader Class Reference

```
#include <MolFileReader.h>
```

Inherits FileReader.

Public Member Functions

- MolFileReader (std::string filename)
- virtual ∼MolFileReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources ()

4.22.1 Constructor & Destructor Documentation

4.22.1.1 MolFileReader::MolFileReader (std::string *filename*)

MolFileReader's constructor.

Parameters

filename	the name of file to work with.

4.22.1.2 MolFileReader::∼MolFileReader ( ) `[virtual]`

Destructor.

4.22.2 Member Function Documentation

4.22.2.1 std::string MolFileReader::getFileName ( ) const [inline], [virtual]

Returns name of file on load.

Returns

a string value giving the complete file name.

Implements FileReader.

4.22.2.2 std::vector< **Molecule** ∗ > ∗ MolFileReader::loadResources ( ) [virtual]

Returns all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implements FileReader.

4.22.2.3 void MolFileReader::setFileName (std::string *filename*) [virtual]

Changes the actual file by a new one.

Implements FileReader.

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

- Collision-Code/reader/MolFileReader.h
- Collision-Code/reader/MolFileReader.cpp

4.23 MonoThreadCalculationOperator Class Reference

```
#include <MonoThreadCalculationOperator.h>
```

Inherits StdCalculationOperator.

Public Member Functions

- MonoThreadCalculationOperator (CalculationState ∗calculationState, Molecule ∗mol, double temperature, double potentialEnergyStart, double timeStepStart, double potentialEnergyCloseCollision, double time←StepCloseCollision, double numberCyclesTM, double numberPointsVelocity, double numberPointsMC←IntegrationTM, double energyConservationThreshold, double numberPointsMCIntegrationEHSSPA)
- virtual ∼MonoThreadCalculationOperator ()

Protected Member Functions

- void calculateTM ()

Additional Inherited Members

4.23.1 Constructor & Destructor Documentation

4.23.1.1 MonoThreadCalculationOperator::MonoThreadCalculationOperator (**CalculationState** ∗ *calculationState,* **Molecule** ∗ *mol,* double *temperature,* double *potentialEnergyStart,* double *timeStepStart,* double *potentialEnergyCloseCollision,* double *timeStepCloseCollision,* double *numberCyclesTM,* double *numberPointsVelocity,* double *numberPointsMCIntegrationTM,* double *energyConservationThreshold,* double *numberPointsMCIntegrationEHSSPA*)

Constructs a mono thread CalculationOperator.

4.23.1.2 MonoThreadCalculationOperator::∼MonoThreadCalculationOperator ( ) `[virtual]`

Destructs allocated resources.

4.23.2 Member Function Documentation

4.23.2.1 void MonoThreadCalculationOperator::calculateTM ( ) `[protected]`, `[virtual]`

Calculates TM and put the results in m_result attribute.

Calculate TM and put the results in m_result attribute.

Implements StdCalculationOperator.

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

- Collision-Code/math/MonoThreadCalculationOperator.h
- Collision-Code/math/MonoThreadCalculationOperator.cpp

4.24 MultiThreadCalculationOperator Class Reference

```
#include <MultiThreadCalculationOperator.h>
```

Inherits StdCalculationOperator.

Public Member Functions

- MultiThreadCalculationOperator (CalculationState ∗calculationState, Molecule ∗mol, int maximalNumber←Threads, double temperature, double potentialEnergyStart, double timeStepStart, double potentialEnergy←CloseCollision, double timeStepCloseCollision, double numberCyclesTM, double numberPointsVelocity, double numberPointsMCIntegrationTM, double energyConservationThreshold, double numberPointsMC←IntegrationEHSSPA)
- virtual ∼MultiThreadCalculationOperator ()

Protected Member Functions

- void calculateTM ()

Additional Inherited Members

4.24.1 Constructor & Destructor Documentation

4.24.1.1 MultiThreadCalculationOperator::MultiThreadCalculationOperator (**CalculationState** ∗ *calculationState,* **Molecule** ∗ *mol,* int *maximalNumberThreads,* double *temperature,* double *potentialEnergyStart,* double *timeStepStart,* double *potentialEnergyCloseCollision,* double *timeStepCloseCollision,* double *numberCyclesTM,* double *numberPointsVelocity,* double *numberPointsMCIntegrationTM,* double *energyConservationThreshold,* double *numberPointsMCIntegrationEHSSPA*)

Constructs a mono thread CalculationOperator.

4.24.1.2 MultiThreadCalculationOperator::∼MultiThreadCalculationOperator ( ) `[virtual]`

Destructs allocated resources.

4.24.2 Member Function Documentation

4.24.2.1 void MultiThreadCalculationOperator::calculateTM ( ) `[protected]`, `[virtual]`

Calculates TM and put the results in m_result attribute.

Calculate TM and put the results in m_result attribute.

Implements StdCalculationOperator.

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

- Collision-Code/math/MultiThreadCalculationOperator.h
- Collision-Code/math/MultiThreadCalculationOperator.cpp

4.25 Observable Class Reference

```
#include <Observable.h>
```

Inherited by CalculationState, and CmdView.

Public Member Functions

- Observable ()
- virtual ∼Observable ()
- void addObserver (Observer ∗obs)
- void removeObserver (Observer ∗obs)
- void notifyObservers (ObservableEvent cond)

Protected Attributes

- std::vector< Observer ∗ > m_observers

4.25.1 Constructor & Destructor Documentation

4.25.1.1 Observable::Observable ( )

Constructor.

4.25.1.2 Observable::∼Observable ( ) [virtual]

Destructor.

4.25.2 Member Function Documentation

4.25.2.1 void Observable::addObserver (**Observer** ∗ ***obs***)

Add an observer to the list of observers to notify.

Parameters

obs	the observer to add to the list.

4.25.2.2 void Observable::notifyObservers (**ObservableEvent** ***cond***)

Notify all observers of the condition cond.

Parameters

cond	the condition for notify observers.

4.25.2.3 void Observable::removeObserver (**Observer** ∗ ***obs***)

Remove an observer from the list of observers to notify.

Parameters

obs	the observer to remove from the list.

4.25.3 Member Data Documentation

4.25.3.1 std::vector<Observer∗> Observable::m_observers [protected]

A list containing all observers.

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

- Collision-Code/observer/Observable.h
- Collision-Code/observer/Observable.cpp

4.26 Observer Class Reference

```
#include <Observer.h>
```

Inherited by CCFrame, and ConsoleView.

Public Member Functions

- Observer ()
- virtual ∼Observer ()
- virtual void update (ObservableEvent cond, Observable ∗obs)=0

4.26.1 Constructor & Destructor Documentation

4.26.1.1 Observer::Observer ( )

Constructor.

4.26.1.2 Observer::∼Observer ( ) [virtual]

Destructor.

4.26.2 Member Function Documentation

4.26.2.1 virtual void Observer::update (ObservableEvent *cond*, Observable ∗ *obs*) [pure virtual]

Updates the observer.

Parameters

cond	the condition that triggered the notification.
obs	the Observable which triggered the call. May be null.

Implemented in CCFrame, and ConsoleView.

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

- Collision-Code/observer/Observer.h
- Collision-Code/observer/Observer.cpp

4.27 PdbFileReader Class Reference

`#include <PdbFileReader.h>`

Inherits FileReader.

Public Member Functions

- PdbFileReader (std::string filename)
- virtual ∼PdbFileReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources ()

4.27.1 Constructor & Destructor Documentation

4.27.1.1 PdbFileReader::PdbFileReader (std::string *filename*)

Constructs a new PdbFileReader.

4.27.1.2 PdbFileReader::∼PdbFileReader ( ) `[virtual]`

Frees all resources.

4.27.2 Member Function Documentation

4.27.2.1 std::string PdbFileReader::getFileName ( ) const `[inline]`, `[virtual]`

Returns name of file on load.

Returns

a string value giving the complete file name.

Implements FileReader.

4.27.2.2 std::vector< Molecule ∗ > ∗ PdbFileReader::loadResources ( ) `[virtual]`

Returns all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implements FileReader.

4.27.2.3 void PdbFileReader::setFileName (std::string *filename*) `[virtual]`

Changes the actual file by a new one.

Parameters

filename	the name of the file to load.

Implements FileReader.

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

- Collision-Code/reader/PdbFileReader.h
- Collision-Code/reader/PdbFileReader.cpp

4.28 qt_meta_stringdata_CCFrame_t Struct Reference

Public Attributes

- QByteArrayData data [40]
- char stringdata0 [800]

4.28.1 Member Data Documentation

4.28.1.1 QByteArrayData qt_meta_stringdata_CCFrame_t::data[40]

4.28.1.2 char qt_meta_stringdata_CCFrame_t::stringdata0[800]

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

- Collision-Code/gui/moc_CCFrame.cpp

4.29 qt_meta_stringdata_Worker_t Struct Reference

Public Attributes

- QByteArrayData data [6]
- char stringdata0 [40]

4.29.1 Member Data Documentation

4.29.1.1 QByteArrayData qt_meta_stringdata_Worker_t::data[6]

4.29.1.2 char qt_meta_stringdata_Worker_t::stringdata0[40]

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

- Collision-Code/gui/moc_CCFrame.cpp

4.30 RandomGenerator Class Reference

```
#include <RandomGenerator.h>
```

Public Member Functions

- virtual ~RandomGenerator ()
- double getRandomNumber ()

Static Public Member Functions

- static RandomGenerator *const getInstance ()

4.30.1 Constructor & Destructor Documentation

4.30.1.1 RandomGenerator::~RandomGenerator ( ) [virtual]

Destructor.

4.30.2 Member Function Documentation

4.30.2.1 static RandomGenerator* const RandomGenerator::getInstance ( ) [inline], [static]

Returns an instance of RandomGenerator.

Returns

a pointer to a RandomGenerator.

4.30.2.2 double RandomGenerator::getRandomNumber ( ) [inline]

Returns a random number between 0 and 1. Random generation is uniform.

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

- Collision-Code/math/RandomGenerator.h
- Collision-Code/math/RandomGenerator.cpp

4.31 Result Class Reference

```
#include <Result.h>
```

Inherited by StdResult.

Public Member Functions

- virtual ∼Result ()
- virtual Molecule ∗ getAssociateMolecule ()=0
- virtual double getEHSS ()=0
- virtual double getPA ()=0
- virtual double getTM ()=0
- virtual double getStructAsymParam ()=0
- virtual double getStandardDeviation ()=0
- virtual int getNumberOfFailedTrajectories ()=0
- virtual bool isEHSSSaved ()=0
- virtual bool isPASaved ()=0
- virtual bool isTMSaved ()=0
- virtual void setEHSS (double ehss)=0
- virtual void setPA (double pa)=0
- virtual void setTM (double tm)=0
- virtual void setStructAsymParam (double asymParam)=0
- virtual void setStandardDeviation (double stdDeviation)=0
- virtual void setNumberOfFailedTrajectories (int nbFailedTraject)=0
- virtual void EHSSNeedsToBePrinted (bool b)=0
- virtual void PANeedsToBePrinted (bool b)=0
- virtual void TMNeedsToBePrinted (bool b)=0
- virtual bool isEHSSPrintable ()=0
- virtual bool isPAPrintable ()=0
- virtual bool isTMPrintable ()=0
- virtual void accept (class FileWriter &fileWriter)=0

4.31.1 Detailed Description

Interface describing how to save results.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 virtual Result::∼Result ( ) `[inline], [virtual]`

4.31.3 Member Function Documentation

4.31.3.1 virtual void Result::accept (class FileWriter & *fileWriter*) `[pure virtual]`

Write the result via the FileWriter.

