Deeto

Generated by Doxygen 1.8.6

Wed Dec 3 2014 18:03:05

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:

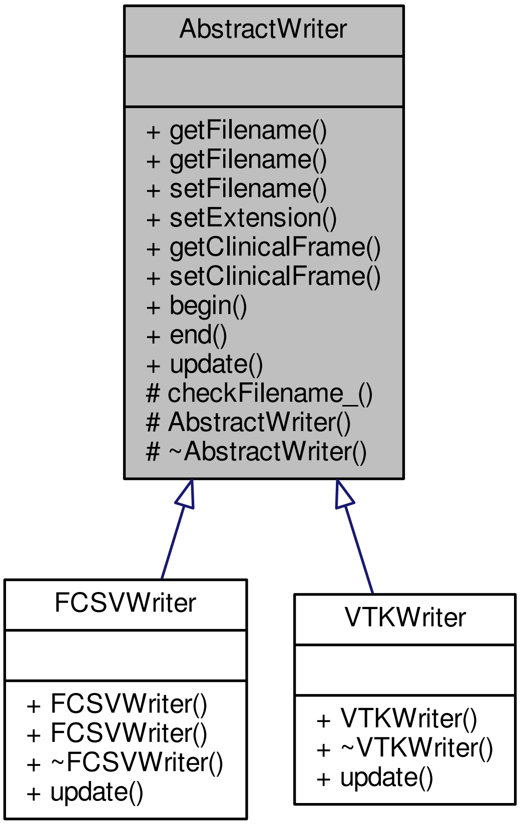
Chapter 4

Class Documentation

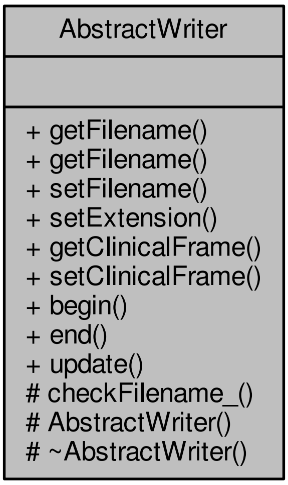
4.1 AbstractWriter Class Reference

```
#include <AbstractWriter.h>
```

Inheritance diagram for AbstractWriter:



Collaboration diagram for AbstractWriter:



Public Member Functions

- const string getFilename () const
- void getFilename (string filename) const
- void setFilename (string filename)
- void setExtension (string ext)
- const ClinicalFrame ∗ getClinicalFrame (void) const
- void setClinicalFrame (ClinicalFrame ∗cf)
- ClinicalFrame::ConstElectrodeIterator begin (void) const
- ClinicalFrame::ConstElectrodeIterator end (void) const
- virtual int update ()=0

Protected Member Functions

- void checkFilename_ (void)
- AbstractWriter ()
- virtual ∼AbstractWriter ()

4.1.1 Detailed Description

AbstractWriter class This class is the base class for each writer. It takes care of filename consistency and it holds the ClinicalFrame pointer.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AbstractWriter::AbstractWriter ( ) [inline], [protected]

Purposely proteced since it is supposed to be istantiated only by its child

4.1.2.2 virtual AbstractWriter::∼AbstractWriter ( ) [inline], [protected], [virtual]

It sets the ClinicalFrame∗ to NULL upon call

4.1.3 Member Function Documentation

4.1.3.1 ClinicalFrame::ConstElectrodeIterator AbstractWriter::begin (void) const [inline]

returns the head of ConstElectrodeIterator to navigate the ClinicalFrame implant details

4.1.3.2 void AbstractWriter::checkFilename_ (void) [inline], [protected]

appends correct extension to filename depending on which subclass has been istantiated

4.1.3.3 ClinicalFrame::ConstElectrodeIterator AbstractWriter::end (void) const [inline]

returns the tail fo ConstElectrodeIterator

4.1.3.4 const ClinicalFrame∗ AbstractWriter::getClinicalFrame (void) const [inline]

4.1.3.5 const string AbstractWriter::getFilename ( ) const [inline]

4.1.3.6 void AbstractWriter::getFilename (string *filename*) const [inline]

4.1.3.7 void AbstractWriter::setClinicalFrame (ClinicalFrame ∗ *cf*) [inline]

4.1.3.8 void AbstractWriter::setExtension (string *ext*) [inline]

4.1.3.9 void AbstractWriter::setFilename (string *filename*) [inline]

4.1.3.10 virtual int AbstractWriter::update ( ) [pure virtual]

pure virtual method that each child should implement depending on file formats

Implemented in FCSVWriter, and VTKWriter.

