PCG Library Reference Manual

1.0

Generated by Doxygen 1.3.9.1

Tue Apr 17 09:39:11 2007

Contents

Chapter 1

PCG Library Module Index

1.1 PCG Library Modules

Here is a list of all modules:

Chapter 2

PCG Library Directory Hierarchy

2.1 PCG Library Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Chapter 3

PCG Library Namespace Index

3.1 PCG Library Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

PCG Library Hierarchical Index

4.1 PCG Library Class Hierarchy

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

Chapter 5

PCG Library Class Index

5.1 PCG Library Class List

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

Chapter 6

PCG Library File Index

6.1 PCG Library File List

Here is a list of all files with brief descriptions:

Chapter 7

PCG Library Module Documentation

7.1 The PCG model layer

The PCG model layer.

Classes

- class model::Analyzer
- class model::Annotation
- class model::AnnotationMapping
- class model::Bone
- class model::Constraint
- class model::Contour
- class model::ExampleModel

  An ExampleModel in PCG.

- class model::Face

  A Face class.

- class model::Fitting
- class model::Landmark
- class model::Mapping
- class model::Model

  A abstract class of a Model in PCG.

- class model::ReferenceModel

An ReferenceModel in PCG.

- class model::Skeleton
- class model::Vertex

  A Vertex class.

7.1.1 Detailed Description

The PCG model layer.

7.2 Functions

The PCG functions layer.

Classes

- class functions::Exporter
- class functions::Importer

  Write brief comment for Importer here.

- class BadConversion

  Write brief comment for BadConversion here.

- class log

  Write brief comment for log here.

Functions

- std::string stringify (const char *c)

  Write brief comment for stringify here.

- template<typename T> void convert (const std::string &s, T &x, bool failIfLeftoverChars=true)

  Write brief comment for convert here.

- template<typename T> T convertTo (const std::string &s, bool failIfLeftoverChars=true)

  Write brief comment for convertTo here.

- void Tokenize (const string &str, vector< string > &tokens, const string &delimiters=" ")

  Write brief comment for Tokenize here.

7.2.1 Detailed Description

The PCG functions layer.

7.2.2 Function Documentation

7.2.2.1 template<typename T> void convert (const std::string & *s*, T & *x*, bool *failIfLeftoverChars* = true) [inline]

Write brief comment for convert here.

Parameters:

s Description of parameter s.

x Description of parameter x.

failIfLeftoverChars Description of parameter failIfLeftoverChars.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for convert here.

Remarks:

Write remarks for convert here.

See also:

Separate items with the '|' character.

Definition at line 91 of file func.h.

Referenced by convertTo().

```
91                                                                                {
92    std::istringstream i(s);
93    char c;
94    if (!(i >> x) || (failIfLeftoverChars && i.get(c))) {
95                  throw BadConversion(s);
96    }
97 }
```

7.2.2.2 template<typename T> T convertTo (const std::string & *s*, bool *failIfLeftoverChars* = true) [inline]

Write brief comment for convertTo here.

Parameters:

s Description of parameter s.

failIfLeftoverChars Description of parameter failIfLeftoverChars.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for convertTo here.

Remarks:
Write remarks for convertTo here.

See also:
Separate items with the '|' character.

Definition at line 124 of file func.h.

References convert().

```
124                                                                          {
125         T x;
126         convert(s, x, failIfLeftoverChars);
127         return x;
128  }
```

Here is the call graph for this function:

7.2.2.3 std::string stringify (const char * *c*) [inline]

Write brief comment for stringify here.

Parameters:
c Description of parameter c.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for stringify here.

Remarks:
Write remarks for stringify here.

See also:
Separate items with the '|' character.

Definition at line 58 of file func.h.

```
59  {
60    std::ostringstream o;
61    if (!(o << c))
62      throw BadConversion("stringify(const char*)");
63    return o.str();
64  }
```

7.2.2.4 void Tokenize (const string & *str*, vector< string > & *tokens*, const string & *delimiters* = " ") [inline]

Write brief comment for Tokenize here.

Parameters:

str Description of parameter str.

tokens Description of parameter tokens.

delimiters Description of parameter delimiters.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Tokenize here.

Remarks:

Write remarks for Tokenize here.

See also:

Separate items with the '|' character.

Definition at line 154 of file func.h.

Referenced by model::Model::load(), model::Analyzer::load(), and functions::Importer::readAttributes().

```
154
155     // Skip delimiters at beginning.
156     string::size_type lastPos = str.find_first_not_of(delimiters, 0);
157     // Find first "non-delimiter".
158     string::size_type pos     = str.find_first_of(delimiters, lastPos);
159
160     while (string::npos != pos || string::npos != lastPos)
161     {
162         // Found a token, add it to the vector.
163                 string tmp = str.substr(lastPos, pos - lastPos);
164                 string::size_type loc = tmp.find( "\n", 0 );
165                 if(loc != string::npos) {
166                         tmp.erase(loc,1);
167                 }
168         tokens.push_back(tmp);
169                 //cout << "Token: " << tokens[tokens.size()-1] << "\n";
170         // Skip delimiters.  Note the "not_of"
171         lastPos = str.find_first_not_of(delimiters, pos);
```

```
172
173         // Find next "non-delimiter"
174         pos = str.find_first_of(delimiters, lastPos);
175
176     }
177 }
```

Chapter 8

PCG Library Directory Documentation

8.1 Functions/ Directory Reference

Directories

- directoryinclude

Files

- file Exporter.cpp
- file Exporter.h
- file Importer.cpp
- file Importer.h
- file log.cpp
- file log.h

8.2 gui/ Directory Reference

Directories

- directorypcggui

Files

- file batch-analyzer.cpp
- file batch-analyzer.h
- file batch-functions.cpp
- file batch-functions.h
- file Batch.cpp
- file Batch.h

8.3 Functions/include/ Directory Reference

Files

- file func.h

8.4 model/ Directory Reference

Files

- file Analyzer.cpp
- file Analyzer.h
- file Annotation.cpp
- file Annotation.h
- file AnnotationMapping.cpp
- file AnnotationMapping.h
- file Bone.cpp
- file Bone.h
- file Constraint.cpp
- file Constraint.h
- file Contour.cpp
- file Contour.h
- file ExampleModel.cpp
- file ExampleModel.h
- file Face.cpp
- file Face.h
- file Fitting.cpp
- file Fitting.h
- file Landmark.cpp
- file Landmark.h
- file Mapping.cpp
- file Mapping.h
- file Model.cpp
- file Model.h
- file ReferenceModel.cpp
- file ReferenceModel.h
- file Skeleton.cpp
- file Skeleton.h
- file Vertex.cpp
- file Vertex.h

8.5 gui/pcggui/ Directory Reference

Files

- file DllEntry.cpp
- file pcggui.cpp
- file pcggui.h
- file resource.h

Chapter 9

PCG Library Namespace Documentation

9.1 functions Namespace Reference

The Functions namespace.

Classes

- class functions::Exporter
- class functions::Importer

  Write brief comment for Importer here.

9.1.1 Detailed Description

The Functions namespace.

9.2 model Namespace Reference

The model namespace.

Classes

- class model::Analyzer
- class model::Annotation
- class model::AnnotationMapping
- class model::Bone
- class model::Constraint
- class model::Contour
- class model::ExampleModel

  An ExampleModel in PCG.

- class model::Face

  A Face class.

- class model::Fitting
- class model::Landmark
- class model::Mapping
- class model::Model

  A abstract class of a Model in PCG.

- class model::ReferenceModel

  An ReferenceModel in PCG.

- class model::Skeleton
- class model::Vertex

  A Vertex class.

9.2.1 Detailed Description

The model namespace.

9.3 std Namespace Reference

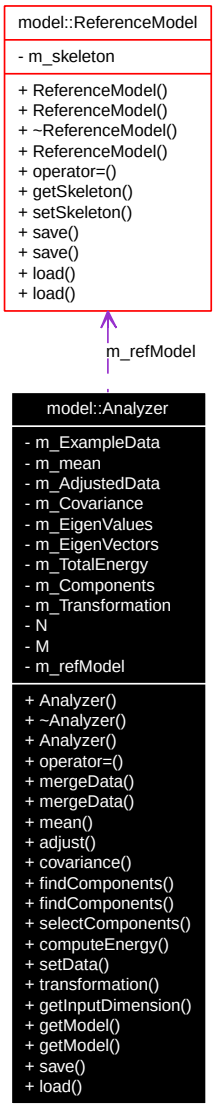
Chapter 10

PCG Library Class Documentation

10.1 model::Analyzer Class Reference

`#include <Analyzer.h>`

Collaboration diagram for model::Analyzer:



Public Types

- enum e_PCAMode { ALL, JACOBI, HOUSEHOLDER, OTHER }

Public Member Functions

- Analyzer ()

  Constructor for the PCA.

- ~Analyzer ()
- Analyzer (const Analyzer &other)

- Analyzer & operator= (const Analyzer &other)
- void mergeData (ReferenceModel refModel, vector< ExampleModel * > lmodels, int size)
- void mergeData (ReferenceModel refModel, vector< ExampleModel * > lmodels)
- void mean ()

  Calculates the emperical mean of a matrix.

- void adjust ()
- void covariance ()
- void findComponents ()
- void findComponents (e_PCAMode mode)
- void selectComponents ()
- void computeEnergy ()
- void setData (Matrix data)
- void transformation ()
- int getInputDimension ()
- ReferenceModel getModel ()
- ReferenceModel getModel (Matrix input)
- bool save (char *filename)
- bool load (char *filename)

Private Attributes

- Matrix m_ExampleData
- Matrix m_mean
- Matrix m_AdjustedData
- Matrix m_Covariance
- SymmetricMatrix m_EigenValues
- Matrix m_EigenVectors
- Matrix m_TotalEnergy
- Matrix m_Components
- Matrix m_Transformation
- int N
- int M
- ReferenceModel m_refModel

10.1.1 Detailed Description

Write brief comment for Analyzer here.

Write detailed description for Analyzer here.

Remarks:

Write remarks for Analyzer here.

See also:
Separate items with the '|' character.

Definition at line 33 of file Analyzer.h.

10.1.2 Member Enumeration Documentation

10.1.2.1 enum model::Analyzer::e_PCAMode

Write brief comment for e_PCAMode here.

Write detailed description for e_PCAMode here.

Remarks:
Write remarks for e_PCAMode here.

See also:
Separate items with the '|' character.

Enumeration values:

ALL All methods

JACOBI Jacobi method

HOUSEHOLDER Householder method

OTHER Other methods

Definition at line 63 of file Analyzer.h.

```
63                                  {
64                          ALL,
65                          JACOBI,
66                          HOUSEHOLDER,
67                          OTHER
68                  };
```

10.1.3 Constructor & Destructor Documentation

10.1.3.1 Analyzer::Analyzer ()

Constructor for the PCA.

This default constructer does stuff and initiates a PCA process

Definition at line 9 of file Analyzer.cpp.

References log::write().

```
10 {
11         log::write("Analyzer Init");
12 }
```

Here is the call graph for this function:

10.1.3.2 Analyzer::~Analyzer ()

Write brief comment for ~Analyzer here.

Exceptions:

<*exception* class> Description of criteria for throwing this exception.

Write detailed description for ~Analyzer here.

Remarks:

Write remarks for ~Analyzer here.

See also:

Separate items with the '|' character.

Definition at line 29 of file Analyzer.cpp.

References m_AdjustedData, m_Covariance, m_EigenValues, m_EigenVectors, m_ExampleData, and m_mean.

```
30 {
31         m_ExampleData = 0;
32         m_mean = 0;
33         m_AdjustedData = 0;
34         m_Covariance = 0;
35         m_EigenValues = 0;
36         m_EigenVectors = 0;
37 }
```

10.1.3.3 Analyzer::Analyzer (const Analyzer & *other*)

Write brief comment for Analyzer here.

Parameters:

other Description of parameter other.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for Analyzer here.

Remarks:
Write remarks for Analyzer here.

See also:
Separate items with the '|' character.

Definition at line 583 of file Analyzer.cpp.

References M, m_AdjustedData, m_Components, m_Covariance, m_EigenValues, m_EigenVectors, m_ExampleData, m_mean, m_refModel, m_TotalEnergy, m_Transformation, and N.

```
583                                         {
584         m_ExampleData = other.m_ExampleData;
585         m_mean = other.m_mean;
586         m_AdjustedData = other.m_AdjustedData;
587         m_Covariance = other.m_Covariance;
588         m_EigenValues = other.m_EigenValues;
589         m_EigenVectors = other.m_EigenVectors;
590         m_TotalEnergy = other.m_TotalEnergy;
591         m_Components = other.m_Components;
592         m_Transformation = other.m_Transformation;
593         N = other.N;
594         M = other.M;
595         m_refModel = other.m_refModel;
596 }
```

10.1.4 Member Function Documentation

10.1.4.1 void Analyzer::adjust ()

Write brief comment for adjust here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for adjust here.

Remarks:
Write remarks for adjust here.

See also:
Separate items with the '|' character.

Definition at line 179 of file Analyzer.cpp.

References m_AdjustedData, m_ExampleData, m_mean, N, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
180 {
181         log::write("Analyzer adjust");
182         Matrix h = Matrix(1,N);
183         h = 1;
184         /*cout << setw(10) << setprecision(5) << m_ExampleData;
185         cout << setw(10) << setprecision(5) << m_mean;
186         cout << setw(10) << setprecision(5) << h;
187         cout << setw(10) << setprecision(5) << (m_mean * h);*/
188         m_AdjustedData = m_ExampleData - (m_mean * h);
189         log::write("Adjusted Data",m_AdjustedData);
190 }
```

Here is the call graph for this function:

10.1.4.2 void Analyzer::computeEnergy ()

Write brief comment for computeEnergy here.

Exceptions:

<*exception* class> Description of criteria for throwing this exception.

Write detailed description for computeEnergy here.

Remarks:

Write remarks for computeEnergy here.

See also:

Separate items with the '|' character.

Definition at line 301 of file Analyzer.cpp.

References M, m_EigenValues, m_TotalEnergy, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
302 {
303         log::write("Analyzer compute energy");
304         //Requires that EigenVectors and Values are sorted pair-wise. Should be done
305         //by the matrix library i believe (this should be tested somehow)
306
307         m_TotalEnergy = Matrix(M,1); // [ M x 1 ]
```

```
308         m_EigenValues = m_EigenValues.Reverse();
309         m_TotalEnergy(1,1) = m_EigenValues(1,1);
310         for(int i=2;i<=M;i++)
311         {
312                 m_TotalEnergy(i,1) = m_TotalEnergy(i-1,1) + m_EigenValues(i,i);
313         }
314         log::write("Total Energy",m_TotalEnergy);
315 }
```

Here is the call graph for this function:

10.1.4.3 void Analyzer::covariance ()

Write brief comment for covariance here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for covariance here.

Remarks:

Write remarks for covariance here.

See also:

Separate items with the '|' character.

Definition at line 207 of file Analyzer.cpp.

References m_AdjustedData, m_Covariance, N, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
208 {
209         log::write("Analyzer covariance");
210         m_Covariance = m_AdjustedData/(N-1) * m_AdjustedData.t();
211         log::write("Covariance",m_Covariance);
212 }
```

Here is the call graph for this function:

10.1.4.4 void Analyzer::findComponents (e_PCAMode *mode*)

Write brief comment for findComponents here.

Parameters:
mode Description of parameter mode.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for findComponents here.

Remarks:
Write remarks for findComponents here.

See also:
Separate items with the '|' character.

Definition at line 252 of file Analyzer.cpp.

References HOUSEHOLDER, JACOBI, M, m_EigenValues, m_EigenVectors, OTHER, and log::write().

```
253 {
254         log::write("Analyzer find components");
255         DiagonalMatrix dm(M);
256         SymmetricMatrix test;
257         test << m_Covariance;
258 
259         switch (mode)
260         {
261           case JACOBI:
262             log::write("Calculating with Jacobi");
263             Jacobi(test, dm, m_EigenVectors);
264             break;
265           case HOUSEHOLDER:
266             log::write("Calculating with Householder");
267             EigenValues(test, dm, m_EigenVectors);
268             break;
269           case OTHER:
270             log::write("Calculating with Jacobi (matched OTHER)");
271             Jacobi(test, dm, m_EigenVectors);
272             break;
273           default:
274             log::write("Calculating with Jacobi (matched default)");
275             Jacobi(test, dm, m_EigenVectors);
276             break;
277         }
278         //Jacobi(test, dm, m_EigenVectors);
279         //EigenValues(test, dm, m_EigenVectors);
280         m_EigenValues << dm;
281 
282         log::write("Eigen Vectors",m_EigenVectors);
283         log::write("Eigen Values",m_EigenValues);
284 }
```

Here is the call graph for this function:

10.1.4.5 void Analyzer::findComponents ()

Write brief comment for findComponents here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for findComponents here.

Remarks:
Write remarks for findComponents here.

See also:
Separate items with the '|' character.

Definition at line 229 of file Analyzer.cpp.

References JACOBI.

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
230 {
231   findComponents(JACOBI);
232 }
```

10.1.4.6 int Analyzer::getInputDimension ()

Write brief comment for getInputDimension here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getInputDimension here.

Remarks:
Write remarks for getInputDimension here.

See also:
Separate items with the '|' character.

Definition at line 450 of file Analyzer.cpp.

References m_Components.

Referenced by Batch::getRandomModels().

```
450                                 {
451         return m_Components.Ncols();
452 }
```

10.1.4.7 ReferenceModel Analyzer::getModel (Matrix *input*)

Write brief comment for getModel here.

Parameters:
input Description of parameter input.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getModel here.

Remarks:
Write remarks for getModel here.

See also:
Separate items with the '|' character.

Definition at line 421 of file Analyzer.cpp.

References m_Components, m_mean, m_refModel, N, model::Model::updateVertices(), and log::write().

```
421                                                 {
422     log::write("Analyzer::getModel\n");
423         Matrix h = Matrix(1,N);
424         h = 1;
425         Matrix in = m_Components*input+(m_mean * h);
426         log::write("Analyzer-in",in);
427         m_refModel.updateVertices(in);
428         log::write("Analyzer-refModel",m_refModel);
429         return m_refModel;
430 }
```

Here is the call graph for this function:

10.1.4.8 ReferenceModel Analyzer::getModel ()

Write brief comment for getModel here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getModel here.

Remarks:
Write remarks for getModel here.

See also:
Separate items with the '|' character.

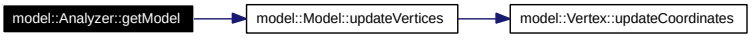
Definition at line 393 of file Analyzer.cpp.

References m_Components, m_mean, m_refModel, m_Transformation, N, and model::Model::updateVertices().

Referenced by Batch::getRandomModels().

```
393                                     {
394         Matrix h = Matrix(1,N);
395         h = 1;
396         m_refModel.updateVertices(m_Components*m_Transformation+(m_mean*h));
397         return m_refModel;
398 }
```

Here is the call graph for this function:



10.1.4.9 bool Analyzer::load (char * *filename*)

Write brief comment for load here.

Parameters:
filename Description of parameter filename.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for load here.

Remarks:
Write remarks for load here.

See also:
Separate items with the '|' character.

Definition at line 513 of file Analyzer.cpp.

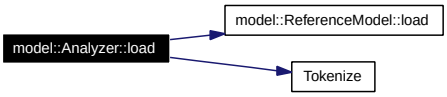
References model::ReferenceModel::load(), m_Components, m_mean, m_refModel, and Tokenize().

Referenced by pcggui::LoadAnalyzer().

```
513                                     {
514         m_mean.CleanUp();
515         m_Components.CleanUp();
516         m_refModel = model::ReferenceModel();
517 
518         string line;
519         vector<string> tokens;
520         vector<string> coords;
521 
522         ifstream loadFile(filename);
523         if(loadFile.is_open()) {
524                 getline(loadFile,line);//Mean Dim
525                 getline(loadFile,line);//Dim
526                 tokens.clear();
527                 Tokenize(line,tokens,",");
528                 m_mean = Matrix(convertTo<int>(tokens[0]),convertTo<int>(tokens[1]));
529 
530                 for(int i=0;i<m_mean.Nrows();i++) {
531                         getline(loadFile,line);//Data
532                         tokens.clear();
533                         Tokenize(line,tokens," ");
534                         for(int n=0;n<m_mean.Ncols();n++) {
535                                 m_mean(i+1,n+1) = convertTo<float>(tokens[n]);
536                         }
537                 }
538 
539                 getline(loadFile,line);//
540                 getline(loadFile,line);//Components Dim
```

```
541                 getline(loadFile,line);//Dim
542                 tokens.clear();
543                 Tokenize(line,tokens,",");
544                 m_Components = Matrix(convertTo<int>(tokens[0]),convertTo<int>(tokens[1]));
545
546                 for(int i=0;i<m_Components.Nrows();i++) {
547                         getline(loadFile,line);//Data
548                         tokens.clear();
549                         Tokenize(line,tokens," ");
550                         for(int n=0;n<m_Components.Ncols();n++) {
551                                 m_Components(i+1,n+1) = convertTo<float>(tokens[n]);
552                         }
553                 }
554
555                 getline(loadFile,line);//
556                 m_refModel.load(&loadFile);
557                 loadFile.close();
558                 return true;
559         } else {
560                 return false;
561         }
562 }
```

Here is the call graph for this function:



10.1.4.10 void Analyzer::mean ()

Calculates the emperical mean of a matrix.

Write brief comment for mean here.

Exceptions:

<*exception* class> Description of criteria for throwing this exception.

Write detailed description for mean here.

Remarks:

Write remarks for mean here.

See also:

Separate items with the '|' character.

Definition at line 150 of file Analyzer.cpp.

References M, m_ExampleData, m_mean, N, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
151 {
152         log::write("Analyzer mean");
153         m_mean = Matrix(M,1); //[M x 1]
154         for(int i=1;i <= M; i++)
155         {
156                 /*    printf("Integer %d\n", i);
157                 printf("Cols: %d\n", data.Ncols());
158                 printf("Sum: %f\n", data.Row(i).Sum());*/
159                 m_mean(i,1) = 1.0/N * m_ExampleData.Row(i).Sum();
160         }
161         log::write("Mean",m_mean);
162 }
```

Here is the call graph for this function:

10.1.4.11 void Analyzer::mergeData (ReferenceModel *refModel*, vector< ExampleModel * > *lmodels*)

Write brief comment for mergeData here.

Parameters:

refModel Description of parameter refModel.

lmodels Description of parameter lmodels.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for mergeData here.

Remarks:

Write remarks for mergeData here.

See also:

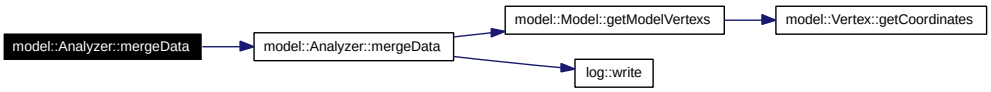
Separate items with the '|' character.

Definition at line 60 of file Analyzer.cpp.

References mergeData().

```
60                                                                      {
61         mergeData(refModel, lmodels,(int)lmodels.size());
62 }
```

Here is the call graph for this function:



10.1.4.12 void Analyzer::mergeData (ReferenceModel *refModel*, vector< ExampleModel * > *lmodels*, int *size*)

Write brief comment for mergeData here.

Parameters:

refModel Description of parameter refModel.

lmodels Description of parameter lmodels.

size Description of parameter size.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for mergeData here.

Remarks:

Write remarks for mergeData here.

See also:

Separate items with the '|' character.

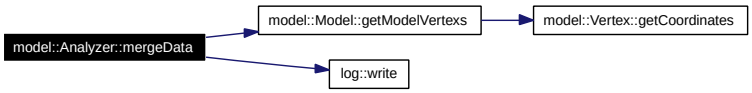
Definition at line 88 of file Analyzer.cpp.

References model::Model::getModelVertexs(), M, m_ExampleData, m_refModel, N, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), mergeData(), and Batch::TestAnalyzer().

```
89 {
90         log::write("Analyzer merge data");
91         log::write("Number of models",size);
92         m_refModel = refModel;
93         for(int i=0;i<size;i++) {
94                 Matrix mv = lmodels[i]->getModelVertexs();
95                 if(i==0) {
96                         m_ExampleData = mv;
97                 } else {
98                         m_ExampleData = m_ExampleData | mv;
99                 }
100          }
101          N = m_ExampleData.Ncols();
102          M = m_ExampleData.Nrows();
103          log::write("N",N);
104          log::write("M",M);
105          log::write("Example Data",m_ExampleData);
106 }
```

Here is the call graph for this function:



10.1.4.13 Analyzer & Analyzer::operator= (const Analyzer & *other*)

Write brief comment for operator = here.

Parameters:

other Description of parameter other.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:

Write remarks for operator = here.

See also:

Separate items with the '|' character.

Definition at line 620 of file Analyzer.cpp.

References M, m_AdjustedData, m_Components, m_Covariance, m_EigenValues, m_EigenVectors, m_ExampleData, m_mean, m_refModel, m_TotalEnergy, m_-Transformation, and N.

```
621 {
622     // if same object
623     if ( this == &other )
624         return *this;
625 
626         m_ExampleData = other.m_ExampleData;
627         m_mean = other.m_mean;
628         m_AdjustedData = other.m_AdjustedData;
629         m_Covariance = other.m_Covariance;
630         m_EigenValues = other.m_EigenValues;
631         m_EigenVectors = other.m_EigenVectors;
632         m_TotalEnergy = other.m_TotalEnergy;
633         m_Components = other.m_Components;
634         m_Transformation = other.m_Transformation;
635         N = other.N;
```

```
636         M = other.M;
637         m_refModel = other.m_refModel;
638
639     return *this;
640 }
```

10.1.4.14 bool Analyzer::save (char * *filename*)

Write brief comment for save here.

Parameters:
filename Description of parameter filename.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for save here.

Remarks:
Write remarks for save here.

See also:
Separate items with the '|' character.

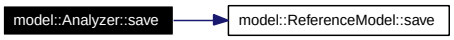
Definition at line 475 of file Analyzer.cpp.

References m_Components, m_mean, m_refModel, and model::ReferenceModel::save().

Referenced by pcggui::SaveAnalyzer().

```
475                                 {
476         ofstream saveFile(filename);
477         if(saveFile.is_open()) {
478                 saveFile << "Mean Dim\n";
479                 saveFile << m_mean.Nrows() << "," << m_mean.Ncols() << "\n";
480                 saveFile << m_mean << "\n";
481                 saveFile << "Components Dim\n";
482                 saveFile << m_Components.Nrows() << "," << m_Components.Ncols() << "\n";
483                 saveFile << m_Components << "\n";
484                 m_refModel.save(&saveFile);
485                 saveFile.close();
486                 return true;
487         } else {
488                 return false;
489         }
490 }
```

Here is the call graph for this function:



10.1.4.15 void Analyzer::selectComponents ()

Write brief comment for selectComponents here.

Exceptions:

<*exception* class> Description of criteria for throwing this exception.

Write detailed description for selectComponents here.

Remarks:

Write remarks for selectComponents here.

See also:

Separate items with the '|' character.

Definition at line 333 of file Analyzer.cpp.

References m_Components, m_EigenVectors, m_TotalEnergy, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
334 {
335         log::write("Analyzer select Components");
336         int index = 1;
337         float percent = 0.0;
338 
339         do
340         {
341                 percent = ((m_TotalEnergy(1,m_TotalEnergy.Ncols()) - m_TotalEnergy(1,index)) * m_TotalE
342                 log::write("Percent calc",percent);
343                 index++;
344         }
345         while(percent <= 90.0 && percent > 0.0);
346 
347         m_Components = m_EigenVectors.Reverse().Columns(1,index-1).Reverse();
348         log::write("Components",m_Components);
349 }
```

Here is the call graph for this function:

10.1.4.16 void Analyzer::setData (Matrix *data*)

Write brief comment for setData here.

Parameters:
data Description of parameter data.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setData here.

Remarks:
Write remarks for setData here.

See also:
Separate items with the '|' character.

Definition at line 126 of file Analyzer.cpp.

References m_ExampleData.

```
127 {
128         m_ExampleData = data;
129 }
```

10.1.4.17 void Analyzer::transformation ()

Write brief comment for transformation here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for transformation here.

Remarks:
Write remarks for transformation here.

See also:
Separate items with the '|' character.

Definition at line 367 of file Analyzer.cpp.

References m_Components, m_Transformation, and log::write().

Referenced by pcggui::Analyze(), Batch::analyzeModel(), and Batch::TestAnalyzer().

```
368 {
369         log::write("Analyzer Transformation Matrix");
370         // m_Covariance should perhaps be normalized - see wikipedia for info.
371         m_Transformation = m_Components.t() * m_AdjustedData;
372         log::write("Transformation",m_Transformation);
373 }
```

Here is the call graph for this function:

10.1.5 Member Data Documentation

10.1.5.1 int model::Analyzer::M [private]

Definition at line 44 of file Analyzer.h.

Referenced by Analyzer(), computeEnergy(), findComponents(), mean(), mergeData(), and operator=().

10.1.5.2 Matrix model::Analyzer::m_AdjustedData [private]

Definition at line 37 of file Analyzer.h.

Referenced by adjust(), Analyzer(), covariance(), operator=(), and ~Analyzer().

10.1.5.3 Matrix model::Analyzer::m_Components [private]

Definition at line 42 of file Analyzer.h.

Referenced by Analyzer(), getInputDimension(), getModel(), load(), operator=(), save(), selectComponents(), and transformation().

10.1.5.4 Matrix model::Analyzer::m_Covariance [private]

Definition at line 38 of file Analyzer.h.

Referenced by Analyzer(), covariance(), operator=(), and ~Analyzer().

10.1.5.5 SymmetricMatrix model::Analyzer::m_EigenValues [private]

Definition at line 39 of file Analyzer.h.

Referenced by Analyzer(), computeEnergy(), findComponents(), operator=(), and ~Analyzer().

10.1.5.6 Matrix model::Analyzer::m_EigenVectors [private]

Definition at line 40 of file Analyzer.h.

Referenced by Analyzer(), findComponents(), operator=(), selectComponents(), and ∼Analyzer().

10.1.5.7 Matrix model::Analyzer::m_ExampleData [private]

Definition at line 35 of file Analyzer.h.

Referenced by adjust(), Analyzer(), mean(), mergeData(), operator=(), setData(), and ∼Analyzer().

10.1.5.8 Matrix model::Analyzer::m_mean [private]

Definition at line 36 of file Analyzer.h.

Referenced by adjust(), Analyzer(), getModel(), load(), mean(), operator=(), save(), and ∼Analyzer().

10.1.5.9 ReferenceModel model::Analyzer::m_refModel [private]

Definition at line 45 of file Analyzer.h.

Referenced by Analyzer(), getModel(), load(), mergeData(), operator=(), and save().

10.1.5.10 Matrix model::Analyzer::m_TotalEnergy [private]

Definition at line 41 of file Analyzer.h.

Referenced by Analyzer(), computeEnergy(), operator=(), and selectComponents().

10.1.5.11 Matrix model::Analyzer::m_Transformation [private]

Definition at line 43 of file Analyzer.h.

Referenced by Analyzer(), getModel(), operator=(), and transformation().

10.1.5.12 int model::Analyzer::N [private]

Definition at line 44 of file Analyzer.h.

Referenced by adjust(), Analyzer(), covariance(), getModel(), mean(), mergeData(), and operator=().

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

- model/Analyzer.h

- model/Analyzer.cpp

10.2 model::Annotation Class Reference

`#include <Annotation.h>`

Public Member Functions

- Annotation ()
- Annotation (int id, int score)
- Annotation (int id, int score, string name)
- ~Annotation ()
- void setId (int id)
- void setScore (int score)
- void setName (string name)
- int getId ()
- int getScore ()
- string getName ()
- Annotation (const Annotation &other)
- Annotation & operator= (const Annotation &other)
- bool operator== (const Annotation &other) const
- bool operator!= (const Annotation &other) const

Private Attributes

- int m_iId
- int m_iScore
- string m_sName

10.2.1 Detailed Description

Write brief comment for Annotation here.

Write detailed description for Annotation here.

Remarks:
Write remarks for Annotation here.

See also:
Separate items with the '|' character.

Definition at line 29 of file Annotation.h.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 model::Annotation::Annotation () [inline]

Write brief comment for Annotation here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Annotation here.

Remarks:
Write remarks for Annotation here.

See also:
Separate items with the '|' character.

Definition at line 51 of file Annotation.h.

Referenced by Annotation().

```
51 {};
```

10.2.2.2 Annotation::Annotation (int *id*, int *score*)

Write brief comment for Annotation here.

Parameters:
id Description of parameter id.

score Description of parameter score.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Annotation here.

Remarks:
Write remarks for Annotation here.

See also:
Separate items with the '|' character.

Definition at line 26 of file Annotation.cpp.

References m_iId, and m_iScore.

```
26                                          {
27         m_iId = id;
28         m_iScore = score;
29 }
```

10.2.2.3 Annotation::Annotation (int *id*, int *score*, string *name*)

Write brief comment for Annotation here.

Parameters:

id Description of parameter id.

score Description of parameter score.

name Description of parameter name.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Annotation here.

Remarks:

Write remarks for Annotation here.

See also:

Separate items with the '|' character.

Definition at line 55 of file Annotation.cpp.

References Annotation(), and m_sName.

```
55                                                  {
56         Annotation(id,score);
57         m_sName = name;
58 }
```

Here is the call graph for this function:

10.2.2.4 model::Annotation::~Annotation () [inline]

Write brief comment for ~Annotation here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Annotation here.

Remarks:
Write remarks for ~Annotation here.

See also:
Separate items with the '|' character.

Definition at line 69 of file Annotation.h.

```
69 {};
```

10.2.2.5 Annotation::Annotation (const Annotation & *other*)

Write brief comment for Annotation here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Annotation here.

Remarks:
Write remarks for Annotation here.

See also:
Separate items with the '|' character.

Definition at line 210 of file Annotation.cpp.

References m_iId, m_iScore, and m_sName.

```
210                                                 {
211         m_iId = other.m_iId;
212         m_iScore = other.m_iScore;
213         m_sName = other.m_sName;
214 }
```

10.2.3 Member Function Documentation

10.2.3.1 int Annotation::getId ()

Write brief comment for getId here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getId here.

Remarks:
Write remarks for getId here.

See also:
Separate items with the '|' character.

Definition at line 144 of file Annotation.cpp.

```
144                         {
145         return m_iId;
146 }
```

10.2.3.2 string Annotation::getName ()

Write brief comment for getName here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getName here.

Remarks:
Write remarks for getName here.

See also:
Separate items with the '|' character.

Definition at line 188 of file Annotation.cpp.

```
188                             {
189         return m_sName;
190 }
```

10.2.3.3 int Annotation::getScore ()

Write brief comment for getScore here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getScore here.

Remarks:
Write remarks for getScore here.

See also:
Separate items with the '|' character.

Definition at line 166 of file Annotation.cpp.

```
166                             {
167         return m_iScore;
168 }
```

10.2.3.4 bool Annotation::operator!= (const Annotation & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 303 of file Annotation.cpp.

References m_iId, m_iScore, and m_sName.

```
304 {
305         if( m_iId == other.m_iId
306         && m_iScore == other.m_iScore
307         && m_sName == other.m_sName ) {
308         return false;
309         }
310     return true;
311 }
```

10.2.3.5 Annotation & Annotation::operator= (const Annotation & *other*)

Write brief comment for operator = here.

Parameters:

other Description of parameter other.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:

Write remarks for operator = here.

See also:

Separate items with the '|' character.

Definition at line 237 of file Annotation.cpp.