Implemented in StdResult.

4.31.3.2 virtual void Result::EHSSNeedsToBePrinted (bool *b*) `[pure virtual]`

Indicates if EHSS needs to be printed.

Parameters

true	if EHSS needs to be printed, false otherwise.

Implemented in StdResult.

4.31.3.3 virtual Molecule∗ Result::getAssociateMolecule ( ) [pure virtual]

Returns the molecule saves with these results.

Returns

the molecule saves with these results.

Implemented in StdResult.

4.31.3.4 virtual double Result::getEHSS ( ) [pure virtual]

Returns the result for EHSS.

Returns

the EHSS result, or 0 is !isEHSSSaved().

Implemented in StdResult.

4.31.3.5 virtual int Result::getNumberOfFailedTrajectories ( ) [pure virtual]

Returns the number of failed trajectories.

Returns

the number of failed trajectories.

Implemented in StdResult.

4.31.3.6 virtual double Result::getPA ( ) [pure virtual]

Returns the result for PA.

Returns

the PA result, or 0 is !isPASaved().

Implemented in StdResult.

4.31.3.7 virtual double Result::getStandardDeviation ( ) [pure virtual]

Returns the standard deviation.

Returns

the standard deviation.

Implemented in StdResult.

4.31.3.8 virtual double Result::getStructAsymParam ( ) [pure virtual]

Returns the structural asymmetry parameter.

Returns

the structural asymmetry parameter.

Implemented in StdResult.

4.31.3.9 virtual double Result::getTM ( ) [pure virtual]

Returns the result for TM.

Returns

the TM result, or 0 is !isTMSaved().

Implemented in StdResult.

4.31.3.10 virtual bool Result::isEHSSPrintable ( ) [pure virtual]

Indicates if EHSS needs to be printed.

Returns

true if EHSS needs to be printed, false otherwise.

Implemented in StdResult.

4.31.3.11 virtual bool Result::isEHSSSaved ( ) [pure virtual]

Returns

true if EHSS was saved, false in the other case.

Implemented in StdResult.

4.31.3.12 virtual bool Result::isPAPrintable ( ) `[pure virtual]`

Indicates if PA needs to be printed.

Returns

true if PA needs to be printed, false otherwise.

Implemented in StdResult.

4.31.3.13 virtual bool Result::isPASaved ( ) `[pure virtual]`

Returns

true if PA was saved, false in the other case.

Implemented in StdResult.

4.31.3.14 virtual bool Result::isTMPrintable ( ) `[pure virtual]`

Indicates if TM needs to be printed.

Returns

true if TM needs to be printed, false otherwise.

Implemented in StdResult.

4.31.3.15 virtual bool Result::isTMSaved ( ) `[pure virtual]`

Returns

true if TM was saved, false in the other case.

Implemented in StdResult.

4.31.3.16 virtual void Result::PANeedsToBePrinted (bool *b*) `[pure virtual]`

Indicates if PA needs to be printed.

Parameters

true	if PA needs to be printed, false otherwise.

Implemented in StdResult.

4.31.3.17 virtual void Result::setEHSS (double *ehss*) `[pure virtual]`

Sets the value of the EHSS result to ehss. Sets isEHSSSaved() to true.

Parameters

ehss	the value of the EHSS result.

Implemented in StdResult.

4.31.3.18 virtual void Result::setNumberOfFailedTrajectories (int *nbFailedTraject*) `[pure virtual]`

Returns the number of failed trajectories.

Parameters

nbFailedTraject	the number of failed trajectories.

Implemented in StdResult.

4.31.3.19 virtual void Result::setPA (double *pa*) `[pure virtual]`

Sets the value of the PA result to pa. Sets isPASaved() to true.

Parameters

pa	the value of the PA result.

Implemented in StdResult.

4.31.3.20 virtual void Result::setStandardDeviation (double *stdDeviation*) `[pure virtual]`

Returns the standard deviation.

Parameters

stdDeviation	the standard deviation.

Implemented in StdResult.

4.31.3.21 virtual void Result::setStructAsymParam (double *asymParam*) `[pure virtual]`

Sets the value of the structural asymmetry parameter to asymParam.

Parameters

asymParam	the value of the structural asymmetry parameter.

Implemented in StdResult.

4.31.3.22 virtual void Result::setTM (double *tm*) `[pure virtual]`

Sets the value of the TM result to tm. Sets isTMSaved() to true.

Parameters

tm	the value of the TM result.

Implemented in StdResult.

4.31.3.23 virtual void Result::TMNeedsToBePrinted (bool *b*) `[pure virtual]`

Indicates if TM needs to be printed.

Parameters

true	if TM needs to be printed, false otherwise.

Implemented in StdResult.

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

- Collision-Code/math/Result.h

4.32 StdAtom Class Reference

```
#include <StdAtom.h>
```

Inherits Atom.

Public Member Functions

- StdAtom (Vector3D ∗pos, std::string symb, double ch=0.0)
- virtual ∼StdAtom ()
- Vector3D ∗ getPosition () const
- Vector3D ∗ getInitialPosition () const
- std::string getSymbol () const
- double getCharge () const
- void setPosition (Vector3D ∗c)
- void setSymbol (std::string s)
- void setCharge (double c)

4.32.1 Constructor & Destructor Documentation

4.32.1.1 StdAtom::StdAtom (Vector3D ∗ *pos*, std::string *symb*, double *ch* = `0.0`)

Creates an atom at position pos of atomic symbol symb. Charge is set to ch.

4.32.1.2 StdAtom::∼StdAtom ( ) `[virtual]`

Releases allocated resources.

4.32.2 Member Function Documentation

4.32.2.1 double StdAtom::getCharge ( ) const `[inline]`, `[virtual]`

Returns charge value of atom.

Implements Atom.

4.32.2.2 Vector3D∗ StdAtom::getInitialPosition ( ) const `[inline]`, `[virtual]`

Returns the initial position of atom.

Implements Atom.

4.32.2.3 Vector3D∗ StdAtom::getPosition ( ) const `[inline]`, `[virtual]`

Returns position of atom.

Implements Atom.

4.32.2.4 std::string StdAtom::getSymbol ( ) const `[inline]`, `[virtual]`

Return symbol of atom.

Implements Atom.

4.32.2.5 void StdAtom::setCharge (double *c*) `[inline]`, `[virtual]`

Sets a new charge value for atom.

Parameters

c	One double.

Implements Atom.

4.32.2.6 void StdAtom::setPosition (Vector3D ∗ *c*) [virtual]

Sets a new position for the atom.

Parameters

c	One coordinate.

Implements Atom.

4.32.2.7 void StdAtom::setSymbol (std::string *s*) [virtual]

Sets a new symbol value for atom.

Parameters

s	A string value.

Implements Atom.

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

- Collision-Code/molecule/StdAtom.h
- Collision-Code/molecule/StdAtom.cpp

4.33 StdCalculationOperator Class Reference

```
#include <StdCalculationOperator.h>
```

Inherits CalculationOperator.

Inherited by MonoThreadCalculationOperator, and MultiThreadCalculationOperator.

Public Member Functions

- StdCalculationOperator (CalculationState ∗calculationState, Molecule ∗mol, double temperature, double potentialEnergyStart, double timeStepStart, double potentialEnergyCloseCollision, double timeStepClose↩ Collision, double numberCyclesTM, double numberPointsVelocity, double numberPointsMCIntegrationTM, double energyConservationThreshold, double numberPointsMCIntegrationEHSSPA)
- virtual ∼StdCalculationOperator ()
- Result ∗ getResults ()
- CalculationState ∗ getCalculationState () const
- void runEHSSAndPA ()
- void runTM ()

Protected Member Functions

- void calculateEHSSAndPA (Molecule ∗mol)
- void che (Molecule ∗mol, int refl, double &halfCos, double cop, double &yRand, double &zRand, bool &kp, Vector3D &initialIncidenceVector)
- virtual void calculateTM ()=0
- void calculateAsymmetryParameter ()
- double calculatePotentials (std::vector< Vector3D > &molPos, const Vector3D &p, Vector3D &dPot, double &dMax)
- double calculateTrajectory (std::vector< Vector3D > &molPos, double v, double b)
- double calculateHamilton (std::vector< Vector3D > &molPos, std::array< double, 6 > &w, std::array< double, 6 > &dw, double &dMax)
- double calculateRKandAM (std::vector< Vector3D > &molPos, int &l, double &tim, double &dt, std::array< double, 6 > &w, std::array< double, 6 > &dw, std::array< std::array< double, 6 >, 6 > &arrayDouble, double &dMax, double &hVar, double &hcVar)

Protected Attributes

- CalculationState ∗ m_calculationState
- Molecule ∗ m_mol
- Result ∗ m_result
- std::vector< double > m_rhsTab
- double m_temperature
- int m_numberCyclesTM
- int m_numberPointsVelocity
- int m_numberPointsMCIntegrationTM
- int m_numberPointsMCIntegrationEHSSPA
- std::vector< double > m_EOLJTab
- std::vector< double > m_ROLJTab
- double m_maxROLJ
- double m_asymmetryParameter
- double m_massConstant
- double m_mobilityConstant
- double m_potentialEnergyStart
- double m_timeStepStart
- double m_potentialEnergyCloseCollision
- double m_timeStepCloseCollision
- double m_energyConservationThreshold
- std::vector< Vector3D > m_molInitPos
- std::vector< Vector3D > m_molPos
- unsigned int m_molNbAtoms
- std::vector< double > m_molChg
- double m_molMass

Static Protected Attributes

- static const int m_MaxSuccRefl = 30
- static const double m_IonInducedDipolePotential

  TM.
- static const double m_XeFromMobcal
- static const double m_XkFromMobcal
- static const double m_XmvFromMobcal
- static const double m_EoFromMobcal
- static const double m_RoFromMobcal
- static const int m_NbIntegrationStep = 1
- static const double m_MaxImpactParameter = 0.0005

4.33.1 Constructor & Destructor Documentation

4.33.1.1 StdCalculationOperator::StdCalculationOperator (**CalculationState** ∗ *calculationState*, **Molecule** ∗ *mol*, double *temperature*, double *potentialEnergyStart*, double *timeStepStart*, double *potentialEnergyCloseCollision*, double *timeStepCloseCollision*, double *numberCyclesTM*, double *numberPointsVelocity*, double *numberPointsMCIntegrationTM*, double *energyConservationThreshold*, double *numberPointsMCIntegrationEHSSPA*)

StdCalculationOperator's constructor.

Parameters

mol	the molecule to work with.

4.33.1.2 StdCalculationOperator::∼StdCalculationOperator ( ) `[virtual]`

Destructs allocated resources.

4.33.2 Member Function Documentation

4.33.2.1 void StdCalculationOperator::calculateAsymmetryParameter ( ) `[protected]`

Calculates the structural asymmetry parameter and puts it in m_asymmetryParameter.

4.33.2.2 void StdCalculationOperator::calculateEHSSAndPA (**Molecule** ∗ *mol*) `[protected]`

Calculates EHSS and PA and put the results in m_result attribute.

Calculate EHSS and PA and put the results in m_result attribute.

4.33.2.3 double StdCalculationOperator::calculateHamilton (std::vector< **Vector3D** > & *molPos*, std::array< double, 6 > & *w*, std::array< double, 6 > & *dw*, double & *dMax*) `[protected]`

Defines Hamilton's equations of motion ad the time derivates of the coordinates and momenta.