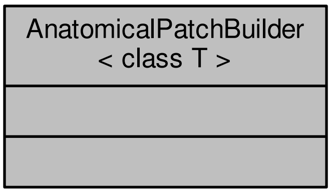
The documentation for this class was generated from the following file:

- AbstractWriter.h

4.2 AnatomicalPatchBuilder< class T > Class Reference

```
#include <AnatomicalPatchBuilder.h>
```

Collaboration diagram for AnatomicalPatchBuilder< class T >:



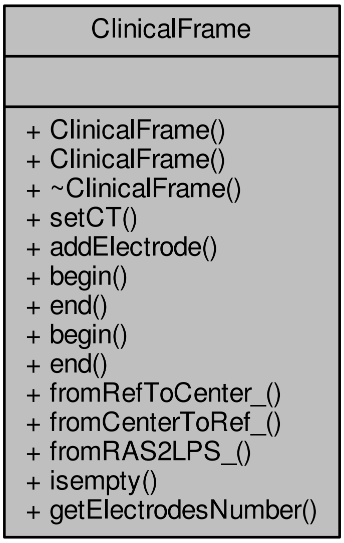
The documentation for this class was generated from the following file:

- AnatomicalPatchBuilder.h

4.3 ClinicalFrame Class Reference

```
#include <ClinicalFrame.h>
```

Collaboration diagram for ClinicalFrame:



Public Types

- typedef vector< Electrode >
  ::iterator ElectrodeIterator

- typedef vector< Electrode >
  ::const_iterator ConstElectrodeIterator

Public Member Functions

- ClinicalFrame (TCLAP::CmdLine ∗)
- ClinicalFrame (void)
- ∼ClinicalFrame (void)
- void setCT (ImagePointerType ct)
- void addElectrode (Electrode e)
- ElectrodeIterator begin ()
- ElectrodeIterator end ()
- ConstElectrodeIterator begin () const
- ConstElectrodeIterator end () const
- void fromRefToCenter_ (PhysicalPointType ∗physicalPoint)
- void fromCenterToRef_ (PhysicalPointType ∗physicalPoint)
- void fromRAS2LPS_ (PhysicalPointType ∗physicalPoint)
- bool isempty (void) const
- int getElectrodesNumber (void) const

4.3.1 Detailed Description

ClinicalFrame class

this class is the central object that holds the information for reconstruction and saves the reconstructed data. It has a pointer to CT data which constitutes the reference space (**Ref. Space**) . Initially, as read from FCSV (3DSlicer format) each Electrode has 2 points entry and target which are represented in a Centered Coordinate system. For this, the class provides methods to transform each point from Ref to Centered and back-

4.3.2 Member Typedef Documentation

4.3.2.1 typedef vector< **Electrode** >::const_iterator **ClinicalFrame::ConstElectrodeIterator**

4.3.2.2 typedef vector< **Electrode** >::iterator **ClinicalFrame::ElectrodeIterator**

4.3.3 Constructor & Destructor Documentation

4.3.3.1 ClinicalFrame::ClinicalFrame (TCLAP::CmdLine ∗) [inline]

4.3.3.2 ClinicalFrame::ClinicalFrame (void) [inline]

4.3.3.3 ClinicalFrame::∼ClinicalFrame (void) [inline]

4.3.4 Member Function Documentation

4.3.4.1 void ClinicalFrame::addElectrode (**Electrode** *e*) [inline]

4.3.4.2 **ElectrodeIterator** ClinicalFrame::begin (void) [inline]

this function returns a pointer to HEAD in vector< Electrode >

4.3.4.3 **ConstElectrodeIterator** ClinicalFrame::begin (void) const [inline]

this function returns a const pointer to HEAD in vector< Electrode >

4.3.4.4 ElectrodeIterator ClinicalFrame::end (void) [inline]

this function returns a pointer to TAIL in vector< Electrode >

4.3.4.5 ConstElectrodeIterator ClinicalFrame::end (void) const [inline]

this function returns a const pointer to TAIL in vector< Electrode >

4.3.4.6 void ClinicalFrame::fromCenterToRef_ (PhysicalPointType $*$ *physicalPoint*)

this function transform a physicalPoint from Centered to Reference space

4.3.4.7 void ClinicalFrame::fromRAS2LPS_ (PhysicalPointType $*$ *physicalPoint*)

this function transform a physicalPoint from LPS to RAS space

4.3.4.8 void ClinicalFrame::fromRefToCenter_ (PhysicalPointType $*$ *physicalPoint*)

this function transform a physicalPoint from Ref to Centered space

4.3.4.9 int ClinicalFrame::getElectrodesNumber (void) const [inline]

4.3.4.10 bool ClinicalFrame::isempty (void) const [inline]

this function returns true or false whether the vector< Electrode> is empty or not

4.3.4.11 void ClinicalFrame::setCT (ImagePointerType *ct*) [inline]

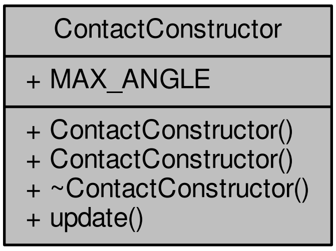
The documentation for this class was generated from the following file:

- ClinicalFrame.h

4.4 ContactConstructor Class Reference

```
#include <ContactConstructor.h>
```

Collaboration diagram for ContactConstructor:



Public Member Functions

- ContactConstructor (ImageType::Pointer ctImage, ClinicalFrame ∗headFrame, TCLAP::CmdLine ∗c)
- ContactConstructor (ImageType::Pointer ctImage, ClinicalFrame ∗headFrame)
- ∼ContactConstructor ()
- void update (void)

Static Public Attributes

- static const double MAX_ANGLE = 0.988

4.4.1 Constructor & Destructor Documentation

4.4.1.1 ContactConstructor::ContactConstructor (ImageType::Pointer *ctImage,* ClinicalFrame ∗ *headFrame,* TCLAP::CmdLine ∗ *c*) [inline]