References m_iId, m_iScore, and m_sName.

```
238 {
239     // if same object
240     if ( this == &other )
241         return *this;
242 
243         m_iId = other.m_iId;
244         m_iScore = other.m_iScore;
245         m_sName = other.m_sName;
246 
247     return *this;
248 }
```

10.2.3.6 bool Annotation::operator== (const Annotation & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 271 of file Annotation.cpp.

References m_iId, m_iScore, and m_sName.

```
272 {
273
274     if( m_iId == other.m_iId
275         && m_iScore == other.m_iScore
276         && m_sName == other.m_sName ) {
277         return true;
278         }
279     return false;
280 }
```

10.2.3.7 void Annotation::setId (int *id*)

Write brief comment for setId here.

Parameters:
id Description of parameter id.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setId here.

Remarks:
Write remarks for setId here.

See also:
Separate items with the '|' character.

Definition at line 78 of file Annotation.cpp.

References m_iId.

```
78                                {
79          m_iId = id;
80 }
```

10.2.3.8 void Annotation::setName (string *name*)

Write brief comment for setName here.

Parameters:
name Description of parameter name.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setName here.

Remarks:
Write remarks for setName here.

See also:
Separate items with the '|' character.

Definition at line 122 of file Annotation.cpp.

References m_sName.

```
122                                      {
123          m_sName = name;
124 }
```

10.2.3.9 void Annotation::setScore (int *score*)

Write brief comment for setScore here.

Parameters:
score Description of parameter score.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setScore here.

Remarks:
Write remarks for setScore here.

See also:
Separate items with the '|' character.

Definition at line 100 of file Annotation.cpp.

References m_iScore.

```
100                                     {
101         m_iScore = score;
102 }
```

10.2.4 Member Data Documentation

10.2.4.1 int model::Annotation::m_iId [private]

Definition at line 31 of file Annotation.h.

Referenced by Annotation(), operator!=(), operator=(), operator==(), and setId().

10.2.4.2 int model::Annotation::m_iScore [private]

Definition at line 32 of file Annotation.h.

Referenced by Annotation(), operator!=(), operator=(), operator==(), and setScore().

10.2.4.3 string model::Annotation::m_sName [private]

Definition at line 33 of file Annotation.h.

Referenced by Annotation(), operator!=(), operator=(), operator==(), and setName().

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

- model/Annotation.h
- model/Annotation.cpp

10.3 model::AnnotationMapping Class Reference

`#include <AnnotationMapping.h>`

10.3.1 Detailed Description

Write brief comment for AnnotationMapping here.

Write detailed description for AnnotationMapping here.

Remarks:
Write remarks for AnnotationMapping here.

See also:
Separate items with the '|' character.

Definition at line 25 of file AnnotationMapping.h.

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

- model/AnnotationMapping.h

10.4 BadConversion Class Reference

Write brief comment for BadConversion here.

`#include <func.h>`

Public Member Functions

- BadConversion (const std::string &s)

10.4.1 Detailed Description

Write brief comment for BadConversion here.

Write detailed description for BadConversion here.

Remarks:
Write remarks for BadConversion here.

See also:
Separate items with the '|' character.

Definition at line 32 of file func.h.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 BadConversion::BadConversion (const std::string & *s*) [inline]

Definition at line 34 of file func.h.

```
34 : std::runtime_error(s) { }
```

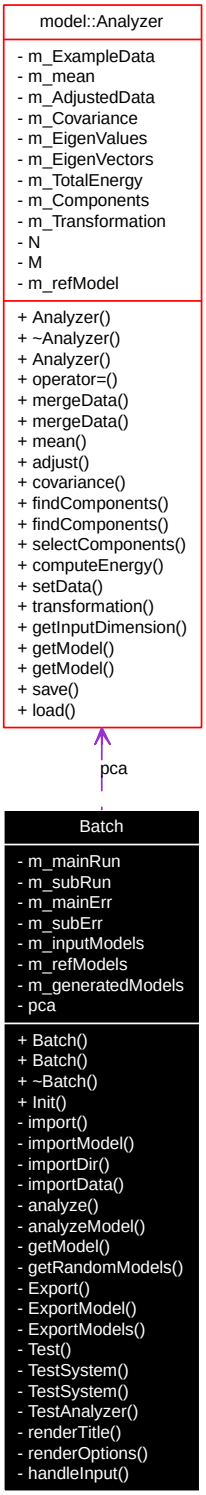
The documentation for this class was generated from the following file:

- Functions/include/func.h

10.5 Batch Class Reference

`#include <Batch.h>`

Collaboration diagram for Batch:


model::Analyzer
- m_ExampleData
- m_mean
- m_AdjustedData
- m_Covariance
- m_EigenValues
- m_EigenVectors
- m_TotalEnergy
- m_Components
- m_Transformation
- N
- M
- m_refModel
+ Analyzer()
+ ~Analyzer()
+ Analyzer()
+ operator=()
+ mergeData()
+ mergeData()
+ mean()
+ adjust()
+ covariance()
+ findComponents()
+ findComponents()
+ selectComponents()
+ computeEnergy()
+ setData()
+ transformation()
+ getInputDimension()
+ getModel()
+ getModel()
+ save()
+ load()
pca
Batch
- m_mainRun
- m_subRun
- m_mainErr
- m_subErr
- m_inputModels
- m_refModels
- m_generatedModels
- pca
+ Batch()
+ Batch()
+ ~Batch()
+ Init()
- import()
- importModel()
- importDir()
- importData()
- analyze()
- analyzeModel()
- getModel()
- getRandomModels()
- Export()
- ExportModel()
- ExportModels()
- Test()
- TestSystem()
- TestSystem()
- TestAnalyzer()
- renderTitle()
- renderOptions()
- handleInput()

Public Types

- enum e_TESTMODE { PCA, STABILITY, ALLTEST, ACCURACY }

Public Member Functions

- Batch ()
- Batch (bool debug)
- ~Batch ()
- void Init ()

Private Member Functions

- void import ()
- void importModel ()
- void importDir ()
- void importData ()
- void analyze ()
- void analyzeModel ()
- void getModel ()
- void getRandomModels ()
- void Export ()
- void ExportModel ()
- void ExportModels ()
- void Test ()
- void TestSystem (e_TESTMODE mode, Analyzer::e_PCAMode method)
- void TestSystem (e_TESTMODE mode)
- void TestAnalyzer (Analyzer::e_PCAMode mode, int size)
- void renderTitle (string title, bool sub=false)
- void renderOptions (string title, vector< string > options, bool back=false)
- int handleInput ()

Private Attributes

- bool m_mainRun
- bool m_subRun
- string m_mainErr
- string m_subErr
- vector< ExampleModel * > m_inputModels
- vector< ReferenceModel > m_refModels
- vector< ReferenceModel > m_generatedModels
- Analyzer pca

10.5.1 Member Enumeration Documentation

10.5.1.1 enum Batch::e_TESTMODE

Enumeration values:

PCA

STABILITY

ALLTEST

ACCURACY

Definition at line 31 of file Batch.h.

```
31 { PCA, STABILITY, ALLTEST, ACCURACY };
```

10.5.2 Constructor & Destructor Documentation

10.5.2.1 Batch::Batch ()

Definition at line 21 of file Batch.cpp.

```
21             {
22         Batch(false);
23 }
```

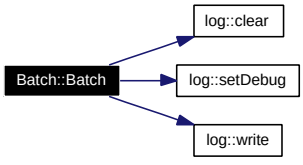
10.5.2.2 Batch::Batch (bool *debug*)

Definition at line 25 of file Batch.cpp.

References log::clear(), m_mainErr, m_subErr, log::setDebug(), and log::write().

```
25                       {
26         m_mainErr = "";
27         m_subErr = "";
28         log::clear();
29         log::setDebug(debug);
30         log::write("Batch started");
31 }
```

Here is the call graph for this function:



10.5.2.3 Batch::~Batch () [inline]

Definition at line 64 of file Batch.h.

```
64 {};
```

10.5.3 Member Function Documentation

10.5.3.1 void Batch::analyze () [private]

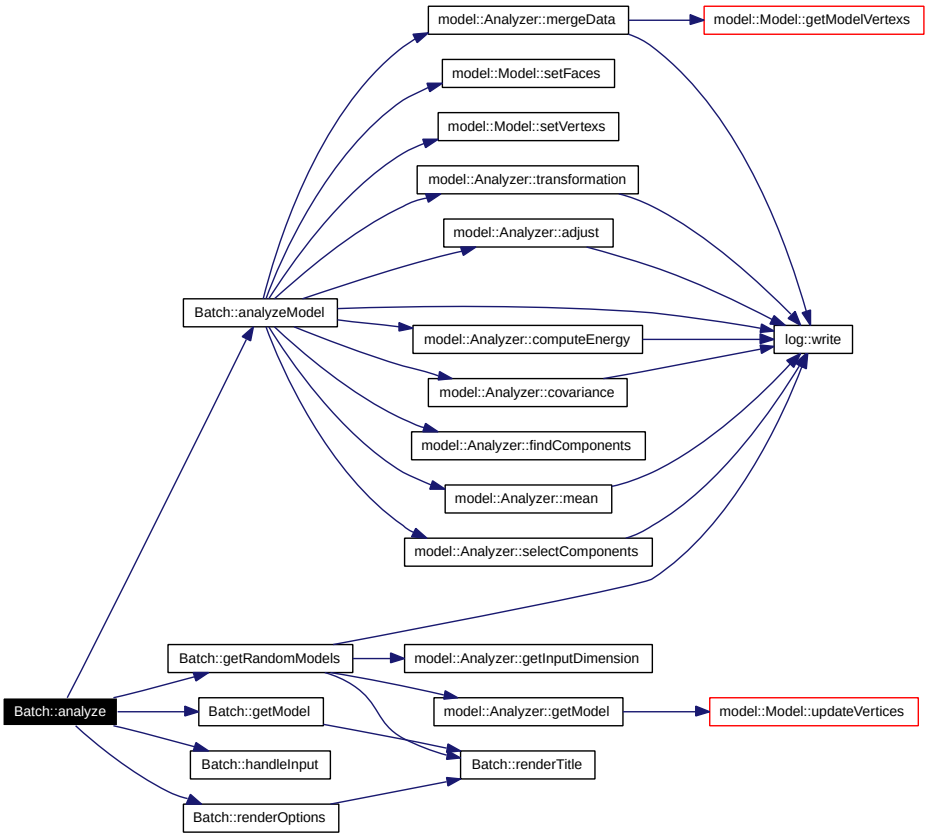
Definition at line 189 of file Batch.cpp.

References analyzeModel(), getModel(), getRandomModels(), handleInput(), m_inputModels, m_subErr, m_subRun, and renderOptions().

Referenced by Init().

```
189                     {
190         m_subRun=true;
191         vector<string> options;
192         char tmp[100];
193         #ifndef __WINDOWS__
194         sprintf(tmp,"Analyze %d models",m_inputModels.size());
195         #else
196         sprintf_s(tmp,"Analyze %d models",m_inputModels.size());
197         #endif
198         string s = string(tmp);
199         options.push_back(s);
200         options.push_back("Get a Model");
201         options.push_back("Get random Model");
202 
203         while(m_subRun) {
204                 renderOptions("Analyze models", options, true);
205                 int o = handleInput();
206                 switch(o) {
207                         case 1: analyzeModel();break;
208                         case 2: getModel();break;
209                         case 3: getRandomModels();break;
210                         default: m_subErr = "Not a valid option!";break;
211                 }
212         }
213 }
```

Here is the call graph for this function:



10.5.3.2 void Batch::analyzeModel () [private]

Definition at line 215 of file Batch.cpp.

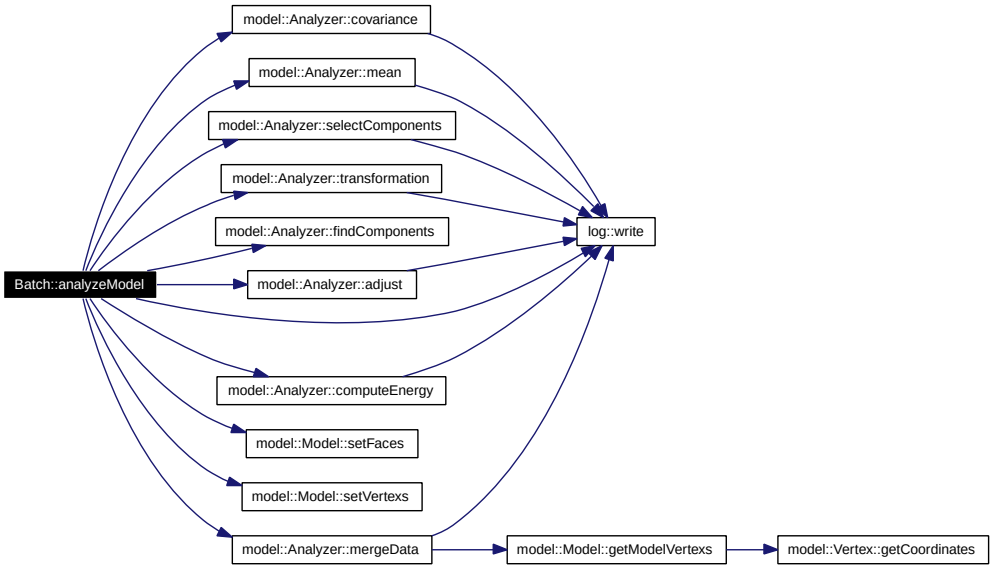
References model::Analyzer::adjust(), model::Analyzer::computeEnergy(), model::Analyzer::covariance(), model::Analyzer::findComponents(), m_inputModels, m_subErr, model::Analyzer::mean(), model::Analyzer::mergeData(), pca, model::Analyzer::selectComponents(), model::Model::setFaces(), model::Model::setVertexs(), model::Analyzer::transformation(), and log::write().

Referenced by analyze().

```
215                         {
216         //pca = Analyzer();
217         ReferenceModel refModel = ReferenceModel();
218         refModel.setVertexs(m_inputModels[0]->getVertexs());
219         refModel.setFaces(m_inputModels[0]->getFaces());
220         log::write("************Starting to Analyze************");
221         printf("start\n");
222         pca.mergeData(refModel,m_inputModels);
223         printf("mergeData done\n");
```

```
224         log::write("Calculating mean");
225         pca.mean();
226         printf("mean done\n");
227         log::write("Adjusting data");
228         pca.adjust();
229         printf("adjust done\n");
230         log::write("Finding Covariance");
231         pca.covariance();
232         printf("covariance done\n");
233         log::write("Begin EigenVector calculations");
234         log::write("Timer Started");
235         clock_t begin = clock();
236         pca.findComponents();
237         clock_t end = clock();
238         log::write("Timer Stopped");
239         log::write("Time elapsed in seconds:", difftime(end, begin));
240         log::write("Time elapsed in clock cycles:",(int)(end-begin));
241         log::write("End EigenVector calculations");
242         printf("finding done\n");
243         pca.computeEnergy();
244         printf("compute energy done\n");
245         pca.selectComponents();
246         printf("select components done\n");
247         pca.transformation();
248         m_subErr = "Models analyzed";
249         log::write("************ANALYSIS ENDED************");
250 }
```

Here is the call graph for this function:



10.5.3.3 void Batch::Export () [private]

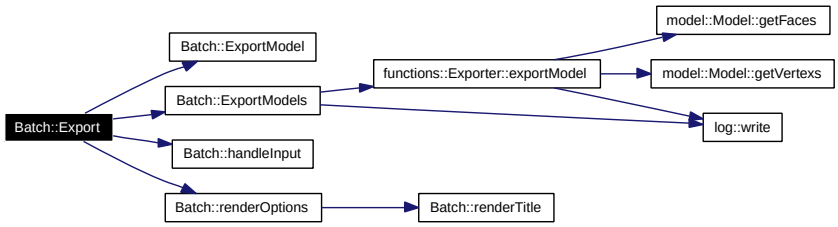
Definition at line 278 of file Batch.cpp.

References ExportModel(), ExportModels(), handleInput(), m_generatedModels, m_-inputModels, m_refModels, m_subErr, m_subRun, and renderOptions().

Referenced by Init().

```
278                     {
279         m_subRun=true;
280         vector<string> options;
281         options.push_back("Export a model");
282         char* optname = "Export all models";
283         int count = m_inputModels.size() + m_refModels.size() + m_generatedModels.size();
284         if(count>0) {
285                 char tmp[100];
286                 #ifndef __WINDOWS__
287                 sprintf(tmp,"%s (%d)",optname,count);
288                 #else
289                 sprintf_s(tmp,"%s (%d)",optname,count);
290                 #endif
291                 optname = tmp;
292         }
293         string s = string(optname);
294         options.push_back(s);
295
296         while(m_subRun) {
297                 renderOptions("Export models", options, true);
298                 int o = handleInput();
299                 switch(o) {
300                         case 1: ExportModel();break;
301                         case 2: ExportModels(); break;
302                         default: m_subErr = "Not a valid option!";break;
303                 }
304         }
305 }
```

Here is the call graph for this function:



10.5.3.4 void Batch::ExportModel () [private]

Definition at line 307 of file Batch.cpp.

Referenced by Export().

```
307                         {
308
309 }
```

10.5.3.5 void Batch::ExportModels () [private]

Definition at line 311 of file Batch.cpp.

References functions::Exporter::exportModel(), m_generatedModels, m_inputModels, m_refModels, m_subErr, and log::write().

Referenced by Export().

```
311                         {
312         log::write("Batch::ExportModels");
313         functions::Exporter ex = functions::Exporter();
314         char tmp[100];
315         log::write("Export Inputmodels");
316         for(int i=0; i<m_inputModels.size(); i++) {
317                 #ifndef __WINDOWS__
318                 sprintf(tmp,"inputModel%d.x3d",i);
319                 #else
320                 sprintf_s(tmp,"inputModel%d.x3d",i);
321                 #endif
322                 ex.exportModel(m_inputModels[i],tmp);
323         }
324         log::write("Export refModels");
325         for(int i=0; i<m_refModels.size(); i++) {
326                 #ifndef __WINDOWS__
327                 sprintf(tmp,"referenceModel%d.x3d",i);
328                 #else
329                 sprintf_s(tmp,"referenceModel%d.x3d",i);
330                 #endif
331
332                 ex.exportModel(&m_refModels[i],tmp);
333         }
334         log::write("Export generatedModels");
335         for(int i=0; i<m_generatedModels.size(); i++) {
336                 #ifndef __WINDOWS__
337                 sprintf(tmp,"generatedModel%d.x3d",i);
338                 #else
339                 sprintf_s(tmp,"generatedModel%d.x3d",i);
340                 #endif
341                 ex.exportModel(&m_generatedModels[i],tmp);
342         }
343         m_subErr = "Models exported";
344 }
```

Here is the call graph for this function:

10.5.3.6 void Batch::getModel () [private]

Definition at line 252 of file Batch.cpp.

References renderTitle().

Referenced by analyze().

```
252                     {
253   renderTitle("Get a model");
254
255 }
```

Here is the call graph for this function:

10.5.3.7 void Batch::getRandomModels () [private]

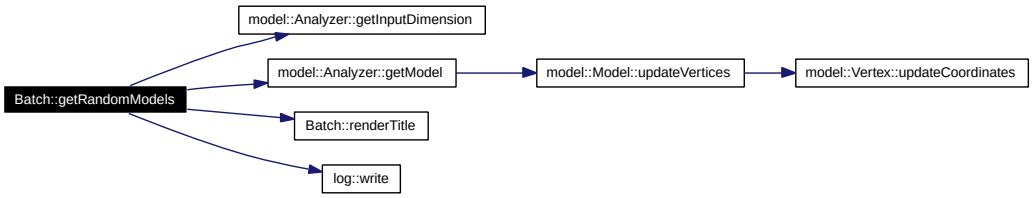
Definition at line 257 of file Batch.cpp.

References model::Analyzer::getInputDimension(), model::Analyzer::getModel(), m_generatedModels, pca, renderTitle(), and log::write().

Referenced by analyze().

```
257                             {
258         renderTitle("Generates 10 random models",true);
259         int dim = pca.getInputDimension();
260         Matrix input(1,dim);
261         int lowest=-10, highest=10;
262         int range=(highest-lowest)+1;
263         for(int i=0; i<10;i++) {
264                 if(dim=1) {
265                         input(1,1) = i-5;
266                 } else {
267                         for(int n=0;n<dim;n++) {
268                                 input(1,n+1) = lowest+int(range*rand()/(RAND_MAX + 1.0));
269                         }
270                 }
271                 log::write("Batch-Input",input);
272                 ReferenceModel refModel = pca.getModel(input);
273                 log::write("ReferenceModel",refModel);
274                 m_generatedModels.push_back(refModel);
275         }
276 }
```

Here is the call graph for this function:



10.5.3.8 int Batch::handleInput () [private]

Definition at line 476 of file Batch.cpp.

References m_mainRun, and m_subRun.

Referenced by analyze(), Export(), import(), Init(), and Test().

```
476                         {
477         //cout << "handel input\n";
478         int opt;
479         char temp[100];
480         cin.getline(temp, 100);
481
482         opt=atoi(temp);
483         if(opt==9) {
484                 m_mainRun = false;
485                 m_subRun = false;
486         } else if(opt == 8) {
487                 m_subRun = false;
488         }
489         //cout << opt << "\n";
490         return opt;
491 }
```

10.5.3.9 void Batch::import () [private]

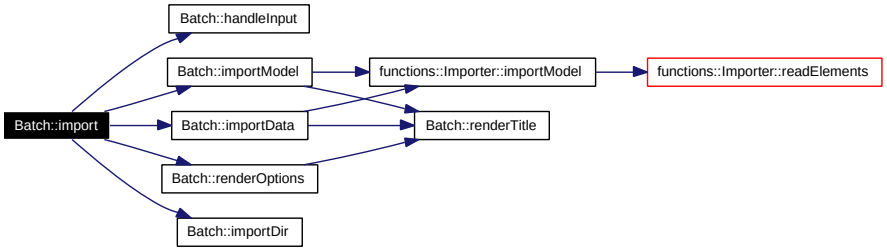
Definition at line 81 of file Batch.cpp.

References handleInput(), importData(), importDir(), importModel(), m_subErr, m_-subRun, and renderOptions().

```
81                      {
82         m_subRun=true;
83         vector<string> options;
84         options.push_back("Import a model");
85         options.push_back("Import models");
86         options.push_back("Import data models");
87
88         while(m_subRun) {
89                 renderOptions("Import models", options, true);
90                 int o = handleInput();
```

```
91                 switch(o) {
92                         case 1: importModel();break;
93                         case 2: importDir(); break;
94                         case 3: importData(); break;
95                         default: m_subErr = "Not a valid option!";break;
96                 }
97         }
98 }
```

Here is the call graph for this function:



10.5.3.10 void Batch::importData () [private]

Definition at line 143 of file Batch.cpp.

References functions::Importer::importModel(), m_inputModels, m_subErr, and renderTitle().

Referenced by import().

```
143                         {
144         functions::Importer im = functions::Importer();
145         string path = "";
146         string name = "";
147         string ending = "";
148         int rangeStart = 0;
149         int rangeEnd = 1;
150         char temp[100];
151         bool run = true;
152         renderTitle("Import data models",true);
153         while(run) {
154                 cout << "Enter path to the data models (../data/): ";
155                 cin.getline(temp, 100);
156                 path.assign(temp);
157                 cout << "Enter start file name (exportFile): ";
158                 cin.getline(temp, 100);
159                 name.assign(temp);
160                 cout << "Enter file ending (.x3d): ";
161                 cin.getline(temp, 100);
162                 ending.assign(temp);
163                 cout << "Enter start range (1): ";
164                 cin.getline(temp, 100);
165                 rangeStart=atoi(temp);
166                 cout << "Enter end range (99): ";
```

```
167                 cin.getline(temp, 100);
168                 rangeEnd=atoi(temp);
169                 for(int i=rangeStart;i<=rangeEnd;i++) {
170                         std::cout << "Importing model " << i << " of " << rangeEnd << "
171                         ExampleModel* em = new ExampleModel();
172                         string filename = path + name;
173                         #ifndef __WINDOWS__
174                         sprintf(temp,"%d",i);
175                         #else
176                         sprintf_s(temp,"%d",i);
177                         #endif
178                         filename.append(string(temp));
179                         filename.append(ending);
180                         run = !im.importModel(filename.c_str(),em);
181                         if(!run) {
182                                 m_subErr = "Data Models loaded";
183                                 m_inputModels.push_back(em);
184                         }
185                 }
186         }
187 }
```

Here is the call graph for this function:

10.5.3.11 void Batch::importDir () [private]

Definition at line 119 of file Batch.cpp.

Referenced by import().

```
119                         {
120 /*      DIR *pdir;
121         struct dirent *pent;
122         string path = "";
123
124         renderTitle("Import a model dir",true);
125         cout << "Enter path to the model file: ";
126         cin >> path;
127
128         pdir=opendir(path); //"." refers to the current dir
129         if (!pdir){
130                 printf ("opendir() failure; terminating");
131         }
132         errno=0;
133         while ((pent=readdir(pdir))){
134                 printf("%s", pent->d_name);
135         }
136         if (errno){
137                 printf ("readdir() failure; terminating");
138                 exit(1);
```

```
139         }
140         closedir(pdir);*/
141 }
```

10.5.3.12 void Batch::importModel () [private]

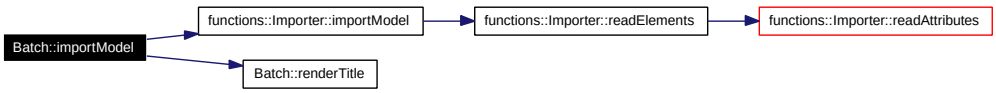
Definition at line 100 of file Batch.cpp.

References functions::Importer::importModel(), m_inputModels, m_subErr, and renderTitle().

Referenced by import().

```
100                         {
101         ExampleModel* em = new ExampleModel();
102         functions::Importer im = functions::Importer();
103         string file = "";
104         bool run = true;
105         renderTitle("Import a model",true);
106         while(run) {
107                 cout << "Enter path to the model file: ";
108                 cin >> file;
109                 char temp[100];
110                 cin.getline(temp, 100);
111                 run = !im.importModel(file.c_str(),em);
112                 if(!run) {
113                         m_subErr = "Model loaded";
114                         m_inputModels.push_back(em);
115                 }
116         }
117 }
```

Here is the call graph for this function:



10.5.3.13 void Batch::Init ()

Definition at line 33 of file Batch.cpp.

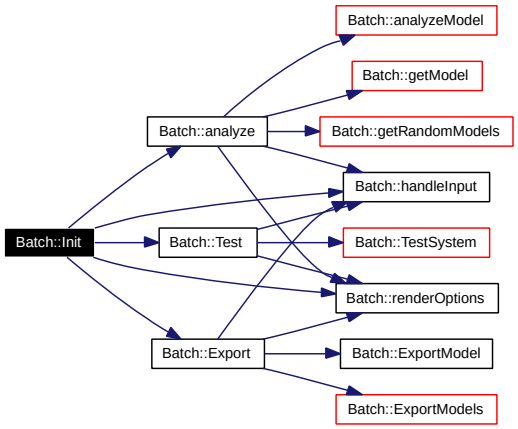
References analyze(), Export(), handleInput(), m_generatedModels, m_inputModels, m_mainErr, m_mainRun, m_refModels, m_subRun, renderOptions(), and Test().

Referenced by main().

```
33                  {
```

```
34         m_mainRun = true;
35         m_subRun = true;
36
37
38         while(m_mainRun) {
39                 vector<string> options;
40                 options.push_back("Import models");
41                 char* optname = "Analyze models";
42                 int count = m_inputModels.size();
43                 if(count>0) {
44                         char tmp[100];
45                         #ifndef __WINDOWS__
46                         sprintf(tmp,"%s (%d)",optname,count);
47                         #else
48                         sprintf_s(tmp,"%s (%d)",optname,count);
49                         #endif
50                         optname = tmp;
51                 }
52                 string s = string(optname);
53                 options.push_back(s);
54                 optname = "Export models";
55                 count = m_inputModels.size() + m_refModels.size() + m_generatedModels.size();
56                 if(count>0) {
57                         char tmp[100];
58                         #ifndef __WINDOWS__
59                         sprintf(tmp,"%s (%d)",optname,count);
60                         #else
61                         sprintf_s(tmp,"%s (%d)",optname,count);
62                         #endif
63                         optname = tmp;
64                 }
65                 s = string(optname);
66                 options.push_back(s);
67                 options.push_back("Test");
68
69                 renderOptions("PCG",options);
70                 int o = handleInput();
71                 switch(o) {
72                         case 1: import();break;
73                         case 2: analyze(); break;
74                         case 3: Export(); break;
75                         case 4: Test(); break;
76                         default: m_mainErr = "Not a valid option!";break;
77                 }
78         }
79 }
```

Here is the call graph for this function:



10.5.3.14 void Batch::renderOptions (string *title*, vector< string > *options*, bool *back* = `false`) `[private]`

Definition at line 462 of file Batch.cpp.

References renderTitle().

Referenced by analyze(), Export(), import(), Init(), and Test().

```
462                                                                                 {
463         renderTitle(title,back);
464         for(int i=0; i<options.size();i++) {
465                 cout << "        " << i+1 << ") " << options[i] << "\n";
466         }
467         cout << "\n\n";
468         if(back) {
469                 cout << "        8) Back\n";
470         }
471         cout << "        9) Quit\n\n";
472         cout << "      Select a option: ";
473
474 }
```

Here is the call graph for this function:

10.5.3.15 void Batch::renderTitle (string *title*, bool *sub* = `false`) `[private]`

Definition at line 446 of file Batch.cpp.

References m_mainErr, and m_subErr.

Referenced by getModel(), getRandomModels(), importData(), importModel(), and renderOptions().

```
446                                                 {
447         #ifdef __WINDOWS__
448                 system("cls");
449         #else
450                 system("clear");
451         #endif
452         string err = m_mainErr;
453         if(sub) {
454                 err = m_subErr;
455         }
456         cout << err << "\n\n\n";
457         cout << "     " << title << "\n\n";
458         m_mainErr = "";
459         m_subErr = "";
460 }
```

10.5.3.16 void Batch::Test () [private]

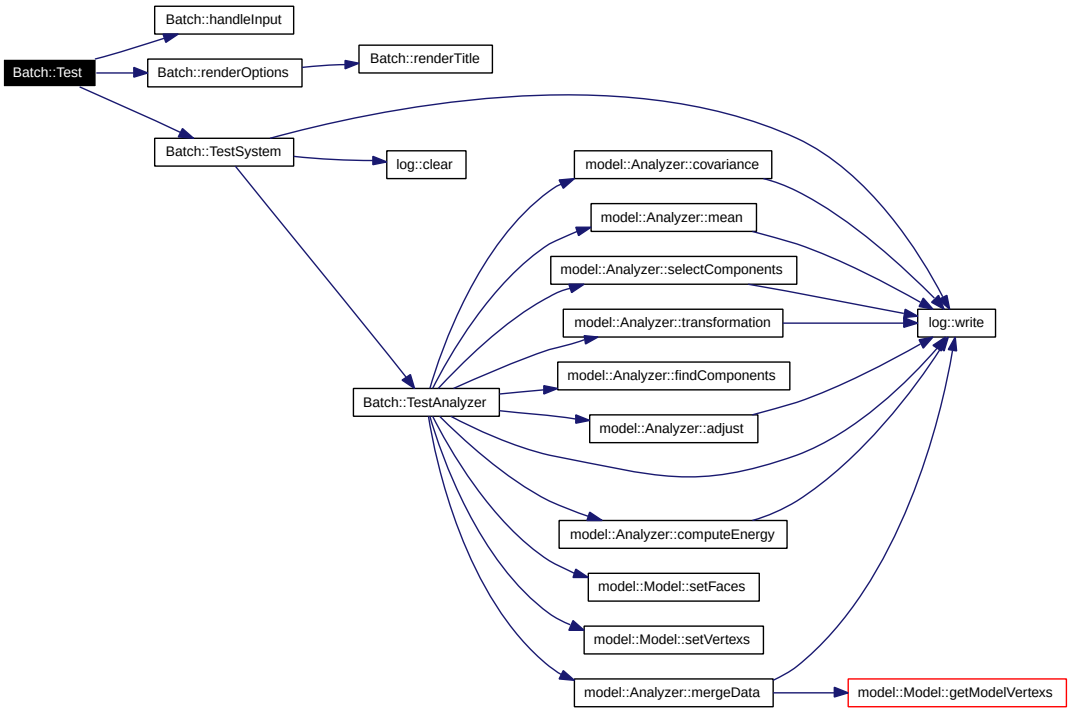
Definition at line 348 of file Batch.cpp.

References ALLTEST, handleInput(), m_subErr, m_subRun, PCA, renderOptions(), STABILITY, and TestSystem().

Referenced by Init().

```
348                 {
349         m_subRun=true;
350         vector<string> options;
351         options.push_back("ALL Methods");
352         options.push_back("JACOBI Method");
353         options.push_back("HOUSEHOLDER Method");
354         options.push_back("Time stability");
355         options.push_back("All tests");
356
357         while(m_subRun) {
358                 renderOptions("Test", options, true);
359                 int o = handleInput();
360                 switch(o) {
361                         case 1: TestSystem(PCA,Analyzer::ALL);break;
362                         case 2: TestSystem(PCA,Analyzer::JACOBI); break;
363                         case 3: TestSystem(PCA,Analyzer::HOUSEHOLDER); break;
364                         case 4: TestSystem(STABILITY); break;
365                         case 5: TestSystem(ALLTEST); break;
366                         default: m_subErr = "Not a valid option!";break;
367                 }
368         }
369 }
```

Here is the call graph for this function:



10.5.3.17 void Batch::TestAnalyzer (Analyzer::e_PCAMode *mode*, int *size*) [private]

Definition at line 410 of file Batch.cpp.

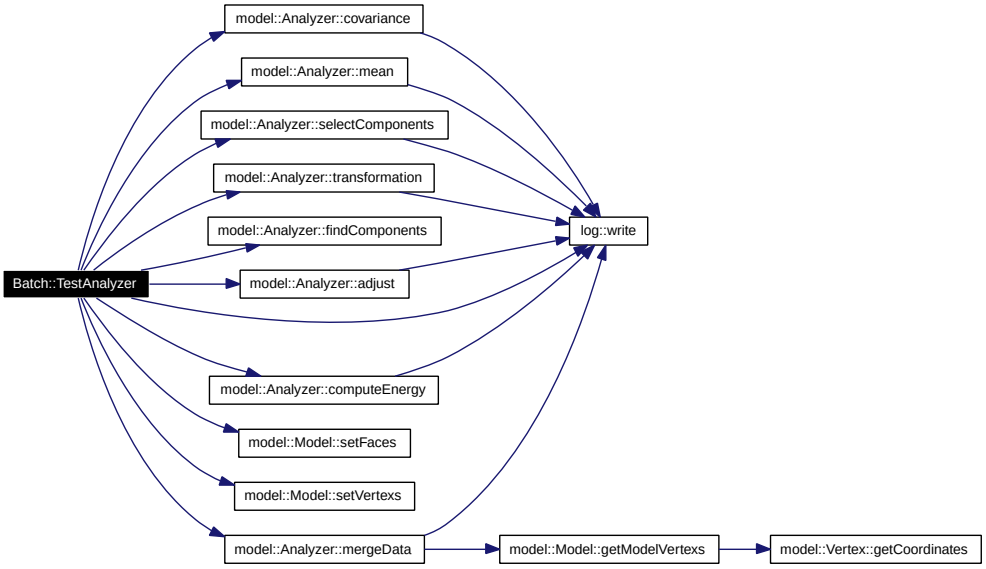
References model::Analyzer::adjust(), model::Analyzer::computeEnergy(), model::Analyzer::covariance(), model::Analyzer::findComponents(), m_inputModels, model::Analyzer::mean(), model::Analyzer::mergeData(), pca, model::Analyzer::selectComponents(), model::Model::setFaces(), model::Model::setVertexs(), model::Analyzer::transformation(), and log::write().