Returns

the potential

4.33.2.4 double StdCalculationOperator::calculatePotentials (std::vector< **Vector3D** > & *molPos*, const **Vector3D** & *p*, **Vector3D** & *dPot*, double & *dMax*) `[protected]`

Calculates the potential and the derivates of the potential. The potential is given by a sum of 6-12 two body ...

Parameters

p	the position for the calculation
dPot	the derivates of the potential

Returns

the potential

4.33.2.5 double StdCalculationOperator::calculateRKandAM (std::vector< Vector3D > & *molPos,* int & *l,* double & *tim,* double & *dt,* std::array< double, 6 > & *w,* std::array< double, 6 > & *dw,* std::array< std::array< double, 6 >, 6 > & *arrayDouble,* double & *dMax,* double & *hVar,* double & *hcVar*) [protected]

Integration method. Uses 5th order Runge-Kutta-Gill to initiate and 5th order Adams-Moulton predictor-corrector to propagate.

Returns

the potential

4.33.2.6 virtual void StdCalculationOperator::calculateTM ( ) [protected], [pure virtual]

Calculates TM and put the results in m_result attribute.

Implemented in MultiThreadCalculationOperator, and MonoThreadCalculationOperator.

4.33.2.7 double StdCalculationOperator::calculateTrajectory (std::vector< Vector3D > & *molPos,* double *v,* double *b*) [protected]

Calculates a trajectory.

Returns

angle of deviation

4.33.2.8 void StdCalculationOperator::che (Molecule * *mol,* int *refl,* double & *halfCos,* double *cop,* double & *yRand,* double & *zRand,* bool & *kp,* Vector3D & *initialIncidenceVector*) [protected]

Guides hard sphere scattering trajectory.

4.33.2.9 CalculationState* StdCalculationOperator::getCalculationState ( ) const [inline], [virtual]

Returns

the calculation state associated with this calculation operator.

Implements CalculationOperator.

4.33.2.10 Result∗ StdCalculationOperator::getResults ( ) [inline], [virtual]

Returns the results.

Returns

a pointer to the results of calculations. Pointer is destroy when the instance of StdCalculationOperator is destroyed.

Implements CalculationOperator.

4.33.2.11 void StdCalculationOperator::runEHSSAndPA ( ) [virtual]

Launches the calculation of EHSS and PA.

Implements CalculationOperator.

4.33.2.12 void StdCalculationOperator::runTM ( ) [virtual]

Launches the calculation of TM.

Implements CalculationOperator.

4.33.3 Member Data Documentation

4.33.3.1 double StdCalculationOperator::m_asymmetryParameter [protected]

Structural asymmetry parameter.

4.33.3.2 CalculationState∗ StdCalculationOperator::m_calculationState [protected]

The state to update during calculations to notify the views.

4.33.3.3 double StdCalculationOperator::m_energyConservationThreshold [protected]

Energy conservation threshold.

4.33.3.4 const double StdCalculationOperator::m_EoFromMobcal [static], [protected]

Initial value:

```
=
  1.34 * pow(10, -3) * 1.60217733 * pow(10, -19)
```

Lennard-Jones scaling parameter.

4.33.3.5 std::vector<double> StdCalculationOperator::m_EOLJTab [protected]

EOLJ for Helium values.

4.33.3.6 const double StdCalculationOperator::m_IonInducedDipolePotential [static], [protected]

Initial value:

```
=
  (0.204956 * pow(10, -30) / (8.0 * M_PI * 8.854187817 * pow(10, -12)))
    * pow(1.60217733 * pow(10, -19), 2)
```

TM.

Constant for ion-induced dipole potential.

4.33.3.7 double StdCalculationOperator::m_massConstant [protected]

Mass constant (mu in Mobcal).

4.33.3.8 const double StdCalculationOperator::m_MaxImpactParameter = 0.0005 [static], [protected]

Determines the maximum impact parameter at each velocity.

4.33.3.9 double StdCalculationOperator::m_maxROLJ [protected]

ROLJ maximum.

4.33.3.10 const int StdCalculationOperator::m_MaxSuccRefl = 30 [static], [protected]

Maximum of successive reflections followed.

4.33.3.11 double StdCalculationOperator::m_mobilityConstant [protected]

Mobility constant (mconst in Mobcal).

4.33.3.12 Molecule* StdCalculationOperator::m_mol [protected]

The molecule to work with.

4.33.3.13 std::vector<double> StdCalculationOperator::m_molChg [protected]

Charges of each atoms of the molecule. For calculations.

4.33.3.14 std::vector< Vector3D > StdCalculationOperator::m_molInitPos [protected]

Initial positions of the atoms of the molecule. For calculations.

4.33.3.15 double StdCalculationOperator::m_molMass [protected]

Mass of the molecule. For calculations.

4.33.3.16 unsigned int StdCalculationOperator::m_molNbAtoms [protected]

Number of atoms in the molecule. For calculations.

4.33.3.17 std::vector< Vector3D > StdCalculationOperator::m_molPos [protected]

Positions of the atoms of the molecule. For calculations.

4.33.3.18 const int StdCalculationOperator::m_NbIntegrationStep = 1 [static], [protected]

Number of integration steps before the program tests to see if the trajectory is done or lost.

4.33.3.19 int StdCalculationOperator::m_numberCyclesTM [protected]

Number of complete cycles for TM calculation.

4.33.3.20 int StdCalculationOperator::m_numberPointsMCIntegrationEHSSPA [protected]

Number of points in Monte-Carlo integration in EHSS/PA methods.

4.33.3.21 int StdCalculationOperator::m_numberPointsMCIntegrationTM [protected]

Number of points in Monte-Carlo integrations of impact parameter and orientation.

4.33.3.22 int StdCalculationOperator::m_numberPointsVelocity [protected]

Number of points in velocity integration.

4.33.3.23 double StdCalculationOperator::m_potentialEnergyCloseCollision [protected]

Potential energy where the trajectory comes close to a collision.

4.33.3.24 double StdCalculationOperator::m_potentialEnergyStart [protected]

Potential energy where the trajectory starts.

4.33.3.25 Result∗ StdCalculationOperator::m_result [protected]

The pointer where the results will be put.

4.33.3.26 std::vector<double> StdCalculationOperator::m_rhsTab [protected]

RHS values.

4.33.3.27 const double StdCalculationOperator::m_RoFromMobcal [static], [protected]

Initial value:

```
=
  3.043 * pow(10, -10)
```

Lennard-Jones scaling parameter.

4.33.3.28 std::vector<double> StdCalculationOperator::m_ROLJTab [protected]

ROLJ for Helium values.

4.33.3.29 double StdCalculationOperator::m_temperature [protected]

The temperature.

4.33.3.30 double StdCalculationOperator::m_timeStepCloseCollision [protected]

Time step when the trajectory comes close to a collision.

4.33.3.31 double StdCalculationOperator::m_timeStepStart [protected]

Time step at the start of the trajectory.

4.33.3.32 const double StdCalculationOperator::m_XeFromMobcal [static], [protected]

Initial value:

```
=
  1.60217733 * pow(10, -19)
```

xe from Mobcal.

4.33.3.33 const double StdCalculationOperator::m_XkFromMobcal [static], [protected]

Initial value:

```
=
  1.380658 * pow(10, -23)
```

xk from Mobcal.

4.33.3.34 const double StdCalculationOperator::m_XmvFromMobcal [static], [protected]

Initial value:

```
=
  0.0224141
```

xmv from Mobcal.

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

- Collision-Code/math/StdCalculationOperator.h
- Collision-Code/math/StdCalculationOperator.cpp

4.34 StdCmdView Class Reference

```
#include <StdCmdView.h>
```

Inherits CmdView.

Public Member Functions

- StdCmdView ()
- virtual ∼StdCmdView ()
- std::vector< std::string > getInputFiles () const
- std::vector< Molecule ∗ > getLoadedGeometries () const
- bool willEHSSBeCalculated () const
- bool willPABeCalculated () const
- bool willTMBeCalculated () const
- void shouldEHSSBeCalculated (bool b)
- void shouldPABeCalculated (bool b)
- void shouldTMBeCalculated (bool b)
- void addInputFile (std::string fileName)
- void removeInputFile (std::string fileName)
- int loadInputFiles ()
- std::string getChargeFile () const
- void setChargeFile (std::string chargeFileName)
- std::string getResultFormat () const
- void setOutputFile (std::string outputFileName)
- std::string getOutputFile () const
- void saveResults ()
- void launch ()

Additional Inherited Members

4.34.1 Constructor & Destructor Documentation

4.34.1.1 StdCmdView::StdCmdView ( )

Creates a new StdCmdView. By default, EHSS, PA and TM will be calculated.

4.34.1.2 StdCmdView::∼StdCmdView ( ) `[virtual]`

Releases all allocated resources.

4.34.2 Member Function Documentation

4.34.2.1 void StdCmdView::addInputFile (std::string *fileName*) `[virtual]`

Indicates a new file to load.

Parameters

fileName	the name of the file to load.

Implements CmdView.

4.34.2.2 std::string StdCmdView::getChargeFile ( ) const `[inline], [virtual]`

Returns

the name of the charge file.

Implements CmdView.

4.34.2.3 std::vector<std::string> StdCmdView::getInputFiles ( ) const `[inline], [virtual]`

Returns

the list of input files.

Implements CmdView.

4.34.2.4 std::vector<Molecule∗> StdCmdView::getLoadedGeometries ( ) const `[inline], [virtual]`

Returns

all loaded geometries.

Implements CmdView.

4.34.2.5 std::string StdCmdView::getOutputFile ( ) const [inline], [virtual]

Returns

the file name of the output file.

Implements CmdView.

4.34.2.6 std::string StdCmdView::getResultFormat ( ) const [virtual]

Returns

a string representing the content of the calculations save.

Implements CmdView.

4.34.2.7 void StdCmdView::launch ( ) [virtual]

Launches all the calculations, on all input files. Write the results in the output file.

Implements CmdView.

4.34.2.8 int StdCmdView::loadInputFiles ( ) [virtual]

Loads all saved input files with charge file if present.

Returns

the number of geometries loaded.

Implements CmdView.

4.34.2.9 void StdCmdView::removeInputFile (std::string *fileName*) [virtual]

Remove a file from the vector of the file to load.

Parameters

fileName	the name of the file to remove of the vector of the file to load.

Implements CmdView.

4.34.2.10 void StdCmdView::saveResults ( ) [virtual]

Save the results in the file named fileName. Delete previous content of the file.

Parameters

fileName	the name of the output file.

Implements CmdView.

4.34.2.11 void StdCmdView::setChargeFile (std::string *chargeFileName*) `[inline]`, `[virtual]`

Indicates the charge file name.

Parameters

chargeFileName	the name of the charge file.

Implements CmdView.

4.34.2.12 void StdCmdView::setOutputFile (std::string *outputFileName*) `[inline]`, `[virtual]`

Sets the output file.

Parameters

outputFileName	the file name of the output file.

Implements CmdView.

4.34.2.13 void StdCmdView::shouldEHSSBeCalculated (bool *b*) `[inline]`, `[virtual]`

Indicates if yes or no, EHSS should be calculated.

Parameters

b	true if EHSS should be calculated, else otherwise.

Implements CmdView.

4.34.2.14 void StdCmdView::shouldPABeCalculated (bool *b*) `[inline]`, `[virtual]`

Indicates if yes or no, PA should be calculated.

Parameters

b	true if PA should be calculated, else otherwise.

Implements CmdView.

4.34.2.15 void StdCmdView::shouldTMBeCalculated (bool *b*) [inline], [virtual]

Indicates if yes or no, TM should be calculated.