4.4.1.2 ContactConstructor::ContactConstructor (ImageType::Pointer *ctImage,* ClinicalFrame ∗ *headFrame*) [inline]

4.4.1.3 ContactConstructor::∼ContactConstructor ( ) [inline]

4.4.2 Member Function Documentation

4.4.2.1 void ContactConstructor::update (void)

4.4.3 Member Data Documentation

4.4.3.1 const double ContactConstructor::MAX_ANGLE = 0.988 [static]

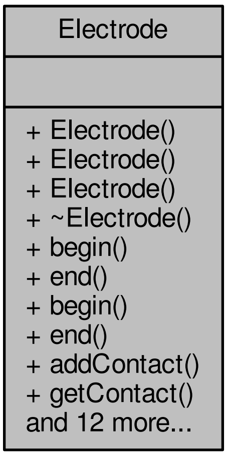
The documentation for this class was generated from the following file:

- ContactConstructor.h

4.5 Electrode Class Reference

```
#include <Electrode.h>
```

Collaboration diagram for Electrode:



Public Types

- typedef PhysicalPointType Contact
- typedef const PhysicalPointType ConstContact
- typedef vector< Contact >::iterator ContactIterator
- typedef vector< Contact >
  ::const_iterator ConstContactIterator

Public Member Functions

- Electrode (string name, Contact &target, Contact &entry, TCLAP::CmdLine ∗c)
- Electrode (string name, Contact &target, Contact &entry)
- Electrode (string id, Contact &target, Contact &entry, ElectrodeModel m)
- ∼Electrode ()
- ContactIterator begin ()
- ContactIterator end ()
- ConstContactIterator begin () const
- ConstContactIterator end () const
- void addContact (Contact c)
- ConstContact ∗ getContact (ulong id) const
- Contact ∗ getContact (ulong id)
- ulong getContactNumber () const
- Contact getTarget () const
- Contact getEntry () const
- void getTargetAsDouble (double ∗t) const
- void getEntryAsDouble (double ∗e) const
- void setTarget (Contact c)
- void setEntry (Contact c)
- string getName () const
- void setName (string name)
- void setModel (ElectrodeModel model)
- ElectrodeModel getModel ()

Friends

- ostream & operator<< (ostream &os, const Electrode &obj)

4.5.1 Detailed Description

This class represents the electrode structure

4.5.2 Member Typedef Documentation

4.5.2.1 typedef const PhysicalPointType Electrode::ConstContact

4.5.2.2 typedef vector< Contact >::const_iterator Electrode::ConstContactIterator

4.5.2.3 typedef PhysicalPointType Electrode::Contact

4.5.2.4 typedef vector< Contact >::iterator Electrode::ContactIterator

4.5.3 Constructor & Destructor Documentation

4.5.3.1 Electrode::Electrode (string *name,* Contact & *target,* Contact & *entry,* TCLAP::CmdLine ∗ *c*) [inline]

4.5.3.2 Electrode::Electrode (string *name,* Contact & *target,* Contact & *entry*) [inline]

4.5.3.3 Electrode::Electrode (string *id,* Contact & *target,* Contact & *entry,* ElectrodeModel *m*)

4.5.3.4 Electrode::∼Electrode ( ) [inline]

4.5.4 Member Function Documentation

4.5.4.1 void Electrode::addContact (Contact *c*) [inline]

4.5.4.2 ContactIterator Electrode::begin (void) [inline]

this method returns a pointer to HEAD of vector< Contact >

4.5.4.3 ConstContactIterator Electrode::begin (void) const [inline]

this method returns a const pointer to HEAD of vector< Contact >

4.5.4.4 ContactIterator Electrode::end (void) [inline]

this method returns a pointer to TAIL of vector< Contact >

4.5.4.5 ConstContactIterator Electrode::end (void) const [inline]

this method returns a const pointer to TAIL of vector< Contact >

4.5.4.6 ConstContact∗ Electrode::getContact (ulong *id*) const [inline]

get const pointer to contact given contact position along vector < Contact >

Parameters

id	the contact index in vector

Returns

NULL pointer in case of overflow (id > vector.size)

4.5.4.7 Contact∗ Electrode::getContact (ulong *id*) [inline]

get pointer to contact given contact position along vector < Contact >

Parameters

id	the contact index in vector

Returns

NULL pointer in case of overflow (id > vector.size)

4.5.4.8 ulong Electrode::getContactNumber ( ) const [inline]

get number of contacts present in vector< Contact >

4.5.4.9 Contact Electrode::getEntry ( ) const [inline]

this function returns the entry point in mm as read from fiducial list

4.5.4.10 void Electrode::getEntryAsDouble (double ∗ *e*) const [inline]

this function converts Contact entry to double[3] entry

4.5.4.11 ElectrodeModel Electrode::getModel ( ) [inline]

4.5.4.12 string Electrode::getName ( ) const [inline]

4.5.4.13 Contact Electrode::getTarget ( ) const [inline]

this function returns the target point in mm as read from fiducial list

4.5.4.14 void Electrode::getTargetAsDouble (double ∗ *t*) const [inline]

this function converts Contact target to double[3] target

4.5.4.15 void Electrode::setEntry (Contact *c*) [inline]

4.5.4.16 void Electrode::setModel (ElectrodeModel *model*)

4.5.4.17 void Electrode::setName (string *name*) [inline]

4.5.4.18 void Electrode::setTarget (Contact *c*) [inline]

4.5.5 Friends And Related Function Documentation

4.5.5.1 ostream& operator<< (ostream & *os*, const Electrode & *obj*) [friend]