Referenced by TestSystem().

```
410                                                     {
411         if(mode == Analyzer::JACOBI) {
412                 printf("Running JACOBI with %d models",size);
413         } else if(mode == Analyzer::HOUSEHOLDER) {
414                 printf("Running HOUSEHOLDER with %d models",size);
415         }
416         ReferenceModel refModel = ReferenceModel();
417         refModel.setVertexs(m_inputModels[0]->getVertexs());
418         refModel.setFaces(m_inputModels[0]->getFaces());
419         log::write("************Starting to Analyze************");
420         pca.mergeData(refModel,m_inputModels,size);
421         log::write("Calculating mean");
422         pca.mean();
```

```
423         log::write("Adjusting data");
424         pca.adjust();
425         log::write("Finding Covariance");
426         pca.covariance();
427         log::write("Begin EigenVector calculations");
428         log::write("Timer Started");
429         clock_t begin = clock();
430         pca.findComponents(mode);
431         clock_t end = clock();
432         float time = (float)(end - begin)/CLOCKS_PER_SEC;
433         printf(" - It took %f seconds\n",time);
434         log::write("Timer Stopped");
435         log::write("Time elapsed in seconds:", time);
436         log::write("Time elapsed in clock cycles:",(int)(end-begin));
437         log::write("End EigenVector calculations");
438         pca.computeEnergy();
439         pca.selectComponents();
440         pca.transformation();
441         log::write("************ANALYSIS ENDED************");
442 }
```

Here is the call graph for this function:



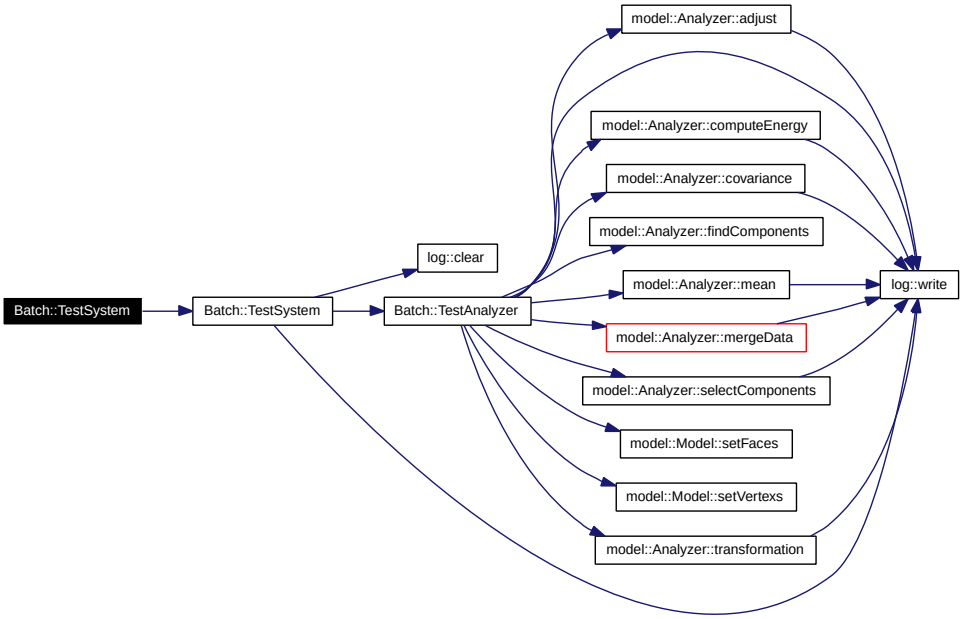
10.5.3.18 void Batch::TestSystem (e_TESTMODE *mode*) [private]

Definition at line 371 of file Batch.cpp.

References TestSystem().

```
371                                     {
372         TestSystem(mode,Analyzer::ALL);
373 }
```

Here is the call graph for this function:



10.5.3.19 void Batch::TestSystem (e_TESTMODE *mode*, Analyzer::e_PCAMode *method*) [private]

Definition at line 375 of file Batch.cpp.

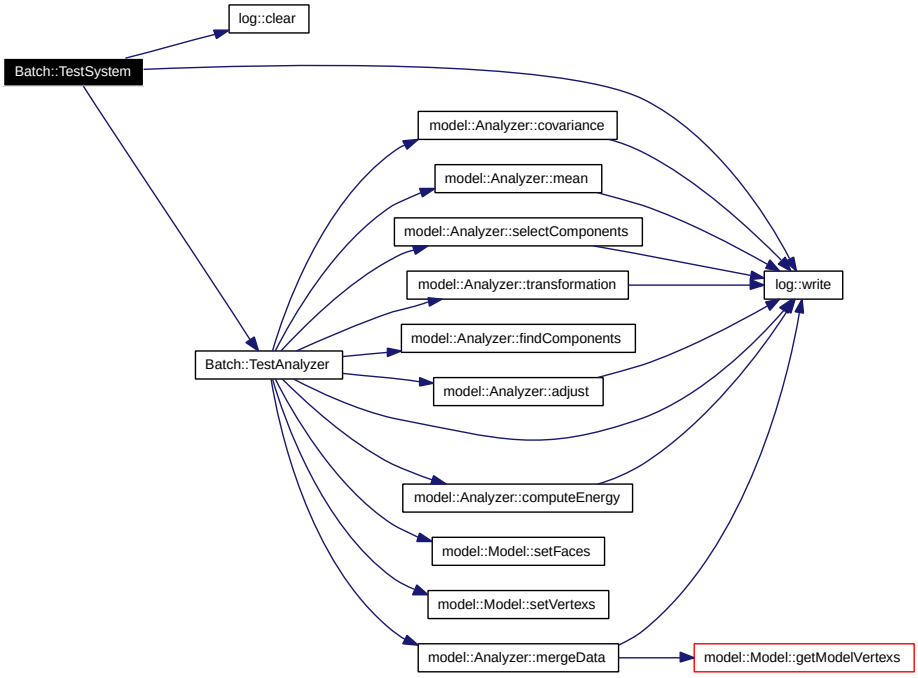
References log::clear(), PCA, STABILITY, TestAnalyzer(), and log::write().

Referenced by Test(), and TestSystem().

```
375                                                                 {
376         #ifdef __WINDOWS__
377                 system("cls");
378         #else
379                 system("clear");
380         #endif
381         log::clear();
382         log::write("TESTING STARTED");
383         if(mode == PCA || mode == ALLTEST) {
384                 log::write("PCA TEST");
385                 int testRunNumbers[4] = {5,10,50,100};
386                 if(method == Analyzer::ALL) {
387                         for(int i=0; i<4; i++) {
388                                 TestAnalyzer(Analyzer::JACOBI, testRunNumbers[i]);
```

```
389                         }
390                         for(int i=0; i<4; i++) {
391                                 TestAnalyzer(Analyzer::HOUSEHOLDER,testRunNumbers[i]);
392                         }
393                 } else {
394                         for(int i=0; i<4; i++) {
395                                 TestAnalyzer(method,testRunNumbers[i]);
396                         }
397                 }
398         } else if(mode == STABILITY || mode == ALLTEST) {
399                 log::write("TIME STABILITY TEST");
400                 int testRunNumbers[4] = {2,3,4,5};
401                 for(int i=0;i<4;i++) {
402                         for(int n=0;n<100;n++) {
403                                 log::write("Run nr:",n+1);
404                                 TestAnalyzer(Analyzer::HOUSEHOLDER,testRunNumbers[i]);
405                         }
406                 }
407         }
408 }
```

Here is the call graph for this function:



10.5.4 Member Data Documentation

10.5.4.1 vector<ReferenceModel> Batch::m_generatedModels [private]

Definition at line 37 of file Batch.h.

Referenced by Export(), ExportModels(), getRandomModels(), and Init().

10.5.4.2 vector<ExampleModel*> Batch::m_inputModels [private]

Definition at line 35 of file Batch.h.

Referenced by analyze(), analyzeModel(), Export(), ExportModels(), importData(), importModel(), Init(), and TestAnalyzer().

10.5.4.3 string Batch::m_mainErr [private]

Definition at line 34 of file Batch.h.

Referenced by Batch(), Init(), and renderTitle().

10.5.4.4 bool Batch::m_mainRun [private]

Definition at line 33 of file Batch.h.

Referenced by handleInput(), and Init().

10.5.4.5 vector<ReferenceModel> Batch::m_refModels [private]

Definition at line 36 of file Batch.h.

Referenced by Export(), ExportModels(), and Init().

10.5.4.6 string Batch::m_subErr [private]

Definition at line 34 of file Batch.h.

Referenced by analyze(), analyzeModel(), Batch(), Export(), ExportModels(), import(), importData(), importModel(), renderTitle(), and Test().

10.5.4.7 bool Batch::m_subRun [private]

Definition at line 33 of file Batch.h.

Referenced by analyze(), Export(), handleInput(), import(), Init(), and Test().

10.5.4.8 Analyzer Batch::pca [private]

Definition at line 38 of file Batch.h.

Referenced by analyzeModel(), getRandomModels(), and TestAnalyzer().

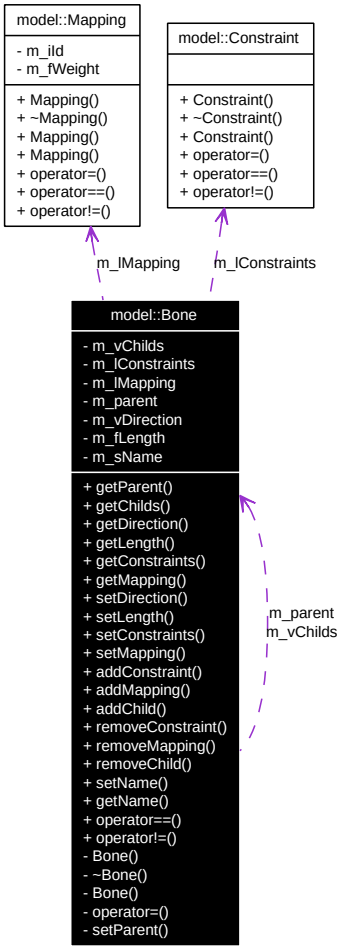
The documentation for this class was generated from the following files:

- gui/Batch.h
- gui/Batch.cpp

10.6 model::Bone Class Reference

`#include <Bone.h>`

Collaboration diagram for model::Bone:



Public Member Functions

- Bone * getParent ()
- vector< Bone * > getChilds ()
- IvVector3 * getDirection ()
- float getLength ()
- vector< Constraint * > getConstraints ()
- vector< Mapping * > getMapping ()

- void setDirection (IvVector3 *vDirection)
- void setLength (float fLength)
- void setConstraints (vector< Constraint * > lConstraints)
- void setMapping (vector< Mapping * > lMapping)
- void addConstraint (Constraint *constraint)
- void addMapping (Mapping *mapping)
- void addChild (Bone *bone)
- void removeConstraint (Constraint *constraint)
- void removeMapping (Mapping *mapping)
- void removeChild (string name)
- void setName (string name)

  Write brief comment for setName here.

- string getName ()

  Write brief comment for getName here.

- bool operator== (const Bone &other) const
- bool operator!= (const Bone &other) const

Private Member Functions

- Bone (Bone *parent, IvVector3 *vDirection, float fLength, string name)
- ~Bone ()
- Bone (const Bone &other)
- Bone & operator= (const Bone &other)
- void setParent (Bone *bone)

Private Attributes

- Bone * m_vChilds
- Constraint * m_lConstraints
- Mapping * m_lMapping
- Bone * m_parent
- IvVector3 * m_vDirection
- float m_fLength
- string m_sName

Friends

- class Skeleton

10.6.1 Detailed Description

Write brief comment for Bone here.

Write detailed description for Bone here.

Remarks:
Write remarks for Bone here.

See also:
Separate items with the '|' character.

Definition at line 30 of file Bone.h.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 Bone::Bone (Bone * *parent*, IvVector3 * *vDirection*, float *fLength*, string *name*) [private]

Write brief comment for Bone here.

Parameters:
origin Description of parameter origin.

vDirection Description of parameter vDirection.

fLength Description of parameter fLength.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Bone here.

Remarks:
Write remarks for Bone here.

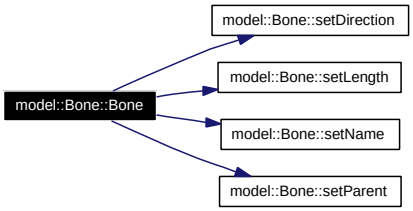
See also:
Separate items with the '|' character.

Definition at line 29 of file Bone.cpp.

References setDirection(), setLength(), setName(), and setParent().

```
29                                                                  {
30         setParent(parent);
31         setDirection(vDirection);
32         setLength(fLength);
33         setName(name);
34 }
```

Here is the call graph for this function:



10.6.2.2 model::Bone::~Bone () [inline, private]

Write brief comment for ~Bone here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Bone here.

Remarks:
Write remarks for ~Bone here.

See also:
Separate items with the '|' character.

Definition at line 64 of file Bone.h.

```
64 {};
```

10.6.2.3 Bone::Bone (const Bone & *other*) [private]

Write brief comment for Bone here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Bone here.

Remarks:
Write remarks for Bone here.

See also:
Separate items with the '|' character.

Definition at line 54 of file Bone.cpp.

References m_fLength, m_lConstraints, m_lMapping, m_parent, m_sName, m_vChilds, and m_vDirection.

```
54                              {
55          m_parent = other.m_parent;
56          m_vDirection = other.m_vDirection;
57          m_fLength = other.m_fLength;
58          m_lConstraints = other.m_lConstraints;
59          m_lMapping = other.m_lMapping;
60          m_vChilds = other.m_vChilds;
61          m_sName = other.m_sName;
62 }
```

10.6.3 Member Function Documentation

10.6.3.1 void Bone::addChild (Bone * *bone*)

Definition at line 328 of file Bone.cpp.

References m_vChilds.

Referenced by model::Skeleton::addBone().

```
328                                 {
329          m_vChilds.push_back(bone);
330 }
```

10.6.3.2 void Bone::addConstraint (Constraint * *constraint*)

Write brief comment for addConstraint here.

Parameters:
constraint Description of parameter constraint.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addConstraint here.

Remarks:
Write remarks for addConstraint here.

See also:
Separate items with the '|' character.

Definition at line 303 of file Bone.cpp.

```
303                                                 {
304
305 }
```

10.6.3.3 void Bone::addMapping (Mapping * *mapping*)

Write brief comment for addMapping here.

Parameters:
mapping Description of parameter mapping.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addMapping here.

Remarks:
Write remarks for addMapping here.

See also:
Separate items with the '|' character.

Definition at line 324 of file Bone.cpp.

Referenced by pcggui::ConvertSkin().

```
324                                         {
325
326 }
```

10.6.3.4 vector<Bone*> model::Bone::getChilds ()

10.6.3.5 vector< Constraint * > Bone::getConstraints ()

Write brief comment for getConstraints here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getConstraints here.

Remarks:
Write remarks for getConstraints here.

See also:
Separate items with the '|' character.

Definition at line 161 of file Bone.cpp.

```
161                                         {
162         return m_lConstraints;
163 }
```

10.6.3.6 IvVector3 * Bone::getDirection ()

Write brief comment for getDirection here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getDirection here.

Remarks:
Write remarks for getDirection here.

See also:
Separate items with the '|' character.

Definition at line 119 of file Bone.cpp.

```
119                                 {
120         return m_vDirection;
121 }
```

10.6.3.7 float Bone::getLength ()

Write brief comment for getLength here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getLength here.

Remarks:
Write remarks for getLength here.

See also:
Separate items with the '|' character.

Definition at line 140 of file Bone.cpp.

```
140                         {
141         return m_fLength;
142 }
```

10.6.3.8 vector< Mapping * > Bone::getMapping ()

Write brief comment for getMapping here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getMapping here.

Remarks:
Write remarks for getMapping here.

See also:
Separate items with the '|' character.

Definition at line 182 of file Bone.cpp.

```
182                                 {
183         return m_lMapping;
184 }
```

10.6.3.9 string Bone::getName ()

Write brief comment for getName here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getName here.

Remarks:
Write remarks for getName here.

See also:
Separate items with the '|' character.

Definition at line 486 of file Bone.cpp.

```
486                     {
487         return m_sName;
488 }
```

10.6.3.10 Bone* model::Bone::getParent ()

10.6.3.11 bool Bone::operator!= (const Bone & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 434 of file Bone.cpp.

References m_fLength, m_lConstraints, m_lMapping, m_sName, and m_vDirection.

```
435 {
436         if(m_vDirection == other.m_vDirection
437         && m_fLength == other.m_fLength
438         && m_lConstraints == other.m_lConstraints
439         && m_lMapping == other.m_lMapping
440         && m_sName.compare(other.m_sName)) {
441         return false;
442         }
443     return true;
444 }
```

10.6.3.12 Bone & Bone::operator= (const Bone & *other*) [private]

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 85 of file Bone.cpp.

References m_fLength, m_lConstraints, m_lMapping, m_parent, m_sName, m_vChilds, and m_vDirection.

```
86 {
87     // if same object
88     if ( this == &other )
89         return *this;
90
91         m_parent = other.m_parent;
```

```
92          m_vDirection = other.m_vDirection;
93          m_fLength = other.m_fLength;
94          m_lConstraints = other.m_lConstraints;
95          m_lMapping = other.m_lMapping;
96          m_vChilds = other.m_vChilds;
97          m_sName = other.m_sName;
98
99      return *this;
100 }
```

10.6.3.13 bool Bone::operator== (const Bone & *other*) const

Write brief comment for operator == here.

Parameters:

other Description of parameter other.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:

Write remarks for operator == here.

See also:

Separate items with the '|' character.

Definition at line 400 of file Bone.cpp.

References m_fLength, m_lConstraints, m_lMapping, m_sName, and m_vDirection.

```
401 {
402
403     if(m_vDirection == other.m_vDirection
404         && m_fLength == other.m_fLength
405         && m_lConstraints == other.m_lConstraints
406         && m_lMapping == other.m_lMapping
407         && m_sName.compare(other.m_sName)) {
408         return true;
409         }
410     return false;
411 }
```

10.6.3.14 void Bone::removeChild (string *name*)

Definition at line 375 of file Bone.cpp.

```
375                                     {
376
377 }
```

10.6.3.15 void Bone::removeConstraint (Constraint * *constraint*)

Write brief comment for removeConstraint here.

Parameters:
constraint Description of parameter constraint.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeConstraint here.

Remarks:
Write remarks for removeConstraint here.

See also:
Separate items with the '|' character.

Definition at line 349 of file Bone.cpp.

```
349                                                     {
350
351 }
```

10.6.3.16 void Bone::removeMapping (Mapping * *mapping*)

Write brief comment for removeMapping here.

Parameters:
mapping Description of parameter mapping.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeMapping here.

Remarks:
Write remarks for removeMapping here.

See also:
Separate items with the '|' character.

Definition at line 371 of file Bone.cpp.

```
371                                             {
372
373 }
```

10.6.3.17 void Bone::setConstraints (vector< Constraint * > *lConstraints*)

Write brief comment for setConstraints here.

Parameters:
lConstraints Description of parameter lConstraints.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setConstraints here.

Remarks:
Write remarks for setConstraints here.

See also:
Separate items with the '|' character.

Definition at line 261 of file Bone.cpp.

References m_lConstraints.

```
261                                                             {
262         m_lConstraints = lConstraints;
263 }
```

10.6.3.18 void Bone::setDirection (IvVector3 * *vDirection*)

Write brief comment for setDirection here.

Parameters:
vDirection Description of parameter vDirection.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setDirection here.

Remarks:
Write remarks for setDirection here.

See also:
Separate items with the '|' character.

Definition at line 218 of file Bone.cpp.

References m_vDirection.

Referenced by Bone().

```
218                                                 {
219         m_vDirection = vDirection;
220 }
```

10.6.3.19 void Bone::setLength (float *fLength*)

Write brief comment for setLength here.

Parameters:
fLength Description of parameter fLength.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setLength here.

Remarks:
Write remarks for setLength here.

See also:
Separate items with the '|' character.

Definition at line 240 of file Bone.cpp.

References m_fLength.

Referenced by Bone().

```
240                                     {
241         m_fLength = fLength;
242 }
```

10.6.3.20 void Bone::setMapping (vector< Mapping * > *lMapping*)

Write brief comment for setMapping here.

Parameters:
lMapping Description of parameter lMapping.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setMapping here.

Remarks:
Write remarks for setMapping here.

See also:
Separate items with the '|' character.

Definition at line 282 of file Bone.cpp.

References m_lMapping.

```
282                                                 {
283         m_lMapping = lMapping;
284 }
```

10.6.3.21 void Bone::setName (string *name*)

Write brief comment for setName here.

Parameters:
name Description of parameter name.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setName here.

Remarks:
Write remarks for setName here.

See also:
Separate items with the '|' character.

Definition at line 464 of file Bone.cpp.

References m_sName.

Referenced by Bone().

```
464                                 {
465         m_sName = name;
466 }
```

10.6.3.22 void Bone::setParent (Bone * *bone*) [private]

Write brief comment for setParent here.

Parameters:
bone Description of parameter origin.

Write detailed description for setParent here.

See also:
Separate items with the '|' character.

Definition at line 197 of file Bone.cpp.

References m_parent.

Referenced by Bone().

```
197                                  {
198         m_parent = bone;
199 }
```

10.6.4 Friends And Related Function Documentation

10.6.4.1 friend class Skeleton [friend]

Definition at line 100 of file Bone.h.

10.6.5 Member Data Documentation

10.6.5.1 float model::Bone::m_fLength [private]

Definition at line 43 of file Bone.h.

Referenced by Bone(), operator!=(), operator=(), operator==(), and setLength().

10.6.5.2 Constraint* model::Bone::m_lConstraints [private]

Definition at line 34 of file Bone.h.

Referenced by Bone(), operator!=(), operator=(), operator==(), and setConstraints().

10.6.5.3 Mapping* model::Bone::m_lMapping [private]

Definition at line 35 of file Bone.h.

Referenced by Bone(), operator!=(), operator=(), operator==(), and setMapping().

10.6.5.4 Bone* model::Bone::m_parent [private]

Definition at line 41 of file Bone.h.

Referenced by Bone(), operator=(), and setParent().

10.6.5.5 string model::Bone::m_sName [private]

Definition at line 44 of file Bone.h.

Referenced by Bone(), operator!=(), operator=(), operator==(), and setName().

10.6.5.6 Bone* model::Bone::m_vChilds [private]

Definition at line 33 of file Bone.h.

Referenced by addChild(), Bone(), and operator=().

10.6.5.7 IvVector3* model::Bone::m_vDirection [private]

Definition at line 42 of file Bone.h.

Referenced by Bone(), operator!=(), operator=(), operator==(), and setDirection().

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

- model/Bone.h
- model/Bone.cpp

10.7 model::Constraint Class Reference

`#include <Constraint.h>`

Public Member Functions

- Constraint ()
- ~Constraint ()
- Constraint (const Constraint &other)
- Constraint & operator= (const Constraint &other)
- bool operator== (const Constraint &other) const
- bool operator!= (const Constraint &other) const

10.7.1 Detailed Description

Write brief comment for Constraint here.

Write detailed description for Constraint here.

Remarks:
Write remarks for Constraint here.

See also:
Separate items with the '|' character.

Definition at line 25 of file Constraint.h.

10.7.2 Constructor & Destructor Documentation

10.7.2.1 model::Constraint::Constraint () [inline]

Write brief comment for Constraint here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Constraint here.

Remarks:
Write remarks for Constraint here.

See also:
Separate items with the '|' character.

Definition at line 45 of file Constraint.h.

```
45 {};
```

10.7.2.2 model::Constraint::~Constraint () [inline]

Write brief comment for ~Constraint here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Constraint here.

Remarks:
Write remarks for ~Constraint here.

See also:
Separate items with the '|' character.

Definition at line 61 of file Constraint.h.

```
61 {};
```

10.7.2.3 Constraint::Constraint (const Constraint & *other*)

Write brief comment for Constraint here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Constraint here.

Remarks:
Write remarks for Constraint here.

See also:
Separate items with the '|' character.

Definition at line 23 of file Constraint.cpp.

```
23                                                  {
24
25 }
```

10.7.3 Member Function Documentation

10.7.3.1 bool Constraint::operator!= (const Constraint & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 109 of file Constraint.cpp.

```
110 {
111         if( true ) {
112         return false;
113         }
114     return true;
115 }
```

10.7.3.2 Constraint & Constraint::operator= (const Constraint & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 48 of file Constraint.cpp.

```
49 {
50     // if same object
51     if ( this == &other )
52         return *this;
53
54
55     return *this;
56 }
```

10.7.3.3 bool Constraint::operator== (const Constraint & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 79 of file Constraint.cpp.

```
80 {
81
82     if( true ) {
83         return true;
84         }
85     return false;
86 }
```

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

- model/Constraint.h
- model/Constraint.cpp

10.8 model::Contour Class Reference

`#include <Contour.h>`

Public Member Functions

- Contour ()
- ~Contour ()
- Contour (const Contour &other)
- Contour & operator= (const Contour &other)
- bool operator== (const Contour &other) const
- bool operator!= (const Contour &other) const

10.8.1 Detailed Description

Write brief comment for Contour here.

Write detailed description for Contour here.

Remarks:
Write remarks for Contour here.

See also:
Separate items with the '|' character.

Definition at line 25 of file Contour.h.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 model::Contour::Contour () [inline]

Write brief comment for Contour here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Contour here.

Remarks:
Write remarks for Contour here.

See also:
Separate items with the '|' character.

Definition at line 44 of file Contour.h.

```
44 {};
```

10.8.2.2 model::Contour::~Contour () [inline]

Write brief comment for ~Contour here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Contour here.

Remarks:
Write remarks for ~Contour here.

See also:
Separate items with the '|' character.

Definition at line 60 of file Contour.h.

```
60 {};
```

10.8.2.3 Contour::Contour (const Contour & *other*)

Write brief comment for Contour here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Contour here.

Remarks:
Write remarks for Contour here.

See also:
Separate items with the '|' character.

Definition at line 23 of file Contour.cpp.

```
23                                      {
24
25 }
```

10.8.3 Member Function Documentation

10.8.3.1 bool Contour::operator!= (const Contour & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 109 of file Contour.cpp.

```
110 {
111         if( true ) {
112         return false;
113         }
114     return true;
115 }
```

10.8.3.2 Contour & Contour::operator= (const Contour & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 48 of file Contour.cpp.

```
49 {
50     // if same object
51     if ( this == &other )
52         return *this;
53
54
55     return *this;
56 }
```

10.8.3.3 bool Contour::operator== (const Contour & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 79 of file Contour.cpp.

```
80 {
81
82     if( true ) {
83         return true;
84         }
85     return false;
86 }
```

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

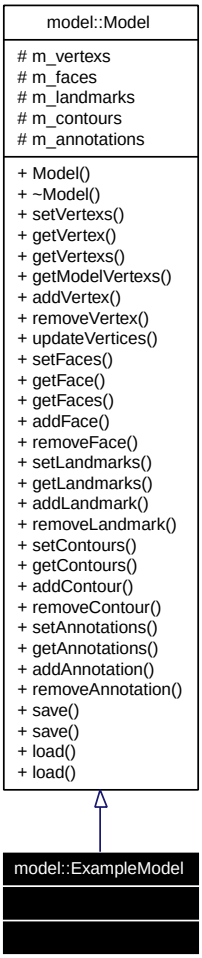
- model/Contour.h
- model/Contour.cpp

10.9 model::ExampleModel Class Reference

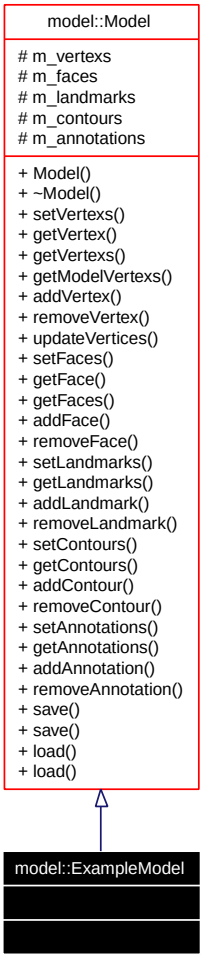
An ExampleModel in PCG.

`#include <ExampleModel.h>`

Inheritance diagram for model::ExampleModel:



Collaboration diagram for model::ExampleModel:



10.9.1 Detailed Description

An ExampleModel in PCG.

Definition at line 20 of file ExampleModel.h.

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

- model/ExampleModel.h

10.10 functions::Exporter Class Reference

`#include <Exporter.h>`

Public Member Functions

- Exporter ()
- ~Exporter ()
- void exportModel (ExampleModel *model, const char *filename)
- void exportModel (ReferenceModel *model, const char *filename)

10.10.1 Detailed Description

Write brief comment for Exporter here.

Write detailed description for Exporter here.

Remarks:
Write remarks for Exporter here.

See also:
Separate items with the '|' character.

Definition at line 48 of file Exporter.h.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 Exporter::Exporter ()

Definition at line 9 of file Exporter.cpp.

```
9                   {
10
11 }
```

10.10.2.2 functions::Exporter::~Exporter () [inline]

Write brief comment for ~Exporter here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for ~Exporter here.

Remarks:
Write remarks for ~Exporter here.

See also:
Separate items with the '|' character.

Definition at line 65 of file Exporter.h.

```
65 {};
```

10.10.3 Member Function Documentation

10.10.3.1 void Exporter::exportModel (ReferenceModel * *model*, const char * *filename*)

Write brief comment for exportModel here.

Parameters:
model Description of parameter model.

filename Description of parameter filename.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for exportModel here.

Remarks:
Write remarks for exportModel here.

See also:
Separate items with the '|' character.

Definition at line 33 of file Exporter.cpp.

References exportModel(), and log::write().

```
33                                                                    {
34   log::write("Exporter::exportModel");
35   log::write("exportModel-ref",*model);
36   exportModel((ExampleModel*) model, filename);
37 }
```

Here is the call graph for this function:

10.10.3.2 void Exporter::exportModel (ExampleModel * *model*, const char * *filename*)

Write brief comment for exportModel here.

Parameters:

model Description of parameter model.

filename Description of parameter filename.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for exportModel here.

Remarks:

Write remarks for exportModel here.

See also:

Separate items with the '|' character.

Definition at line 59 of file Exporter.cpp.

References model::Model::getFaces(), model::Model::getVertexs(), and log::write().

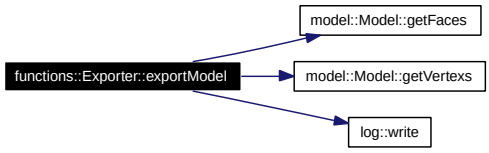
Referenced by pcggui::ExportAll(), exportModel(), and Batch::ExportModels().

```
59                                                                    {
60         log::write("exportModel-Ex",*model);
61         //Make data
62         string coordIndex = "";
63         string normalIndex = "";
64         string points = "";
65         string vectors = "";
66         char tmp[100];
```

```
67          log::write("Exporter get faces");
68          vector<Face*> faces = model->getFaces();
69          for(int i=0; i<(int)faces.size(); i++) {
70                  #ifndef __WINDOWS__
71                  sprintf(tmp,"%d, %d, %d,-1,",faces[i]->getVertex(0)->getIndex(),faces[i]->getVertex(1)->
72                  #else
73                  sprintf_s(tmp,"%d, %d, %d,-1,",faces[i]->getVertex(0)->getIndex(),faces[i]->getVertex(1)
74                  #endif
75                  coordIndex.append(tmp);
76                  normalIndex.append(tmp);
77                  /*cout << "Value:" << vertexs[i]->getCoordinates()->x << "," << vertexs[i]->getCoordinat
78                  cout << "Tmp: " << tmp << "\n";
79                  cout << "Point: " << points << "\n";
80                  system("PAUSE");*/
81          }
82          log::write("Exporter get vertex");
83          vector<Vertex*> vertexs = model->getVertexs();
84          for(int i=0; i<(int)vertexs.size(); i++) {
85                  #ifndef __WINDOWS__
86                  sprintf(tmp,"%.4f %.4f %.4f,",vertexs[i]->getCoordinates()->x,vertexs[i]->getCoordinates
87                  #else
88                  sprintf_s(tmp,"%.4f %.4f %.4f,",vertexs[i]->getCoordinates()->x,vertexs[i]->getCoordinat
89                  #endif
90                  points.append(tmp);
91                  #ifndef __WINDOWS__
92                  sprintf(tmp,"%.4f %.4f %.4f,",vertexs[i]->getNormal()->x,vertexs[i]->getNormal()->y,vert
93                  #else
94                  sprintf_s(tmp,"%.4f %.4f %.4f,",vertexs[i]->getNormal()->x,vertexs[i]->getNormal()->y,ve
95                  #endif
96                  vectors.append(tmp);
97                  /*cout << "Value:" << vertexs[i]->getCoordinates()->x << "," << vertexs[i]->getCoordinat
98                  cout << "Tmp: " << tmp << "\n";
99                  cout << "Point: " << points << "\n";
100                  system("PAUSE");*/
101         }
102         log::write("Exporter make xml");
103         //Make XML doc
104         TiXmlDocument doc;
105         TiXmlDeclaration * decl = new TiXmlDeclaration( "1.0", "UTF-8", "" );
106         doc.LinkEndChild( decl );
107
108         TiXmlElement * X3D = new TiXmlElement( "X3D" );
109         X3D->SetAttribute("profile","Immersive");
110         doc.LinkEndChild( X3D );
111
112         TiXmlElement * scene = new TiXmlElement( "Scene" );
113         X3D->LinkEndChild( scene );
114
115         TiXmlElement * transform = new TiXmlElement( "Transform" );
116         scene->LinkEndChild( transform );
117
118         TiXmlElement * shape = new TiXmlElement( "Shape" );
119         transform->LinkEndChild( shape );
120
121         TiXmlElement * indexedFaceSet = new TiXmlElement( "IndexedFaceSet" );
122         indexedFaceSet->SetAttribute("normalPerVertex","true");
123         indexedFaceSet->SetAttribute("coordIndex",coordIndex.c_str());
124         indexedFaceSet->SetAttribute("normalIndex",normalIndex.c_str());
125         shape->LinkEndChild( indexedFaceSet );
126
127         TiXmlElement * coordinate = new TiXmlElement( "Coordinate" );
128         coordinate->SetAttribute("point",points.c_str());
```

```
129         indexedFaceSet->LinkEndChild( coordinate );
130
131         TiXmlElement * normal = new TiXmlElement( "Normal" );
132         normal->SetAttribute("vector",vectors.c_str());
133         indexedFaceSet->LinkEndChild( normal );
134
135         doc.SaveFile( filename );
136 }
```

Here is the call graph for this function:



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

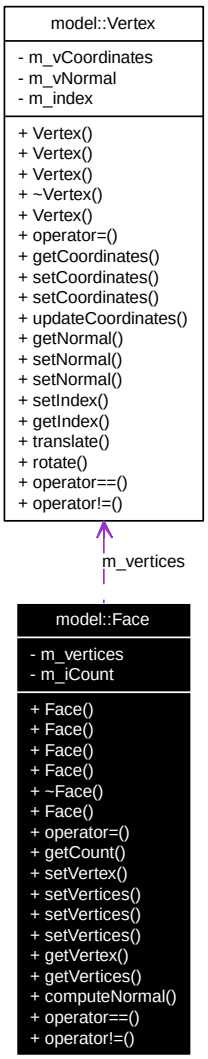
- Functions/Exporter.h
- Functions/Exporter.cpp

10.11 model::Face Class Reference

A Face class.