Parameters

b	true if TM should be calculated, else otherwise.

Implements CmdView.

4.34.2.16 bool StdCmdView::willEHSSBeCalculated ( ) const [inline], [virtual]

Returns

true if EHSS should be calculated, else otherwise.

Implements CmdView.

4.34.2.17 bool StdCmdView::willPABeCalculated ( ) const [inline], [virtual]

Returns

true if PA should be calculated, else otherwise.

Implements CmdView.

4.34.2.18 bool StdCmdView::willTMBeCalculated ( ) const [inline], [virtual]

Returns

true if TM should be calculated, else otherwise.

Implements CmdView.

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

- Collision-Code/general/StdCmdView.h
- Collision-Code/general/StdCmdView.cpp

4.35 StdExtractFactory Class Reference

```
#include <StdExtractFactory.h>
```

Inherits ExtractFactory.

Public Member Functions

- StdExtractFactory ()
- virtual ∼StdExtractFactory ()
- FileReader ∗ getReader (std::string fileName)

4.35.1 Constructor & Destructor Documentation

4.35.1.1 StdExtractFactory::StdExtractFactory ( )

Creates a new StdExtractFactory.

4.35.1.2 StdExtractFactory::∼StdExtractFactory ( ) [virtual]

Releases allocated resources.

4.35.2 Member Function Documentation

4.35.2.1 FileReader ∗ StdExtractFactory::getReader (std::string *fileName*) [virtual]

Returns the FileReader necessary to read the file.

Parameters

fileName	the file name.

Returns

a pointer to a FileReader which can read the file, or null if the file can't be read.

Implements ExtractFactory.

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

- Collision-Code/reader/StdExtractFactory.h
- Collision-Code/reader/StdExtractFactory.cpp

4.36 StdExtractResources Class Reference

```
#include <StdExtractResources.h>
```

Inherits ExtractResources.

Public Member Functions

- StdExtractResources ()
- virtual ∼StdExtractResources ()
- std::vector< Molecule ∗ > ∗ getGeometriesFromFile (std::string fileName)

4.36.1 Constructor & Destructor Documentation

4.36.1.1 StdExtractResources::StdExtractResources ( )

Creates a new StdExtractResources.

4.36.1.2 StdExtractResources::∼StdExtractResources ( ) [virtual]

Releases allocated resources.

4.36.2 Member Function Documentation

4.36.2.1 std::vector< **Molecule** ∗ > ∗ StdExtractResources::getGeometriesFromFile (std::string *fileName*) [virtual]

Returns a vector of molecules loaded from the file.

Parameters

fileName	the name of the file in which are the molecules.

Returns

a pointer to a vector containing the loaded molecules, or null if the file can't be loaded.

Implements ExtractResources.

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

- Collision-Code/reader/StdExtractResources.h
- Collision-Code/reader/StdExtractResources.cpp

4.37 StdFileWriter Class Reference

```
#include <StdFileWriter.h>
```

Inherits FileWriter.

Public Member Functions

- StdFileWriter (std::ostream &stream)
- virtual ∼StdFileWriter ()
- void visitResult (Result ∗result)
- void visitMean (Mean ∗mean)

4.37.1 Constructor & Destructor Documentation

4.37.1.1 StdFileWriter::StdFileWriter (std::ostream & *stream*)

Creates a new StdFileWriter.

4.37.1.2 StdFileWriter::∼StdFileWriter ( ) `[virtual]`

Releases all allocated resources.

4.37.2 Member Function Documentation

4.37.2.1 void StdFileWriter::visitMean (Mean ∗ *mean*) `[virtual]`

Writes a mean of results in a file.

Implements FileWriter.

4.37.2.2 void StdFileWriter::visitResult (Result ∗ *result*) `[virtual]`

Writes a result in a stream.

Implements FileWriter.

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

- Collision-Code/writer/StdFileWriter.h
- Collision-Code/writer/StdFileWriter.cpp

4.38 StdGeometryCalculator Class Reference

`#include <StdGeometryCalculator.h>`

Inherits GeometryCalculator.

Public Member Functions

- StdGeometryCalculator ()
- virtual ∼StdGeometryCalculator ()
- bool willEHSSBeCalculated () const
- bool willPABeCalculated () const
- bool willTMBeCalculated () const
- bool areCalculationsFinished (Molecule ∗mol) const
- Result ∗ getResults (Molecule ∗mol) const
- struct CalculationValues getCalculationValues () const
- void saveCalculationValues ()
- void shouldEHSSBeCalculated (bool b)
- void shouldPABeCalculated (bool b)
- void shouldTMBeCalculated (bool b)
- void setGeometries (std::vector< Molecule ∗ > ∗geometries)
- void takeObservers (std::vector< Observer ∗ > obs)
- void launchCalculations ()

4.38.1 Constructor & Destructor Documentation

4.38.1.1 StdGeometryCalculator::StdGeometryCalculator ( )

Constructs an instance of StdGeometryCalculator. By default, EHSS, PA and TM are calculated.

4.38.1.2 StdGeometryCalculator::∼StdGeometryCalculator ( ) [virtual]

Releases all allocated resources.

4.38.2 Member Function Documentation

4.38.2.1 bool StdGeometryCalculator::areCalculationsFinished (Molecule ∗ *mol*) const [virtual]

Indicates if calculations are finished for the molecule.

Parameters

mol	the molecule.

Returns

true if calculations are finished for the molecule, false otherwise.

Implements GeometryCalculator.

4.38.2.2 struct CalculationValues StdGeometryCalculator::getCalculationValues ( ) const [inline], [virtual]

Returns the values used for the calculations.

Returns

the values used for the calculations.

Implements GeometryCalculator.

4.38.2.3 Result $*$ StdGeometryCalculator::getResults (Molecule $*$ *mol*) const [virtual]

Returns the results of CCS calculation of a molecule.

Parameters

mol	the molecule.

Returns

the results for the molecule.

Implements GeometryCalculator.

4.38.2.4 void StdGeometryCalculator::launchCalculations ( ) [virtual]

Launches all the calculations, on all geometries.

Implements GeometryCalculator.

4.38.2.5 void StdGeometryCalculator::saveCalculationValues ( ) [virtual]

Forces the GeometryCalculator to save the calculation values from GlobalParameters at this moment.

Implements GeometryCalculator.

4.38.2.6 void StdGeometryCalculator::setGeometries (std::vector$<$ Molecule $*$ $>$ $*$ *geometries*) [virtual]

Sets a vector of molecules (geometries) for CCS calculation.

Parameters

geometries	a vector of geometries.

Implements GeometryCalculator.

4.38.2.7 void StdGeometryCalculator::shouldEHSSBeCalculated (bool *b*) [inline], [virtual]

Indicates if yes or no, EHSS should be calculated.

Parameters

b	true if EHSS should be calculated, else otherwise.

Implements GeometryCalculator.

4.38.2.8 void StdGeometryCalculator::shouldPABeCalculated (bool *b*) [inline], [virtual]

Indicates if yes or no, PA should be calculated.

Parameters

b	true if PA should be calculated, else otherwise.

Implements GeometryCalculator.

4.38.2.9 void StdGeometryCalculator::shouldTMBeCalculated (bool *b*) [inline], [virtual]

Indicates if yes or no, TM should be calculated.

Parameters

b	true if TM should be calculated, else otherwise.

Implements GeometryCalculator.

4.38.2.10 void StdGeometryCalculator::takeObservers (std::vector< Observer * > *obs*) [inline], [virtual]

Indicates that these Observers want to be notified about the calculations.

Parameters

obs	the observers list.

Implements GeometryCalculator.

4.38.2.11 bool StdGeometryCalculator::willEHSSBeCalculated ( ) const [inline], [virtual]

Returns

true if EHSS will be calculated, false otherwise.

Implements GeometryCalculator.

4.38.2.12 bool StdGeometryCalculator::willPABeCalculated ( ) const [inline], [virtual]

Returns

true if PA will be calculated, false otherwise.

Implements GeometryCalculator.

4.38.2.13 bool StdGeometryCalculator::willTMBeCalculated ( ) const [inline], [virtual]

Returns

true if TM will be calculated, false otherwise.

Implements GeometryCalculator.

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

- Collision-Code/general/StdGeometryCalculator.h
- Collision-Code/general/StdGeometryCalculator.cpp

4.39 StdMathLib Class Reference

```
#include <StdMathLib.h>
```

Inherits MathLib.

Public Member Functions

- StdMathLib ()
- virtual ∼StdMathLib ()
- void rotate (Molecule ∗mol, double angleX, double angleY, double angleZ)
- void rotate (const std::vector< Vector3D > &initPos, std::vector< Vector3D > &pos, double angleX, double angleY, double angleZ)
- void randomRotation (Molecule ∗mol)
- void randomRotation (const std::vector< Vector3D > &initPos, std::vector< Vector3D > &pos)
- Vector3D calculateMassCenter (const Molecule &mol)
- Atom ∗ findFarthestAtom (const Molecule &mol)
- double monteCarloIntegration (double(∗f)(double), double minLimit, double maxLimit, int n)

4.39.1 Constructor & Destructor Documentation

4.39.1.1 StdMathLib::StdMathLib ( )

4.39.1.2 StdMathLib::∼StdMathLib ( ) [virtual]

4.39.2 Member Function Documentation

4.39.2.1 Vector3D StdMathLib::calculateMassCenter (const Molecule & *mol*) [virtual]

Calculates the center of mass of a molecule.

Parameters

mol	the molecule.

Returns

the coordinates of the center of mass of the molecule mol.

Implements MathLib.

4.39.2.2 Atom ∗ StdMathLib::findFarthestAtom (const Molecule & *mol*) [virtual]

Finds the atom the farthest of the center of mass.

Parameters

mol	the molecule.

Returns

a pointer to the atom which is the farthest of the center of mass of mol.

Implements MathLib.

4.39.2.3 double StdMathLib::monteCarloIntegration (double(∗)(double) *f*, double *minLimit*, double *maxLimit*, int *n*) [virtual]

Implementation of the Monte-Carlo method for calculating integrals.

Parameters

f	the function to integrate.
minLimit	the lower limit of the integral.
maxLimit	the upper limit of the integral.
n	the number of points generate to calculate the integral. More points increase the result precision.

Returns

the result of the integration.

Implements MathLib.

4.39.2.4 void StdMathLib::randomRotation (Molecule ∗ *mol*) [virtual]

Rotates the molecule by random angles on each axis.

Parameters

mol	the molecule to rotate.

Implements MathLib.

4.39.2.5 void StdMathLib::randomRotation (const std::vector< Vector3D > & *initPos*, std::vector< Vector3D > & *pos*) [virtual]

Rotates the positions by random angles on each axis.

Parameters

pos	the positions to rotate.

Implements MathLib.

4.39.2.6 void StdMathLib::rotate (Molecule ∗ *mol*, double *angleX*, double *angleY*, double *angleZ*) [virtual]

Rotates the molecule by angles specified on each axis.

Parameters

mol	the molecule to rotate.
angleX	the angle of rotation one the X axis.
angleY	the angle of rotation one the Y axis.
angleZ	the angle of rotation one the Z axis.

Implements MathLib.

4.39.2.7 void StdMathLib::rotate (const std::vector< Vector3D > & *initPos*, std::vector< Vector3D > & *pos*, double *angleX*, double *angleY*, double *angleZ*) [virtual]

Rotates the position by angles specified on each axis.