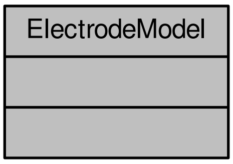
The documentation for this class was generated from the following file:

- Electrode.h

4.6 ElectrodeModel Class Reference

`#include <ElectrodeModel.h>`

Collaboration diagram for ElectrodeModel:



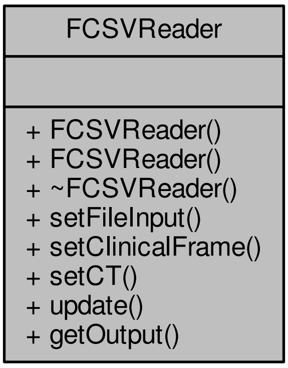
The documentation for this class was generated from the following file:

- ElectrodeModel.h

4.7 FCSVReader Class Reference

`#include <FCSVReader.h>`

Collaboration diagram for FCSVReader:



Public Member Functions

- FCSVReader (TCLAP::CmdLine ∗c)
- FCSVReader (string ∗filein, TCLAP::CmdLine ∗c)
- ∼FCSVReader ()
- void setFileInput (string ∗filein)
- void setClinicalFrame (ClinicalFrame ∗cf)
- void setCT (ImageType::Pointer ctImage)
- int update (void)
- ClinicalFrame ∗ getOutput ()

4.7.1 Detailed Description

ASSUME: CT is nifti, so must be RAS and Ref. for this reason we assume that the fiducial list *have* to be in RAS and Ref format.

NOTICE: ITKReader transform automatically the CT into LPS, so we need to do the same for the fiducial lit.

TODO: FCSVReader legge il file direttamente dalla command line, per cui l'opzione di file reader e' lui che deve aggiungerla. This class reads entry and target points from fiducial file and outputs the clinical frame. this assumes that fiducial data are represented in LPS - Centered space. Usually file constructed with 3DSlicer are defined in this space.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 FCSVReader::FCSVReader (TCLAP::CmdLine ∗ *c*) `[inline]`

4.7.2.2 FCSVReader::FCSVReader (string ∗ *filein,* TCLAP::CmdLine ∗ *c*) `[inline]`

4.7.2.3 FCSVReader::∼FCSVReader ( ) `[inline]`

4.7.3 Member Function Documentation

4.7.3.1 ClinicalFrame∗ FCSVReader::getOutput ( ) `[inline]`

returns a pointer to the constructed ClinicalFrame

4.7.3.2 void FCSVReader::setClinicalFrame (ClinicalFrame ∗ *cf*) `[inline]`

4.7.3.3 void FCSVReader::setCT (ImageType::Pointer *ctImage*)

4.7.3.4 void FCSVReader::setFileInput (string ∗ *filein*) `[inline]`

4.7.3.5 int FCSVReader::update (void)

This function actually reads the fiducial file and populate the ClinicalFrame information it's not important the order of entry/target points till they are represented in LPS-Centered space. This function computes the distance from the center (0,0,0) to understand whether a coordiante triplet is a target or entry point. Comments at the beginning of file are ignored.

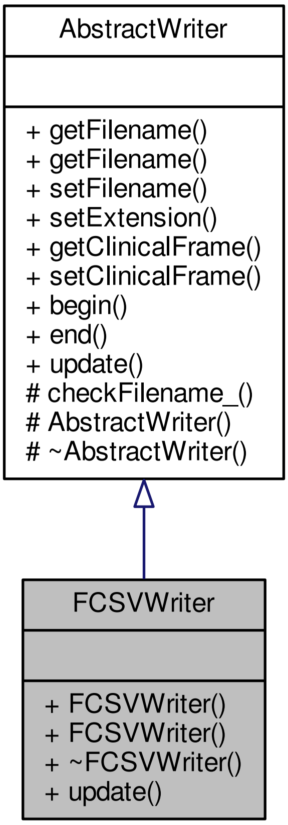
The documentation for this class was generated from the following file:

- FCSVReader.h

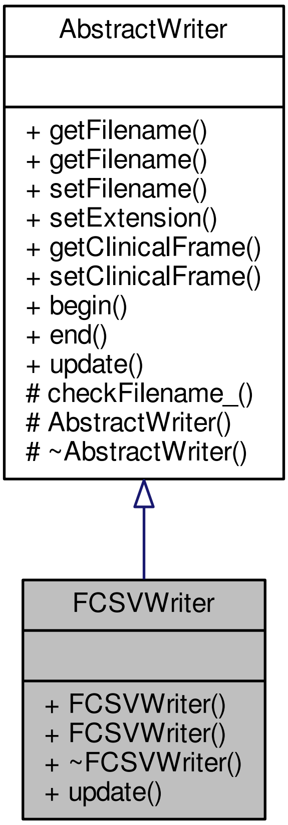
4.8 FCSVWriter Class Reference

```
#include <FCSVWriter.h>
```

Inheritance diagram for FCSVWriter:



Collaboration diagram for FCSVWriter:



Public Member Functions

- FCSVWriter (string filename, TCLAP::CmdLine &cmd)
- FCSVWriter (string filename)
- virtual ∼FCSVWriter (void)
- int update ()

Additional Inherited Members

4.8.1 Detailed Description

FCSVWriter class This class implements 3DSlicer fiducial list for reconstructed data. ATM, only v3 standard is supported. v4 support (which consists in a single file for fiducial point will be handled in next stable release (since v4 support v3 std as retro-comp)

4.8.2 Constructor & Destructor Documentation

4.8.2.1 FCSVWriter::FCSVWriter (string *filename*, TCLAP::CmdLine & *cmd*) [inline]

4.8.2.2 FCSVWriter::FCSVWriter (string *filename*) [inline]

4.8.2.3 virtual FCSVWriter::∼FCSVWriter (void) [inline], [virtual]

4.8.3 Member Function Documentation

4.8.3.1 int FCSVWriter::update (void) [virtual]

it writes down the recontructed data

Implements AbstractWriter.