`#include <Face.h>`

Collaboration diagram for model::Face:



Public Member Functions

- Face ()

- Face (Vertex *vertex1, Vertex *vertex2)
- Face (Vertex *vertex1, Vertex *vertex2, Vertex *vertex3)
- Face (Vertex *vertex1, Vertex *vertex2, Vertex *vertex3, Vertex *vertex4)
- ~Face ()
- Face (const Face &other)
- Face & operator= (const Face &other)
- int getCount ()
- void setVertex (int iIndex, Vertex *vertex)
- void setVertices (Vertex *vertex1, Vertex *vertex2)
- void setVertices (Vertex *vertex1, Vertex *vertex2, Vertex *vertex3)
- void setVertices (Vertex *vertex1, Vertex *vertex2, Vertex *vertex3, Vertex *vertex4)
- Vertex * getVertex (int iIndex)
- vector< Vertex * > getVertices ()
- IvVector3 computeNormal ()
- bool operator== (const Face &other) const
- bool operator!= (const Face &other) const

Private Attributes

- Vertex * m_vertices
- int m_iCount

10.11.1 Detailed Description

A Face class.

Constition of Vertex's.

See also:
Vertex

Definition at line 33 of file Face.h.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 model::Face::Face () [inline]

Definition at line 43 of file Face.h.

```
43 {};
```

10.11.2.2 Face::Face (Vertex * *vertex1*, Vertex * *vertex2*)

Write brief comment for Face here.

Parameters:

vertex1 Description of parameter vertex1.

vertex2 Description of parameter vertex2.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Face here.

Remarks:

Write remarks for Face here.

See also:

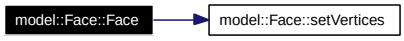
Separate items with the '|' character.

Definition at line 27 of file Face.cpp.

References setVertices().

```
27                                              {
28         setVertices(vertex1, vertex2);
29 }
```

Here is the call graph for this function:



10.11.2.3 Face::Face (Vertex * *vertex1*, Vertex * *vertex2*, Vertex * *vertex3*)

Write brief comment for Face here.

Parameters:

vertex1 Description of parameter vertex1.

vertex2 Description of parameter vertex2.

vertex3 Description of parameter vertex3.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Face here.

Remarks:
Write remarks for Face here.

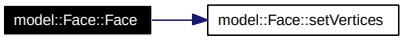
See also:
Separate items with the '|' character.

Definition at line 55 of file Face.cpp.

References setVertices().

```
55                                                                  {
56         setVertices(vertex1, vertex2, vertex3);
57 }
```

Here is the call graph for this function:



10.11.2.4 Face::Face (Vertex * *vertex1*, Vertex * *vertex2*, Vertex * *vertex3*, Vertex * *vertex4*)

Write brief comment for Face here.

Parameters:
vertex1 Description of parameter vertex1.
vertex2 Description of parameter vertex2.
vertex3 Description of parameter vertex3.
vertex4 Description of parameter vertex4.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Face here.

Remarks:
Write remarks for Face here.

See also:
Separate items with the '|' character.

Definition at line 86 of file Face.cpp.

References setVertices().

```
86                                                                  {
87         setVertices(vertex1, vertex2, vertex3, vertex4);
88 }
```

Here is the call graph for this function:

10.11.2.5 model::Face::~Face () [inline]

Definition at line 47 of file Face.h.

```
47 {};
```

10.11.2.6 Face::Face (const Face & *other*)

Write brief comment for Face here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Face here.

Remarks:
Write remarks for Face here.

See also:
Separate items with the '|' character.

Definition at line 108 of file Face.cpp.

References m_iCount, and m_vertices.

```
108                             {
109     m_vertices = other.m_vertices;
110         m_iCount = other.m_iCount;
111 }
```

10.11.3 Member Function Documentation

10.11.3.1 IvVector3 Face::computeNormal ()

Write brief comment for computeNormal here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for computeNormal here.

Remarks:
Write remarks for computeNormal here.

See also:
Separate items with the '|' character.

Definition at line 307 of file Face.cpp.

```
307                                 {
308         return IvVector3();
309 }
```

10.11.3.2 int Face::getCount ()

Write brief comment for getCount here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getCount here.

Remarks:
Write remarks for getCount here.

See also:
Separate items with the '|' character.

Definition at line 395 of file Face.cpp.

```
395                     {
396         return m_iCount;
397 }
```

10.11.3.3 Vertex * Face::getVertex (int *iIndex*)

Write brief comment for getVertex here.

Parameters:
iIndex Description of parameter iIndex.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getVertex here.

Remarks:
Write remarks for getVertex here.

See also:
Separate items with the '|' character.

Definition at line 285 of file Face.cpp.

References m_vertices.

Referenced by functions::Importer::readAttributes(), pcggui::Render(), and model::Model::save().

```
285                                 {
286         return m_vertices[iIndex];
287 }
```

10.11.3.4 vector< Vertex * > Face::getVertices ()

Write brief comment for getVertices here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getVertices here.

Remarks:
Write remarks for getVertices here.

See also:
Separate items with the '|' character.

Definition at line 417 of file Face.cpp.

```
417                                     {
418         return m_vertices;
419 }
```

10.11.3.5 bool Face::operator!= (const Face & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 365 of file Face.cpp.

References m_iCount, and m_vertices.

```
366 {
367         if(other.m_vertices[0] == m_vertices[0]
368         && other.m_vertices[1] == m_vertices[1]
369         && other.m_vertices[2] == m_vertices[2]
370         && other.m_vertices[3] == m_vertices[3]
371         && other.m_iCount == m_iCount) {
372         return false;
373         }
374     return true;
375 }
```

10.11.3.6 Face & Face::operator= (const Face & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 134 of file Face.cpp.

References m_iCount, and m_vertices.

```
135 {
136     // if same object
137     if ( this == &other )
138         return *this;
139 
140     m_vertices = other.m_vertices;
141         m_iCount = other.m_iCount;
142 
143     return *this;
144 }
```

10.11.3.7 bool Face::operator== (const Face & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 332 of file Face.cpp.

References m_iCount, and m_vertices.

```
333 {
334     if(other.m_vertices[0] == m_vertices[0]
335         && other.m_vertices[1] == m_vertices[1]
336         && other.m_vertices[2] == m_vertices[2]
337         && other.m_vertices[3] == m_vertices[3]
338         && other.m_iCount == m_iCount) {
339         return true;
340         }
341     return false;
342 }
```

10.11.3.8 void Face::setVertex (int *iIndex*, Vertex * *vertex*)

Write brief comment for setVertex here.

Parameters:
iIndex Description of parameter iIndex.

vertex Description of parameter vertex.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setVertex here.

Remarks:
Write remarks for setVertex here.

See also:
Separate items with the '|' character.

Definition at line 167 of file Face.cpp.

References m_vertices.

```
167                                                 {
168                 m_vertices[iIndex] = vertex;
169 }
```

10.11.3.9 void Face::setVertices (Vertex * *vertex1*, Vertex * *vertex2*, Vertex * *vertex3*, Vertex * *vertex4*)

Write brief comment for setVertices here.

Parameters:

vertex1 Description of parameter vertex1.

vertex2 Description of parameter vertex2.

vertex3 Description of parameter vertex3.

vertex4 Description of parameter vertex4.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setVertices here.

Remarks:

Write remarks for setVertices here.

See also:

Separate items with the '|' character.

Definition at line 256 of file Face.cpp.

References m_iCount, and m_vertices.

```
256                                                                                    {
257         m_iCount = 4;
258         m_vertices.push_back(vertex1);
259         m_vertices.push_back(vertex2);
260         m_vertices.push_back(vertex3);
261         m_vertices.push_back(vertex4);
262 }
```

10.11.3.10 void Face::setVertices (Vertex * *vertex1*, Vertex * *vertex2*, Vertex * *vertex3*)

Write brief comment for setVertices here.

Parameters:

vertex1 Description of parameter vertex1.

vertex2 Description of parameter vertex2.

vertex3 Description of parameter vertex3.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setVertices here.

Remarks:
Write remarks for setVertices here.

See also:
Separate items with the '|' character.

Definition at line 222 of file Face.cpp.

References m_iCount, and m_vertices.

```
222                                                                    {
223         m_iCount = 3;
224         m_vertices.push_back(vertex1);
225         m_vertices.push_back(vertex2);
226         m_vertices.push_back(vertex3);
227 }
```

10.11.3.11 void Face::setVertices (Vertex * *vertex1*, Vertex * *vertex2*)

Write brief comment for setVertices here.

Parameters:
vertex1 Description of parameter vertex1.

vertex2 Description of parameter vertex2.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setVertices here.

Remarks:
Write remarks for setVertices here.

See also:
Separate items with the '|' character.

Definition at line 192 of file Face.cpp.

References m_iCount, and m_vertices.

Referenced by pcggui::CovertScene(), Face(), and model::Model::load().

```
192                                                    {
193         m_iCount = 2;
194         m_vertices.push_back(vertex1);
195         m_vertices.push_back(vertex2);
196 }
```

10.11.4 Member Data Documentation

10.11.4.1 int model::Face::m_iCount [private]

Definition at line 40 of file Face.h.

Referenced by Face(), operator!=(), operator=(), operator==(), and setVertices().

10.11.4.2 Vertex* model::Face::m_vertices [private]

Definition at line 36 of file Face.h.

Referenced by Face(), getVertex(), operator!=(), operator=(), operator==(), setVertex(), and setVertices().

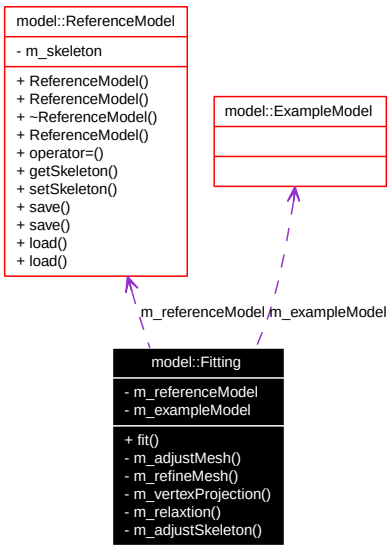
The documentation for this class was generated from the following files:

- model/Face.h
- model/Face.cpp

10.12 model::Fitting Class Reference

`#include <Fitting.h>`

Collaboration diagram for model::Fitting:



Public Member Functions

- void fit (ReferenceModel referenceModel, ExampleModel exampleModel)

Private Member Functions

- void m_adjustMesh ()
- void m_refineMesh ()
- void m_vertexProjection ()
- void m_relaxtion ()
- void m_adjustSkeleton ()

Private Attributes

- ReferenceModel * m_referenceModel
- ExampleModel * m_exampleModel

10.12.1 Member Function Documentation

10.12.1.1 void Fitting::fit (ReferenceModel *referenceModel*, ExampleModel *exampleModel*)

Write brief comment for fit here.

Parameters:

referenceModel Description of parameter referenceModel.

exampleModel Description of parameter exampleModel.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for fit here.

Remarks:

Write remarks for fit here.

See also:

Separate items with the '|' character.

Definition at line 121 of file Fitting.cpp.

```
121                                                                    {
122
123 }
```

10.12.1.2 void Fitting::m_adjustMesh () [private]

Write brief comment for m_adjustMesh here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for m_adjustMesh here.

Remarks:

Write remarks for m_adjustMesh here.

See also:

Separate items with the '|' character.

Definition at line 20 of file Fitting.cpp.

```
20                              {
21
22 }
```

10.12.1.3 void Fitting::m_adjustSkeleton () [private]

Write brief comment for m_adjustSkeleton here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for m_adjustSkeleton here.

Remarks:
Write remarks for m_adjustSkeleton here.

See also:
Separate items with the '|' character.

Definition at line 96 of file Fitting.cpp.

```
96                                  {
97
98 }
```

10.12.1.4 void Fitting::m_refineMesh () [private]

Write brief comment for m_refineMesh here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for m_refineMesh here.

Remarks:
Write remarks for m_refineMesh here.

See also:
Separate items with the '|' character.

Definition at line 39 of file Fitting.cpp.

```
39                              {
40
41 }
```

10.12.1.5 void Fitting::m_relaxtion () [private]

Write brief comment for m_relaxtion here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for m_relaxtion here.

Remarks:
Write remarks for m_relaxtion here.

See also:
Separate items with the '|' character.

Definition at line 77 of file Fitting.cpp.

```
77                              {
78
79 }
```

10.12.1.6 void Fitting::m_vertexProjection () [private]

Write brief comment for m_vertexProjection here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for m_vertexProjection here.

Remarks:
Write remarks for m_vertexProjection here.

See also:
Separate items with the '|' character.

Definition at line 58 of file Fitting.cpp.

```
58                                      {
59
60 }
```

10.12.2 Member Data Documentation

10.12.2.1 ExampleModel* model::Fitting::m_exampleModel [private]

Definition at line 19 of file Fitting.h.

10.12.2.2 ReferenceModel* model::Fitting::m_referenceModel [private]

Definition at line 18 of file Fitting.h.

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

- model/Fitting.h
- model/Fitting.cpp

10.13 functions::Importer Class Reference

Write brief comment for Importer here.

`#include <Importer.h>`

Public Member Functions

- Importer ()
- ~Importer ()

  Write brief comment for ~Importer here.

- bool importModel (const char *filename, ExampleModel *am)
- bool importDirectory (const char *dirname, vector< ExampleModel * > vem)

Private Member Functions

- void readElements (TiXmlNode *doc, ExampleModel *em)
- void readAttributes (TiXmlElement *pElement, ExampleModel *em)

Private Attributes

- char * m_dir
- int globalCIndex
- int globalNIndex
- vector< int > globalCoordIndexs
- vector< int > globalNormalIndexs
- vector< float > translation
- vector< float > rotation
- vector< float > scale
- vector< float > scaleOrientation
- vector< float > center

10.13.1 Detailed Description

Write brief comment for Importer here.

Write detailed description for Importer here.

Remarks:

Write remarks for Importer here.

See also:
Separate items with the '|' character.

Definition at line 38 of file Importer.h.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 Importer::Importer ()

Write brief comment for Importer here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Importer here.

Remarks:
Write remarks for Importer here.

See also:
Separate items with the '|' character.

Definition at line 22 of file Importer.cpp.

```
22                      {
23
24 }
```

10.13.2.2 functions::Importer::~Importer () [inline]

Write brief comment for ~Importer here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Importer here.

Remarks:
Write remarks for ~Importer here.

See also:
Separate items with the '|' character.

Definition at line 65 of file Importer.h.

```
65 {};
```

10.13.3 Member Function Documentation

10.13.3.1 bool functions::Importer::importDirectory (const char * *dirname*, vector< ExampleModel * > *vem*)

10.13.3.2 bool Importer::importModel (const char * *filename*, ExampleModel * *em*)

Write brief comment for importModel here.

Parameters:

filename Description of parameter filename.

am Description of parameter am.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for importModel here.

Remarks:

Write remarks for importModel here.

See also:

Separate items with the '|' character.

Definition at line 49 of file Importer.cpp.

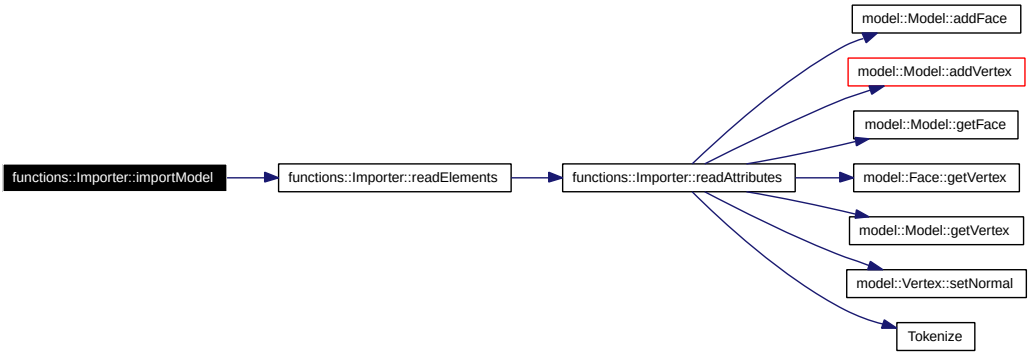
References center, globalCIndex, globalNIndex, readElements(), rotation, scale, scaleOrientation, and translation.

Referenced by Batch::importData(), Batch::importModel(), and main().

```
49                                                                  {
50         //ExampleModel* em = new ExampleModel();
51         globalCIndex=0;
52         globalNIndex=0;
53         translation = vector<float>(4,0);
54         rotation = vector<float>(4,0);
55         rotation[2] = 1;
56         scale = vector<float>(4,1);
57         scaleOrientation = vector<float>(4,0);
58         scaleOrientation[2] = 1;
59         center = vector<float>(4,0);
```

```
60
61          //New code
62          TiXmlDocument doc(filename);
63          bool loadOkay = doc.LoadFile();
64          if (loadOkay)
65          {
66                  //printf("\n%s:\n", filename);
67                  readElements(&doc,em);
68          }
69          else
70          {
71                  printf("Failed to load file \"%s\"\n", filename);
72          }
73
74          return loadOkay;
75 }
```

Here is the call graph for this function:



10.13.3.3 void Importer::readAttributes (TiXmlElement * *pElement*, ExampleModel * *em*) [private]

Write brief comment for readAttributes here.

Parameters:

pElement Description of parameter pElement.

em Description of parameter em.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for readAttributes here.

Remarks:

Write remarks for readAttributes here.

See also:
Separate items with the '|' character.

Definition at line 140 of file Importer.cpp.

References model::Model::addFace(), model::Model::addVertex(), center, model::Model::getFace(), model::Face::getVertex(), model::Model::getVertex(), globalCIndex, globalCoordIndexs, globalNIndex, globalNormalIndexs, rotation, scale, scaleOrientation, model::Vertex::setNormal(), Tokenize(), and translation.

Referenced by readElements().

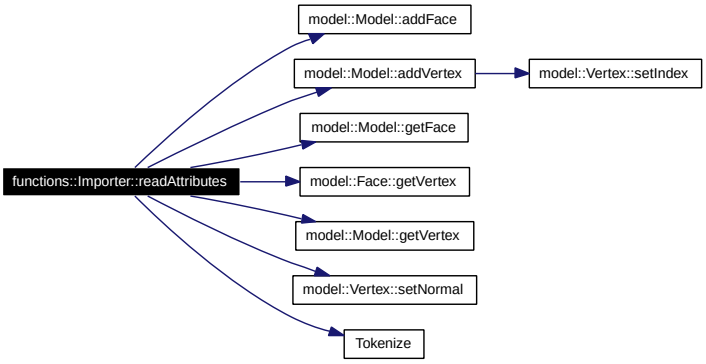
```
141 {
142         if ( pElement ) {
143 
144                 TiXmlAttribute* pAttrib=pElement->FirstAttribute();
145 
146 
147                 while (pAttrib)
148                 {
149                         if(!strcmp(pAttrib->Name(),"coordIndex")) {
150                                 globalCoordIndexs.clear();
151                                 vector<string> tokens;
152                                 string s = string(pAttrib->Value());
153                                 //cout << "String: " << s << "\n";
154                                 Tokenize(s,tokens,",");
155                                 //printf("Tokens: %d\n",tokens.size());
156                                 for(int i=0;i<(int)tokens.size();i++) {
157                                         //cout << i << " | " << tokens[i] << "\n";
158                                         globalCoordIndexs.push_back(convertTo<int>(tokens[i]));
159                                 }
160                         } else if(!strcmp(pAttrib->Name(),"normalIndex")) {
161                                 globalNormalIndexs.clear();
162                                 vector<string> tokens;
163                                 string s = string(pAttrib->Value());
164                                 Tokenize(s,tokens,",");
165                                 //printf("Tokens: %d\n",tokens.size());
166                                 for(int i=0;i<(int)tokens.size();i++) {
167                                         //cout << i << " | " << tokens[i] << "\n";
168                                         globalNormalIndexs.push_back(convertTo<int>(tokens[i]));
169                                 }
170                         } else if(!strcmp(pAttrib->Name(),"point")) {
171                                 vector<string> tokens;
172                                 string str = string(pAttrib->Value());
173                                 Tokenize(str,tokens,",");
174                                 Matrix vertexs;
175                                 //printf("Tokens: %d\n",tokens.size());
176                                 for(int i=0;i<(int)tokens.size();i++) {
177                                         vector<string> tokens2;
178                                         Tokenize(tokens[i],tokens2," ");
179                                         //printf("Tokens: %d - ",tokens2.size());
180                                         //cout << i << " | " << tokens[i] << "\n";
181 
182                                         Matrix tmp(4,1);
183                                         tmp << convertTo<float>(tokens2[0]) << convertTo<float>(tokens2
184                                         if(i==0) {
185                                                 vertexs = tmp;
186                                         } else {
187                                                 vertexs = vertexs | tmp;
188                                         }
189                                 }
```

```
190                             //Generate transforamtion matrix
191                             Matrix translationMatrix(4,4);
192                             Matrix rotationMatrix(4,4);
193                             Matrix scaleMatrix(4,4);
194                             Matrix centerMatrix(4,4);
195                             Matrix scaleOrientationMatrix(4,4);
196                             Matrix minusCenterMatrix(4,4);
197                             Matrix minusScaleOrientationMatrix(4,4);
198
199                             //Translation
200                             translationMatrix       << 1.0 << 0.0 << 0.0 << translation[0
201                                                                  << 0.0 << 1.0 << 0.0
202                                                                  << 0.0 << 0.0 << 1.0
203                                                                  << 0.0 << 0.0 << 0.0
204
205                             //Rotation
206                             //float angle = (rotation[3]*(180/3.14159265))/2;
207                             //float w = cos((angle*3.14159265)/180);
208                             float w = rotation[3];
209                             float x = rotation[0];
210                             float y = rotation[1];
211                             float z = rotation[2];
212                             float c = cos(w);
213                             float s = sin(w);
214                             float t = 1 - c;
215
216
217                             rotationMatrix  << t*pow(x,2)+c << t*x*y-s*z    << t*x*z+s*y
218                                             << t*x*y+s*z    << t*pow(y,2)
219                                             << t*x*z-s*y    << t*y*z+s*x
220                                             << 0.0                  << 0.
221
222                             //Scale
223                             scaleMatrix     = 0.0;
224                             scaleMatrix(1,1) = scale[0];
225                             scaleMatrix(2,2) = scale[1];
226                             scaleMatrix(3,3) = scale[2];
227                             scaleMatrix(4,4) = 1.0;
228
229                             //centerMatrix
230                             centerMatrix    << 1.0 << 0.0 << 0.0 << center[0]
231                                             << 0.0 << 1.0 << 0.0 << cente
232                                             << 0.0 << 0.0 << 1.0 << cente
233                                             << 0.0 << 0.0 << 0.0 << 1.0;
234
235                             //MinuScenterMatrix
236                             minusCenterMatrix       << 1.0 << 0.0 << 0.0 << -center[0]
237                                                     << 0.0 << 1.0 << 0.0
238                                                     << 0.0 << 0.0 << 1.0
239                                                     << 0.0 << 0.0 << 0.0
240
241                             //scaleOrientationMatrix
242                             scaleOrientationMatrix  << 1.0 << 0.0 << 0.0 << scaleOrientat
243                                                                          << 0.0 << 1.0
244                                                                          << 0.0 << 0.0
245                                                                          << 0.0 << 0.0
246
247                             //MinusScaleOrientationMatrix
248                             minusScaleOrientationMatrix     << 1.0 << 0.0 << 0.0 << -scal
249                                                                                  << 0.
250                                                                                  << 0.
251                                                                                  << 0.
```

```
252                             //vertexs = vertexs.t();
253                             //multiply the matrixs
254                             //P' = T * C * R * SR * S * -SR * -C * P
255                             vertexs = translationMatrix * centerMatrix * rotationMatrix * scaleOrie
256                             //vertexs =   rotationMatrix * scaleMatrix* vertexs;
257                             //vertexs =    rotationMatrix* vertexs;
258
259
260
261
262                             //vertexs = translationMatrix * rotationMatrix *  scaleMatrix  * vertex
263                             //vertexs = vertexs * minusCenterMatrix * minusScaleOrientationMatrix
264                             //vertexs = centerMatrix * rotationMatrix * scaleOrientationMatrix * sc
265                             //vertexs = translationMatrix * vertexs;
266                             /*cout << "Scale:\n";
267                             cout << setw(10) << setprecision(5) << scaleMatrix;
268                             cout << "Rotate:\n";
269                             cout << setw(10) << setprecision(5) << rotationMatrix;
270                             cout << "Translation:\n";
271                             cout << setw(10) << setprecision(5) << translationMatrix;*/
272                             //vertexs = vertexs.t();
273
274
275
276                             //Extract each vertex
277                             for(int i=1; i<=vertexs.Ncols();i++) {
278                                     //printf("Vertex add %d\n",i);
279                                     Vertex* v = new Vertex((float)vertexs(1,i),(float)vertexs(2,i),
280                                     em->addVertex(v);
281                             }
282                             //printf("Adding faces - %d\n",globalCoordIndexs.size());
283                             for(int i=0;i<(int)globalCoordIndexs.size();i=i+4) {
284                                     //printf("Face: %d < %d\n",i,globalCoordIndexs.size());
285                                     //printf("VertexI: %d,%d,%d\n",globalCoordIndexs[i],globalCoord
286                                     Face* f = new Face(em->getVertex(globalCIndex+globalCoordIndexs
287                                     em->addFace(f);
288                             }
289                             globalCIndex += tokens.size();
290                             translation = vector<float>(4,0);
291                             rotation = vector<float>(4,0);
292                             rotation[2] = 1;
293                             scale = vector<float>(4,1);
294                             scaleOrientation = vector<float>(4,0);
295                             scaleOrientation[2] = 1;
296                             center = vector<float>(4,0);
297                     } else if(!strcmp(pAttrib->Name(),"vector")) {
298                             vector<IvVector3*> normals;
299                             vector<string> tokens;
300                             string s = string(pAttrib->Value());
301                             Tokenize(s,tokens,",");
302                             //printf("Tokens: %d\n",tokens.size());
303                             for(int i=0;i<(int)tokens.size();i++) {
304                                     vector<string> tokens2;
305                                     Tokenize(tokens[i],tokens2," ");
306                                     //printf("Tokens: %d - ",tokens2.size());
307                                     //cout << i << " | " << tokens[i] << "\n";
308                                     IvVector3* tempVector = new IvVector3(convertTo<float>(tokens2[
309                                     normals.push_back(tempVector);
310                             }
311                             //printf("Adding normals - %d - %d\n",globalNormalIndexs.size(),normals
312                             for(int i=0;i<(int)globalNormalIndexs.size();i=i+4) {
313                                     //printf("Normal: %d < %d --- %d < %d\n",i,globalNormalIndexs.s
```

```
314                                     //printf("globalNormalIndexs: %d < %d\n",globalNormal
315                                     //printf("globalNormalIndexs: %d < %d\n",globalNormal
316                                     //printf("globalNormalIndexs: %d < %d\n",globalNormal
317                                     em->getFace(globalNIndex+i/4)->getVertex(0)->setNorma
318                                     em->getFace(globalNIndex+i/4)->getVertex(1)->setNorma
319                                     em->getFace(globalNIndex+i/4)->getVertex(2)->setNorma
320                             }
321                             globalNIndex += tokens.size();
322                             //printf("Index: %d\n",globalNIndex);
323                     } else if(      !strcmp(pAttrib->Name(),"translation")  ||
324                                     !strcmp(pAttrib->Name(),"rotation")
325                                     !strcmp(pAttrib->Name(),"scale")
326                                     !strcmp(pAttrib->Name(),"center")
327                                     !strcmp(pAttrib->Name(),"scaleOrientation"))
328                             vector<string> tokens;
329                             string s = string(pAttrib->Value());
330                             Tokenize(s,tokens," ");
331                             //printf("Tokens: %d\n",tokens.size());
332                             for(int i=0;i<(int)tokens.size();i++) {
333                                     if(!strcmp(pAttrib->Name(),"translation")) {
334                                             translation[i] = convertTo<float>(tokens[i]);
335                                     } else if(!strcmp(pAttrib->Name(),"rotation")) {
336                                             rotation[i] = convertTo<float>(tokens[i]);
337                                     }else if(!strcmp(pAttrib->Name(),"scale")) {
338                                             scale[i] = convertTo<float>(tokens[i]);
339                                     }else if(!strcmp(pAttrib->Name(),"center")) {
340                                             center[i] = convertTo<float>(tokens[i]);
341                                     }else if(!strcmp(pAttrib->Name(),"scaleOrientation")
342                                             scaleOrientation[i] = convertTo<float>(tokens
343                                     }
344                             }
345                     }
346                     pAttrib=pAttrib->Next();
347             }
348     }
349 }
```

Here is the call graph for this function:



10.13.3.4 void Importer::readElements (TiXmlNode * *pParent*, ExampleModel * *em*) [private]

Write brief comment for readElements here.

Parameters:

doc Description of parameter doc.

em Description of parameter em.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for readElements here.

Remarks:

Write remarks for readElements here.

See also:

Separate items with the '|' character.

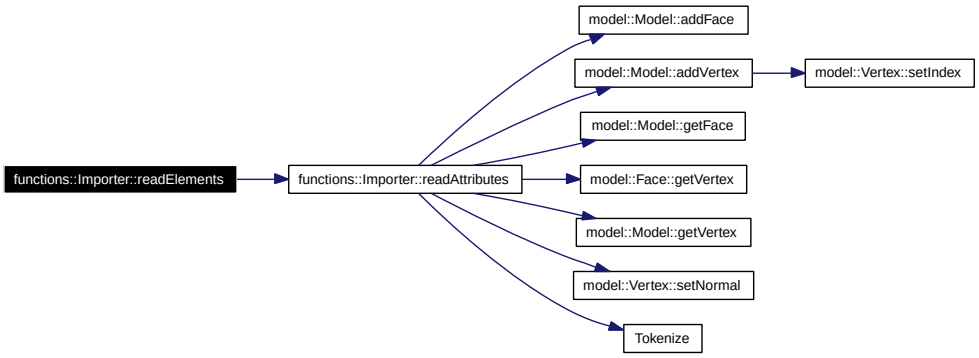
Definition at line 97 of file Importer.cpp.

References readAttributes().

Referenced by importModel().

```
97                                                                  {
98         if ( !pParent ) return;
99
100         TiXmlNode* pChild;
101         int t = pParent->Type();
102
103         if(t == TiXmlNode::ELEMENT) {
104                 //printf( "Element [%s]\n", pParent->Value() );
105
106                 if(     !strcmp(pParent->Value(),"IndexedFaceSet") ||
107                         !strcmp(pParent->Value(),"Coordinate") ||
108                         !strcmp(pParent->Value(),"Normal")) {
109                         readAttributes(pParent->ToElement(),em);
110                 } else if(!strcmp(pParent->Value(),"Transform")) {
111                         readAttributes(pParent->ToElement(),em);
112                 }
113         }
114         for ( pChild = pParent->FirstChild(); pChild != 0; pChild = pChild->NextSibling())
115         {
116                 readElements( pChild, em );
117         }
118 }
```

Here is the call graph for this function:



10.13.4 Member Data Documentation

10.13.4.1 vector<float> functions::Importer::center [private]

Definition at line 44 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.2 int functions::Importer::globalCIndex [private]

Definition at line 41 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.3 vector<int> functions::Importer::globalCoordIndexs [private]

Definition at line 42 of file Importer.h.

Referenced by readAttributes().

10.13.4.4 int functions::Importer::globalNIndex [private]

Definition at line 41 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.5 vector<int> functions::Importer::globalNormalIndexs [private]

Definition at line 43 of file Importer.h.

Referenced by readAttributes().

10.13.4.6 char* functions::Importer::m_dir [private]

Definition at line 40 of file Importer.h.

10.13.4.7 vector<float> functions::Importer::rotation [private]

Definition at line 44 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.8 vector<float> functions::Importer::scale [private]

Definition at line 44 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.9 vector<float> functions::Importer::scaleOrientation [private]

Definition at line 44 of file Importer.h.

Referenced by importModel(), and readAttributes().

10.13.4.10 vector<float> functions::Importer::translation [private]

Definition at line 44 of file Importer.h.

Referenced by importModel(), and readAttributes().

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

- Functions/Importer.h
- Functions/Importer.cpp

10.14 ISkin Class Reference

`#include <pcggui.h>`

Public Member Functions

- ISkin ()
- ~ISkin ()
- virtual int GetBoneInitTM (INode *pNode, Matrix3 &InitTM, bool bObjOffset=false)=0
- virtual int GetSkinInitTM (INode *pNode, Matrix3 &InitTM, bool bObjOffset=false)=0
- virtual int GetNumBones ()=0
- virtual INode * GetBone (int idx)=0
- virtual DWORD GetBoneProperty (int idx)=0
- virtual ISkinContextData * GetContextInterface (INode *pNode)=0
- virtual BOOL AddBone (INode *bone)=0
- virtual BOOL AddBones (INodeTab *bones)=0
- virtual BOOL RemoveBone (INode *bone)=0
- virtual void Invalidate ()=0

10.14.1 Constructor & Destructor Documentation

10.14.1.1 ISkin::ISkin () [inline]

Definition at line 115 of file pcggui.h.

```
115 {}
```

10.14.1.2 ISkin::~ISkin () [inline]

Definition at line 116 of file pcggui.h.

```
116 {}
```

10.14.2 Member Function Documentation

10.14.2.1 virtual BOOL ISkin::AddBone (INode * *bone*) [pure virtual]

10.14.2.2 virtual BOOL ISkin::AddBones (INodeTab * *bones*) [pure virtual]

10.14.2.3 virtual INode* ISkin::GetBone (int *idx*) [pure virtual]

Referenced by pcggui::ConvertSkin().

10.14.2.4 virtual int ISkin::GetBoneInitTM (INode * *pNode*, Matrix3 & *InitTM*, bool *bObjOffset* = false) [pure virtual]

10.14.2.5 virtual DWORD ISkin::GetBoneProperty (int *idx*) [pure virtual]

10.14.2.6 virtual ISkinContextData* ISkin::GetContextInterface (INode * *pNode*) [pure virtual]

Referenced by pcggui::ConvertSkin().

10.14.2.7 virtual int ISkin::GetNumBones () [pure virtual]

10.14.2.8 virtual int ISkin::GetSkinInitTM (INode * *pNode*, Matrix3 & *InitTM*, bool *bObjOffset* = false) [pure virtual]

10.14.2.9 virtual void ISkin::Invalidate () [pure virtual]

10.14.2.10 virtual BOOL ISkin::RemoveBone (INode * *bone*) [pure virtual]

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

- gui/pcggui/pcggui.h

10.15 ISkinContextData Class Reference

`#include <pcggui.h>`

Public Member Functions

- virtual int GetNumPoints ()=0
- virtual int GetNumAssignedBones (int vertexIdx)=0
- virtual int GetAssignedBone (int vertexIdx, int boneIdx)=0
- virtual float GetBoneWeight (int vertexIdx, int boneIdx)=0
- virtual int GetSubCurveIndex (int vertexIdx, int boneIdx)=0
- virtual int GetSubSegmentIndex (int vertexIdx, int boneIdx)=0
- virtual float GetSubSegmentDistance (int vertexIdx, int boneIdx)=0
- virtual Point3 GetTangent (int vertexIdx, int boneIdx)=0
- virtual Point3 GetOPoint (int vertexIdx, int boneIdx)=0
- virtual void SetWeight (int vertexIdx, int boneIdx, float weight)=0
- virtual void SetWeight (int vertexIdx, INode *bone, float weight)=0
- virtual void SetWeights (int vertexIdx, Tab< int > boneIdx, Tab< float > weights)=0
- virtual void SetWeights (int vertexIdx, INodeTab boneIdx, Tab< float > weights)=0

10.15.1 Member Function Documentation

10.15.1.1 virtual int ISkinContextData::GetAssignedBone (int *vertexIdx*, int *boneIdx*) [pure virtual]

Referenced by pcggui::ConvertSkin().