Parameters

pos	the positions to rotate.
angleX	the angle of rotation one the X axis.
angleY	the angle of rotation one the Y axis.
angleZ	the angle of rotation one the Z axis.

Implements MathLib.

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

- Collision-Code/math/StdMathLib.h
- Collision-Code/math/StdMathLib.cpp

4.40 StdMean Class Reference

```
#include <StdMean.h>
```

Inherits Mean.

Public Member Functions

- StdMean ()
- virtual ∼StdMean ()
- double getMeanEHSS ()
- double getMeanPA ()
- double getMeanTM ()
- double getMeanStructAsymParam ()
- double getMeanStandardDeviation ()
- int getMeanNumberOfFailedTrajectories ()
- bool isEHSSSaved ()
- bool isPASaved ()
- bool isTMSaved ()
- bool isEHSSPrintable ()
- bool isPAPrintable ()
- bool isTMPrintable ()
- void addResult (Result ∗r)
- void accept (class FileWriter &fileWriter)

4.40.1 Constructor & Destructor Documentation

4.40.1.1 StdMean::StdMean ( )

Constructor.

4.40.1.2 StdMean::∼StdMean ( ) `[virtual]`

Destructor.

4.40.2 Member Function Documentation

4.40.2.1 void StdMean::accept (class FileWriter & *fileWriter*) `[virtual]`

Write the mean object via the FileWriter.

Implements Mean.

4.40.2.2 void StdMean::addResult (Result ∗ *r*) `[inline]`, `[virtual]`

Add a result to the results used to calculate the means.

Parameters

r	the result to add to the list.

Implements Mean.

4.40.2.3 double StdMean::getMeanEHSS ( ) [virtual]

Returns the mean of EHSS results.

Returns

the mean of EHSS results, or 0 is !isEHSSSaved().

Implements Mean.

4.40.2.4 int StdMean::getMeanNumberOfFailedTrajectories ( ) [virtual]

Returns the mean of the numbers of failed trajectories.

Returns

the mean of the numbers of failed trajectories.

Implements Mean.

4.40.2.5 double StdMean::getMeanPA ( ) [virtual]

Returns the mean of PA results.

Returns

the mean of PA results, or 0 is !isPASaved().

Implements Mean.

4.40.2.6 double StdMean::getMeanStandardDeviation ( ) [virtual]

Returns the mean of the standard deviations.

Returns

the mean of the standard deviation.

Implements Mean.

4.40.2.7 double StdMean::getMeanStructAsymParam ( ) [virtual]

Returns the mean of the structural asymmetry parameters.

Returns

the mean of the structural asymmetry parameters.

Implements Mean.

4.40.2.8 double StdMean::getMeanTM ( ) [virtual]

Returns the mean of TM results.

Returns

the mean of TM results, or 0 is !isTMSaved().

Implements Mean.

4.40.2.9 bool StdMean::isEHSSPrintable ( ) [virtual]

Indicates if EHSS needs to be printed.

Returns

true if EHSS needs to be printed, false otherwise.

Implements Mean.

4.40.2.10 bool StdMean::isEHSSSaved ( ) [virtual]

Returns

true if EHSS was saved, false in the other case.

Implements Mean.

4.40.2.11 bool StdMean::isPAPrintable ( ) [virtual]

Indicates if PA needs to be printed.

Returns

true if PA needs to be printed, false otherwise.

Implements Mean.

4.40.2.12 bool StdMean::isPASaved ( ) [virtual]

Returns

true if PA was saved, false in the other case.

Implements Mean.

4.40.2.13 bool StdMean::isTMPrintable ( ) [virtual]

Indicates if TM needs to be printed.

Returns

true if TM needs to be printed, false otherwise.

Implements Mean.

4.40.2.14 bool StdMean::isTMSaved ( ) [virtual]

Returns

true if TM was saved, false in the other case.

Implements Mean.

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

- Collision-Code/math/StdMean.h
- Collision-Code/math/StdMean.cpp

4.41 StdMolecule Class Reference

`#include <StdMolecule.h>`

Inherits Molecule.

Public Member Functions

- StdMolecule ()
- virtual ∼StdMolecule ()
- std::string getName ()
- unsigned int getAtomNumber () const
- double getTotalMass () const
- std::vector< Atom ∗ > ∗ getAllAtoms () const
- Atom ∗ getAtom (const Vector3D &c) const
- void toInitialPosition ()
- void setName (std::string n)
- void addAtom (Atom ∗a)
- void deleteAtom (Atom ∗a)
- void deleteAtom (const Vector3D &c)

4.41.1 Constructor & Destructor Documentation

4.41.1.1 StdMolecule::StdMolecule ( )

Creates an empty molecule.

4.41.1.2 StdMolecule::∼StdMolecule ( ) `[virtual]`

Releases allocates resources.

4.41.2 Member Function Documentation

4.41.2.1 void StdMolecule::addAtom (**Atom** ∗ *a*) `[virtual]`

Adds an atom on the molecule.

Parameters

a	a pointer on an atom.

Implements Molecule.

4.41.2.2 void StdMolecule::deleteAtom (**Atom** ∗ *a*) `[virtual]`

Deletes the specified atom.

Parameters

a	a pointer on an atom.

Implements Molecule.

4.41.2.3 void StdMolecule::deleteAtom (const **Vector3D** & *c*) `[virtual]`

Deletes the atom at specified position.

Parameters

c	a coordinate.

Implements Molecule.

4.41.2.4 std::vector<**Atom**∗>∗ StdMolecule::getAllAtoms ( ) const `[inline]`, `[virtual]`

Returns

a pointer on atom collection.

Implements Molecule.

4.41.2.5 Atom ∗ StdMolecule::getAtom (const Vector3D & *c*) const `[virtual]`

Parameters

c	a coordinate

Returns

the atom from the specified position.

Implements Molecule.

4.41.2.6 unsigned int StdMolecule::getAtomNumber ( ) const `[inline]`, `[virtual]`

Returns

the total number of atom forming molecule composition.

Implements Molecule.

4.41.2.7 std::string StdMolecule::getName ( ) `[virtual]`

Returns

the name of molecule.

Implements Molecule.

4.41.2.8 double StdMolecule::getTotalMass ( ) const `[virtual]`

Returns

the mass of the molecule.

Implements Molecule.

4.41.2.9 void StdMolecule::setName (std::string *n*) `[inline]`, `[virtual]`

Replaces the current name of molecule by a new one.

Parameters

n	a string value.

Implements Molecule.

4.41.2.10 void StdMolecule::toInitialPosition ( ) [virtual]

Replaces the molecule at its initial position.

Implements Molecule.

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

- Collision-Code/molecule/StdMolecule.h
- Collision-Code/molecule/StdMolecule.cpp

4.42 StdResult Class Reference

`#include <StdResult.h>`

Inherits Result.

Public Member Functions

- StdResult (Molecule ∗mol)
- virtual ∼StdResult ()
- Molecule ∗ getAssociateMolecule ()
- double getEHSS ()
- double getPA ()
- double getTM ()
- double getStructAsymParam ()
- double getStandardDeviation ()
- int getNumberOfFailedTrajectories ()
- bool isEHSSSaved ()
- bool isPASaved ()
- bool isTMSaved ()
- void setEHSS (double ehss)
- void setPA (double pa)
- void setTM (double tm)
- void setStructAsymParam (double asymParam)
- void setStandardDeviation (double stdDeviation)
- void setNumberOfFailedTrajectories (int nbFailedTraject)
- void EHSSNeedsToBePrinted (bool b)
- void PANeedsToBePrinted (bool b)
- void TMNeedsToBePrinted (bool b)
- bool isEHSSPrintable ()
- bool isPAPrintable ()
- bool isTMPrintable ()
- void accept (FileWriter &fileWriter)

4.42.1 Constructor & Destructor Documentation

4.42.1.1 StdResult::StdResult (**Molecule** ∗ *mol*)

StdResult's constructor.

Parameters

mol	the molecule relative to the results.

4.42.1.2 StdResult::∼StdResult ( ) `[virtual]`

4.42.2 Member Function Documentation

4.42.2.1 void StdResult::accept (FileWriter & *fileWriter*) `[virtual]`

Write the result via the FileWriter.

Implements Result.

4.42.2.2 void StdResult::EHSSNeedsToBePrinted (bool *b*) `[inline]`, `[virtual]`

Indicates if EHSS needs to be printed.

Parameters

true	if EHSS needs to be printed, false otherwise.

Implements Result.

4.42.2.3 Molecule∗ StdResult::getAssociateMolecule ( ) `[inline]`, `[virtual]`

Returns the molecule saves with these results.

Returns

the molecule saves with these results.

Implements Result.

4.42.2.4 double StdResult::getEHSS ( ) `[inline]`, `[virtual]`

Returns the result for EHSS.

Returns

the EHSS result, or 0 is !isEHSSSaved().

Implements Result.

4.42.2.5 int StdResult::getNumberOfFailedTrajectories ( ) [inline], [virtual]

Returns the number of failed trajectories.

Returns

the number of failed trajectories.

Implements Result.

4.42.2.6 double StdResult::getPA ( ) [inline], [virtual]

Returns the result for PA.

Returns

the PA result, or 0 is !isPASaved().

Implements Result.

4.42.2.7 double StdResult::getStandardDeviation ( ) [inline], [virtual]

Returns the standard deviation.

Returns

the standard deviation.

Implements Result.

4.42.2.8 double StdResult::getStructAsymParam ( ) [inline], [virtual]

Returns the structural asymmetry parameter.

Returns

the structural asymmetry parameter.

Implements Result.

4.42.2.9 double StdResult::getTM ( ) [inline], [virtual]

Returns the result for TM.

Returns

the TM result, or 0 is !isTMSaved().

Implements Result.

4.42.2.10 bool StdResult::isEHSSPrintable ( ) [inline], [virtual]

Indicates if EHSS needs to be printed.

Returns

true if EHSS needs to be printed, false otherwise.

Implements Result.

4.42.2.11 bool StdResult::isEHSSSaved ( ) [inline], [virtual]

Returns

true if EHSS was saved, false in the other case.

Implements Result.

4.42.2.12 bool StdResult::isPAPrintable ( ) [inline], [virtual]

Indicates if PA needs to be printed.

Returns

true if PA needs to be printed, false otherwise.

Implements Result.

4.42.2.13 bool StdResult::isPASaved ( ) [inline], [virtual]

Returns

true if PA was saved, false in the other case.

Implements Result.

4.42.2.14 bool StdResult::isTMPrintable ( ) [inline], [virtual]

Indicates if TM needs to be printed.

Returns

true if TM needs to be printed, false otherwise.

Implements Result.

4.42.2.15 bool StdResult::isTMSaved ( ) [inline], [virtual]

Returns

true if TM was saved, false in the other case.

Implements Result.

4.42.2.16 void StdResult::PANeedsToBePrinted (bool *b*) [inline], [virtual]

Indicates if PA needs to be printed.

Parameters

true	if PA needs to be printed, false otherwise.

Implements Result.

4.42.2.17 void StdResult::setEHSS (double *ehss*) [inline], [virtual]

Sets the value of the EHSS result to ehss. Sets isEHSSSaved() to true.

Parameters

ehss	the value of the EHSS result.

Implements Result.

4.42.2.18 void StdResult::setNumberOfFailedTrajectories (int *nbFailedTraject*) [inline], [virtual]

Returns the number of failed trajectories.

Parameters

nbFailedTraject	the number of failed trajectories.

Implements Result.

4.42.2.19 void StdResult::setPA (double *pa*) [inline], [virtual]

Sets the value of the PA result to pa. Sets isPASaved() to true.