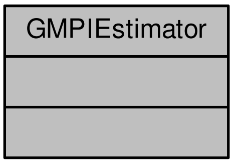
The documentation for this class was generated from the following file:

- FCSVWriter.h

4.9 GMPIEstimator Class Reference

```
#include <GMPIEstimator.h>
```

Collaboration diagram for GMPIEstimator:



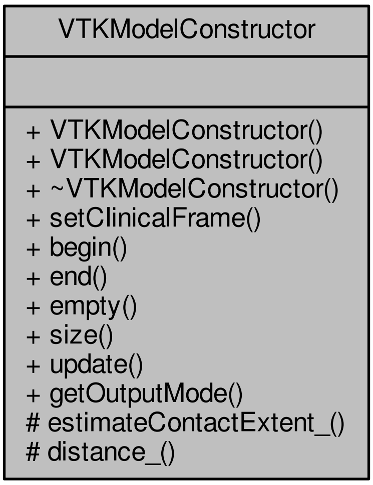
The documentation for this class was generated from the following file:

- GMPIEstimator.h

4.10 VTKModelConstructor Class Reference

```
#include <VTKModelConstructor.h>
```

Collaboration diagram for VTKModelConstructor:



Public Types

- typedef vector
  < vtkSmartPointer< vtkPolyData >
  >::const_iterator ConstModelIterator
- typedef vector
  < vtkSmartPointer< vtkPolyData >
  >::iterator ModelIterator

Public Member Functions

- VTKModelConstructor (const ClinicalFrame ∗cf, bool s)
- VTKModelConstructor (const ClinicalFrame ∗cf)
- ∼VTKModelConstructor (void)
- void setClinicalFrame (ClinicalFrame ∗cf)
- ModelIterator begin (void)
- ModelIterator end (void)
- bool empty (void)
- int size (void) const
- int update ()
- bool getOutputMode (void) const

Protected Member Functions

- void estimateContactExtent_ (double ∗, double ∗, vtkLineSource ∗)
- double distance_ (double ∗p1, double ∗p2)

4.10.1 Detailed Description

VTKModelConstructor class

This class constructs VTK 3D model based on reconstructed information

4.10.2 Member Typedef Documentation

4.10.2.1 typedef vector< vtkSmartPointer<vtkPolyData> >::const_iterator **VTKModelConstructor::ConstModelIterator**

4.10.2.2 typedef vector< vtkSmartPointer<vtkPolyData> >::iterator **VTKModelConstructor::ModelIterator**

4.10.3 Constructor & Destructor Documentation

4.10.3.1 VTKModelConstructor::VTKModelConstructor (const **ClinicalFrame** * *cf*, bool *s*) `[inline]`

Parameters

cf	holds the pointer to ClinicalFrame and to vector< Electrode >
c	is a pointer to TCLAP::CmdLine class to set/get command line options

4.10.3.2 VTKModelConstructor::VTKModelConstructor (const **ClinicalFrame** * *cf*) `[inline]`

4.10.3.3 VTKModelConstructor::∼VTKModelConstructor (void) `[inline]`

4.10.4 Member Function Documentation

4.10.4.1 **ModelIterator** VTKModelConstructor::begin (void) `[inline]`

methods that returns a pointer to the HEAD of VTK model vector

4.10.4.2 double VTKModelConstructor::distance_ (double * *p1*, double * *p2*) `[protected]`

function that computes the euclidean distance between two points

4.10.4.3 bool VTKModelConstructor::empty (void) `[inline]`

methods that check whether VTK model vector is empty

4.10.4.4 **ModelIterator** VTKModelConstructor::end (void) `[inline]`

methods that returns a pointer to the TAIL of VTK model vector

4.10.4.5 void VTKModelConstructor::estimateContactExtent_ (double * *p1*, double * *p2*, vtkLineSource * *line*) `[protected]`

for each contact it estimates its position along the line that connectes contact1 and contact2 ATM the function assumes that p1 and p2 are 3.5 mm apart, better estimation needs to be used in order to create more physically reliable models

4.10.4.6 bool VTKModelConstructor::getOutputMode (void) const [inline]

this function returns whether the user requested a single vtk file (ie all the electrodes together as output or multiple files (ie each electrode as separate vtk file)

4.10.4.7 void VTKModelConstructor::setClinicalFrame (ClinicalFrame ∗ *cf*) [inline]

4.10.4.8 int VTKModelConstructor::size (void) const [inline]

4.10.4.9 int VTKModelConstructor::update (void)

function that computes the euclidean distance between two points core method that navigate the clinical frame, extracts the centroids for each contact, estimates the contact as well as the electrode axes, and builds the vtk model (cylinder + spline) based on reconstructed information