10.15.1.2 virtual float ISkinContextData::GetBoneWeight (int *vertexIdx*, int *boneIdx*) [pure virtual]

Referenced by pcggui::ConvertSkin().

10.15.1.3 virtual int ISkinContextData::GetNumAssignedBones (int *vertexIdx*) [pure virtual]

Referenced by pcggui::ConvertSkin().

10.15.1.4 virtual int ISkinContextData::GetNumPoints () [pure virtual]

Referenced by pcggui::ConvertSkin().

10.15.1.5 virtual Point3 ISkinContextData::GetOPoint (int *vertexIdx*, int *boneIdx*) [pure virtual]

10.15.1.6 virtual int ISkinContextData::GetSubCurveIndex (int *vertexIdx*, int *boneIdx*) [pure virtual]

10.15.1.7 virtual float ISkinContextData::GetSubSegmentDistance (int *vertexIdx*, int *boneIdx*) [pure virtual]

10.15.1.8 virtual int ISkinContextData::GetSubSegmentIndex (int *vertexIdx*, int *boneIdx*) [pure virtual]

10.15.1.9 virtual Point3 ISkinContextData::GetTangent (int *vertexIdx*, int *boneIdx*) [pure virtual]

10.15.1.10 virtual void ISkinContextData::SetWeight (int *vertexIdx*, INode * *bone*, float *weight*) [pure virtual]

10.15.1.11 virtual void ISkinContextData::SetWeight (int *vertexIdx*, int *boneIdx*, float *weight*) [pure virtual]

10.15.1.12 virtual void ISkinContextData::SetWeights (int *vertexIdx*, INodeTab *boneIdx*, Tab< float > *weights*) [pure virtual]

10.15.1.13 virtual void ISkinContextData::SetWeights (int *vertexIdx*, Tab< int > *boneIdx*, Tab< float > *weights*) [pure virtual]

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

- gui/pcggui/pcggui.h

10.16 model::Landmark Class Reference

`#include <Landmark.h>`

Public Member Functions

- Landmark ()
- ~Landmark ()
- Landmark (const Landmark &other)
- Landmark & operator= (const Landmark &other)
- bool operator== (const Landmark &other) const
- bool operator!= (const Landmark &other) const

10.16.1 Detailed Description

Write brief comment for Landmark here.

Write detailed description for Landmark here.

Remarks:
Write remarks for Landmark here.

See also:
Separate items with the '|' character.

Definition at line 25 of file Landmark.h.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 model::Landmark::Landmark () [inline]

Write brief comment for Landmark here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Landmark here.

Remarks:
Write remarks for Landmark here.

See also:
Separate items with the '|' character.

Definition at line 44 of file Landmark.h.

```
44 {};
```

10.16.2.2 model::Landmark::~Landmark () [inline]

Write brief comment for ~Landmark here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Landmark here.

Remarks:
Write remarks for ~Landmark here.

See also:
Separate items with the '|' character.

Definition at line 60 of file Landmark.h.

```
60 {};
```

10.16.2.3 Landmark::Landmark (const Landmark & *other*)

Write brief comment for Landmark here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Landmark here.

Remarks:
Write remarks for Landmark here.

See also:
Separate items with the '|' character.

Definition at line 23 of file Landmark.cpp.

```
23                                      {
24
25 }
```

10.16.3 Member Function Documentation

10.16.3.1 bool Landmark::operator!= (const Landmark & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 109 of file Landmark.cpp.

```
110 {
111         if( true ) {
112         return false;
113         }
114     return true;
115 }
```

10.16.3.2 Landmark & Landmark::operator= (const Landmark & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 48 of file Landmark.cpp.

```
49 {
50     // if same object
51     if ( this == &other )
52         return *this;
53
54
55     return *this;
56 }
```

10.16.3.3 bool Landmark::operator== (const Landmark & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 79 of file Landmark.cpp.

```
80 {
81
82     if( true ) {
83         return true;
84         }
85     return false;
86 }
```

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

- model/Landmark.h
- model/Landmark.cpp

10.17 log Class Reference

Write brief comment for log here.

`#include <log.h>`

Static Public Member Functions

- void write (string log)
- void write (string log, double data)
- void write (string log, int data)
- void write (string log, Matrix data)
- void write (string log, ExampleModel data)
- void write (string log, ReferenceModel data)
- void setDebug (bool debug)
- void clear ()

Static Public Attributes

- const char * m_filename = "PCG.log"
- bool m_debug = false

10.17.1 Detailed Description

Write brief comment for log here.

Write detailed description for log here.

Remarks:
Write remarks for log here.

See also:
Separate items with the '|' character.

Definition at line 47 of file log.h.

10.17.2 Member Function Documentation

10.17.2.1 void log::clear () [static]

Definition at line 141 of file log.cpp.

References m_filename.

Referenced by Batch::Batch(), and Batch::TestSystem().

```
141                 {
142         ofstream logfile;
143         logfile.open (m_filename, ios::out);
144         logfile << "";
145         logfile.close();
146 }
```

10.17.2.2 void log::setDebug (bool *debug*) [static]

Definition at line 148 of file log.cpp.

References m_debug.

Referenced by Batch::Batch().

```
148                                 {
149         m_debug = debug;
150 }
```

10.17.2.3 void log::write (string *log*, ReferenceModel *data*) [static]

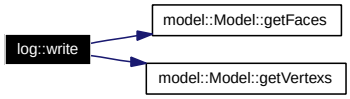
Definition at line 110 of file log.cpp.

References model::Model::getFaces(), model::Model::getVertexs(), and m_filename.

```
110                                                 {
111         ofstream logfile;
112         char timeStr [19];
113         #ifndef __WINDOWS__
114         _strtime( timeStr );
115         #else
116         _strtime_s( timeStr );
117         #endif
118         logfile.open (m_filename, ios::out | ios::app);
119         logfile << timeStr << " " << log << "\n";
120         logfile << "Vertices: " << data.getVertexs().size()  << "\n";
121         logfile << "Faces:    " << data.getFaces().size()  << "\n";
122         logfile.close();
123         if(m_debug) {
124                 ofstream modelfile;
125                 modelfile.open (log.append("-MODEL.log").c_str());
126                 modelfile << "Vertices: " << data.getVertexs().size() << "\n";
127                 modelfile << "faces:    " << data.getFaces().size() << "\n";
```

```
128                 for(int i=0;i<(int)data.getVertexs().size();i++) {
129                         modelfile       << "Index: " << data.getVertexs()[i]->getIndex() << "\n";
130                         modelfile       << "Normal: (" << data.getVertexs()[i]->getNormal()->x;
131                         modelfile       << "," << data.getVertexs()[i]->getNormal()->y;
132                         modelfile       << "," << data.getVertexs()[i]->getNormal()->z << ")\n";
133                         modelfile       << "Coordinate: (" << data.getVertexs()[i]->getCoordinates()->x
134                         modelfile       << "," << data.getVertexs()[i]->getCoordinates()->y;
135                         modelfile       << "," << data.getVertexs()[i]->getCoordinates()->z << ")\n";
136                 }
137                 modelfile.close();
138         }
139 }
```

Here is the call graph for this function:



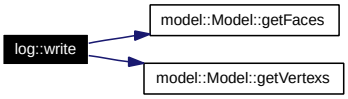
10.17.2.4 void log::write (string *log*, ExampleModel *data*) [static]

Definition at line 81 of file log.cpp.

References model::Model::getFaces(), model::Model::getVertexs(), and m_filename.

```
81                                                      {
82         ofstream logfile;
83         char timeStr [19];
84         #ifndef __WINDOWS__
85         _strtime( timeStr );
86         #else
87         _strtime_s( timeStr );
88         #endif
89         logfile.open (m_filename, ios::out | ios::app);
90         logfile << timeStr << " " << log << "\n";
91         logfile << "Vertices: " << data.getVertexs().size()  << "\n";
92         logfile << "Faces:    " << data.getFaces().size()  << "\n";
93         logfile.close();
94         if(m_debug) {
95                 ofstream modelfile;
96                 modelfile.open (log.append("-MODEL.log").c_str());
97                 for(int i=0;i<(int)data.getVertexs().size();i++) {
98                         modelfile       << "Index: " << data.getVertexs()[i]->getIndex() << "\n";
99                         modelfile       << "Normal: (" << data.getVertexs()[i]->getNormal()->x;
100                         modelfile       << "," << data.getVertexs()[i]->getNormal()->y;
101                         modelfile       << "," << data.getVertexs()[i]->getNormal()->z << ")\n";
102                         modelfile       << "Coordinate: (" << data.getVertexs()[i]->getCoordinates()->x
103                         modelfile       << "," << data.getVertexs()[i]->getCoordinates()->y;
104                         modelfile       << "," << data.getVertexs()[i]->getCoordinates()->z << ")\n";
105                 }
106                 modelfile.close();
107         }
108 }
```

Here is the call graph for this function:



10.17.2.5 void log::write (string *log*, Matrix *data*) [static]

Definition at line 61 of file log.cpp.

References m_filename.

```
61                                          {
62         ofstream logfile;
63         char timeStr [19];
64         #ifndef __WINDOWS__
65         _strtime( timeStr );
66         #else
67         _strtime_s( timeStr );
68         #endif
69         logfile.open (m_filename, ios::out | ios::app);
70         logfile << timeStr << " " << log << "\n";
71         logfile << "Dim: [" << data.Nrows() << "," << data.Ncols() << "]\n";
72         logfile.close();
73         if(m_debug) {
74                 ofstream matrixfile;
75                 matrixfile.open (log.append("-MATRIX.log").c_str());
76                 matrixfile << setw(10) << setprecision(5) << data;
77                 matrixfile.close();
78         }
79 }
```

10.17.2.6 void log::write (string *log*, int *data*) [static]

Definition at line 48 of file log.cpp.

References m_filename.

```
48                                        {
49         ofstream logfile;
50         char timeStr [19];
51         #ifndef __WINDOWS__
52         _strtime( timeStr );
53         #else
54         _strtime_s( timeStr );
55         #endif
56         logfile.open (m_filename, ios::out | ios::app);
57         logfile << timeStr << " " << log << ": " << data << "\n";
58         logfile.close();
59 }
```

10.17.2.7 void log::write (string *log*, double *data*) [static]

Definition at line 35 of file log.cpp.

References m_filename.

```
35                                      {
36         ofstream logfile;
37         char timeStr [19];
38         #ifndef __WINDOWS__
39         _strtime( timeStr );
40         #else
41         _strtime_s( timeStr );
42         #endif
43         logfile.open (m_filename, ios::out | ios::app);
44         logfile << timeStr << " " << log << ": " << data << "\n";
45         logfile.close();
46 }
```

10.17.2.8 void log::write (string *log*) [static]

Definition at line 21 of file log.cpp.

References m_filename.

Referenced by model::Analyzer::adjust(), Batch::analyzeModel(), model::Analyzer::Analyzer(), Batch::Batch(), model::Analyzer::computeEnergy(), model::Analyzer::covariance(), functions::Exporter::exportModel(), Batch::ExportModels(), model::Analyzer::findComponents(), model::Analyzer::getModel(), Batch::getRandomModels(), model::Analyzer::mean(), model::Analyzer::mergeData(), model::Analyzer::selectComponents(), Batch::TestAnalyzer(), Batch::TestSystem(), and model::Analyzer::transformation().

```
21                             {
22         ofstream logfile;
23         char timeStr [19];
24         #ifndef __WINDOWS__
25         _strtime( timeStr );
26         #else
27         _strtime_s( timeStr );
28         #endif
29
30         logfile.open (m_filename, ios::out | ios::app);
31         logfile << timeStr << " " << log << "\n";
32         logfile.close();
33 }
```

10.17.3 Member Data Documentation

10.17.3.1 bool log::m_debug = false [static]

Definition at line 153 of file log.cpp.

Referenced by setDebug().

10.17.3.2 const char * log::m_filename = "PCG.log" [static]

Definition at line 152 of file log.cpp.

Referenced by clear(), and write().

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

- Functions/log.h
- Functions/log.cpp

10.18 model::Mapping Class Reference

`#include <Mapping.h>`

Public Member Functions

- Mapping ()
- ~Mapping ()
- Mapping (int id, float weight)
- Mapping (const Mapping &other)
- Mapping & operator= (const Mapping &other)
- bool operator== (const Mapping &other) const
- bool operator!= (const Mapping &other) const

Private Attributes

- int m_iId
- float m_fWeight

10.18.1 Constructor & Destructor Documentation

10.18.1.1 model::Mapping::Mapping () [inline]

Write brief comment for Mapping here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for Mapping here.

Remarks:
Write remarks for Mapping here.

See also:
Separate items with the '|' character.

Definition at line 34 of file Mapping.h.

```
34 {};
```

10.18.1.2 model::Mapping::~Mapping () [inline]

Write brief comment for ~Mapping here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for ~Mapping here.

Remarks:
Write remarks for ~Mapping here.

See also:
Separate items with the '|' character.

Definition at line 50 of file Mapping.h.

```
50 {};
```

10.18.1.3 Mapping::Mapping (int *id*, float *weight*)

Definition at line 5 of file Mapping.cpp.

References m_fWeight, and m_iId.

```
5                                       {
6         m_iId = id;
7         m_fWeight = weight;
8 }
```

10.18.1.4 Mapping::Mapping (const Mapping & *other*)

Write brief comment for Mapping here.

Parameters:
other Description of parameter other.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for Mapping here.

Remarks:
Write remarks for Mapping here.

See also:
Separate items with the '|' character.

Definition at line 27 of file Mapping.cpp.

```
27                                      {
28
29 }
```

10.18.2 Member Function Documentation

10.18.2.1 bool Mapping::operator!= (const Mapping & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 113 of file Mapping.cpp.

```
114 {
115         if( true ) {
116         return false;
117         }
118     return true;
119 }
```

10.18.2.2 Mapping & Mapping::operator= (const Mapping & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 52 of file Mapping.cpp.

```
53 {
54     // if same object
55     if ( this == &other )
56         return *this;
57
58
59     return *this;
60 }
```

10.18.2.3 bool Mapping::operator== (const Mapping & *other*) const

Write brief comment for operator == here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:
Write remarks for operator == here.

See also:
Separate items with the '|' character.

Definition at line 83 of file Mapping.cpp.

```
84 {
85
86     if( true ) {
87         return true;
88         }
89     return false;
90 }
```

10.18.3 Member Data Documentation

10.18.3.1 float model::Mapping::m_fWeight [private]

Definition at line 16 of file Mapping.h.

Referenced by Mapping().

10.18.3.2 int model::Mapping::m_iId [private]

Definition at line 15 of file Mapping.h.

Referenced by Mapping().

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

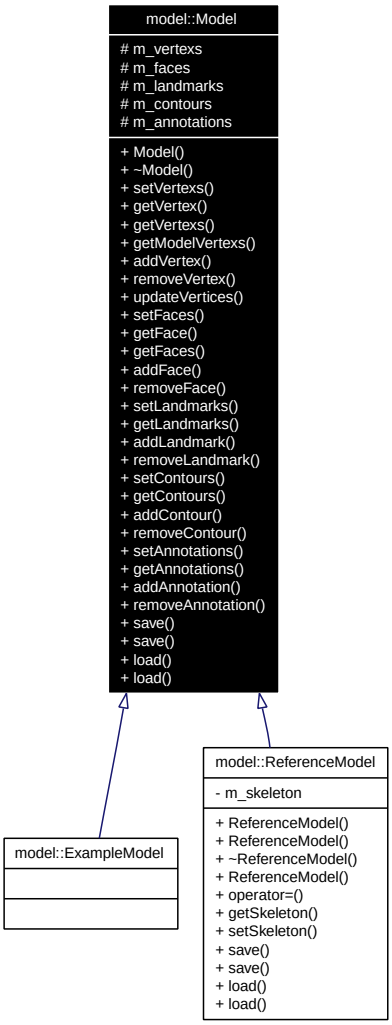
- model/Mapping.h
- model/Mapping.cpp

10.19 model::Model Class Reference

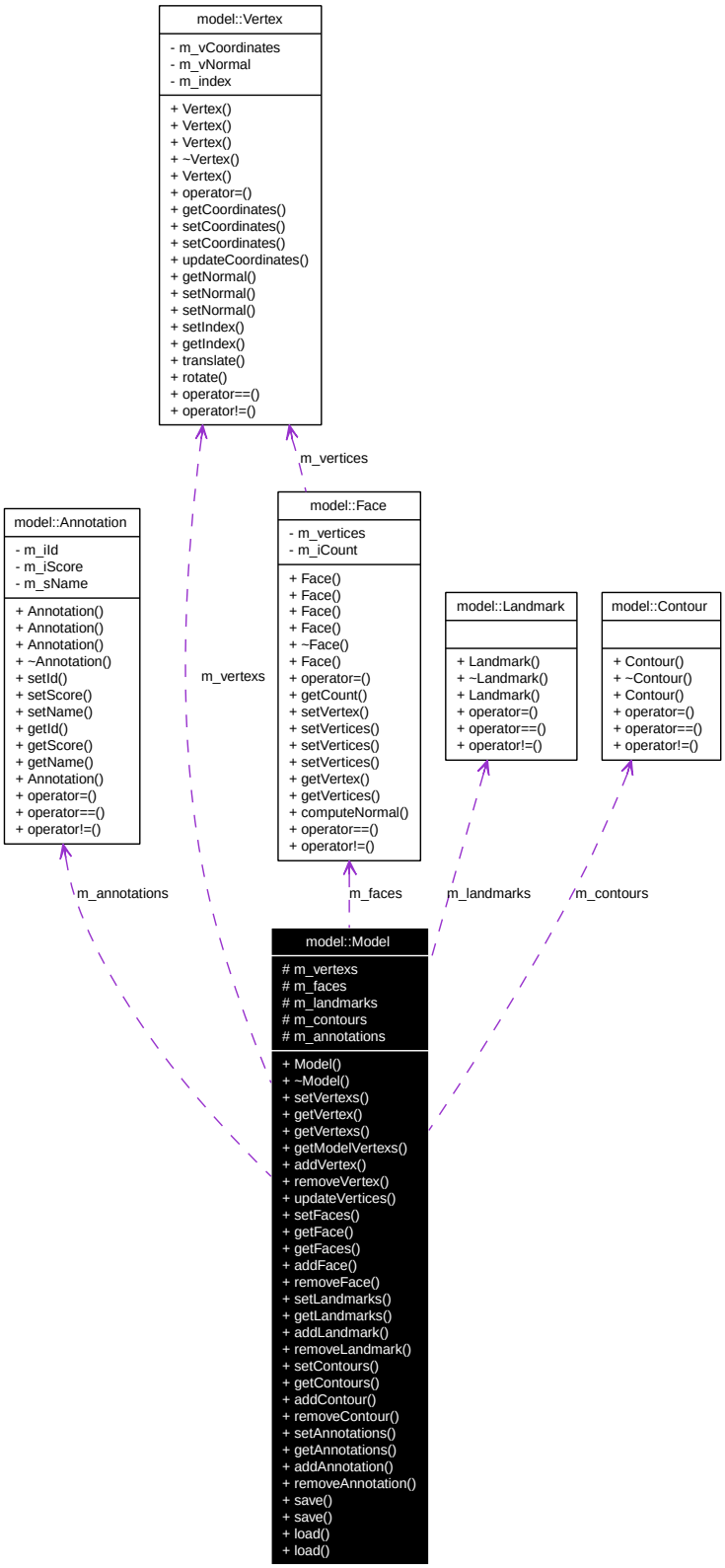
A abstract class of a Model in PCG.

`#include <Model.h>`

Inheritance diagram for model::Model:



Collaboration diagram for model::Model:


model::Vertex
- m_vCoordinates
- m_vNormal
- m_index
+ Vertex()
+ Vertex()
+ Vertex()
+ ~Vertex()
+ Vertex()
+ operator=()
+ getCoordinates()
+ setCoordinates()
+ setCoordinates()
+ updateCoordinates()
+ getNormal()
+ setNormal()
+ setNormal()
+ setIndex()
+ getIndex()
+ translate()
+ rotate()
+ operator==()
+ operator!=()
m_vertices
model::Face
- m_vertices
- m_iCount
+ Face()
+ Face()
+ Face()
+ Face()
+ ~Face()
+ Face()
+ operator=()
+ getCount()
+ setVertex()
+ setVertices()
+ setVertices()
+ setVertices()
+ getVertex()
+ getVertices()
+ computeNormal()
+ operator==()
+ operator!=()
model::Annotation
- m_iId
- m_iScore
- m_sName
+ Annotation()
+ Annotation()
+ Annotation()
+ ~Annotation()
+ setId()
+ setScore()
+ setName()
+ getId()
+ getScore()
+ getName()
+ Annotation()
+ operator=()
+ operator==()
+ operator!=()
m_vertexs
model::Landmark
+ Landmark()
+ ~Landmark()
+ Landmark()
+ operator=()
+ operator==()
+ operator!=()
model::Contour
+ Contour()
+ ~Contour()
+ Contour()
+ operator=()
+ operator==()
+ operator!=()
m_annotations
m_faces
m_landmarks
m_contours
model::Model
m_vertexs
m_faces
m_landmarks
m_contours
m_annotations
+ Model()
+ ~Model()
+ setVertexs()
+ getVertex()
+ getVertexs()
+ getModelVertexs()
+ addVertex()
+ removeVertex()
+ updateVertices()
+ setFaces()
+ getFace()
+ getFaces()
+ addFace()
+ removeFace()
+ setLandmarks()
+ getLandmarks()
+ addLandmark()
+ removeLandmark()
+ setContours()
+ getContours()
+ addContour()
+ removeContour()
+ setAnnotations()
+ getAnnotations()
+ addAnnotation()
+ removeAnnotation()
+ save()
+ save()
+ load()
+ load()

Public Member Functions

- Model ()
- virtual ~Model ()=0
- void setVertexs (vector< Vertex * > vertexs)
- Vertex * getVertex (int i)
- vector< Vertex * > getVertexs ()
- Matrix getModelVertexs ()
- void addVertex (Vertex *vertex)
- void removeVertex (Vertex *vertex)
- void updateVertices (Matrix vertexs)
- void setFaces (vector< Face * > faces)
- Face * getFace (int i)
- vector< Face * > getFaces ()
- void addFace (Face *face)
- void removeFace (Face *face)
- void setLandmarks (vector< Landmark * > landmarks)
- vector< Landmark * > getLandmarks ()
- void addLandmark (Landmark *landmark)
- void removeLandmark (Landmark *landmark)
- void setContours (vector< Contour * > contours)
- vector< Contour * > getContours ()
- void addContour (Contour *contour)
- void removeContour (Contour *contour)
- void setAnnotations (vector< Annotation * > annotations)
- vector< Annotation * > getAnnotations ()
- void addAnnotation (Annotation *annotation)
- void removeAnnotation (Annotation *annotation)
- bool save (char *filename)
- bool save (ofstream *saveFile)
- bool load (char *filename)
- bool load (ifstream *loadFile)

Protected Attributes

- Vertex * m_vertexs
- Face * m_faces
- Landmark * m_landmarks
- Contour * m_contours
- Annotation * m_annotations

10.19.1 Detailed Description

A abstract class of a Model in PCG.

It contains Vertex's, Face's, Landmark's, Contour's, and Annotation's.

See also:
Vertex | Face | Landmark | Contour | Annotation

Definition at line 35 of file Model.h.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 model::Model::Model () [inline]

Write brief comment for Model here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Model here.

Remarks:
Write remarks for Model here.

See also:
Separate items with the '|' character.

Definition at line 66 of file Model.h.

```
66 {};
```

10.19.2.2 Model::~Model () [pure virtual]

Write brief comment for ~Model here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Model here.

Remarks:
Write remarks for ~Model here.

See also:
Separate items with the '|' character.

Definition at line 20 of file Model.cpp.

```
20                {
21
22 }
```

10.19.3 Member Function Documentation

10.19.3.1 void Model::addAnnotation (Annotation * *annotation*)

Write brief comment for addAnnotation here.

Parameters:
annotation Description of parameter annotation.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addAnnotation here.

Remarks:
Write remarks for addAnnotation here.

See also:
Separate items with the '|' character.

Definition at line 548 of file Model.cpp.

References m_annotations.

```
548                                                 {
549         m_annotations.push_back(annotation);
550 }
```

10.19.3.2 void Model::addContour (Contour * *contour*)

Write brief comment for addContour here.

Parameters:
contour Description of parameter contour.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addContour here.

Remarks:
Write remarks for addContour here.

See also:
Separate items with the '|' character.

Definition at line 460 of file Model.cpp.

References m_contours.

```
460                                         {
461         m_contours.push_back(contour);
462 }
```

10.19.3.3 void Model::addFace (Face * *face*)

Write brief comment for addFace here.

Parameters:
face Description of parameter face.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addFace here.

Remarks:
Write remarks for addFace here.

See also:
Separate items with the '|' character.

Definition at line 284 of file Model.cpp.

References m_faces.

Referenced by pcggui::CovertScene(), load(), and functions::Importer::readAttributes().

```
284                                 {
285         m_faces.push_back(face);
286 }
```

10.19.3.4 void Model::addLandmark (Landmark * *landmark*)

Write brief comment for addLandmark here.

Parameters:
landmark Description of parameter landmark.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addLandmark here.

Remarks:
Write remarks for addLandmark here.

See also:
Separate items with the '|' character.

Definition at line 372 of file Model.cpp.

References m_landmarks.

```
372                                             {
373         m_landmarks.push_back(landmark);
374 }
```

10.19.3.5 void Model::addVertex (Vertex * *vertex*)

Write brief comment for addVertex here.

Parameters:
vertex Description of parameter vertex.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addVertex here.

Remarks:
Write remarks for addVertex here.

See also:
Separate items with the '|' character.

Definition at line 141 of file Model.cpp.

References m_vertexs, and model::Vertex::setIndex().

Referenced by pcggui::CovertScene(), load(), and functions::Importer::readAttributes().

```
141                                     {
142         vertex->setIndex(m_vertexs.size());
143         m_vertexs.push_back(vertex);
144 }
```

Here is the call graph for this function:

10.19.3.6 vector< Annotation * > Model::getAnnotations ()

Write brief comment for getAnnotations here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getAnnotations here.

Remarks:
Write remarks for getAnnotations here.

See also:
Separate items with the '|' character.

Definition at line 526 of file Model.cpp.

```
526                                             {
527         return m_annotations;
528 }
```

10.19.3.7 vector< Contour * > Model::getContours ()

Write brief comment for getContours here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getContours here.

Remarks:
Write remarks for getContours here.

See also:
Separate items with the '|' character.

Definition at line 438 of file Model.cpp.

```
438                                     {
439         return m_contours;
440 }
```

10.19.3.8 Face * Model::getFace (int *i*)

Write brief comment for getFace here.

Parameters:
i Description of parameter i.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getFace here.

Remarks:
Write remarks for getFace here.

See also:
Separate items with the '|' character.

Definition at line 240 of file Model.cpp.

References m_faces.

Referenced by functions::Importer::readAttributes(), and pcggui::Render().

```
240                             {
241         return m_faces[i];
242 }
```

10.19.3.9 vector< Face * > Model::getFaces ()

Write brief comment for getFaces here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getFaces here.

Remarks:
Write remarks for getFaces here.

See also:
Separate items with the '|' character.

Definition at line 262 of file Model.cpp.

Referenced by functions::Exporter::exportModel(), pcggui::Render(), and log::write().

```
262                                 {
263         return m_faces;
264 }
```

10.19.3.10 vector< Landmark * > Model::getLandmarks ()

Write brief comment for getLandmarks here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getLandmarks here.

Remarks:
Write remarks for getLandmarks here.

See also:
Separate items with the '|' character.

Definition at line 350 of file Model.cpp.

```
350                                         {
351         return m_landmarks;
352 }
```

10.19.3.11 Matrix Model::getModelVertexs ()

Write brief comment for getModelVertexs here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getModelVertexs here.

Remarks:
Write remarks for getModelVertexs here.

See also:
Separate items with the '|' character.

Definition at line 111 of file Model.cpp.

References model::Vertex::getCoordinates(), and m_vertexs.

Referenced by model::Analyzer::mergeData().

```
111                                {
112         int size = m_vertexs.size();
113         Matrix v(size*3,1);
114         for(int i=0; i<(int)m_vertexs.size(); i++) {
115                 IvVector3* tmp = m_vertexs[i]->getCoordinates();
116                 v((i*3)+1,1) = tmp->x;
117                 v((i*3)+2,1) = tmp->y;
118                 v((i*3)+3,1) = tmp->z;
119         }
120         return v;
121 }
```

Here is the call graph for this function:

10.19.3.12 Vertex * Model::getVertex (int *i*)

Write brief comment for getVertex here.

Parameters:
i Description of parameter i.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getVertex here.

Remarks:
Write remarks for getVertex here.

See also:
Separate items with the '|' character.

Definition at line 67 of file Model.cpp.

References m_vertexs.

Referenced by pcggui::CovertScene(), load(), functions::Importer::readAttributes(), and pcggui::Render().

```
67                                {
68         return m_vertexs[i];
69 }
```

10.19.3.13 vector< Vertex * > Model::getVertexs ()

Write brief comment for getVertexs here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getVertexs here.

Remarks:
Write remarks for getVertexs here.

See also:
Separate items with the '|' character.

Definition at line 89 of file Model.cpp.

Referenced by functions::Exporter::exportModel(), pcggui::Render(), and log::write().

```
89                                    {
90         return m_vertexs;
91 }
```

10.19.3.14 bool Model::load (ifstream * *loadFile*)

Write brief comment for load here.

Parameters:
loadFile Description of parameter loadFile.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for load here.

Remarks:
Write remarks for load here.

See also:
Separate items with the '|' character.

Reimplemented in model::ReferenceModel.

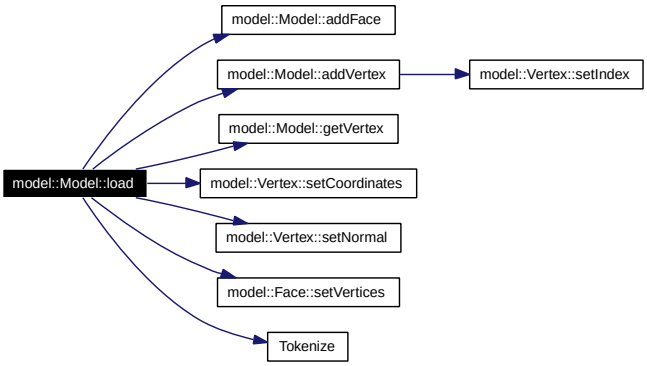
Definition at line 711 of file Model.cpp.

References addFace(), addVertex(), getVertex(), m_annotations, m_contours, m_faces, m_landmarks, m_vertexs, model::Vertex::setCoordinates(), model::Vertex::setNormal(), model::Face::setVertices(), and Tokenize().

```
711                                     {
712         m_vertexs.clear();
713         m_faces.clear();
714         m_landmarks.clear();
715         m_contours.clear();
716         m_annotations.clear();
717
718
719         string line;
720         vector<string> tokens;
721         vector<string> coords;
722         if(loadFile->is_open()) {
723                 getline(*loadFile,line);//Vertex
724                 getline(*loadFile,line);//Coords
725                 tokens.clear();
726                 Tokenize(line,tokens,",");
727                 for(int i=0;i<(int)tokens.size();i++) {
728                         coords.clear();
729                         Tokenize(tokens[i],coords," ");
730                         Vertex* vertex = new Vertex();
731                         vertex->setCoordinates(convertTo<float>(coords[0]),convertTo<float>(
732                         addVertex(vertex);
733                 }
734                 getline(*loadFile,line);//Normals
735                 tokens.clear();
736                 Tokenize(line,tokens,",");
```

```
737                 for(int i=0;i<(int)tokens.size();i++) {
738                         coords.clear();
739                         Tokenize(tokens[i],coords," ");
740                         Vertex* vertex = getVertex(i);
741                         vertex->setNormal(convertTo<float>(coords[0]),convertTo<float>(coords[1]),conve
742                 }
743                 getline(*loadFile,line);//Face
744                 getline(*loadFile,line);//Indexs
745                 tokens.clear();
746                 Tokenize(line,tokens,",");
747                 for(int i=0;i<(int)tokens.size();i++) {
748                         coords.clear();
749                         Tokenize(tokens[i],coords," ");
750                         Face* face = new Face();
751                         face->setVertices(m_vertexs[convertTo<int>(coords[0])],m_vertexs[convertTo<int>
752                         addFace(face);
753                 }
754                 getline(*loadFile,line);//Landmark
755                 getline(*loadFile,line);//Data
756                 tokens.clear();
757                 Tokenize(line,tokens,",");
758                 for(int i=0;i<(int)tokens.size();i++) {
759                         coords.clear();
760                         Tokenize(tokens[i],coords," ");
761                         //addLandmark
762                 }
763                 getline(*loadFile,line);//Contour
764                 getline(*loadFile,line);//Data
765                 tokens.clear();
766                 Tokenize(line,tokens,",");
767                 for(int i=0;i<(int)tokens.size();i++) {
768                         coords.clear();
769                         Tokenize(tokens[i],coords," ");
770                 }
771                 getline(*loadFile,line);//Annotation
772                 getline(*loadFile,line);//Data
773                 tokens.clear();
774                 Tokenize(line,tokens,",");
775                 for(int i=0;i<(int)tokens.size();i++) {
776                         coords.clear();
777                         Tokenize(tokens[i],coords," ");
778                 }
779                 return true;
780         } else {
781                 return false;
782         }
783 }
```

Here is the call graph for this function:



10.19.3.15 bool Model::load (char * *filename*)

Write brief comment for load here.

Parameters:
filename Description of parameter filename.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for load here.

Remarks:
Write remarks for load here.

See also:
Separate items with the '|' character.

Reimplemented in model::ReferenceModel.

Definition at line 683 of file Model.cpp.

Referenced by model::ReferenceModel::load(), and pcggui::LoadAll().

```
683                                {
684         ifstream loadFile(filename);
685         bool res = load(&loadFile);
686         loadFile.close();
687         return res;
688 }
```

10.19.3.16 void Model::removeAnnotation (Annotation * *annotation*)

Write brief comment for removeAnnotation here.

Parameters:
annotation Description of parameter annotation.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeAnnotation here.

Remarks:
Write remarks for removeAnnotation here.

See also:
Separate items with the '|' character.

Definition at line 570 of file Model.cpp.

```
570                                                 {
571         //m_annotations.remove(annotation);
572 }
```

10.19.3.17 void Model::removeContour (Contour * *contour*)

Write brief comment for removeContour here.

Parameters:
contour Description of parameter contour.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeContour here.

Remarks:
Write remarks for removeContour here.

See also:
Separate items with the '|' character.

Definition at line 482 of file Model.cpp.

```
482                                         {
483         //m_contours.remove(contour);
484 }
```

10.19.3.18 void Model::removeFace (Face * *face*)

Write brief comment for removeFace here.