Parameters

pa	the value of the PA result.

Implements Result.

4.42.2.20 void StdResult::setStandardDeviation (double *stdDeviation*) [inline], [virtual]

Returns the standard deviation.

Parameters

stdDeviation	the standard deviation.

Implements Result.

4.42.2.21 void StdResult::setStructAsymParam (double *asymParam*) `[inline]`, `[virtual]`

Sets the value of the structural asymmetry parameter to asymParam.

Parameters

asymParam	the value of the structural asymmetry parameter.

Implements Result.

4.42.2.22 void StdResult::setTM (double *tm*) `[inline]`, `[virtual]`

Sets the value of the TM result to tm. Sets isTMSaved() to true.

Parameters

tm	the value of the TM result.

Implements Result.

4.42.2.23 void StdResult::TMNeedsToBePrinted (bool *b*) `[inline]`, `[virtual]`

Indicates if TM needs to be printed.

Parameters

true	if TM needs to be printed, false otherwise.

Implements Result.

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

- Collision-Code/math/StdResult.h
- Collision-Code/math/StdResult.cpp

4.43 SystemParameters Class Reference

```
#include <SystemParameters.h>
```

Public Member Functions

- virtual ∼SystemParameters ()
- unsigned int getMaximalNumberThreads () const
- void setMaximalNumberThreads (int n)

Static Public Member Functions

- static SystemParameters ∗ getInstance ()

4.43.1 Constructor & Destructor Documentation

4.43.1.1 SystemParameters::∼SystemParameters ( ) [virtual]

Destructor.

4.43.2 Member Function Documentation

4.43.2.1 static SystemParameters∗ SystemParameters::getInstance ( ) [inline], [static]

Returns

an instance of SystemParameters to work with.

4.43.2.2 unsigned int SystemParameters::getMaximalNumberThreads ( ) const [inline]

Returns the maximal number of threads.

Returns

the maximal number of threads.

4.43.2.3 void SystemParameters::setMaximalNumberThreads (int *n*) [inline]

Sets the maximal number of threads to n.

Parameters

n	the new maximal number of threads.

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

- Collision-Code/general/SystemParameters.h
- Collision-Code/general/SystemParameters.cpp

4.44 Vector3D Class Reference

```
#include <Vector3D.h>
```

Public Member Functions

- Vector3D (double x=0.0, double y=0.0, double z=0.0)
- Vector3D (const Vector3D &vec)
- ∼Vector3D ()
- void operator= (const Vector3D &vec)
- bool operator== (const Vector3D &vec) const

Public Attributes

- double x
- double y
- double z

Friends

- std::ostream & operator<< (std::ostream &out, const Vector3D &vec)

4.44.1 Constructor & Destructor Documentation

4.44.1.1 Vector3D::Vector3D (double *x* = `0.0`, double *y* = `0.0`, double *z* = `0.0`)

Create a coordinate with three points specified in input.

Parameters

x	the value on the X axis.
y	the value on the Y axis.
z	the value on the Z axis.

4.44.1.2 Vector3D::Vector3D (const Vector3D & *vec*)

Copy constructor.

4.44.1.3 Vector3D::∼Vector3D ( )

Release allocates resources.

4.44.2 Member Function Documentation

4.44.2.1 void Vector3D::operator= (const Vector3D & *vec*) `[inline]`

Overloading of the assignment operator.

4.44.2.2 bool Vector3D::operator== (const Vector3D & *vec*) const `[inline]`

Overloading of the equality operator.

4.44.3 Friends And Related Function Documentation

4.44.3.1 std::ostream& operator<< (std::ostream & *out*, const Vector3D & *vec*) `[friend]`

Overload of the << operator.

4.44.4 Member Data Documentation

4.44.4.1 double Vector3D::x

The value on the X axis.

4.44.4.2 double Vector3D::y

The value on the Y axis.

4.44.4.3 double Vector3D::z

The value on the Z axis.

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

- Collision-Code/math/Vector3D.h
- Collision-Code/math/Vector3D.cpp

4.45 Worker Class Reference

```
#include <CCFrame.h>
```

Inherits QObject.

Public Slots

- void doWork (StdCmdView *cmd)

Signals

- void finished ()

4.45.1 Detailed Description

A worker class to launch calculations on a different thread.

4.45.2 Member Function Documentation

4.45.2.1 void Worker::doWork (StdCmdView ∗ *cmd*) `[inline]`, `[slot]`

Launches the calculations.

4.45.2.2 void Worker::finished ( ) `[signal]`

Indicates that the calculations are finished.

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

- Collision-Code/gui/CCFrame.h
- Collision-Code/gui/moc_CCFrame.cpp

4.46 XyzFileReader Class Reference

```
#include <XyzFileReader.h>
```

Inherits FileReader.

Public Member Functions

- XyzFileReader (std::string filename)
- virtual ∼XyzFileReader ()
- std::string getFileName () const
- void setFileName (std::string filename)
- std::vector< Molecule ∗ > ∗ loadResources ()

4.46.1 Constructor & Destructor Documentation

4.46.1.1 XyzFileReader::XyzFileReader (std::string *filename*)

XyzFileReader's constructor.

Parameters

filename	the name of file to work with.

4.46.1.2 XyzFileReader::∼XyzFileReader ( ) [virtual]

Destructor.

4.46.2 Member Function Documentation

4.46.2.1 std::string XyzFileReader::getFileName ( ) const [inline], [virtual]

Returns name of file onload.

Returns

a string value giving the complete file name.

Implements FileReader.

4.46.2.2 std::vector< Molecule ∗ > ∗ XyzFileReader::loadResources ( ) [virtual]

Returns all molecule from the actual file.

Returns

a pointer to a molecule vector extract from file.

Implements FileReader.

4.46.2.3 void XyzFileReader::setFileName (std::string *filename*) [virtual]

Changes the actual file by a new one.

Implements FileReader.

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

- Collision-Code/reader/XyzFileReader.h
- Collision-Code/reader/XyzFileReader.cpp

Chapter 5

File Documentation

5.1 Collision-Code/console/ConsoleView.cpp File Reference

```
#include "ConsoleView.h"
#include <iostream>
#include <vector>
#include <string>
#include <cstring>
#include <chrono>
#include <sstream>
#include <iomanip>
#include "../general/StdCmdView.h"
#include "../general/AtomInformations.h"
#include "../general/SystemParameters.h"
#include "../general/GlobalParameters.h"
#include "../observer/Event.h"
#include "../observer/state/CalculationState.h"
```

Functions

- std::string getCmdStr ()

5.1.1 Function Documentation

5.1.1.1 std::string getCmdStr ( )

Returns

a string describing the command parameters.

5.2 Collision-Code/console/ConsoleView.h File Reference

Describes the console view and parses the arguments from the command line.

```
#include "../observer/Observer.h"
#include "../general/CmdView.h"
```

Classes

- class ConsoleView

5.2.1 Detailed Description

Describes the console view and parses the arguments from the command line.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.3 Collision-Code/general/AtomInformations.cpp File Reference

```
#include "AtomInformations.h"
#include <fstream>
#include <sstream>
#include <cstdlib>
#include <cmath>
#include <iostream>
#include <string>
```

5.4 Collision-Code/general/AtomInformations.h File Reference

Class implementing a singleton to access data on atoms.

```
#include <string>
#include <map>
#include <vector>
```

Classes

- class AtomInformations

5.4.1 Detailed Description

Class implementing a singleton to access data on atoms.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

Data contains atomic number, mass, hard sphere radius, parameters of Lennard-Jones and color.

5.5 Collision-Code/general/CmdView.h File Reference

Interface describing the general model.

```
#include "../observer/Observer.h"
#include "../molecule/Molecule.h"
#include <string>
#include <vector>
```

Classes

- class CmdView

5.5.1 Detailed Description

Interface describing the general model.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

A facade which oversees method calls.

5.6 Collision-Code/general/GeometryCalculator.h File Reference

Interface describing calculations and results.

```
#include <vector>
#include "../observer/Observer.h"
#include "../molecule/Molecule.h"
#include "../math/Result.h"
```

Classes

- class GeometryCalculator
- struct GeometryCalculator::CalculationValues

5.6.1 Detailed Description

Interface describing calculations and results.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

A facade which oversees calculation and results methods.

5.7 Collision-Code/general/GlobalParameters.cpp File Reference

```
#include "GlobalParameters.h"
```

5.8 Collision-Code/general/GlobalParameters.h File Reference

Class implementing a singleton to access global parameters.

Classes

- class GlobalParameters

5.8.1 Detailed Description

Class implementing a singleton to access global parameters.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

08 may 2016

5.9 Collision-Code/general/StdCmdView.cpp File Reference

```
#include "StdCmdView.h"
#include "GlobalParameters.h"
#include "StdGeometryCalculator.h"
#include "../reader/StdExtractResources.h"
#include "../reader/ChargesReader.h"
#include "../reader/ChgChargesReader.h"
#include "../molecule/Molecule.h"
#include "../writer/FileWriter.h"
#include "../writer/StdFileWriter.h"
#include "../observer/Event.h"
#include "../math/Mean.h"
#include "../math/StdMean.h"
#include <sstream>
#include <fstream>
#include <iostream>
```

Functions

- void doLines (std::ostringstream &oStream, bool EHSS, bool PA, bool TM)
- void doEntete (std::ostringstream &oStream, bool EHSS, bool PA, bool TM)

5.9.1 Function Documentation

5.9.1.1 void doEntete (std::ostringstream & *oStream,* bool *EHSS,* bool *PA,* bool *TM*)

5.9.1.2 void doLines (std::ostringstream & *oStream,* bool *EHSS,* bool *PA,* bool *TM*)

5.10 Collision-Code/general/StdCmdView.h File Reference

Class implementing the interface CmdView.h.

```
#include "CmdView.h"
#include "GeometryCalculator.h"
#include "../reader/ExtractResources.h"
#include "../observer/Observable.h"
#include <map>
```

Classes

- class StdCmdView

5.10.1 Detailed Description

Class implementing the interface CmdView.h.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

A facade which oversees method calls.

5.11 Collision-Code/general/StdGeometryCalculator.cpp File Reference

```
#include "StdGeometryCalculator.h"
#include "GlobalParameters.h"
#include "SystemParameters.h"
#include "../math/CalculationOperator.h"
#include "../math/MonoThreadCalculationOperator.h"
#include "../math/MultiThreadCalculationOperator.h"
#include "../observer/state/CalculationState.h"
#include <string>
```

5.12 Collision-Code/general/StdGeometryCalculator.h File Reference

Class implementing the interface GeometryCalculator.h.

```
#include "GeometryCalculator.h"
#include <map>
```

Classes

- class StdGeometryCalculator

5.12.1 Detailed Description

Class implementing the interface GeometryCalculator.h.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

A class which oversees calculations and results.

5.13 Collision-Code/general/SystemParameters.cpp File Reference

```
#include "SystemParameters.h"
```

5.14 Collision-Code/general/SystemParameters.h File Reference

Class implementing a singleton to access system parameters.