Returns

this function returns 0 or 1 upon failure or success, respectively

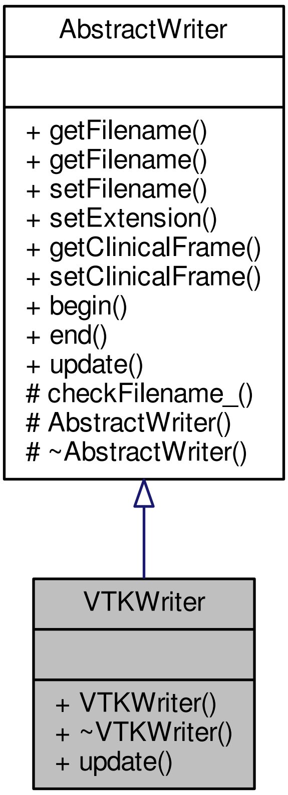
The documentation for this class was generated from the following file:

- VTKModelConstructor.h

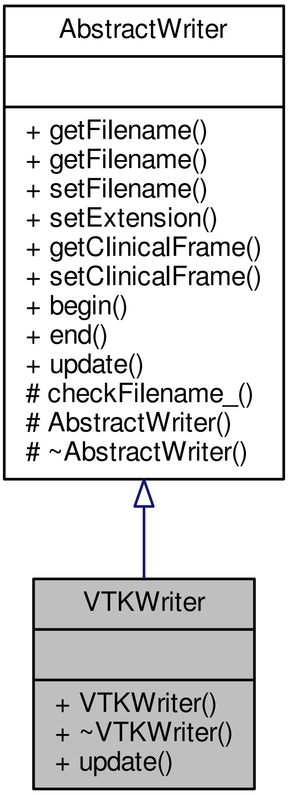
4.11 VTKWriter Class Reference

```
#include <VTKWriter.h>
```

Inheritance diagram for VTKWriter:



Collaboration diagram for VTKWriter:



Public Member Functions

- VTKWriter (string filename, TCLAP::CmdLine &c)
- virtual ∼VTKWriter (void)
- int update ()

Additional Inherited Members

4.11.1 Detailed Description

VTKWriter class

4.11.2 Constructor & Destructor Documentation

4.11.2.1 VTKWriter::VTKWriter (string *filename*, TCLAP::CmdLine & *c*) [inline]

4.11.2.2 virtual VTKWriter::∼VTKWriter (void) [inline], [virtual]

4.11.3 Member Function Documentation

4.11.3.1 int VTKWriter::update (void) [virtual]

implementation of virtual AbstractFileWriter::update This function navigate through vtkModelConstructor output and writes them down

Implements AbstractWriter.

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

- VTKWriter.h

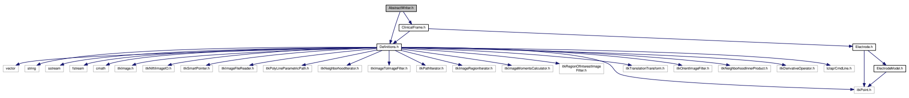
Chapter 5

File Documentation

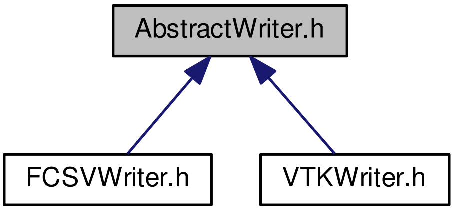
5.1 AbstractWriter.h File Reference

```
#include "Definitions.h"
#include "ClinicalFrame.h"
```

Include dependency graph for AbstractWriter.h:



This graph shows which files directly or indirectly include this file:



Classes

- class AbstractWriter

5.2 AnatomicalPatchBuilder.h File Reference

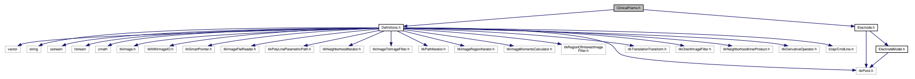
Classes

- class AnatomicalPatchBuilder< class T >

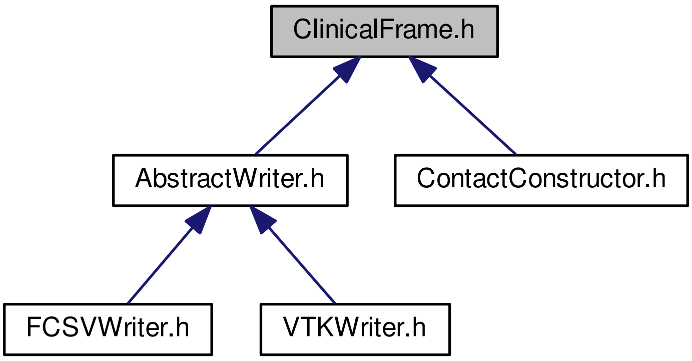
5.3 ClinicalFrame.h File Reference

```
#include <Definitions.h>
#include <Electrode.h>
```

Include dependency graph for ClinicalFrame.h:



This graph shows which files directly or indirectly include this file:



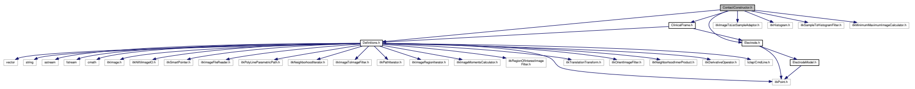
Classes

- class ClinicalFrame

5.4 ContactConstructor.h File Reference

```
#include "Definitions.h"
#include "Electrode.h"
#include "ClinicalFrame.h"
#include "itkImageToListSampleAdaptor.h"
#include "itkHistogram.h"
#include "itkSampleToHistogramFilter.h"
#include "itkMinimumMaximumImageCalculator.h"
```

Include dependency graph for ContactConstructor.h:



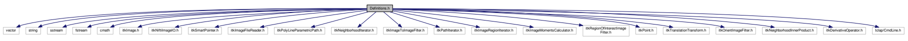
Classes

- class ContactConstructor

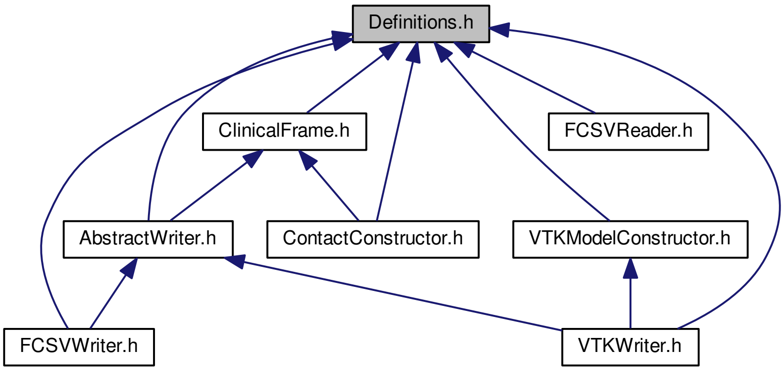
5.5 Definitions.h File Reference

```
#include <vector>
#include <string>
#include <sstream>
#include <fstream>
#include <cmath>
#include <itkImage.h>
#include <itkNiftiImageIO.h>
#include <itkSmartPointer.h>
#include <itkImageFileReader.h>
#include <itkPolyLineParametricPath.h>
#include <itkNeighborhoodIterator.h>
#include <itkImageToImageFilter.h>
#include <itkPathIterator.h>
#include <itkImageRegionIterator.h>
#include <itkImageMomentsCalculator.h>
#include <itkRegionOfInterestImageFilter.h>
#include <itkPoint.h>
#include <itkTranslationTransform.h>
#include <itkOrientImageFilter.h>
#include <itkNeighborhoodInnerProduct.h>
#include <itkDerivativeOperator.h>
#include <tclap/CmdLine.h>
```

Include dependency graph for Definitions.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- typedef itk::Image< short, 3 > ImageType
- typedef ImageType::Pointer ImagePointerType
- typedef itk::Point< double, 3 > PhysicalPointType
- typedef ImageType::IndexType VoxelPointType
- typedef itk::ImageFileReader
  < ImageType > ImageReaderType
- typedef
  itk::ImageMomentsCalculator
  < ImageType > CalculatorType
- typedef ImageType::SizeType SizeType
- typedef ImageType::RegionType RegionType
- typedef ImageType::SpacingType SpacingType
- typedef
  itk::RegionOfInterestImageFilter
  < ImageType, ImageType > FilterType

5.5.1 Typedef Documentation

5.5.1.1 typedef itk::ImageMomentsCalculator<ImageType> CalculatorType

5.5.1.2 typedef itk::RegionOfInterestImageFilter<ImageType,ImageType> FilterType

5.5.1.3 typedef ImageType::Pointer ImagePointerType

5.5.1.4 typedef itk::ImageFileReader< ImageType > ImageReaderType

5.5.1.5 typedef itk::Image<short, 3> ImageType

5.5.1.6 typedef itk::Point<double,3> PhysicalPointType

5.5.1.7 typedef ImageType::RegionType RegionType

5.5.1.8 typedef ImageType::SizeType SizeType

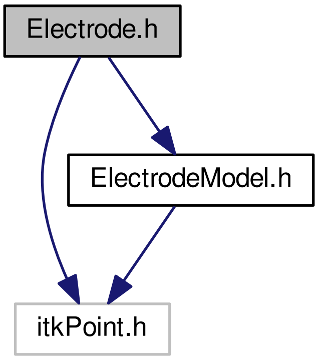
5.5.1.9 typedef ImageType::SpacingType SpacingType

5.5.1.10 typedef ImageType::IndexType VoxelPointType

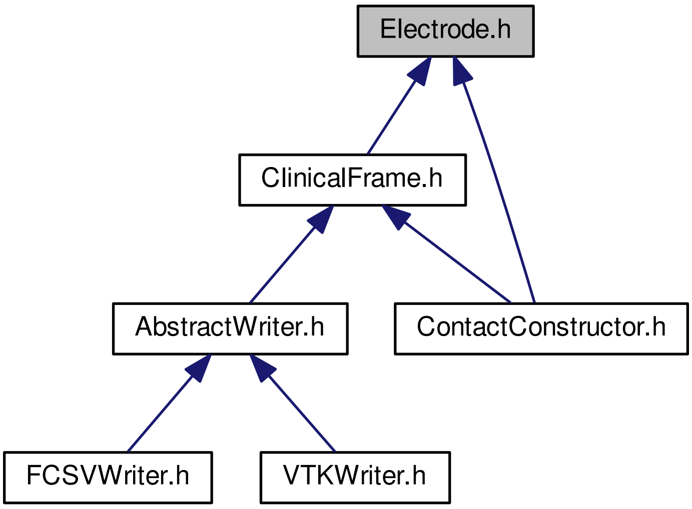
5.6 Electrode.h File Reference

```
#include <itkPoint.h>
#include "ElectrodeModel.h"
```