Parameters:
face Description of parameter face.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeFace here.

Remarks:
Write remarks for removeFace here.

See also:
Separate items with the '|' character.

Definition at line 306 of file Model.cpp.

```
306                                 {
307         //m_faces.remove(face);
308 }
```

10.19.3.19 void Model::removeLandmark (Landmark * *landmark*)

Write brief comment for removeLandmark here.

Parameters:
landmark Description of parameter landmark.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeLandmark here.

Remarks:
Write remarks for removeLandmark here.

See also:
Separate items with the '|' character.

Definition at line 394 of file Model.cpp.

```
394                                             {
395         //m_landmarks.remove(landmark);
396 }
```

10.19.3.20 void Model::removeVertex (Vertex * *vertex*)

Write brief comment for removeVertex here.

Parameters:
vertex Description of parameter vertex.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeVertex here.

Remarks:
Write remarks for removeVertex here.

See also:
Separate items with the '|' character.

Definition at line 164 of file Model.cpp.

```
164                                     {
165         //m_vertexs.erase(vertex);
166 }
```

10.19.3.21 bool Model::save (ofstream * *saveFile*)

Write brief comment for save here.

Parameters:
saveFile Description of parameter saveFile.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for save here.

Remarks:
Write remarks for save here.

See also:
Separate items with the '|' character.

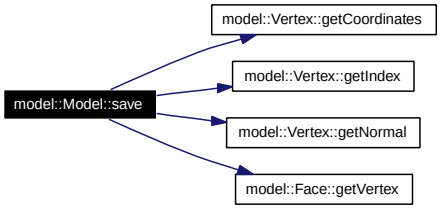
Reimplemented in model::ReferenceModel.

Definition at line 623 of file Model.cpp.

References model::Vertex::getCoordinates(), model::Vertex::getIndex(), model::Vertex::getNormal(), model::Face::getVertex(), m_annotations, m_contours, m_faces, m_landmarks, and m_vertexs.

```
623                                     {
624         if(saveFile->is_open()) {
625                 *saveFile << "Vertex\n";
626                 for(int i=0;i<(int)m_vertexs.size();i++) {
627                         *saveFile << m_vertexs[i]->getCoordinates()->x << " ";
628                         *saveFile << m_vertexs[i]->getCoordinates()->y << " ";
629                         *saveFile << m_vertexs[i]->getCoordinates()->z << ",";
630                 }
631                 *saveFile << "\n";
632                 for(int i=0;i<(int)m_vertexs.size();i++) {
633                         *saveFile << m_vertexs[i]->getNormal()->x << " ";
634                         *saveFile << m_vertexs[i]->getNormal()->y << " ";
635                         *saveFile << m_vertexs[i]->getNormal()->z << ",";
636                 }
637                 *saveFile << "\nFace\n";
638                 for(int i=0;i<(int)m_faces.size();i++) {
639                         *saveFile << m_faces[i]->getVertex(0)->getIndex() << " ";
640                         *saveFile << m_faces[i]->getVertex(1)->getIndex() << " ";
641                         *saveFile << m_faces[i]->getVertex(2)->getIndex() << ",";
642                 }
643                 *saveFile << "\nLandmark\n";
644                 for(int i=0;i<(int)m_landmarks.size();i++) {
645                         //saveFile << m_landmarks[i] << " ";
646                 }
647                 *saveFile << "\nContour\n";
648                 for(int i=0;i<(int)m_contours.size();i++) {
649                         //saveFile << m_contours[i] << " ";
650                 }
651                 *saveFile << "\nAnnotation\n";
652                 for(int i=0;i<(int)m_annotations.size();i++) {
653                         //saveFile << m_annotations[i] << " ";
654                 }
655                 *saveFile << "\n";
656                 return true;
657         } else {
658                 return false;
659         }
660 }
```

Here is the call graph for this function:



10.19.3.22 bool Model::save (char * *filename*)

Write brief comment for save here.

Parameters:
filename Description of parameter filename.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for save here.

Remarks:
Write remarks for save here.

See also:
Separate items with the '|' character.

Reimplemented in model::ReferenceModel.

Definition at line 595 of file Model.cpp.

Referenced by model::ReferenceModel::save().

```
595                                {
596         ofstream saveFile(filename);
597         bool res = save(&saveFile);
598         saveFile.close();
599         return res;
600 }
```

10.19.3.23 void Model::setAnnotations (vector< Annotation * > *annotations*)

Write brief comment for setAnnotations here.

Parameters:
annotations Description of parameter annotations.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setAnnotations here.

Remarks:
Write remarks for setAnnotations here.

See also:
Separate items with the '|' character.

Definition at line 504 of file Model.cpp.

References m_annotations.

```
504                                                         {
505         m_annotations = annotations;
506 }
```

10.19.3.24 void Model::setContours (vector< Contour * > *contours*)

Write brief comment for setContours here.

Parameters:
contours Description of parameter contours.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setContours here.

Remarks:
Write remarks for setContours here.

See also:
Separate items with the '|' character.

Definition at line 416 of file Model.cpp.

References m_contours.

```
416                                                 {
417         m_contours = contours;
418 }
```

10.19.3.25 void Model::setFaces (vector< Face * > *faces*)

Write brief comment for setFaces here.

Parameters:
faces Description of parameter faces.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setFaces here.

Remarks:
Write remarks for setFaces here.

See also:
Separate items with the '|' character.

Definition at line 215 of file Model.cpp.

References m_faces.

Referenced by Batch::analyzeModel(), and Batch::TestAnalyzer().

```
215                                             {
216         m_faces = faces;
217 }
```

10.19.3.26 void Model::setLandmarks (vector< Landmark * > *landmarks*)

Write brief comment for setLandmarks here.

Parameters:
landmarks Description of parameter landmarks.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setLandmarks here.

Remarks:
Write remarks for setLandmarks here.

See also:
Separate items with the '|' character.

Definition at line 328 of file Model.cpp.

References m_landmarks.

```
328                                                     {
329         m_landmarks = landmarks;
330 }
```

10.19.3.27 void Model::setVertexs (vector< Vertex * > *vertexs*)

Write brief comment for setVertexs here.

Parameters:
vertexs Description of parameter vertexs.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setVertexs here.

Remarks:
Write remarks for setVertexs here.

See also:
Separate items with the '|' character.

Definition at line 42 of file Model.cpp.

References m_vertexs.

Referenced by Batch::analyzeModel(), and Batch::TestAnalyzer().

```
42                                                  {
43         m_vertexs = vertexs;
44 }
```

10.19.3.28 void Model::updateVertices (Matrix *vertexs*)

Write brief comment for updateVertices here.

Parameters:
vertexs Description of parameter vertexs.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for updateVertices here.

Remarks:
Write remarks for updateVertices here.

See also:
Separate items with the '|' character.

Definition at line 186 of file Model.cpp.

References m_vertexs, and model::Vertex::updateCoordinates().

Referenced by model::Analyzer::getModel().

```
186                                         {
187         printf("Model::updateVertices\n");
188         for(int i=0; i<(int)m_vertexs.size();i++) {
189                 float x = (float)vertexs(i*3+1,1);
190                 float y = (float)vertexs(i*3+2,1);
191                 float z = (float)vertexs(i*3+3,1);
192                 m_vertexs[i]->updateCoordinates(x,y,z);
193         }
194 }
```

Here is the call graph for this function:

10.19.4 Member Data Documentation

10.19.4.1 Annotation* model::Model::m_annotations [protected]

A Collection of Annotation's

Definition at line 42 of file Model.h.

Referenced by addAnnotation(), load(), model::ReferenceModel::operator=(), model::ReferenceModel::ReferenceModel(), save(), and setAnnotations().

10.19.4.2 Contour* model::Model::m_contours [protected]

A Collection of Contour's

Definition at line 41 of file Model.h.

Referenced by addContour(), load(), model::ReferenceModel::operator=(), model::ReferenceModel::ReferenceModel(), save(), and setContours().

10.19.4.3 Face* model::Model::m_faces [protected]

A Collection of Face's

Definition at line 39 of file Model.h.

Referenced by addFace(), getFace(), load(), model::ReferenceModel::operator=(), model::ReferenceModel::ReferenceModel(), save(), and setFaces().

10.19.4.4 Landmark* model::Model::m_landmarks [protected]

A Collection of Landmark's

Definition at line 40 of file Model.h.

Referenced by addLandmark(), load(), model::ReferenceModel::operator=(), model::ReferenceModel::ReferenceModel(), save(), and setLandmarks().

10.19.4.5 Vertex* model::Model::m_vertexs [protected]

A Collection of Vertex's

Definition at line 38 of file Model.h.

Referenced by addVertex(), getModelVertexs(), getVertex(), load(), model::ReferenceModel::operator=(), model::ReferenceModel::ReferenceModel(), save(), setVertexs(), and updateVertices().

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

- model/Model.h
- model/Model.cpp

10.20 NullView Class Reference

Public Member Functions

- Point2 ViewToScreen (Point3 p)
- NullView ()

10.20.1 Constructor & Destructor Documentation

10.20.1.1 NullView::NullView () [inline]

Definition at line 125 of file pcggui.cpp.

```
125 { worldToView.IdentityMatrix(); screenW=640.0f; screenH = 480.0f; }
```

10.20.2 Member Function Documentation

10.20.2.1 Point2 NullView::ViewToScreen (Point3 *p*) [inline]

Definition at line 124 of file pcggui.cpp.

```
124 { return Point2(p.x,p.y); }
```

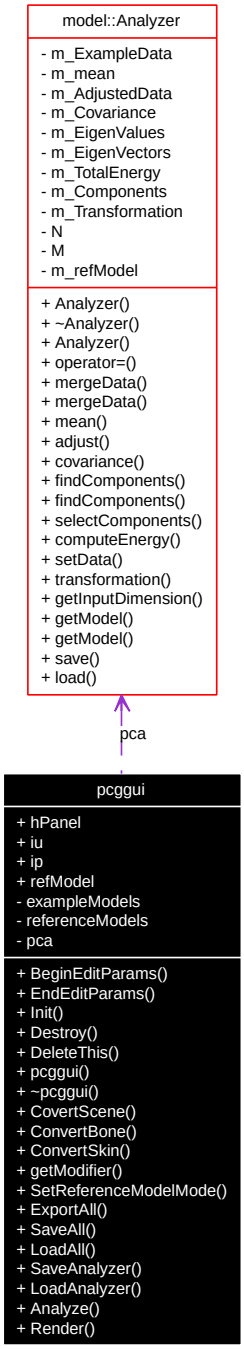
The documentation for this class was generated from the following file:

- gui/pcggui/pcggui.cpp

10.21 pcggui Class Reference

`#include <pcggui.h>`

Collaboration diagram for pcggui:



Public Types

- enum Skin_Status { NoError, InvalidSkeleton, VertexWithoutWeight }

Public Member Functions

- void BeginEditParams (Interface *ip, IUtil *iu)
- void EndEditParams (Interface *ip, IUtil *iu)
- void Init (HWND hWnd)
- void Destroy (HWND hWnd)
- void DeleteThis ()
- pcggui ()
- ~pcggui ()
- void CovertScene ()
- void ConvertBone (INode *pParent, model::ReferenceModel *rm)
- uint ConvertSkin (INode *node, model::ReferenceModel *rm)
- Modifier * getModifier (INode *pNode, Class_ID modCID)
- void SetReferenceModelMode (BOOL rm)
- void ExportAll ()
- void SaveAll ()
- void LoadAll ()
- void SaveAnalyzer ()
- void LoadAnalyzer ()
- void Analyze ()
- void Render ()

Public Attributes

- HWND hPanel
- IUtil * iu
- Interface * ip
- BOOL refModel

Private Attributes

- vector< model::ExampleModel * > exampleModels
- vector< model::ReferenceModel * > referenceModels
- model::Analyzer pca

10.21.1 Member Enumeration Documentation

10.21.1.1 enum pcggui::Skin_Status

Enumeration values:

NoError

InvalidSkeleton

VertexWithoutWeight

Definition at line 54 of file pcggui.h.

```
54                          {
55                  NoError,
56                  InvalidSkeleton,
57                  VertexWithoutWeight,
58          };
```

10.21.2 Constructor & Destructor Documentation

10.21.2.1 pcggui::pcggui ()

Definition at line 130 of file pcggui.cpp.

References hPanel, ip, and iu.

```
131 {
132         iu = NULL;
133         ip = NULL;
134         hPanel = NULL;
135 }
```

10.21.2.2 pcggui::~pcggui ()

Definition at line 137 of file pcggui.cpp.

```
138 {
139
140 }
```

10.21.3 Member Function Documentation

10.21.3.1 void pcggui::Analyze ()

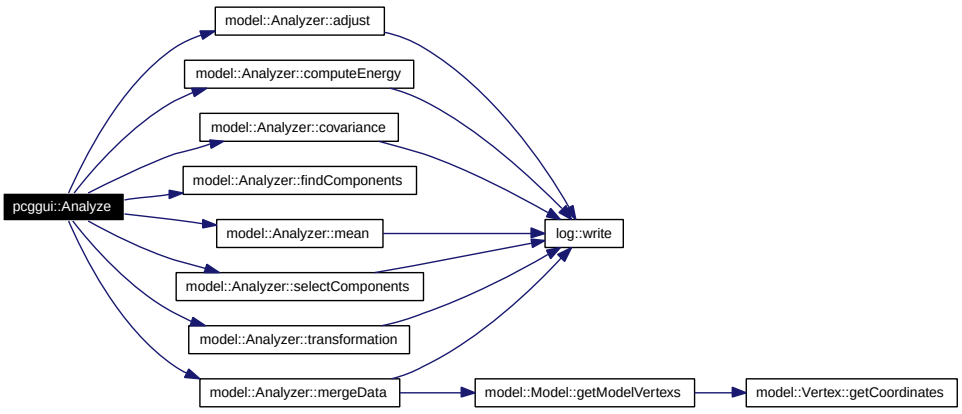
Definition at line 354 of file pcggui.cpp.

References model::Analyzer::adjust(), model::Analyzer::computeEnergy(), model::Analyzer::covariance(), exampleModels, model::Analyzer::findComponents(), model::Analyzer::mean(), model::Analyzer::mergeData(), pca, referenceModels, model::Analyzer::selectComponents(), and model::Analyzer::transformation().

Referenced by pcgguiDlgProc().

```
354                     {
355         DebugPrint("*************Starting to Analyze************\n");
356         pca.mergeData(*referenceModels[0],exampleModels);
357         DebugPrint("Calculating mean\n");
358         pca.mean();
359         DebugPrint("Adjusting data\n");
360         pca.adjust();
361         DebugPrint("Finding Covariance\n");
362         pca.covariance();
363         DebugPrint("Begin EigenVector calculations\n");
364         DebugPrint("Timer Started\n");
365         clock_t begin = clock();
366         pca.findComponents(model::Analyzer::HOUSEHOLDER);
367         clock_t end = clock();
368         DebugPrint("Timer Stopped\n");
369         DebugPrint("Time elapsed in seconds: %d\n", difftime(end, begin));
370         DebugPrint("End EigenVector calculations\n");
371         pca.computeEnergy();
372         pca.selectComponents();
373         pca.transformation();
374 }
```

Here is the call graph for this function:



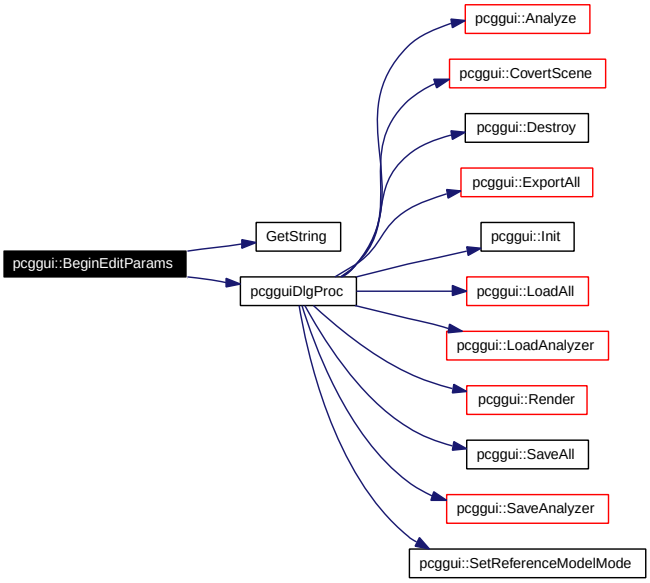
10.21.3.2 void pcggui::BeginEditParams (Interface * *ip*, IUtil * *iu*)

Definition at line 142 of file pcggui.cpp.

References file, GetString(), hInstance, hPanel, IDD_PANEL, IDS_PARAMS, and pcgguiDlgProc().

```
143 {
144         fopen_s(&file,"c:\\testdebug.txt","w");
145         fprintf(file,"");
146         fclose(file);
147         fopen_s(&file,"c:\\testdebug.txt","a");
148         fprintf(file,"\nFile Opened\n");
149         this->iu = iu;
150         this->ip = ip;
151         hPanel = ip->AddRollupPage(
152                 hInstance,
153                 MAKEINTRESOURCE(IDD_PANEL),
154                 pcgguiDlgProc,
155                 GetString(IDS_PARAMS),
156                 0);
157 }
```

Here is the call graph for this function:



10.21.3.3 void pcggui::ConvertBone (INode * *pParent*, model::ReferenceModel * *rm*)

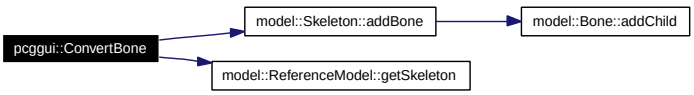
Definition at line 282 of file pcggui.cpp.

References model::Skeleton::addBone(), and model::ReferenceModel::getSkeleton().

Referenced by CovertScene().

```
282                                                                     {
283         DebugPrint("Sub Named Object: \"%s\"\n",pParent->GetName());
284
285         rm->getSkeleton()->addBone(NULL,NULL,10,pParent->GetName());
286
287         for(int q=0;q<pParent->NumberOfChildren();q++) {
288                 ConvertBone(pParent->GetChildNode(q),rm);
289         }
290 }
```

Here is the call graph for this function:



10.21.3.4 uint pcggui::ConvertSkin (INode * *node*, model::ReferenceModel * *rm*)

Definition at line 475 of file pcggui.cpp.

References model::Bone::addMapping(), ISkinContextData::GetAssignedBone(), ISkin::GetBone(), model::Skeleton::getBone(), ISkinContextData::GetBoneWeight(), ISkin::GetContextInterface(), getModifier(), ISkinContextData::GetNumAssigned-Bones(), ISkinContextData::GetNumPoints(), model::ReferenceModel::getSkeleton(), SKIN_CLASS_ID, SKIN_INTERFACE, and uint.

Referenced by CovertScene().

```
475                                                                     {
476         // Return success
477         uint ok=NoError;
478
479         // Get the skin modifier
480         Modifier* skin=getModifier (node, SKIN_CLASS_ID);
481
482         // Found it ?
483         if (skin)
484         {
485                 // **********    COMSKIN EXPORT    **********
486
487                 // Get a com_skin2 interface
488                 ISkin *comSkinInterface=(ISkin*)skin->GetInterface (SKIN_INTERFACE);
489
490                 // Should been controled with isSkin before.
491                 assert (comSkinInterface);
492
493                 // Found com_skin2 ?
```

```
494                 if (comSkinInterface)
495                 {
496                         // Get local data
497                         ISkinContextData *localData=comSkinInterface->GetContextInterface(node);
498 
499                         // Should been controled with isSkin before.
500                         assert (localData);
501 
502                         // Found ?
503                         if (localData)
504                         {
505                                 // Check same vertices count
506                                 uint vertCount=localData->GetNumPoints();
507 
508 
509                                 // For each vertex
510                                 for (uint vert=0; vert<vertCount; vert++)
511                                 {
512                                         // Get bones count for this vertex
513                                         uint boneCount=localData->GetNumAssignedBones (vert);
514 
515                                         // No bones, can't export
516                                         if (boneCount==0)
517                                         {
518                                                 // Error
519                                                 ok=VertexWithoutWeight;
520                                                 break;
521                                         }
522 
523                                         // A map of float / string
524                                         //std::multimap<float, uint> weightMap;
525                                         std::map<float, uint> weightMap;
526 
527                                         // For each bones
528                                         for (uint bone=0; bone<boneCount; bone++)
529                                         {
530                                                 // Get the bone weight
531                                                 float weight=localData->GetBoneWeight (vert, bone);
532 
533                                                 // Get bone number
534                                                 uint boneId=localData->GetAssignedBone (vert, bone);
535 
536                                                 // Insert in the map
537                                                 weightMap.insert (std::map<float, uint>::value_type (we
538                                         }
539 
540                                         // Sum the NL3D_MESH_SKINNING_MAX_MATRIX highest bones
541                                         float sum=0.f;
542                                         std::map<float, uint>::iterator ite=weightMap.begin();
543                                         while (ite!=weightMap.end())
544                                         {
545                                                 // Add to the sum
546                                                 sum+=ite->first;
547 
548                                                 // Next value
549                                                 ite++;
550                                         }
551 
552                                         // For each bones in the list, build the skin information
553                                         uint id=0;
554                                         ite=weightMap.end();
555                                         while (ite!=weightMap.begin())
```

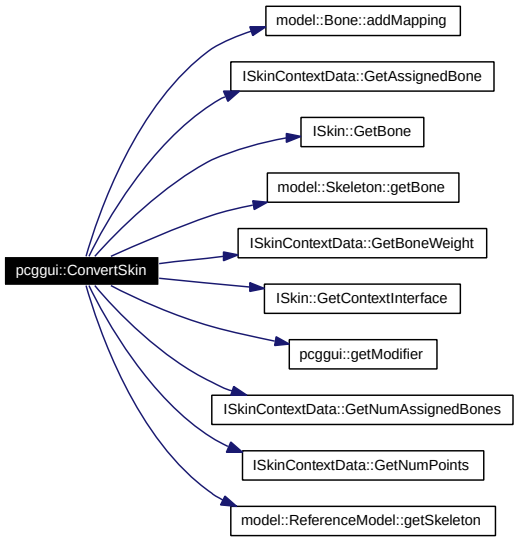
```
556                                         {
557                                                 // Previous value
558                                                 ite--;
559 
560                                                 // Get the bones ID
561                                                 Bone* bone = rm->getSkeleton()->getBone(comSk
562 
563                                                 // Set the weight
564                                                 //DebugPrint("Bone: %s - Vertex: %d - %f\n",b
565                                                 bone->addMapping(new model::Mapping(vert,ite-
566 
567                                                 // Next Id
568                                                 id++;
569                                         }
570                                         // Breaked ?
571                                         if (ite!=weightMap.begin())
572                                         {
573                                                 // break again to exit
574                                                 break;
575                                         }
576                                 }
577                         }
578                 }
579 
580                 // Release the interface
581                 skin->ReleaseInterface (SKIN_INTERFACE, comSkinInterface);
582         }
583         /*else
584         {
585                 // **********    PHYSIQUE EXPORT    **********
586 
587                 // Physique mode
588                 Modifier* skin=getModifier (node, PHYSIQUE_CLASS_ID);
589 
590                 // Must exist
591                 assert (skin);
592 
593                 // Get a com_skin2 interface
594                 IPhysiqueExport *physiqueInterface=(IPhysiqueExport *)skin->GetInterface (I_P
595 
596                 // Should been controled with isSkin before.
597                 assert (physiqueInterface);
598 
599                 // Found com_skin2 ?
600                 if (physiqueInterface)
601                 {
602                         // Get local data
603                         IPhyContextExport *localData=physiqueInterface->GetContextInterface(&
604 
605                         // Should been controled with isSkin before.
606                         assert (localData);
607 
608                         // Found ?
609                         if (localData)
610                         {
611                                 // Use rigid export
612                                 localData->ConvertToRigid (TRUE);
613 
614                                 // Allow blending
615                                 localData->AllowBlending (TRUE);
616 
617                                 // Check same vertices count
```

```
618                             uint vertCount=localData->GetNumberVertices();
619
620                             // Ctrl we have the same number of vertices in the mesh and in the modi
621                             if (rm->getVertexs().size()!=vertCount)
622                             {
623                                     ok=InvalidSkeleton;
624                             }
625                             else
626                             {
627                                     // If not the same count, return false (perhaps, the modifier i
628                                     if (rm->getVertexs().size()==vertCount)
629                                     {
630                                             // Rebuild the array
631                                             buildMesh.SkinWeights.resize (vertCount);
632
633                                             // For each vertex
634                                             for (uint vert=0; vert<vertCount; vert++)
635                                             {
636                                                     // Get a vertex interface
637                                                     IPhyVertexExport *vertexInterface=localData->Ge
638
639                                                     // Check if it is a rigid vertex or a blended v
640                                                     IPhyRigidVertex                 *rigidInterface
641                                                     IPhyBlendedRigidVertex  *blendedInterface=NULL;
642                                                     int type=vertexInterface->GetVertexType ();
643                                                     if (type==RIGID_TYPE)
644                                                     {
645                                                             // this is a rigid vertex
646                                                             rigidInterface=(IPhyRigidVertex*)vertex
647                                                     }
648                                                     else
649                                                     {
650                                                             // It must be a blendable vertex
651                                                             assert (type==RIGID_BLENDED_TYPE);
652                                                             blendedInterface=(IPhyBlendedRigidVerte
653                                                     }
654
655                                                     // Get bones count for this vertex
656                                                     uint boneCount;
657                                                     if (blendedInterface)
658                                                     {
659                                                             // If blenvertex, only one bone
660                                                             boneCount=blendedInterface->GetNumberNo
661                                                     }
662                                                     else
663                                                     {
664                                                             // If rigid vertex, only one bone
665                                                             boneCount=1;
666                                                     }
667
668                                                     // No bones, can't export
669                                                     if (boneCount==0)
670                                                     {
671                                                             // Error
672                                                             ok=VertexWithoutWeight;
673                                                             break;
674                                                     }
675
676                                                     // A map of float / string
677                                                     //std::multimap<float, INode*> weightMap;
678                                                     std::map<float, INode*> weightMap;
679
```

```
680                                                     // For each bones
681                                                     for (uint bone=0; bone<boneCount; bon
682                                                     {
683                                                             if (blendedInterface)
684                                                             {
685                                                                     // Get the bone weigh
686                                                                     float weight=blendedI
687
688                                                                     // Get node
689                                                                     INode *node=blendedIn
690                                                                     assert (node);
691
692                                                                     // Insert in the map
693                                                                     weightMap.insert (std
694                                                             }
695                                                             else
696                                                             {
697                                                                     // Get node
698                                                                     INode *node=rigidInte
699                                                                     assert (node);
700
701                                                                     // Insert in the map
702                                                                     weightMap.insert (std
703                                                             }
704                                                     }
705
706                                                     // Sum the NL3D_MESH_SKINNING_MAX_MAT
707                                                     float sum=0.f;
708                                                     std::map<float, INode*>::iterator ite
709                                                     while (ite!=weightMap.end())
710                                                     {
711                                                             // Add to the sum
712                                                             sum+=ite->first;
713
714                                                             // Next value
715                                                             ite++;
716                                                     }
717
718                                                     // For each bones in the list, build
719                                                     uint id=0;
720                                                     ite=weightMap.end();
721                                                     while (ite!=weightMap.begin())
722                                                     {
723                                                             // Previous value
724                                                             ite--;
725
726                                                             // Get the bones ID
727                                                             sint32 matrixId=-1;
728                                                             TInodePtrInt::const_iterator
729                                                             if (itId!=skeletonShape.end()
730                                                                     matrixId=itId->second
731
732                                                             // Find the bone ?
733                                                             if (matrixId==-1)
734                                                             {
735                                                                     // no, error, wrong s
736                                                                     ok=InvalidSkeleton;
737                                                                     break;
738                                                             }
739
740                                                             // Set the weight
741                                                             buildMesh.SkinWeights[vert].M
```

```
742                                                             buildMesh.SkinWeights[vert].Weights[id]
743
744                                                             // Next Id
745                                                             id++;
746                                                     }
747
748                                                     // Breaked ?
749                                                     if (ite!=weightMap.begin())
750                                                     {
751                                                             // break again to exit
752                                                             break;
753                                                     }
754
755                                                     // Release vertex interfaces
756                                                     localData->ReleaseVertexInterface (vertexInterf
757                                             }
758                                     }
759                             }
760
761                             // Release locaData interface
762                             physiqueInterface->ReleaseContextInterface (localData);
763                     }
764
765             }
766
767             // Release the interface
768             skin->ReleaseInterface (I_PHYINTERFACE, physiqueInterface);
769     }*/
770
771     return ok;
772 }
```

Here is the call graph for this function:



10.21.3.5 void pcggui::CovertScene ()

Definition at line 178 of file pcggui.cpp.

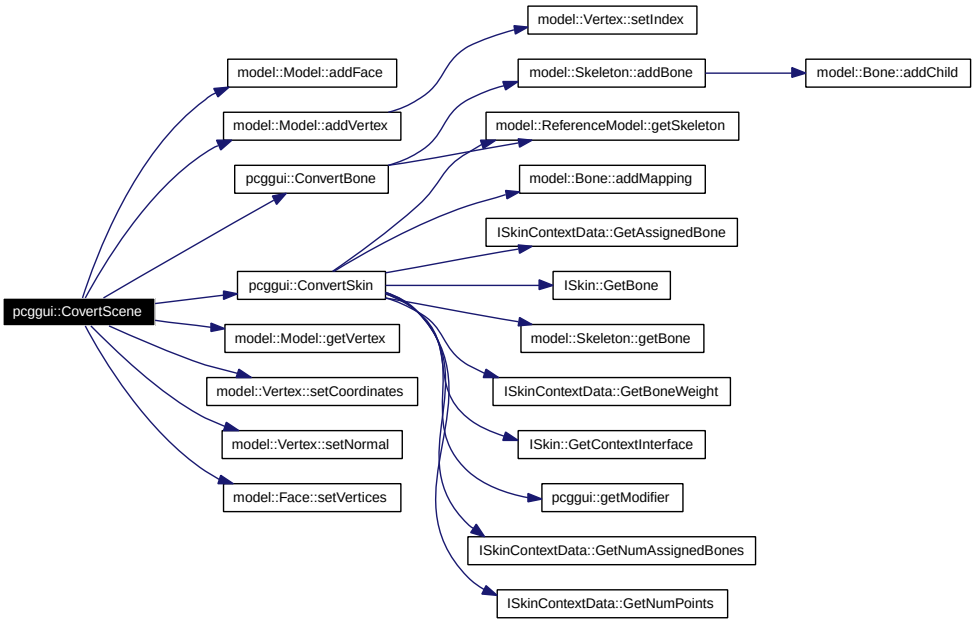
References model::Model::addFace(), model::Model::addVertex(), ConvertBone(), ConvertSkin(), exampleModels, model::Model::getVertex(), ip, referenceModels, model::Vertex::setCoordinates(), model::Vertex::setNormal(), model::Face::setVertices(), and thepcggui.

Referenced by pcgguiDlgProc().

```
178                         {
179         model::ExampleModel* em = new model::ExampleModel();
180         model::ReferenceModel* rm = new model::ReferenceModel();
181         TimeValue currtime = thepcggui.ip->GetTime();
182         INode* node;
183         BOOL needDel;
184         NullView nullView;
185         unsigned long offset = 0;
186         unsigned long tmpOffset = 0;
187 
188         if(thepcggui.ip->GetRootNode()->NumberOfChildren() > 0) {
189                 for(int n=0;n<thepcggui.ip->GetRootNode()->NumberOfChildren();n++) {
190 
191                         node = thepcggui.ip->GetRootNode()->GetChildNode(n);
192                         ObjectState os = node->EvalWorldState(currtime);
193 
195                         // find and export modifier data if
196                         // exists for this node
197 
198                         Object* pObj = node->GetObjectRef();
199                         // make sure we have a derived obj, which implies
200                         // modifiers are avail, and not a base obj
201                         while (pObj && pObj->SuperClassID() == GEN_DERIVOB_CLASS_ID) {
202                                 IDerivedObject* pDerObj = (IDerivedObject*) pObj;
203                                 // Iterate over all entries of the modifier stack.
204                                 for (int Idx = 0; Idx < pDerObj->NumModifiers(); Idx++) {
205                                         Modifier* mod = pDerObj->GetModifier(Idx);
206                                         DebugPrint("Mod Name: %s\n",mod->GetName());
207                                         DebugPrint("Mod class ID: 0x%x - %d\n",mod->ClassID()
208                                         /*if(mod->ClassID() == 9845843) {
209                                                 DebugPrint("SKIN\n");
210                                         }*/
211 
212                                         //if (mod->ClassID() == Class_ID(BENDOSM_CLASS_ID,0)
213                                 }
214                                 // get next object in stack -- could be base obj,
215                                 // or another derived obj
216                                 pObj = pDerObj->GetObjRef();
217                         }
218 
219 
220                         if(os.obj->SuperClassID()==GEOMOBJECT_CLASS_ID) {
221                                 if(node->GetBoneNodeOnOff()) {
222                                         if(refModel) {
223                                                 ConvertBone(node,rm);
224                                         }
225                                 } else {
226                                         Mesh *mesh = ((GeomObject*)os.obj)->GetRenderMesh(the
227                                         if (!mesh) return;
228 
```

```
229                                     mesh->buildNormals();
230                                     Matrix3 tm = node->GetObjTMAfterWSM(thepcggui.ip->GetTime());
231                                     for (int i=0; i<mesh->getNumVerts(); i++) {
232                                             Point3 v = (tm * mesh->verts[i]);
233                                             model::Vertex* vertex = new model::Vertex();
234                                             vertex->setCoordinates(v.x, v.y, v.z);
235                                             vertex->setNormal(mesh->getNormal(i).x,mesh->getNormal(
236                                             if(refModel) {
237                                                     rm->addVertex(vertex);
238                                             } else {
239                                                     em->addVertex(vertex);
240                                             }
241                                             tmpOffset = i;
242                                     }
243                                     for (int i=0; i<mesh->getNumFaces(); i++) {
244                                             model::Face* face = new model::Face();
245                                             if(refModel) {
246                                                     face->setVertices(
247                                                             rm->getVertex(offset + (int)mesh->faces
248                                                             rm->getVertex(offset + (int)mesh->faces
249                                                             rm->getVertex(offset + (int)mesh->faces
250                                                     rm->addFace(face);
251                                             } else {
252                                                     face->setVertices(
253                                                             em->getVertex(offset + (int)mesh->faces
254                                                             em->getVertex(offset + (int)mesh->faces
255                                                             em->getVertex(offset + (int)mesh->faces
256                                                     em->addFace(face);
257                                             }
258                                     }
259                                     offset = tmpOffset;
260                             }
261                     }
262             }
263             for(int n=0;n<thepcggui.ip->GetRootNode()->NumberOfChildren();n++) {
264                     node = thepcggui.ip->GetRootNode()->GetChildNode(n);
265                     ObjectState os = node->EvalWorldState(currtime);
266                     if(os.obj->SuperClassID()==GEOMOBJECT_CLASS_ID) {
267                             if(!node->GetBoneNodeOnOff()) {
268                                     if(refModel) {
269                                             DebugPrint("Error Code: %d\n",ConvertSkin(node,rm));
270                                     }
271                             }
272                     }
273             }
274             if(refModel) {
275                     referenceModels.push_back(rm);
276             } else {
277                     exampleModels.push_back(em);
278             }
279     }
280 }
```

Here is the call graph for this function:



10.21.3.6 void pcggui::DeleteThis () [inline]

Definition at line 67 of file pcggui.h.

```
67 { }
```

10.21.3.7 void pcggui::Destroy (HWND *hWnd*)

Definition at line 173 of file pcggui.cpp.