Classes

- class SystemParameters

5.14.1 Detailed Description

Class implementing a singleton to access system parameters.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

08 may 2016

5.15 Collision-Code/gui/CCFrame.cpp File Reference

```
#include "CCFrame.h"
```

Functions

- QString modifyResult (QString method, double d)
- void initializeSpinBox (QSpinBox ∗sb, int min, int max, int step, int value)
- void initializeDoubleSpinBox (QDoubleSpinBox ∗dsb, double min, double max, double step, int decimals, double value)
- QString obtainRelativePath (QString filePath)
- void launch (StdCmdView ∗cmd)

5.15.1 Function Documentation

5.15.1.1 void initializeDoubleSpinBox (QDoubleSpinBox ∗ *dsb,* double *min,* double *max,* double *step,* int *decimals,* double *value*)

5.15.1.2 void initializeSpinBox (QSpinBox ∗ *sb,* int *min,* int *max,* int *step,* int *value*)

5.15.1.3 void launch (StdCmdView ∗ *cmd*)

5.15.1.4 QString modifyResult (QString *method,* double *d*)

5.15.1.5 QString obtainRelativePath (QString *filePath*)

5.16 Collision-Code/gui/CCFrame.h File Reference

Implements a graphical user interface to use the calculation model.

```
#include <QtWidgets>
#include <string>
#include <array>
#include <map>
#include "../general/GlobalParameters.h"
#include "../general/SystemParameters.h"
#include "../general/StdCmdView.h"
#include "../general/AtomInformations.h"
#include "../molecule/StdAtom.h"
#include "../molecule/StdMolecule.h"
#include "../observer/Observer.h"
#include "../observer/Event.h"
#include "../observer/state/CalculationState.h"
```

Classes

- class Worker
- class CCFrame

5.16.1 Detailed Description

Implements a graphical user interface to use the calculation model.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.17 Collision-Code/gui/moc_CCFrame.cpp File Reference

```
#include "CCFrame.h"
#include <QtCore/qbytearray.h>
#include <QtCore/qmetatype.h>
```

Classes

- struct qt_meta_stringdata_Worker_t
- struct qt_meta_stringdata_CCFrame_t

Macros

- #define QT_MOC_LITERAL(idx, ofs, len)
- #define QT_MOC_LITERAL(idx, ofs, len)

5.17.1 Macro Definition Documentation

5.17.1.1 #define QT_MOC_LITERAL(*idx,* *ofs,* *len*)

Value:

```
Q_STATIC_BYTE_ARRAY_DATA_HEADER_INITIALIZER_WITH_OFFSET(len, \
    qptrdiff(offsetof(qt_meta_stringdata_Worker_t, stringdata0) + ofs \
        - idx * sizeof(QByteArrayData)) \
    )
```

5.17.1.2 #define QT_MOC_LITERAL(*idx,* *ofs,* *len*)

Value:

```
Q_STATIC_BYTE_ARRAY_DATA_HEADER_INITIALIZER_WITH_OFFSET(len, \
    qptrdiff(offsetof(qt_meta_stringdata_CCFrame_t, stringdata0) + ofs \
        - idx * sizeof(QByteArrayData)) \
    )
```

5.18 Collision-Code/main.cpp File Reference

```
#include "console/ConsoleView.h"
```

Functions

- int main (int argc, char ∗const argv[ ])

5.18.1 Function Documentation

5.18.1.1 int main (int *argc,* char ∗const *argv[ ]*)

5.19 Collision-Code/mainQt.cpp File Reference

```
#include <QtWidgets>
#include "gui/CCFrame.h"
```

Functions

- int main (int argc, char ∗argv[ ])

5.19.1 Function Documentation

5.19.1.1 int main (int *argc,* char ∗ *argv[ ]*)

5.20 Collision-Code/math/CalculationOperator.h File Reference

Interface describing methods which will launch calculations on EHSS, PA and TM methods.

```
#include "Result.h"
#include "../observer/state/CalculationState.h"
```

Classes

- class CalculationOperator

5.20.1 Detailed Description

Interface describing methods which will launch calculations on EHSS, PA and TM methods.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.21 Collision-Code/math/MathLib.h File Reference

Interface describing mathematic operations.

```
#include <cstdlib>
#include "Vector3D.h"
#include "../molecule/Atom.h"
#include "../molecule/Molecule.h"
#include <vector>
```

Classes

- class MathLib

5.21.1 Detailed Description

Interface describing mathematic operations.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.22 Collision-Code/math/Mean.h File Reference

Interface describing a way of save mean of calculations results.

```
#include "Result.h"
```

Classes

- class Mean

5.22.1 Detailed Description

Interface describing a way of save mean of calculations results.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.23 Collision-Code/math/MonoThreadCalculationOperator.cpp File Reference

```
#include "MonoThreadCalculationOperator.h"
#include "../general/AtomInformations.h"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "StdResult.h"
#include "MathLib.h"
#include "StdMathLib.h"
#include "RandomGenerator.h"
#include <cmath>
#include <vector>
#include <string>
#include <iostream>
#include <cstdlib>
#include <boost/math/special_functions/pow.hpp>
```

Macros

- #define M_PI 3.14159265358979323846

5.23.1 Macro Definition Documentation

5.23.1.1 #define M_PI 3.14159265358979323846

5.24 Collision-Code/math/MonoThreadCalculationOperator.h File Reference

Implements the operations for calculating cross-section with EHSS, PA and TM methods and one thread.

```
#include "StdCalculationOperator.h"
#include "../molecule/Molecule.h"
#include "Vector3D.h"
#include <array>
#include <vector>
```

Classes

- class MonoThreadCalculationOperator

5.24.1 Detailed Description

Implements the operations for calculating cross-section with EHSS, PA and TM methods and one thread.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.25 Collision-Code/math/MultiThreadCalculationOperator.cpp File Reference

```
#include "MultiThreadCalculationOperator.h"
#include "../general/AtomInformations.h"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "StdResult.h"
#include "MathLib.h"
#include "StdMathLib.h"
#include "RandomGenerator.h"
#include <omp.h>
#include <cmath>
#include <vector>
#include <string>
#include <iostream>
#include <cstdlib>
#include <boost/math/special_functions/pow.hpp>
#include <boost/multiprecision/miller_rabin.hpp>
```

Macros

- #define M_PI 3.14159265358979323846

5.25.1 Macro Definition Documentation

5.25.1.1 #define M_PI 3.14159265358979323846

5.26 Collision-Code/math/MultiThreadCalculationOperator.h File Reference

Implements the operations for calculating cross-section with EHSS, PA and TM methods and many threads.

```
#include "StdCalculationOperator.h"
#include "../molecule/Molecule.h"
#include "Vector3D.h"
#include <array>
#include <vector>
```

Classes

- class MultiThreadCalculationOperator

5.26.1 Detailed Description

Implements the operations for calculating cross-section with EHSS, PA and TM methods and many threads.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.27 Collision-Code/math/RandomGenerator.cpp File Reference

```
#include "RandomGenerator.h"
#include <ctime>
```

5.28 Collision-Code/math/RandomGenerator.h File Reference

A singleton for generating random numbers with an uniform distribution.

```
#include "../lib/boost/random.hpp"
```

Classes

- class RandomGenerator

5.28.1 Detailed Description

A singleton for generating random numbers with an uniform distribution.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.29 Collision-Code/math/Result.h File Reference

A interface describing how to save the results of cross-section calculations.

```
#include "../molecule/Molecule.h"
```

Classes

- class Result

5.29.1 Detailed Description

A interface describing how to save the results of cross-section calculations.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.30 Collision-Code/math/StdCalculationOperator.cpp File Reference

```
#include "StdCalculationOperator.h"
#include "../general/AtomInformations.h"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "StdResult.h"
#include "MathLib.h"
#include "StdMathLib.h"
#include "RandomGenerator.h"
#include <cmath>
#include <array>
#include <vector>
#include <string>
#include <iostream>
#include <cstdlib>
#include <boost/math/special_functions/pow.hpp>
```

Macros

- #define M_PI 3.14159265358979323846
- #define ANGSTROMTOMETER (1e-10)

Variables

- const double var = 2.97013888888
- const double cvar = 0.990972222222
- const double acst = 0.332866152768
- const double a [ ]
- const double b [ ]
- const double c [ ]
- const double ampc [ ]
- const double amcc [ ]

5.30.1 Macro Definition Documentation

5.30.1.1 #define ANGSTROMTOMETER (1e-10)

5.30.1.2 #define M_PI 3.14159265358979323846

5.30.2 Variable Documentation

5.30.2.1 const double a[ ]

Initial value:

```
= {
  0.5,
  0.292893218814,
  1.70710678118,
  0.1666666666667
}
```

5.30.2.2 const double acst = 0.332866152768

5.30.2.3 const double amcc[ ]

Initial value:

```
= {
  0.0189208128941,
  -0.121233356692,
  0.337771548703,
  -0.55921513665
}
```

5.30.2.4 const double ampc[ ]

Initial value:

```
= {
  -0.111059153612,
  0.672667757774,
  -1.70633621697,
  2.33387888707,
  -1.8524668225
}
```

5.30.2.5 const double b[ ]

Initial value:

```
= {
  2.0,
  1.0,
  1.0,
  2.0
}
```

5.30.2.6 const double c[ ]

Initial value:

```
= {
  -0.5,
  -0.292893218814,
  -1.70710678118,
  -0.5
}
```

5.30.2.7 const double cvar = 0.990972222222

5.30.2.8 const double var = 2.97013888888

5.31 Collision-Code/math/StdCalculationOperator.h File Reference

Implements methods which will launch calculations on EHSS, PA and TM methods.

```
#include "CalculationOperator.h"
#include "../molecule/Molecule.h"
#include "Vector3D.h"
#include <array>
#include <vector>
```

Classes

- class StdCalculationOperator

5.31.1 Detailed Description

Implements methods which will launch calculations on EHSS, PA and TM methods.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.32 Collision-Code/math/StdMathLib.cpp File Reference

```
#include "StdMathLib.h"
#include "../general/AtomInformations.h"
#include "RandomGenerator.h"
#include <vector>
#include <cmath>
```

Macros

- #define M_PI 3.14159265358979323846

5.32.1 Macro Definition Documentation

5.32.1.1 #define M_PI 3.14159265358979323846

5.33 Collision-Code/math/StdMathLib.h File Reference

Implements mathematic operations.

```
#include "MathLib.h"
#include "Vector3D.h"
#include "../molecule/Atom.h"
#include "../molecule/Molecule.h"
#include <vector>
```

Classes

- class StdMathLib

5.33.1 Detailed Description

Implements mathematic operations.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.34 Collision-Code/math/StdMean.cpp File Reference

```
#include "StdMean.h"
```

5.35 Collision-Code/math/StdMean.h File Reference

Implements a way of save mean of calculations results.

```
#include "Mean.h"
#include "../writer/FileWriter.h"
#include <vector>
```

Classes

- class StdMean

5.35.1 Detailed Description

Implements a way of save mean of calculations results.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.36 Collision-Code/math/StdResult.cpp File Reference

```
#include "StdResult.h"
```

5.37 Collision-Code/math/StdResult.h File Reference

A class implementing a way to save the results of cross-section calculations.

```
#include "Result.h"
#include "../molecule/Molecule.h"
#include "../writer/FileWriter.h"
```

Classes

- class StdResult

5.37.1 Detailed Description

A class implementing a way to save the results of cross-section calculations.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.38 Collision-Code/math/Vector3D.cpp File Reference

```
#include "Vector3D.h"
```

5.38.1 Detailed Description

Implements a Vector3D.

5.39 Collision-Code/math/Vector3D.h File Reference

Implements a way to save a vector of three variables of type "double".

```
#include <iostream>
```

Classes

- class Vector3D

Functions

- std::ostream & operator<< (std::ostream &out, const Vector3D &vec)

5.39.1 Detailed Description

Implements a way to save a vector of three variables of type "double".

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.39.2 Function Documentation

5.39.2.1 std::ostream& operator<< (std::ostream & *out,* const Vector3D & *vec*) `[inline]`

Overload of the << operator.