Include dependency graph for Electrode.h:



This graph shows which files directly or indirectly include this file:



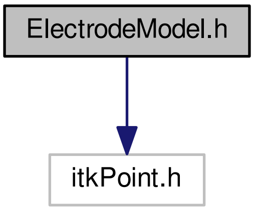
Classes

- class Electrode

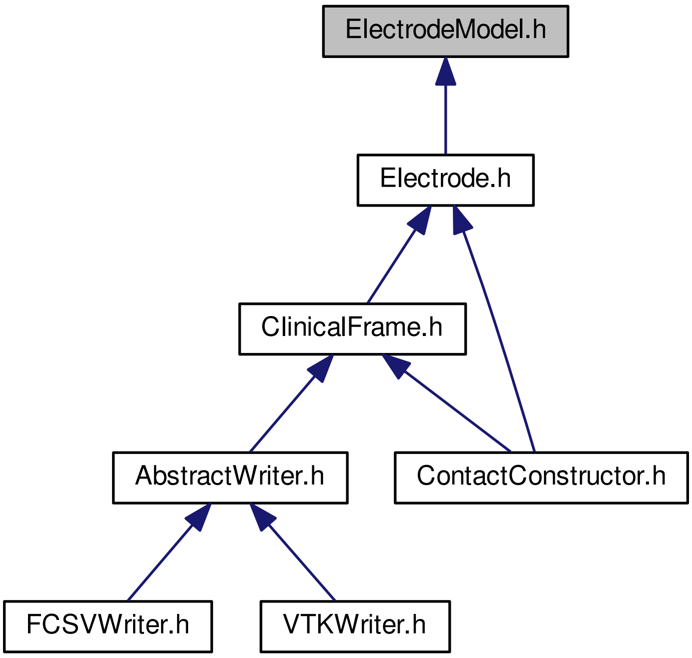
5.7 ElectrodeModel.h File Reference

```
#include <itkPoint.h>
```

Include dependency graph for ElectrodeModel.h:



This graph shows which files directly or indirectly include this file:



Classes

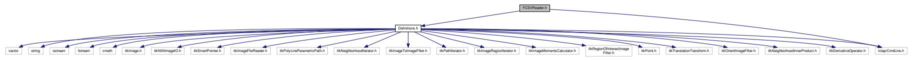
- class ElectrodeModel

5.8 FCSVReader.h File Reference

```
#include <Definitions.h>
```

```
#include <tclap/CmdLine.h>
```

Include dependency graph for FCSVReader.h:



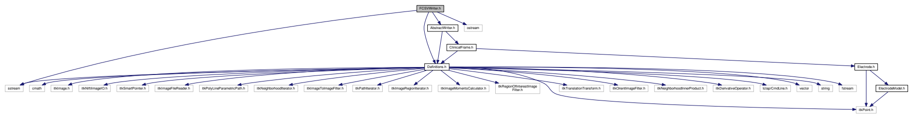
Classes

- class FCSVReader

5.9 FCSVWriter.h File Reference

```
#include "Definitions.h"
#include "AbstractWriter.h"
#include <ostream>
#include <sstream>
```

Include dependency graph for FCSVWriter.h:



Classes

- class FCSVWriter

5.10 GMPIEstimator.h File Reference

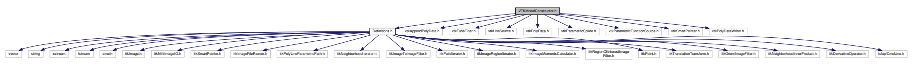
Classes

- class GMPIEstimator

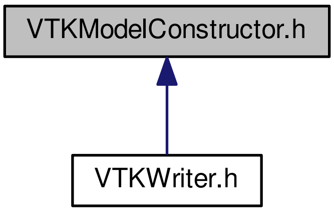
5.11 VTKModelConstructor.h File Reference

```
#include "Definitions.h"
#include <vtkAppendPolyData.h>
#include <vtkTubeFilter.h>
#include <vtkLineSource.h>
#include <vtkPolyData.h>
#include <vtkParametricSpline.h>
#include <vtkParametricFunctionSource.h>
#include <vtkSmartPointer.h>
#include <vtkPolyDataWriter.h>
```

Include dependency graph for VTKModelConstructor.h:



This graph shows which files directly or indirectly include this file:



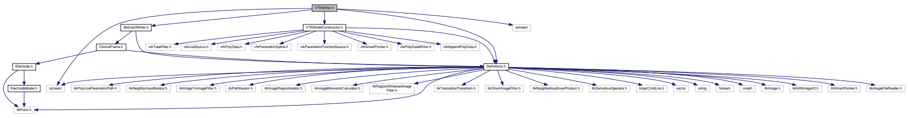
Classes

- class VTKModelConstructor

5.12 VTKWriter.h File Reference

```
#include "Definitions.h"
#include "AbstractWriter.h"
#include "VTKModelConstructor.h"
#include <ostream>
#include <sstream>
```

Include dependency graph for VTKWriter.h:



Classes

- class VTKWriter

Index