References file.

Referenced by pcgguiDlgProc().

```
174 {
175         fprintf(file,"Destroy\n");
176 }
```

10.21.3.8 void pcggui::EndEditParams (Interface * *ip*, IUtil * *iu*)

Definition at line 159 of file pcggui.cpp.

References file, and hPanel.

```
160 {
161         this->iu = NULL;
162         this->ip = NULL;
163         ip->DeleteRollupPage(hPanel);
164         hPanel = NULL;
165         fclose(file);
166 }
```

10.21.3.9 void pcggui::ExportAll ()

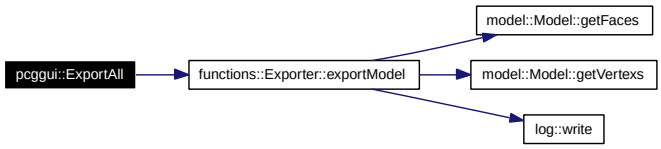
Definition at line 292 of file pcggui.cpp.

References exampleModels, functions::Exporter::exportModel(), and referenceModels.

Referenced by pcgguiDlgProc().

```
292                         {
293         functions::Exporter ex = functions::Exporter();
294         char tmp[100];
295
296         for(int i=0; i<exampleModels.size(); i++) {
297                 sprintf_s(tmp,"c:\\exampleModel%d.x3d",i);
298                 ex.exportModel(exampleModels[i],tmp);
299         }
300
301         for(int i=0; i<referenceModels.size(); i++) {
302                 sprintf_s(tmp,"c:\\referenceModel%d.x3d",i);
303                 ex.exportModel(referenceModels[i],tmp);
304         }
305
306 }
```

Here is the call graph for this function:



10.21.3.10 Modifier * pcggui::getModifier (INode * *pNode*, Class_ID *modCID*)

Definition at line 437 of file pcggui.cpp.

Referenced by ConvertSkin().

```
438 {
439         Object* pObj = pNode->GetObjectRef();
440 
441         if (!pObj)
442                 return NULL;
443 
444         ObjectState os = pNode->EvalWorldState(0);
445         if (os.obj && (os.obj->SuperClassID() != GEOMOBJECT_CLASS_ID) && (os.obj->SuperClassI
446         {
447                 return NULL;
448         }
449 
450         // For all derived objects (can be > 1)
451         while (pObj && (pObj->SuperClassID() == GEN_DERIVOB_CLASS_ID))
452         {
453                 IDerivedObject* pDObj = (IDerivedObject*)pObj;
454                 int m;
455                 int nNumMods = pDObj->NumModifiers();
456                 // Step through all modififers and verify the class id
457                 for (m=0; m<nNumMods; ++m)
458                 {
459                         Modifier* pMod = pDObj->GetModifier(m);
460                         if (pMod)
461                         {
462                                 if (pMod->ClassID() == modCID)
463                                 {
464                                         // Match! Return it
465                                         return pMod;
466                                 }
467                         }
468                 }
469                 pObj = pDObj->GetObjRef();
470         }
471 
472         return NULL;
473 }
```

10.21.3.11 void pcggui::Init (HWND *hWnd*)

Definition at line 168 of file pcggui.cpp.

References file.

Referenced by pcgguiDlgProc().

```
169 {
170         fprintf(file,"Init\n");
171 }
```

10.21.3.12 void pcggui::LoadAll ()

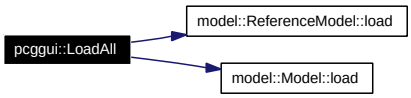
Definition at line 322 of file pcggui.cpp.

References exampleModels, model::ReferenceModel::load(), model::Model::load(), and referenceModels.

Referenced by pcgguiDlgProc().

```
322                     {
323         char tmp[100];
324         int number = 10;
325
326         exampleModels.clear();
327         referenceModels.clear();
328
329         for(int i=0; i<number; i++) {
330                 sprintf_s(tmp,"c:\\savedExampleModel%d.pcg",i);
331                 model::ExampleModel* em = new model::ExampleModel();
332                 if(em->load(tmp)) {
333                         exampleModels.push_back(em);
334                 }
335         }
336
337         for(int i=0; i<number; i++) {
338                 sprintf_s(tmp,"c:\\savedReferenceModel%d.pcg",i);
339                 model::ReferenceModel* rm = new model::ReferenceModel();
340                 if(rm->load(tmp)) {
341                         referenceModels.push_back(rm);
342                 }
343         }
344 }
```

Here is the call graph for this function:



10.21.3.13 void pcggui::LoadAnalyzer ()

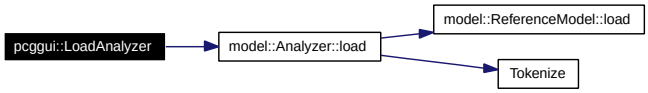
Definition at line 350 of file pcggui.cpp.

References model::Analyzer::load(), and pca.

Referenced by pcgguiDlgProc().

```
350                         {
351         pca.load("c:\\PCG-ANALYZER");
352 }
```

Here is the call graph for this function:



10.21.3.14 void pcggui::Render ()

Definition at line 376 of file pcggui.cpp.

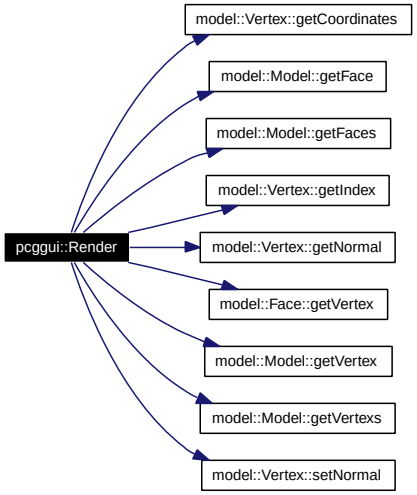
References exampleModels, model::Vertex::getCoordinates(), model::Model::getFace(), model::Model::getFaces(), model::Vertex::getIndex(), model::Vertex::getNormal(), model::Face::getVertex(), model::Model::getVertex(), model::Model::getVertexs(), ip, and model::Vertex::setNormal().

Referenced by pcgguiDlgProc().

```
376                     {
377         Mesh mesh;
378         model::ExampleModel* em = exampleModels[0];
379         mesh.setNumVerts(em->getVertexs().size());
380         mesh.setNumFaces(em->getFaces().size());
381         for(int i=0;i<em->getVertexs().size();i++) {
382                 Vertex* v = em->getVertex(i);
383                 mesh.setVert(i,Point3(v->getCoordinates()->x,v->getCoordinates()->y,v->getCo
384                 mesh.setNormal(i,Point3(v->getNormal()->x,v->getNormal()->y,v->getNormal()->z
385         }
386         for(int i=0;i<em->getFaces().size();i++) {
387                 model::Face* f = em->getFace(i);
388                 mesh.faces[i].setVerts(f->getVertex(0)->getIndex(),f->getVertex(1)->getIndex(
389         }
390 
391         //TriObject *obj = CreateNewTriObject();
392         //RegisterEditTriObjDesc( GetEditTriObjDesc() );
393 
394         // Create a new object through the CreateInstance() API
395         GeomObject *obj = (GeomObject*)ip->CreateInstance(GEOMOBJECT_CLASS_ID, Class_ID(BOXOB
396         assert(obj);
397 
398 
399         //((TriObject*)obj)->mesh = mesh;
400         //RegisterEditTriObjDesc( GetTriObjDescriptor() );
401 
402 
403 
404         // Create a derived object that references the cylinder
405         IDerivedObject *dobj = CreateDerivedObject(obj);
406 
407         // Create a node in the scene that references the derived object
408         INode *node = ip->CreateObjectNode(dobj);
409 
410         // Name the node and make the name unique.
411         TSTR name(_T("MyNode"));
412         ip->MakeNameUnique(name);
413         node->SetName(name);
414 
415         // Eval the node's object (exclude WSMs)
416         Object *oldObj = node->GetObjectRef();
417         ObjectState os = oldObj->Eval(ip->GetTime());
418 
419         // Set the result to be the node's new object (make a clone)
420         Object *newobj = (Object*)os.obj->Clone();
421         newobj->SetSubSelState(0); // Reset the selection level to object level
422         oldObj->SetAFlag(A_LOCK_TARGET);
```

```
423         node->SetObjectRef(newobj);
424         node->NotifyDependents(FOREVER,0,REFMSG_SUBANIM_STRUCTURE_CHANGED);
425         oldObj->ClearAFlag(A_LOCK_TARGET);
426         oldObj->MaybeAutoDelete();
427
428
429         // Redraw the viewports
430         ip->RedrawViews(ip->GetTime());
431
432 }
```

Here is the call graph for this function:



10.21.3.15 void pcggui::SaveAll ()

Definition at line 308 of file pcggui.cpp.

References exampleModels, and referenceModels.

Referenced by pcgguiDlgProc().

```
308                     {
309         char tmp[100];
310
311         for(int i=0; i<exampleModels.size(); i++) {
312                 sprintf_s(tmp,"c:\\savedExampleModel%d.pcg",i);
313                 exampleModels[i]->save(tmp);
314         }
315
316         for(int i=0; i<referenceModels.size(); i++) {
317                 sprintf_s(tmp,"c:\\savedReferenceModel%d.pcg",i);
318                 referenceModels[i]->save(tmp);
319         }
320 }
```

10.21.3.16 void pcggui::SaveAnalyzer ()

Definition at line 346 of file pcggui.cpp.

References pca, and model::Analyzer::save().

Referenced by pcgguiDlgProc().

```
346                             {
347         pca.save("c:\\PCG-ANALYZER");
348 }
```

Here is the call graph for this function:

10.21.3.17 void pcggui::SetReferenceModelMode (BOOL *rm*) [inline]

Definition at line 77 of file pcggui.h.

References refModel.

Referenced by pcgguiDlgProc().

```
77 { refModel = rm; };
```

10.21.4 Member Data Documentation

10.21.4.1 vector<model::ExampleModel*> pcggui::exampleModels [private]

Definition at line 43 of file pcggui.h.

Referenced by Analyze(), CovertScene(), ExportAll(), LoadAll(), Render(), and SaveAll().

10.21.4.2 HWND pcggui::hPanel

Definition at line 49 of file pcggui.h.

Referenced by BeginEditParams(), EndEditParams(), and pcggui().

10.21.4.3 Interface* pcggui::ip

Definition at line 51 of file pcggui.h.

Referenced by CovertScene(), pcggui(), pcgguiDlgProc(), and Render().

10.21.4.4 IUtil* pcggui::iu

Definition at line 50 of file pcggui.h.

Referenced by pcggui().

10.21.4.5 model::Analyzer pcggui::pca [private]

Definition at line 45 of file pcggui.h.

Referenced by Analyze(), LoadAnalyzer(), and SaveAnalyzer().

10.21.4.6 vector<model::ReferenceModel*> pcggui::referenceModels [private]

Definition at line 44 of file pcggui.h.

Referenced by Analyze(), CovertScene(), ExportAll(), LoadAll(), and SaveAll().

10.21.4.7 BOOL pcggui::refModel

Definition at line 52 of file pcggui.h.

Referenced by SetReferenceModelMode().

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

- gui/pcggui/pcggui.h
- gui/pcggui/pcggui.cpp

10.22 pcgguiClassDesc Class Reference

Public Member Functions

- int IsPublic ()
- void * Create (BOOL loading=FALSE)
- const TCHAR * ClassName ()
- SClass_ID SuperClassID ()
- Class_ID ClassID ()
- const TCHAR * Category ()
- const TCHAR * InternalName ()
- HINSTANCE HInstance ()

10.22.1 Member Function Documentation

10.22.1.1 const TCHAR* pcgguiClassDesc::Category () [inline]

Definition at line 39 of file pcggui.cpp.

References GetString(), and IDS_CATEGORY.

```
39 { return GetString(IDS_CATEGORY); }
```

Here is the call graph for this function:

10.22.1.2 Class_ID pcgguiClassDesc::ClassID () [inline]

Definition at line 38 of file pcggui.cpp.

```
38 { return pcggui_CLASS_ID; }
```

10.22.1.3 const TCHAR* pcgguiClassDesc::ClassName () [inline]

Definition at line 36 of file pcggui.cpp.

References GetString(), and IDS_CLASS_NAME.

```
36 { return GetString(IDS_CLASS_NAME); }
```

Here is the call graph for this function:

10.22.1.4 void* pcgguiClassDesc::Create (BOOL *loading* = FALSE) [inline]

Definition at line 35 of file pcggui.cpp.

```
35 { return &thepcggui; }
```

10.22.1.5 HINSTANCE pcgguiClassDesc::HInstance () [inline]

Definition at line 42 of file pcggui.cpp.

```
42 { return hInstance; }                                    // returns owning module handle
```

10.22.1.6 const TCHAR* pcgguiClassDesc::InternalName () [inline]

Definition at line 41 of file pcggui.cpp.

```
41 { return _T("pcggui"); }        // returns fixed parsable name (scripter-visible name)
```

10.22.1.7 int pcgguiClassDesc::IsPublic () [inline]

Definition at line 34 of file pcggui.cpp.

```
34 { return TRUE; }
```

10.22.1.8 SClass_ID pcgguiClassDesc::SuperClassID () [inline]

Definition at line 37 of file pcggui.cpp.

```
37 { return UTILITY_CLASS_ID; }
```

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

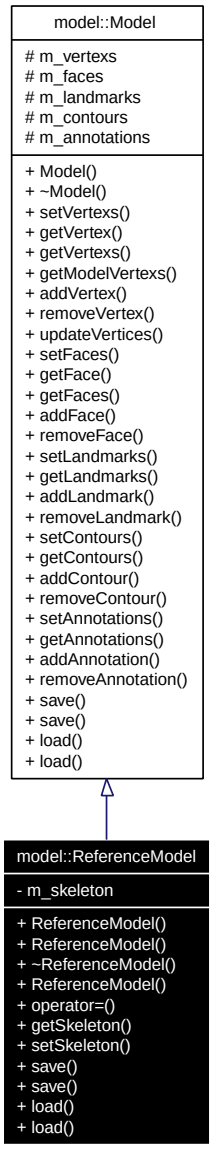
- gui/pcggui/pcggui.cpp

10.23 model::ReferenceModel Class Reference

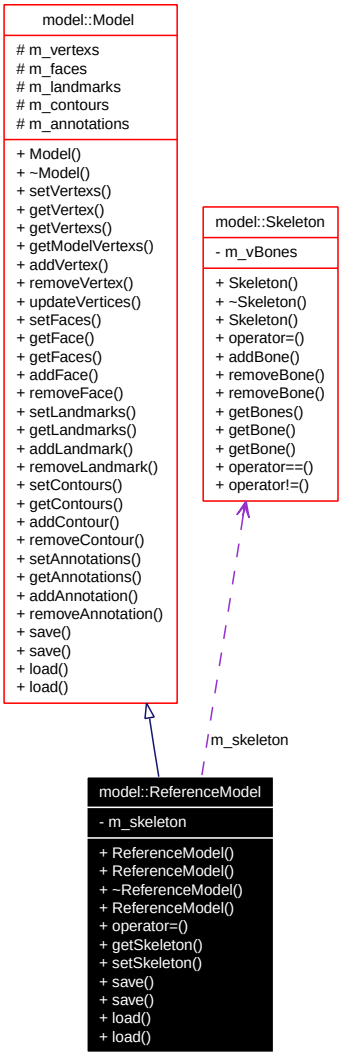
An ReferenceModel in PCG.

`#include <ReferenceModel.h>`

Inheritance diagram for model::ReferenceModel:



Collaboration diagram for model::ReferenceModel:



Public Member Functions

- ReferenceModel ()
- ReferenceModel (Skeleton ∗skeleton)
- ∼ReferenceModel ()
- ReferenceModel (const ReferenceModel &other)
- ReferenceModel & operator= (const ReferenceModel &other)
- Skeleton ∗ getSkeleton ()
- void setSkeleton (Skeleton ∗skeleton)
- bool save (char ∗filename)
- bool save (ofstream ∗saveFile)
- bool load (char ∗filename)
- bool load (ifstream ∗loadFile)

Private Attributes

- Skeleton * m_skeleton

10.23.1 Detailed Description

An ReferenceModel in PCG.

Definition at line 19 of file ReferenceModel.h.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 ReferenceModel::ReferenceModel ()

Definition at line 5 of file ReferenceModel.cpp.

References m_skeleton.

```
5                                {
6         m_skeleton = new Skeleton();
7 }
```

10.23.2.2 ReferenceModel::ReferenceModel (Skeleton * *skeleton*)

Write brief comment for ReferenceModel here.

Parameters:
skeleton Description of parameter skeleton.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ReferenceModel here.

Remarks:
Write remarks for ReferenceModel here.

See also:
Separate items with the '|' character.

Definition at line 26 of file ReferenceModel.cpp.

References m_skeleton.

```
26                                                  {
27         m_skeleton = skeleton;
28 }
```

10.23.2.3 model::ReferenceModel::~ReferenceModel () [inline]

Write brief comment for ~ReferenceModel here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~ReferenceModel here.

Remarks:
Write remarks for ~ReferenceModel here.

See also:
Separate items with the '|' character.

Definition at line 42 of file ReferenceModel.h.

```
42 {};
```

10.23.2.4 ReferenceModel::ReferenceModel (const ReferenceModel & *other*)

Write brief comment for ReferenceModel here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ReferenceModel here.

Remarks:
Write remarks for ReferenceModel here.

See also:
Separate items with the '|' character.

Definition at line 47 of file ReferenceModel.cpp.

References model::Model::m_annotations, model::Model::m_contours, model::Model::m_faces, model::Model::m_landmarks, m_skeleton, and model::Model::m_vertexs.

```
47                                                          {
48         m_skeleton = other.m_skeleton;
49         m_vertexs = other.m_vertexs;
50         m_faces = other.m_faces;
51         m_landmarks = other.m_landmarks;
52         m_contours = other.m_contours;
53         m_annotations = other.m_annotations;
54 }
```

10.23.3 Member Function Documentation

10.23.3.1 Skeleton * ReferenceModel::getSkeleton ()

Write brief comment for getSkeleton here.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getSkeleton here.

Remarks:
Write remarks for getSkeleton here.

See also:
Separate items with the '|' character.

Definition at line 111 of file ReferenceModel.cpp.

Referenced by pcggui::ConvertBone(), and pcggui::ConvertSkin().

```
111                                         {
112         return m_skeleton;
113 }
```

10.23.3.2 bool ReferenceModel::load (ifstream * *loadFile*)

Write brief comment for load here.

Parameters:
loadFile Description of parameter loadFile.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for load here.

Remarks:
Write remarks for load here.

See also:
Separate items with the '|' character.

Reimplemented from model::Model.

Definition at line 250 of file ReferenceModel.cpp.

References model::Model::load().

```
250                                             {
251         bool res = Model::load(loadFile);
252         if(res) {
253                 string line;
254                 vector<string> tokens;
255                 vector<string> coords;
256
257                 if(loadFile->is_open()) {
258                         getline(*loadFile,line);//Skeleton
259                         getline(*loadFile,line);//Data
260                         return true;
261                 } else {
262                         return false;
263                 }
264         } else {
265                 return false;
266         }
267 }
```

Here is the call graph for this function:

10.23.3.3 bool ReferenceModel::load (char * *filename*)

Write brief comment for load here.

Parameters:
filename Description of parameter filename.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for load here.

Remarks:
Write remarks for load here.

See also:
Separate items with the '|' character.

Reimplemented from model::Model.

Definition at line 222 of file ReferenceModel.cpp.

Referenced by model::Analyzer::load(), and pcggui::LoadAll().

```
222                                         {
223         ifstream loadFile(filename);
224         bool res = load(&loadFile);
225         loadFile.close();
226         return res;
227 }
```

10.23.3.4 ReferenceModel & ReferenceModel::operator= (const ReferenceModel & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 77 of file ReferenceModel.cpp.

References model::Model::m_annotations, model::Model::m_contours, model::Model::m_faces, model::Model::m_landmarks, m_skeleton, and model::Model::m_vertexs.

```
78 {
79     // if same object
80     if ( this == &other )
81         return *this;
82 
83         m_skeleton = other.m_skeleton;
84         m_vertexs = other.m_vertexs;
85         m_faces = other.m_faces;
86         m_landmarks = other.m_landmarks;
87         m_contours = other.m_contours;
88         m_annotations = other.m_annotations;
89 
90     return *this;
91 }
```

10.23.3.5 bool ReferenceModel::save (ofstream * *saveFile*)

Write brief comment for save here.

Parameters:

saveFile Description of parameter saveFile.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for save here.

Remarks:

Write remarks for save here.

See also:

Separate items with the '|' character.

Reimplemented from model::Model.

Definition at line 186 of file ReferenceModel.cpp.

References model::Model::save().

```
186                                             {
187         bool res = Model::save(saveFile);
188         if(res) {
```

```
189                 if(saveFile->is_open()) {
190                         *saveFile << "Skeleton\n";
191                         //m_skeleton->getBones();
192                         return true;
193                 } else {
194                         return false;
195                 }
196         } else {
197                 return false;
198         }
199 }
```

Here is the call graph for this function:

10.23.3.6 bool ReferenceModel::save (char * *filename*)

Write brief comment for save here.

Parameters:

filename Description of parameter filename.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for save here.

Remarks:

Write remarks for save here.

See also:

Separate items with the '|' character.

Reimplemented from model::Model.

Definition at line 158 of file ReferenceModel.cpp.

Referenced by model::Analyzer::save().

```
158                                         {
159         ofstream saveFile(filename);
160         bool res = save(&saveFile);
161         saveFile.close();
162         return res;
163 }
```

10.23.3.7 void ReferenceModel::setSkeleton (Skeleton * *skeleton*)

Write brief comment for setSkeleton here.

Parameters:
skeleton Description of parameter skeleton.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for setSkeleton here.

Remarks:
Write remarks for setSkeleton here.

See also:
Separate items with the '|' character.

Definition at line 133 of file ReferenceModel.cpp.

References m_skeleton.

```
133                                                     {
134         m_skeleton = skeleton;
135 }
```

10.23.4 Member Data Documentation

10.23.4.1 Skeleton* model::ReferenceModel::m_skeleton [private]

Definition at line 21 of file ReferenceModel.h.

Referenced by operator=(), ReferenceModel(), and setSkeleton().

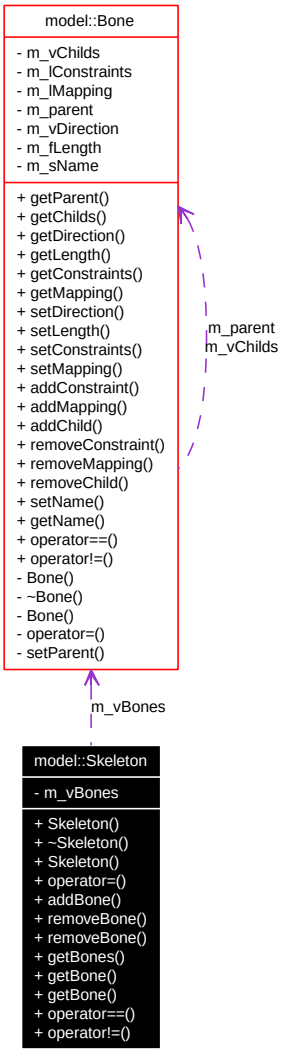
The documentation for this class was generated from the following files:

- model/ReferenceModel.h
- model/ReferenceModel.cpp

10.24 model::Skeleton Class Reference

`#include <Skeleton.h>`

Collaboration diagram for model::Skeleton:



Public Member Functions

- Skeleton ()
- ~Skeleton ()
- Skeleton (const Skeleton &other)

- Skeleton & operator= (const Skeleton &other)
- void addBone (Bone *parent, IvVector3 *direction, float length, string name)

  Write brief comment for addBone here.

- void removeBone (int id)
- void removeBone (string name)
- vector< Bone * > getBones ()
- Bone * getBone (int id)
- Bone * getBone (string name)
- bool operator== (const Skeleton &other) const
- bool operator!= (const Skeleton &other) const

Private Attributes

- Bone * m_vBones

10.24.1 Detailed Description

Write brief comment for Skeleton here.

Write detailed description for Skeleton here.

Remarks:
Write remarks for Skeleton here.

See also:
Separate items with the '|' character.

Definition at line 29 of file Skeleton.h.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 model::Skeleton::Skeleton () [inline]

Write brief comment for Skeleton here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Skeleton here.

Remarks:
Write remarks for Skeleton here.

See also:
Separate items with the '|' character.

Definition at line 52 of file Skeleton.h.

```
52 {};
```

10.24.2.2 model::Skeleton::~Skeleton () [inline]

Write brief comment for ~Skeleton here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for ~Skeleton here.

Remarks:
Write remarks for ~Skeleton here.

See also:
Separate items with the '|' character.

Definition at line 68 of file Skeleton.h.

```
68 {};
```

10.24.2.3 Skeleton::Skeleton (const Skeleton & *other*)

Write brief comment for Skeleton here.

Parameters:
other Description of parameter other.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for Skeleton here.

Remarks:
Write remarks for Skeleton here.

See also:
Separate items with the '|' character.

Definition at line 22 of file Skeleton.cpp.

References m_vBones.

```
22                                          {
23         m_vBones = other.m_vBones;
24 }
```

10.24.3 Member Function Documentation

10.24.3.1 void Skeleton::addBone (Bone * *parent*, IvVector3 * *direction*, float *length*, string *name*)

Write brief comment for addBone here.

Parameters:

bone Description of parameter bone.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for addBone here.

Remarks:

Write remarks for addBone here.

See also:

Separate items with the '|' character.

Definition at line 205 of file Skeleton.cpp.

References model::Bone::addChild(), and m_vBones.

Referenced by pcggui::ConvertBone().

```
205                                                                                                 {
206         Bone* bone = new Bone(parent,direction,length,name);
207         parent->addChild(bone);
208         m_vBones.push_back(bone);
209 }
```

Here is the call graph for this function:

10.24.3.2 Bone * Skeleton::getBone (string *name*)

Write brief comment for getBone here.

Parameters:
name Description of parameter name.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getBone here.

Remarks:
Write remarks for getBone here.

See also:
Separate items with the '|' character.

Definition at line 154 of file Skeleton.cpp.

References m_vBones.

```
154                                     {
155         for(int i=0; i<m_vBones.size(); i++) {
156                 if(name.compare(m_vBones[i]->getName()) == 0) {
157                         return m_vBones[i];
158                 }
159         }
160         return NULL;
161 }
```

10.24.3.3 Bone * Skeleton::getBone (int *id*)

Write brief comment for getBone here.

Parameters:
id Description of parameter id.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for getBone here.

Remarks:
Write remarks for getBone here.

See also:
Separate items with the '|' character.

Definition at line 183 of file Skeleton.cpp.

References m_vBones.

Referenced by pcggui::ConvertSkin().

```
183                                 {
184         return m_vBones[id];
185 }
```

10.24.3.4 vector< Bone * > Skeleton::getBones ()

Write brief comment for getBones here.

Returns:
Write description of return value here.

Exceptions:
<*exception* class> Description of criteria for throwing this exception.

Write detailed description for getBones here.

Remarks:
Write remarks for getBones here.

See also:
Separate items with the '|' character.

Definition at line 130 of file Skeleton.cpp.

```
130                                 {
131         return m_vBones;
132 }
```

10.24.3.5 bool Skeleton::operator!= (const Skeleton & *other*) const

Write brief comment for operator != here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator != here.

Remarks:
Write remarks for operator != here.

See also:
Separate items with the '|' character.

Definition at line 105 of file Skeleton.cpp.

References m_vBones.

```
106 {
107         if(m_vBones != other.m_vBones) {
108                 return false;
109         }
110         return true;
111 }
```

10.24.3.6 Skeleton & Skeleton::operator= (const Skeleton & *other*)

Write brief comment for operator = here.

Parameters:
other Description of parameter other.

Returns:
Write description of return value here.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator = here.

Remarks:
Write remarks for operator = here.

See also:
Separate items with the '|' character.

Definition at line 46 of file Skeleton.cpp.

References m_vBones.

```
47 {
48         // if same object
49         if ( this == &other )
50                 return *this;
51
52         m_vBones = other.m_vBones;
53
54         return *this;
55 }
```

10.24.3.7 bool Skeleton::operator== (const Skeleton & *other*) const

Write brief comment for operator == here.

Parameters:

other Description of parameter other.

Returns:

Write description of return value here.

Exceptions:

<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for operator == here.

Remarks:

Write remarks for operator == here.

See also:

Separate items with the '|' character.

Definition at line 77 of file Skeleton.cpp.

References m_vBones.

```
78 {
79         if(m_vBones == other.m_vBones) {
80                 return true;
81         }
82         return false;
83 }
```

10.24.3.8 void Skeleton::removeBone (std::string *name*)

Write brief comment for removeBone here.

Parameters:
name Description of parameter name.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeBone here.

Remarks:
Write remarks for removeBone here.

See also:
Separate items with the '|' character.

Definition at line 249 of file Skeleton.cpp.

```
249                                         {
250
251 }
```

10.24.3.9 void Skeleton::removeBone (int *id*)

Write brief comment for removeBone here.

Parameters:
id Description of parameter id.

Exceptions:
<***exception*** class> Description of criteria for throwing this exception.

Write detailed description for removeBone here.

Remarks:
Write remarks for removeBone here.

See also:
Separate items with the '|' character.

Definition at line 228 of file Skeleton.cpp.

```
228                                 {
229
230 }
```

10.24.4 Member Data Documentation

10.24.4.1 Bone* model::Skeleton::m_vBones [private]

Definition at line 32 of file Skeleton.h.

Referenced by addBone(), getBone(), operator!=(), operator=(), operator==(), and Skeleton().

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

- model/Skeleton.h
- model/Skeleton.cpp

10.25 model::Vertex Class Reference

A Vertex class.

`#include <Vertex.h>`

Public Member Functions

- Vertex ()
  Default constructor.
- Vertex (IvVector3 *vCoordinates)
  Constructor taking a coordinates vector.
- Vertex (float fX, float fY, float fZ)
  Constructor taking 3 floats as coorninates.
- ~Vertex ()
  Default deconstructor.
- Vertex (const Vertex &other)
  Copy constructor.
- Vertex & operator= (const Vertex &other)
  Copy assignmed.
- IvVector3 * getCoordinates ()
  Get coordinates.
- void setCoordinates (IvVector3 *vCoordinates)
  Set coordinates.
- void setCoordinates (float fX, float fY, float fZ)
  Sets the coordinates.
- void updateCoordinates (float fX, float fY, float fZ)
  Sets the coordinates.
- IvVector3 * getNormal ()
  Get normal.
- void setNormal (IvVector3 *vNormal)
  Sets the normal.

- void setNormal (float fX, float fY, float fZ)
  Sets the normal.
- void setIndex (int index)
  Sets the index.
- int getIndex ()
  Get the index.
- void translate (IvVector3 *vTargetPos)
  Translates the Vertex.
- void rotate (float fAngleX, float fAngleY, float fAngleZ)
  Rotate the Vertex.
- bool operator== (const Vertex &other) const
  Comparition operator.
- bool operator!= (const Vertex &other) const
  Comparition operator.

Private Attributes

- IvVector3 * m_vCoordinates
  Vector holding the coordinates of the vertex.
- IvVector3 * m_vNormal
- int m_index

10.25.1 Detailed Description

A Vertex class.

A data holding class for a Vertex. It has functions to calculate rotation and translation.

See also:
Face | Bone

Definition at line 22 of file Vertex.h.

10.25.2 Constructor & Destructor Documentation

10.25.2.1 model::Vertex::Vertex () [inline]

Default constructor.

Creates a empty Vertex

Definition at line 35 of file Vertex.h.

```
35 {};
```

10.25.2.2 Vertex::Vertex (IvVector3 * *vCoordinates*)

Constructor taking a coordinates vector.

A constructor that takes a vector and creates a vertex with the vector as it's coordinates.

Parameters:

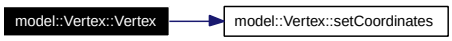
vCoordinates The vertexs coordinates vector.

Definition at line 16 of file Vertex.cpp.

References m_index, and setCoordinates().

```
16                                      {
17         setCoordinates(vCoordinates);
18         m_index = 0;
19 }
```

Here is the call graph for this function:



10.25.2.3 Vertex::Vertex (float *fX*, float *fY*, float *fZ*)

Constructor taking 3 floats as coorninates.

Parameters:

fX Coordinate X.

fY Coordinate Y.

fZ Coordinate Z.

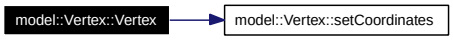
Creates a vertex with the floats as its coordinates.

Definition at line 36 of file Vertex.cpp.

References m_index, and setCoordinates().

```
36                                                  {
37         setCoordinates(fX,fY,fZ);
38         m_index = 0;
39 }
```

Here is the call graph for this function:



10.25.2.4 model::Vertex::~Vertex () [inline]

Default deconstructor.

It does nothing.

Definition at line 46 of file Vertex.h.

```
46 {};
```

10.25.2.5 Vertex::Vertex (const Vertex & *other*)

Copy constructor.

Copying all the variables to the new Vertex.

Parameters:

other A other Vertex

Definition at line 50 of file Vertex.cpp.

References m_index.

```
50                                      :
51 m_vCoordinates( other.m_vCoordinates ),
52 m_vNormal( other.m_vNormal )
53 {
54         m_index = 0;
55 }
```

10.25.3 Member Function Documentation

10.25.3.1 IvVector3 * Vertex::getCoordinates ()

Get coordinates.

Returns the coordiantes of the Vertex.

Returns:
A vector containg the corrdiantes of the Vertex.

Definition at line 136 of file Vertex.cpp.

Referenced by model::Model::getModelVertexs(), pcggui::Render(), and model::Model::save().

```
136                                     {
137         return m_vCoordinates;
138 }
```

10.25.3.2 int Vertex::getIndex ()

Get the index.

Returns the index of the Vertex.

Returns:
The index

Definition at line 272 of file Vertex.cpp.

Referenced by pcggui::Render(), and model::Model::save().

```
272                         {
273         return m_index;
274 }
```

10.25.3.3 IvVector3 * Vertex::getNormal ()

Get normal.

Returns the normal of the Vertex.

Returns:
A vector containing the normal of the Vertex.

Definition at line 148 of file Vertex.cpp.

Referenced by pcggui::Render(), and model::Model::save().

```
148                             {
149         return m_vNormal;
150 }
```

10.25.3.4 bool Vertex::operator!= (const Vertex & *other*) const

Comparition operator.

Compare two Vertexs to see if they are not equal.

Parameters:
other A other Vertex

Returns:
Returns true if the two Vertexs is not equal.

Definition at line 244 of file Vertex.cpp.

References m_vCoordinates, and m_vNormal.

```
245 {
246         if(other.m_vCoordinates == m_vCoordinates && other.m_vNormal == m_vNormal) {
247                 return false;
248         }
249         return true;
250 }
```

10.25.3.5 Vertex & Vertex::operator= (const Vertex & *other*)

Copy assignmed.

Copying all the variables to the new Vertex.

Parameters:
other A other Vertex

Definition at line 66 of file Vertex.cpp.

References m_vCoordinates, and m_vNormal.

```
67 {
68         // if same object
69         if ( this == &other )
70                 return *this;
71
72         m_vCoordinates = other.m_vCoordinates;
73         m_vNormal = other.m_vNormal;
74
75         return *this;
76 }
```

10.25.3.6 bool Vertex::operator== (const Vertex & *other*) const

Comparition operator.