5.40 Collision-Code/molecule/Atom.h File Reference

Interface describing the Atom model.

```
#include <string>
#include "../math/Vector3D.h"
```

Classes

- class Atom

5.40.1 Detailed Description

Interface describing the Atom model.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

An atom is representing by a vector in a three-dimensional coordinate, a charge and a symbol.

5.41 Collision-Code/molecule/Molecule.h File Reference

An interface describing a way of representing a molecule.

```
#include "Atom.h"
#include "../math/Vector3D.h"
#include <vector>
#include <string>
```

Classes

- class Molecule

5.41.1 Detailed Description

An interface describing a way of representing a molecule.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.42 Collision-Code/molecule/StdAtom.cpp File Reference

```
#include "StdAtom.h"
#include "../math/Vector3D.h"
#include "../general/AtomInformations.h"
#include <sstream>
#include <string>
```

5.43 Collision-Code/molecule/StdAtom.h File Reference

Class implementing the interface Atom.h.

```
#include "Atom.h"
#include "../math/Vector3D.h"
```

Classes

- class StdAtom

5.43.1 Detailed Description

Class implementing the interface Atom.h.

Implements an atom.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

11 mars 2016

An atom is representing by a vector in a three-dimensional coordinate, a charge and a symbol.

5.44 Collision-Code/molecule/StdMolecule.cpp File Reference

```
#include "StdMolecule.h"
#include "../general/AtomInformations.h"
#include <sstream>
#include <map>
```

Functions

- const std::string intToString (int number)

5.44.1 Detailed Description

Implements a molecule.

5.44.2 Function Documentation

5.44.2.1 const std::string intToString (int *number*)

Converts an integer to a string.

Parameters

number,the	integer to convert.

Returns

the converted integer in string

5.45 Collision-Code/molecule/StdMolecule.h File Reference

Implements a way of representing a molecule.

```
#include "Atom.h"
#include "Molecule.h"
#include "../math/Vector3D.h"
#include <vector>
#include <string>
```

Classes

- class StdMolecule

5.45.1 Detailed Description

Implements a way of representing a molecule.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.46 Collision-Code/observer/Event.h File Reference

Describes all events that can be launched by the model.

Enumerations

- enum ObservableEvent {
ObservableEvent::GEOMETRIES_LOADED, ObservableEvent::CHARGES_LOADED, ObservableEvent::↩
FILE_SAVED, ObservableEvent::CALCULATIONS_FINISHED,
ObservableEvent::EHSS_STARTED, ObservableEvent::PA_STARTED, ObservableEvent::TM_STARTED,
ObservableEvent::TRAJECTORY_NUMBER_UPDATE,
ObservableEvent::EHSS_ENDED, ObservableEvent::PA_ENDED, ObservableEvent::TM_ENDED, Observable↩
Event::ONE_CALCULATION_FINISHED }

5.46.1 Detailed Description

Describes all events that can be launched by the model.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.46.2 Enumeration Type Documentation

5.46.2.1 enum **ObservableEvent** `[strong]`

Enumerator

GEOMETRIES_LOADED	Launched when geometries are loaded by the model.
CHARGES_LOADED	Launched when a file containing charges is loaded by the model.
FILE_SAVED	Launched when results are saved in a file.
CALCULATIONS_FINISHED	Launched when all calculations on all geometries are finished.
EHSS_STARTED	Launched when a calculation by EHSS method starts.
PA_STARTED	Launched when a calculation by PA method starts.
TM_STARTED	Launched when a calculation by TM method starts.
TRAJECTORY_NUMBER_UPDATE	Launched when some trajectories are calculated by TM method.
EHSS_ENDED	Launched when a calculation by EHSS method ends.
PA_ENDED	Launched when a calculation by PA method ends.
TM_ENDED	Launched when a calculation by TM method ends.
ONE_CALCULATION_FINISHED	Launched when a calculation is finished.

5.47 Collision-Code/observer/Observable.cpp File Reference

```
#include "Observable.h"
#include "Observer.h"
```

5.48 Collision-Code/observer/Observable.h File Reference

Implements the "Observable" part of the pattern Observer/Observable.

```
#include <vector>
#include <algorithm>
#include "Event.h"
```

Classes

- class Observable

5.48.1 Detailed Description

Implements the "Observable" part of the pattern Observer/Observable.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.49 Collision-Code/observer/Observer.cpp File Reference

```
#include "Observer.h"
```

5.50 Collision-Code/observer/Observer.h File Reference

Implements the "Observer" part of the pattern Observer/Observable.

```
#include "Observable.h"
#include "Event.h"
```

Classes

- class Observer

5.50.1 Detailed Description

Implements the "Observer" part of the pattern Observer/Observable.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.51 Collision-Code/observer/state/CalculationState.cpp File Reference

```
#include "CalculationState.h"
#include <cmath>
```

5.52 Collision-Code/observer/state/CalculationState.h File Reference

Describes a state of a calculation by a CalculationOperator instance.

```
#include "../Observable.h"
#include "../../molecule/Molecule.h"
```

Classes

- class CalculationState

5.52.1 Detailed Description

Describes a state of a calculation by a CalculationOperator instance.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

14 may 2016

Allows access to data like finished methods booleans or percentage of progression for TM method.

5.53 Collision-Code/reader/ChargesReader.h File Reference

Interface describing a way to read and charge charges from a .chg file.

```
#include "../molecule/Molecule.h"
#include <vector>
#include <string>
```

Classes

- class ChargesReader

5.53.1 Detailed Description

Interface describing a way to read and charge charges from a .chg file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.54 Collision-Code/reader/ChgChargesReader.cpp File Reference

```
#include "ChgChargesReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include <algorithm>
#include <iterator>
#include "../lib/boost/tokenizer.hpp"
#include "../molecule/Molecule.h"
#include "../molecule/StdMolecule.h"
#include "../molecule/Atom.h"
#include "../molecule/StdAtom.h"
```

5.55 Collision-Code/reader/ChgChargesReader.h File Reference

Implements a way to read and charge charges from a .chg file.

```
#include "ChargesReader.h"
```

Classes

- class ChgChargesReader

5.55.1 Detailed Description

Implements a way to read and charge charges from a .chg file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.56 Collision-Code/reader/ExtractFactory.h File Reference

Interface describing a factory permitting to return the good FileReader to read a certain type of file.

```
#include <string>
#include "FileReader.h"
```

Classes

- class ExtractFactory

5.56.1 Detailed Description

Interface describing a factory permitting to return the good FileReader to read a certain type of file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.57 Collision-Code/reader/ExtractResources.h File Reference

Interface describing a way of read geometries from a file.

```
#include <vector>
#include <string>
#include "../molecule/Molecule.h"
```

Classes

- class ExtractResources

5.57.1 Detailed Description

Interface describing a way of read geometries from a file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.58 Collision-Code/reader/FileReader.h File Reference

An interface describing a way of loading geometries files.

```
#include "../molecule/Molecule.h"
#include <vector>
#include <string>
```

Classes

- class FileReader

5.58.1 Detailed Description

An interface describing a way of loading geometries files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.59 Collision-Code/reader/LogFileReader.cpp File Reference

```
#include "LogFileReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include <vector>
#include "../lib/boost/tokenizer.hpp"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "../general/AtomInformations.h"
```

5.60 Collision-Code/reader/LogFileReader.h File Reference

Implements a way of loading geometries files from .log/.out files.

```
#include "FileReader.h"
```

Classes

- class LogFileReader

5.60.1 Detailed Description

Implements a way of loading geometries files from .log/.out files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.61 Collision-Code/reader/MfjFileReader.cpp File Reference

```
#include "MfjFileReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include "../lib/boost/tokenizer.hpp"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "../general/AtomInformations.h"
```

5.62 Collision-Code/reader/MfjFileReader.h File Reference

Implements a way of loading geometries files from .mfj files.

```
#include "FileReader.h"
```

Classes

- class MfjFileReader

5.62.1 Detailed Description

Implements a way of loading geometries files from .mfj files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.63 Collision-Code/reader/MolFileReader.cpp File Reference

```
#include "MolFileReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include "../lib/boost/tokenizer.hpp"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "../general/AtomInformations.h"
```

5.64 Collision-Code/reader/MolFileReader.h File Reference

Implements a way of loading geometries files from .mol files.

```
#include "FileReader.h"
```

Classes

- class MolFileReader

5.64.1 Detailed Description

Implements a way of loading geometries files from .mol files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.65 Collision-Code/reader/PdbFileReader.cpp File Reference

```
#include "PdbFileReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include <boost/tokenizer.hpp>
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "../general/AtomInformations.h"
```

5.66 Collision-Code/reader/PdbFileReader.h File Reference

Implements a way of loading geometries files from .pdb files.

```
#include "FileReader.h"
```

Classes

- class PdbFileReader

5.66.1 Detailed Description

Implements a way of loading geometries files from .pdb files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.67 Collision-Code/reader/StdExtractFactory.cpp File Reference

```
#include "StdExtractFactory.h"
#include "MolFileReader.h"
#include "PdbFileReader.h"
#include "LogFileReader.h"
#include "XyzFileReader.h"
#include "MfjFileReader.h"
```

Functions

- std::string getFileExt (const std::string &s)

5.67.1 Function Documentation

5.67.1.1 std::string getFileExt (const std::string & *s*)

5.68 Collision-Code/reader/StdExtractFactory.h File Reference

Implements a factory permitting to return the good FileReader to read a certain type of file.

```
#include "ExtractFactory.h"
```

Classes

- class StdExtractFactory

5.68.1 Detailed Description

Implements a factory permitting to return the good FileReader to read a certain type of file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.69 Collision-Code/reader/StdExtractResources.cpp File Reference

```
#include "StdExtractResources.h"
#include "StdExtractFactory.h"
```

5.70 Collision-Code/reader/StdExtractResources.h File Reference

Implements a way of read geometries from a file.

```
#include "ExtractResources.h"
#include "ExtractFactory.h"
```

Classes

- class StdExtractResources

5.70.1 Detailed Description

Implements a way of read geometries from a file.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.71 Collision-Code/reader/XyzFileReader.cpp File Reference

```
#include "XyzFileReader.h"
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <sstream>
#include "../lib/boost/tokenizer.hpp"
#include "../molecule/StdMolecule.h"
#include "../molecule/StdAtom.h"
#include "../general/AtomInformations.h"
```

5.72 Collision-Code/reader/XyzFileReader.h File Reference

Implements a way of loading geometries files from .xyz files.

```
#include "FileReader.h"
```

Classes

- class XyzFileReader

5.72.1 Detailed Description

Implements a way of loading geometries files from .xyz files.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.73 Collision-Code/writer/FileWriter.h File Reference

Interface describing a way of write results and means of results in a stream.

```
#include "../math/Result.h"
#include "../math/Mean.h"
```

Classes

- class FileWriter

5.73.1 Detailed Description

Interface describing a way of write results and means of results in a stream.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

5.74 Collision-Code/writer/StdFileWriter.cpp File Reference

```
#include "StdFileWriter.h"
```

5.75 Collision-Code/writer/StdFileWriter.h File Reference

Implements a way of write results and means of results in a stream.

```
#include "FileWriter.h"
#include <iostream>
```

Classes

- class StdFileWriter

5.75.1 Detailed Description

Implements a way of write results and means of results in a stream.

Author

Anthony Breant, Clement Poinsot, Jeremie Pantin, Mohamed Takhtoukh, Thomas Capet

Version

1.0

Date

23 may 2016

Index