Compare two Vertexs to see if thay are equal.

Parameters:

other A other Vertex

Returns:

Returns true if the two Vertexs is equal.

Definition at line 225 of file Vertex.cpp.

References m_vCoordinates, and m_vNormal.

```
226 {
227         if(other.m_vCoordinates == m_vCoordinates && other.m_vNormal == m_vNormal) {
228                 return true;
229         }
230         return false;
231 }
```

10.25.3.7 void Vertex::rotate (float *fAngleX*, float *fAngleY*, float *fAngleZ*)

Rotate the Vertex.

Changes the normal of the vertex giving 3 angles.

Parameters:

fAngleX The rotation around the x-axis.

fAngleY The rotation around the y-axis.

fAngleZ The rotation around the z-axis.

Definition at line 208 of file Vertex.cpp.

References m_vNormal.

```
208                                                                 {
209         IvMatrix33 euler = IvMatrix33();
210         euler.Rotation(fAngleZ, fAngleY, fAngleX);
211         m_vNormal = &(*m_vNormal * euler);
212 }
```

10.25.3.8 void Vertex::setCoordinates (float *fX*, float *fY*, float *fZ*)

Sets the coordinates.

Sets the coordinates of the Vertex giving 3 floats.

Parameters:

fX The X coordinate

fY The Y coordiante

fZ The Z coordinate

Definition at line 92 of file Vertex.cpp.

References m_vCoordinates.

```
92                                                          {
93         m_vCoordinates = new IvVector3(fX,fY,fZ);
94 }
```

10.25.3.9 void Vertex::setCoordinates (IvVector3 * *vCoordinates*)

Set coordinates.

Sets the coordnates of the Vertex giving a vector.

Parameters:

vCoordinates The vector with the coordinates of the Vertex.

Definition at line 104 of file Vertex.cpp.

References m_vCoordinates.

Referenced by pcggui::CovertScene(), model::Model::load(), and Vertex().

```
104                                                         {
105         m_vCoordinates = vCoordinates;
106 }
```

10.25.3.10 void Vertex::setIndex (int *index*)

Sets the index.

Sets the index of the Vertex giving a int.

Parameters:

index The new index

Definition at line 260 of file Vertex.cpp.

References m_index.

Referenced by model::Model::addVertex().

```
260                                 {
261         m_index = index;
262 }
```

10.25.3.11 void Vertex::setNormal (float *fX*, float *fY*, float *fZ*)

Sets the normal.

Sets the normal of the Vertex giving 3 floats.

Parameters:

fX The X element of the normal.

fY The Y element of the normal.

fZ The Z element of the normal.

Definition at line 178 of file Vertex.cpp.

References m_vNormal.

```
178                                                 {
179         m_vNormal = new IvVector3(fX,fY,fZ);
180 }
```

10.25.3.12 void Vertex::setNormal (IvVector3 * *vNormal*)

Sets the normal.

Sets the normal of the Vertex giving a vector.

Parameters:

vNormal A vector with the new normal

Definition at line 160 of file Vertex.cpp.

References m_vNormal.

Referenced by pcggui::CovertScene(), model::Model::load(), functions::Importer::readAttributes(), and pcggui::Render().

```
160                                         {
161         m_vNormal = vNormal;
162 }
```

10.25.3.13 void Vertex::translate (IvVector3 * *vTargetpos*)

Translates the Vertex.

Giving a vector the Vertex is translated as denoted by the vector.

Parameters:

vTargetpos The vector that are translated with.

Definition at line 190 of file Vertex.cpp.

References m_vCoordinates.

```
190                                                 {
191         m_vCoordinates=&(*m_vCoordinates + *vTargetpos);
192 }
```

10.25.3.14 void Vertex::updateCoordinates (float *fX*, float *fY*, float *fZ*)

Sets the coordinates.

Sets the coordinates of the Vertex giving 3 floats.

Parameters:

fX The X coordinate

fY The Y coordiante

fZ The Z coordinate

Definition at line 122 of file Vertex.cpp.

References m_vCoordinates.

Referenced by model::Model::updateVertices().

```
122                                                                 {
123         m_vCoordinates->x = fX;
124         m_vCoordinates->y = fY;
125         m_vCoordinates->z = fZ;
126 }
```

10.25.4 Member Data Documentation

10.25.4.1 int model::Vertex::m_index [private]

Definition at line 27 of file Vertex.h.

Referenced by setIndex(), and Vertex().

10.25.4.2 IvVector3* model::Vertex::m_vCoordinates [private]

Vector holding the coordinates of the vertex.

Definition at line 25 of file Vertex.h.

Referenced by operator!=(), operator=(), operator==(), setCoordinates(), translate(), and updateCoordinates().

10.25.4.3 IvVector3* model::Vertex::m_vNormal [private]

Definition at line 26 of file Vertex.h.

Referenced by operator!=(), operator=(), operator==(), rotate(), and setNormal().

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

- model/Vertex.h
- model/Vertex.cpp

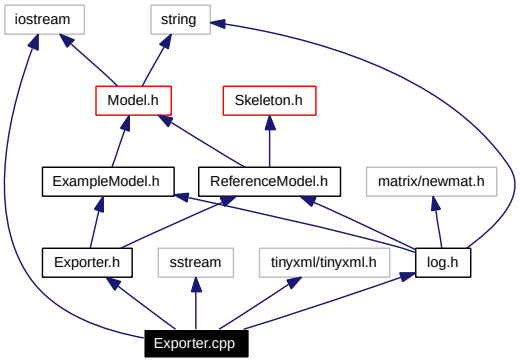
Chapter 11

PCG Library File Documentation

11.1 Functions/Exporter.cpp File Reference

```
#include "Exporter.h"
#include <iostream>
#include <sstream>
#include <tinyxml/tinyxml.h>
#include <log.h>
```

Include dependency graph for Exporter.cpp:

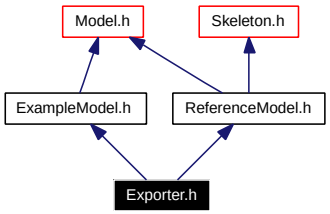


11.2 Functions/Exporter.h File Reference

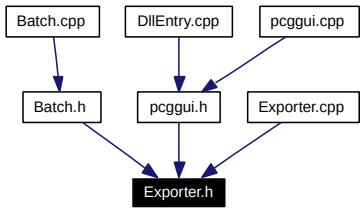
`#include <ExampleModel.h>`

`#include <ReferenceModel.h>`

Include dependency graph for Exporter.h:



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



Namespaces

- namespace functions

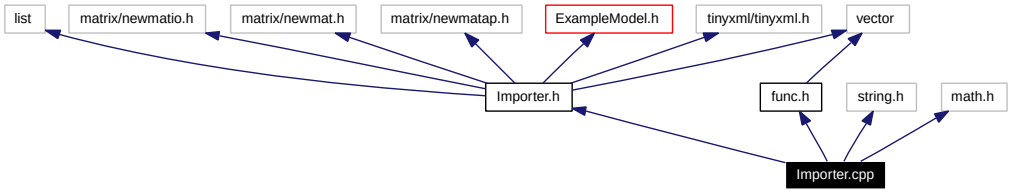
Classes

- class functions::Exporter

11.3 Functions/Importer.cpp File Reference

```
#include "Importer.h"
#include <string.h>
#include "include/func.h"
#include <math.h>
```

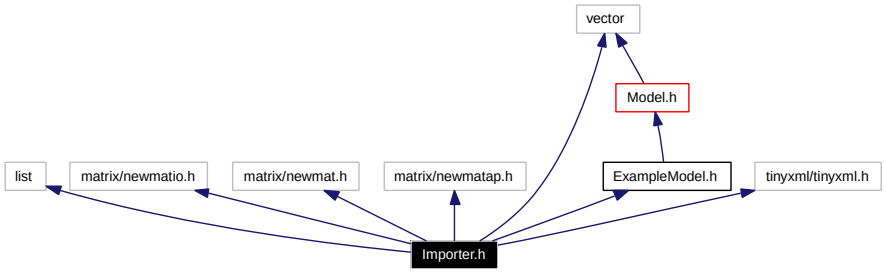
Include dependency graph for Importer.cpp:



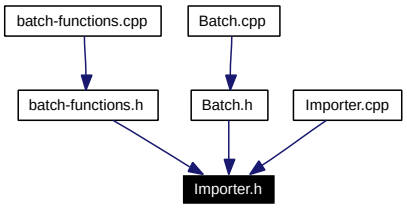
11.4 Functions/Importer.h File Reference

```
#include <list>
#include <matrix/newmatio.h>
#include <matrix/newmat.h>
#include <matrix/newmatap.h>
#include <vector>
#include <ExampleModel.h>
#include <tinyxml/tinyxml.h>
```

Include dependency graph for Importer.h:



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



Namespaces

- namespace functions

Classes

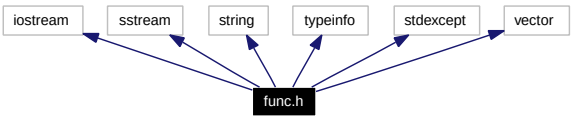
- class functions::Importer

  Write brief comment for Importer here.

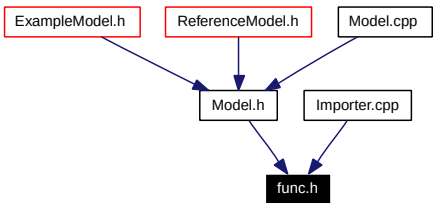
11.5 Functions/include/func.h File Reference

```
#include <iostream>
#include <sstream>
#include <string>
#include <typeinfo>
#include <stdexcept>
#include <vector>
```

Include dependency graph for func.h:



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



Classes

- class BadConversion

  Write brief comment for BadConversion here.

Functions

- std::string stringify (const char *c)

  Write brief comment for stringify here.

- template<typename T> void convert (const std::string &s, T &x, bool failIfLeftoverChars=true)

  Write brief comment for convert here.

- template<typename T> T convertTo (const std::string &s, bool failIfLeftoverChars=true)

  Write brief comment for convertTo here.

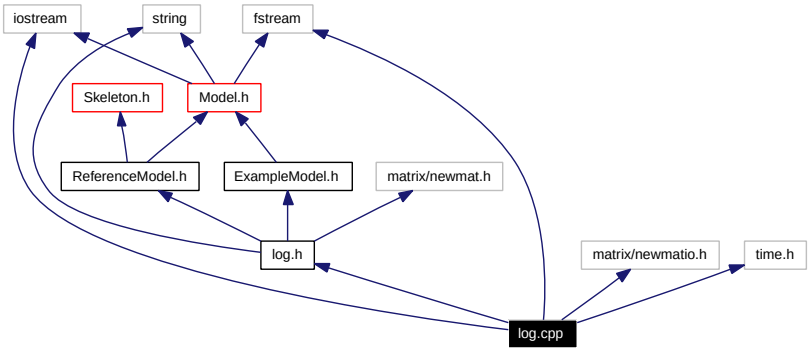
- void Tokenize (const string &str, vector< string > &tokens, const string &delimiters=" ")

  Write brief comment for Tokenize here.

11.6 Functions/log.cpp File Reference

```
#include "log.h"
#include <iostream>
#include <matrix/newmatio.h>
#include <fstream>
#include <time.h>
```

Include dependency graph for log.cpp:



Defi nes

- #define WANT_STREAM

11.6.1 Defi ne Documentation

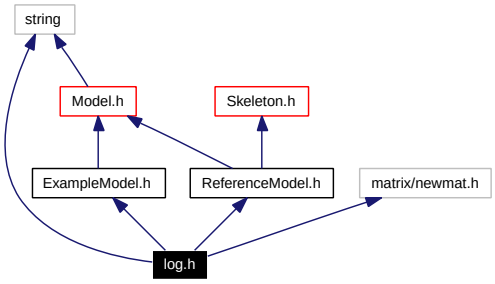
11.6.1.1 #define WANT_STREAM

Definition at line 1 of file log.cpp.

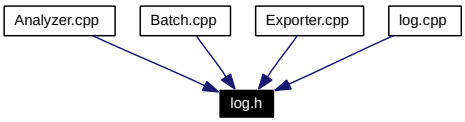
11.7 Functions/log.h File Reference

```
#include <string>
#include <matrix/newmat.h>
#include <ExampleModel.h>
#include <ReferenceModel.h>
```

Include dependency graph for log.h:



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



Classes

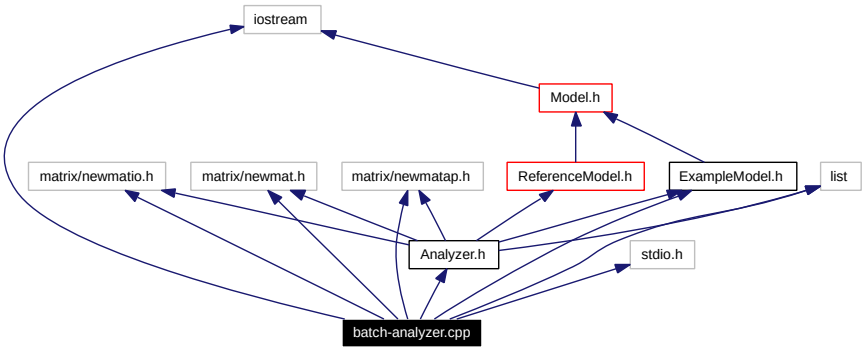
- class log

  Write brief comment for log here.

11.8 gui/batch-analyzer.cpp File Reference

```
#include <iostream>
#include <matrix/newmatio.h>
#include <matrix/newmat.h>
#include <matrix/newmatap.h>
#include <list>
#include <ExampleModel.h>
#include <Analyzer.h>
#include <stdio.h>
```

Include dependency graph for batch-analyzer.cpp:



Functions

- int main ()

11.8.1 Function Documentation

11.8.1.1 int main ()

Definition at line 11 of file batch-analyzer.cpp.

```
12 {
13   Matrix data(2, 10);
14   Real temp[] = { 2.5, 0.5, 2.2, 1.9, 3.1, 2.3, 2, 1, 1.5, 1.1, 2.4, 0.7, 2.9, 2.2, 3.0, 2.7, 1.6, 1.1,
```

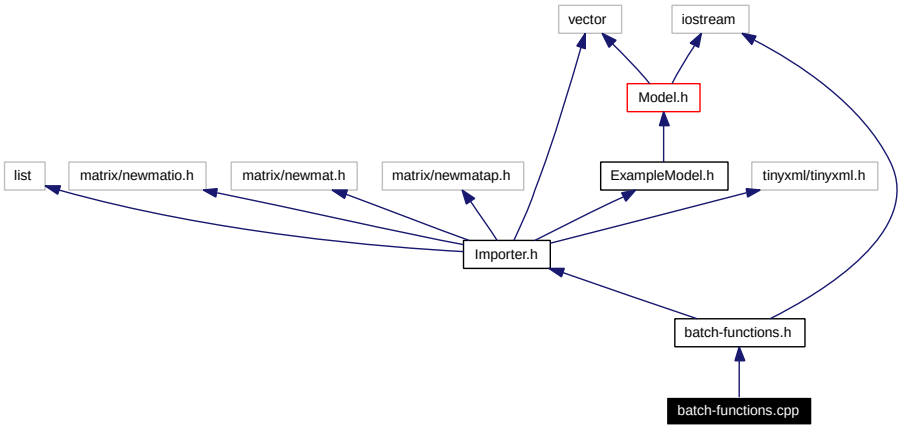
```
15   data << temp;
16
17
18   Analyzer pca;
19
20   pca.setData(data);
21   printf("setData done\n");
22   pca.mean(data);
23   printf("mean done\n");
24   pca.adjust();
25   printf("adjust done\n");
26   pca.covariance();
27   printf("covariance done\n");
28   pca.findComponents();
29   printf("finding done\n");
30   pca.computeEnergy();
31   printf("compute energy done\n");
32   pca.selectComponents();
33   printf("select components done\n");
34   pca.getTransformationMatrix();
35
36
37   //Application to test the Analyzer - should be expanded to a batch program.
38 }
```

11.9 gui/batch-analyzer.h File Reference

11.10 gui/batch-functions.cpp File Reference

```
#include "batch-functions.h"
```

Include dependency graph for batch-functions.cpp:



Functions

- int main ()

11.10.1 Function Documentation

11.10.1.1 int main ()

Definition at line 3 of file batch-functions.cpp.

References functions::Importer::importModel().

```
4 {
5         functions::Importer im = functions::Importer();
6         im.importModel("test2.x3d");
7   return 0;
8 }
```

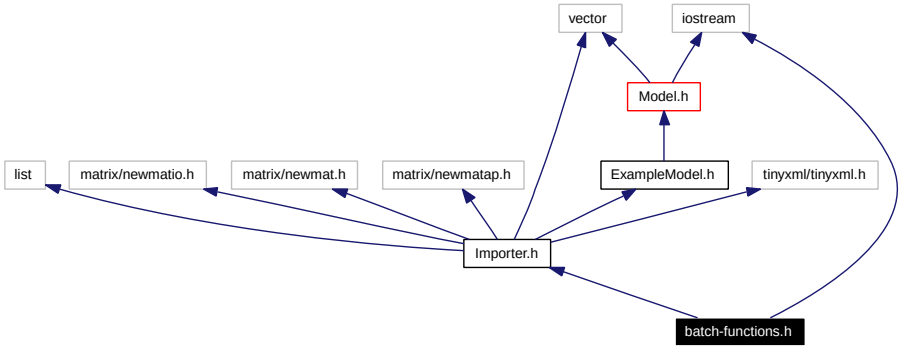
Here is the call graph for this function:

11.11 gui/batch-functions.h File Reference

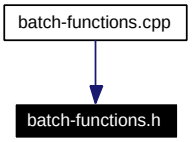
`#include <Importer.h>`

`#include <iostream>`

Include dependency graph for batch-functions.h:



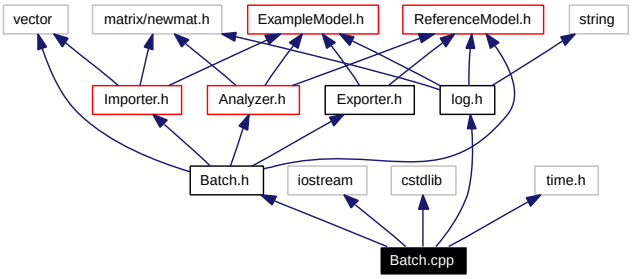
This graph shows which files directly or indirectly include this file:



11.12 gui/Batch.cpp File Reference

```
#include "Batch.h"
#include <iostream>
#include <cstdlib>
#include <log.h>
#include <time.h>
```

Include dependency graph for Batch.cpp:



Functions

- int main (int argc, char *argv[ ])

11.12.1 Function Documentation

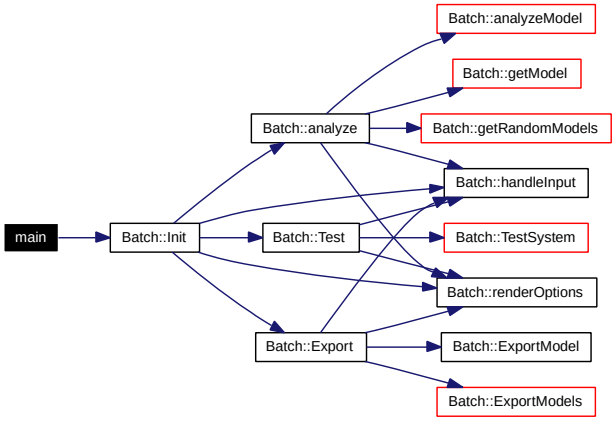
11.12.1.1 int main (int *argc*, char * *argv*[ ])

Definition at line 11 of file Batch.cpp.

References Batch::Init().

```
11                                  {
12         bool debug = false;
13         if(argc==2) {
14                 if(string(argv[1]) == "debug") debug = true;
15         }
16         Batch b = Batch(debug);
17         b.Init();
18         return 0;
19 }
```

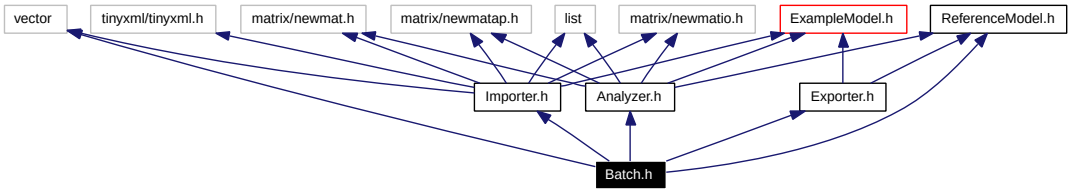
Here is the call graph for this function:



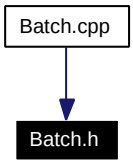
11.13 gui/Batch.h File Reference

```
#include <Importer.h>
#include <Exporter.h>
#include <Analyzer.h>
#include <vector>
#include "ReferenceModel.h"
```

Include dependency graph for Batch.h:



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



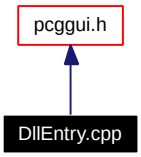
Classes

- class Batch

11.14 gui/pcggui/DllEntry.cpp File Reference

```
#include "pcggui.h"
```

Include dependency graph for DllEntry.cpp:



Functions

- ClassDesc2 * GetpcgguiDesc ()
- BOOL WINAPI DllMain (HINSTANCE hinstDLL, ULONG fdwReason, LPVOID lpvReserved)
- __declspec (dllexport) const TCHAR *LibDescription()
- TCHAR * GetString (int id)

Variables

- HINSTANCE hInstance
- int controlsInit = FALSE

11.14.1 Function Documentation

11.14.1.1 __declspec (dllexport) const

Definition at line 41 of file DllEntry.cpp.

References GetString(), and IDS_LIBDESCRIPTION.

```
42 {
43         return GetString(IDS_LIBDESCRIPTION);
44 }
```

Here is the call graph for this function:

11.14.1.2 BOOL WINAPI DllMain (HINSTANCE *hinstDLL*, ULONG *fdwReason*, LPVOID *lpvReserved*)

Definition at line 26 of file DllEntry.cpp.

References controlsInit, and hInstance.

```
27 {
28         hInstance = hinstDLL;                           // Hang on to this DLL's instance hand
29
30         if (!controlsInit) {
31                 controlsInit = TRUE;
32                 InitCustomControls(hInstance);  // Initialize MAX's custom controls
33                 InitCommonControls();                   // Initialize Win95 controls
34         }
35
36         return (TRUE);
37 }
```

11.14.1.3 ClassDesc2* GetpcgguiDesc ()

Definition at line 48 of file pcggui.cpp.

```
48 { return &pcgguiDesc; }
```

11.14.1.4 TCHAR* GetString (int *id*)

Definition at line 70 of file DllEntry.cpp.

References hInstance.

Referenced by __declspec(), pcggui::BeginEditParams(), pcgguiClassDesc::Category(), and pcgguiClassDesc::ClassName().

```
71 {
72         static TCHAR buf[256];
73
74         if (hInstance)
75                 return LoadString(hInstance, id, buf, sizeof(buf)) ? buf : NULL;
76         return NULL;
77 }
```

11.14.2 Variable Documentation

11.14.2.1 int controlsInit = FALSE

Definition at line 18 of file DllEntry.cpp.

Referenced by DllMain().

11.14.2.2 HINSTANCE hInstance

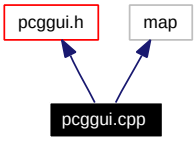
Definition at line 17 of file DllEntry.cpp.

Referenced by pcggui::BeginEditParams(), DllMain(), and GetString().

11.15 gui/pcggui/pcggui.cpp File Reference

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

Include dependency graph for pcggui.cpp:



Classes

- class pcgguiClassDesc
- class NullView

Defi nes

- #define pcggui_CLASS_ID Class_ID(0x67910a43, 0xc56b6dbc)
- #define SKIN_INTERFACE 0x00010000
- #define SKIN_CLASS_ID Class_ID(9815843,87654)
- #define PHYSIQUE_CLASS_ID Class_ID(PHYSIQUE_CLASS_ID_A, PHYSIQUE_CLASS_ID_B)

Typedefs

- typedef signed int sint32
- typedef std::map< INode ∗, uint > TInodePtrInt

Functions

- ClassDesc2 ∗ GetpcgguiDesc ()
- BOOL CALLBACK pcgguiDlgProc (HWND hWnd, UINT msg, WPARAM w-Param, LPARAM lParam)

Variables

- pcggui thepcggui
- FILE ∗ file
- pcgguiClassDesc pcgguiDesc

11.15.1 Define Documentation

11.15.1.1 #define pcggui_CLASS_ID Class_ID(0x67910a43, 0xc56b6dbc)

Definition at line 17 of file pcggui.cpp.

11.15.1.2 #define PHYSIQUE_CLASS_ID Class_ID(PHYSIQUE_CLASS_ID_A, PHYSIQUE_CLASS_ID_B)

Definition at line 24 of file pcggui.cpp.

11.15.1.3 #define SKIN_CLASS_ID Class_ID(9815843,87654)

Definition at line 23 of file pcggui.cpp.

Referenced by pcggui::ConvertSkin().

11.15.1.4 #define SKIN_INTERFACE 0x00010000

Definition at line 22 of file pcggui.cpp.

Referenced by pcggui::ConvertSkin().

11.15.2 Typedef Documentation

11.15.2.1 typedef signed int sint32

Definition at line 26 of file pcggui.cpp.

11.15.2.2 typedef std::map<INode*, uint> TInodePtrInt

Definition at line 27 of file pcggui.cpp.

11.15.3 Function Documentation

11.15.3.1 ClassDesc2* GetpcgguiDesc ()

Definition at line 48 of file pcggui.cpp.

```
48 { return &pcgguiDesc; }
```

11.15.3.2 BOOL CALLBACK pcgguiDlgProc (HWND *hWnd*, UINT *msg*, WPARAM *wParam*, LPARAM *lParam*) [static]

Definition at line 53 of file pcggui.cpp.

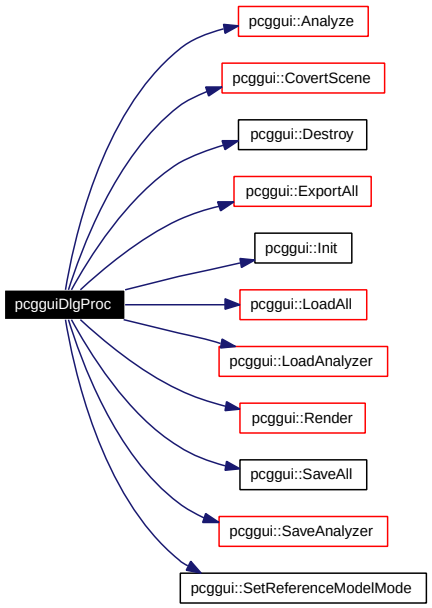
References pcggui::Analyze(), pcggui::CovertScene(), pcggui::Destroy(), pcggui::ExportAll(), IDC_BUTTON_ANALYZE, IDC_BUTTON_CONVERT, IDC_BUTTON_EXPORT, IDC_BUTTON_LOAD, IDC_BUTTON_LOAD_ANALYZER, IDC_BUTTON_RENDER, IDC_BUTTON_SAVE, IDC_BUTTON_SAVE_ANALYZER, IDC_REFERENCE_MODEL, pcggui::Init(), pcggui::ip, pcggui::LoadAll(), pcggui::LoadAnalyzer(), pcggui::Render(), pcggui::SaveAll(), pcggui::SaveAnalyzer(), pcggui::SetReferenceModelMode(), and thepcggui.

Referenced by pcggui::BeginEditParams().

```
55 {
56         switch (msg) {
57                 case WM_INITDIALOG:
58                         thepcggui.Init(hWnd);
59                         break;
60
61                 case WM_DESTROY:
62                         thepcggui.Destroy(hWnd);
63                         break;
64
65                 case WM_COMMAND:
66                         switch(LOWORD(wParam)) { //switch on ID
67                                 case IDC_BUTTON_CONVERT: // A specific button's ID.
68                                         thepcggui.CovertScene();
69                                         break;
70
71                                 case IDC_REFERENCE_MODEL:
72                                         thepcggui.SetReferenceModelMode(IsDlgButtonChecked(hW
73                                         break;
74
75                                 case IDC_BUTTON_EXPORT:
76                                         thepcggui.ExportAll();
77                                         break;
78
79                                 case IDC_BUTTON_SAVE:
80                                         thepcggui.SaveAll();
81                                         break;
82
83                                 case IDC_BUTTON_LOAD:
84                                         thepcggui.LoadAll();
85                                         break;
86
87                                 case IDC_BUTTON_SAVE_ANALYZER:
88                                         thepcggui.SaveAnalyzer();
```

```
89                                      break;
90
91                                  case IDC_BUTTON_LOAD_ANALYZER:
92                                      thepcggui.LoadAnalyzer();
93                                      break;
94
95                                  case IDC_BUTTON_ANALYZE:
96                                      thepcggui.Analyze();
97                                      break;
98
99                                  case IDC_BUTTON_RENDER:
100                                     thepcggui.Render();
101                                     break;
102
103                                 default:
104                                     break;
105                         };
106                         break;
107
108
109                 case WM_LBUTTONDOWN:
110                 case WM_LBUTTONUP:
111                 case WM_MOUSEMOVE:
112                         thepcggui.ip->RollupMouseMessage(hWnd,msg,wParam,lParam);
113                         break;
114
115                 default:
116                         return FALSE;
117         }
118         return TRUE;
119 }
```

Here is the call graph for this function:



11.15.4 Variable Documentation

11.15.4.1 FILE* file [static]

Definition at line 20 of file pcggui.cpp.

Referenced by pcggui::BeginEditParams(), pcggui::Destroy(), pcggui::EndEditParams(), and pcggui::Init().

11.15.4.2 pcgguiClassDesc pcgguiDesc [static]

Definition at line 47 of file pcggui.cpp.

11.15.4.3 pcggui thepcggui [static]

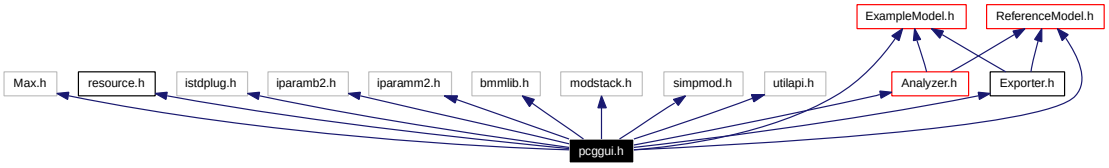
Definition at line 19 of file pcggui.cpp.

Referenced by pcggui::CovertScene(), and pcgguiDlgProc().

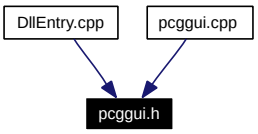
11.16 gui/pcggui/pcggui.h File Reference

```
#include "Max.h"
#include "resource.h"
#include "istdplug.h"
#include "iparamb2.h"
#include "iparamm2.h"
#include "bmmlib.h"
#include "modstack.h"
#include "simpmod.h"
#include "utilapi.h"
#include <ExampleModel.h>
#include <ReferenceModel.h>
#include <Exporter.h>
#include <Analyzer.h>
```

Include dependency graph for pcggui.h:



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



Classes

- class pcggui
- class ISkinContextData
- class ISkin

Typedefs

- typedef unsigned int uint

Functions

- TCHAR * GetString (int id)

Variables

- HINSTANCE hInstance

11.16.1 Typedef Documentation

11.16.1.1 typedef unsigned int uint

Definition at line 35 of file pcggui.h.

Referenced by pcggui::ConvertSkin().

11.16.2 Function Documentation

11.16.2.1 TCHAR* GetString (int *id*)

Definition at line 70 of file DllEntry.cpp.

References hInstance.

Referenced by __declspec(), pcggui::BeginEditParams(), pcgguiClassDesc::Category(), and pcgguiClassDesc::ClassName().

```
71 {
72         static TCHAR buf[256];
73
74         if (hInstance)
75                 return LoadString(hInstance, id, buf, sizeof(buf)) ? buf : NULL;
76         return NULL;
77 }
```

11.16.3 Variable Documentation

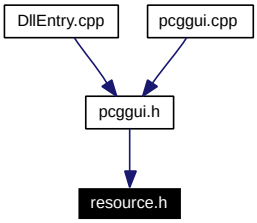
11.16.3.1 HINSTANCE hInstance

Definition at line 17 of file DllEntry.cpp.

Referenced by pcggui::BeginEditParams(), DllMain(), and GetString().

11.17 gui/pcggui/resource.h File Reference

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



Defines

- #define IDS_LIBDESCRIPTION 1
- #define IDS_CATEGORY 2
- #define IDS_CLASS_NAME 3
- #define IDS_PARAMS 4
- #define IDS_SPIN 5
- #define IDS_CONVERT 5
- #define IDD_PANEL 101
- #define IDC_CLOSEBUTTON 1000
- #define IDC_DOSTUFF 1000
- #define IDC_BUTTON_CONVERT 1001
- #define IDC_REFERENCE_MODEL 1004
- #define IDC_BUTTON_EXPORT 1005
- #define IDC_BUTTON_SAVE 1006
- #define IDC_BUTTON_LOAD 1008
- #define IDC_BUTTON_SAVE_ANALYZER 1009
- #define IDC_BUTTON_ANALYZE 1010
- #define IDC_BUTTON_LOAD_ANALYZER 1011
- #define IDC_BUTTON1 1012
- #define IDC_BUTTON_RENDER 1012
- #define IDC_COLOR 1456
- #define IDC_EDIT 1490
- #define IDC_SPIN 1496

11.17.1 Define Documentation

11.17.1.1 #define IDC_BUTTON1 1012

Definition at line 22 of file resource.h.

11.17.1.2 #define IDC_BUTTON_ANALYZE 1010

Definition at line 20 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.3 #define IDC_BUTTON_CONVERT 1001

Definition at line 14 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.4 #define IDC_BUTTON_EXPORT 1005

Definition at line 16 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.5 #define IDC_BUTTON_LOAD 1008

Definition at line 18 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.6 #define IDC_BUTTON_LOAD_ANALYZER 1011

Definition at line 21 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.7 #define IDC_BUTTON_RENDER 1012

Definition at line 23 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.8 #define IDC_BUTTON_SAVE 1006

Definition at line 17 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.9 #define IDC_BUTTON_SAVE_ANALYZER 1009

Definition at line 19 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.10 #define IDC_CLOSEBUTTON 1000

Definition at line 12 of file resource.h.

11.17.1.11 #define IDC_COLOR 1456

Definition at line 24 of file resource.h.

11.17.1.12 #define IDC_DOSTUFF 1000

Definition at line 13 of file resource.h.

11.17.1.13 #define IDC_EDIT 1490

Definition at line 25 of file resource.h.

11.17.1.14 #define IDC_REFERENCE_MODEL 1004

Definition at line 15 of file resource.h.

Referenced by pcgguiDlgProc().

11.17.1.15 #define IDC_SPIN 1496

Definition at line 26 of file resource.h.

11.17.1.16 #define IDD_PANEL 101

Definition at line 11 of file resource.h.

Referenced by pcggui::BeginEditParams().

11.17.1.17 #define IDS_CATEGORY 2

Definition at line 6 of file resource.h.

Referenced by pcgguiClassDesc::Category().

11.17.1.18 #define IDS_CLASS_NAME 3

Definition at line 7 of file resource.h.

Referenced by pcgguiClassDesc::ClassName().

11.17.1.19 #define IDS_CONVERT 5

Definition at line 10 of file resource.h.

11.17.1.20 #define IDS_LIBDESCRIPTION 1

Definition at line 5 of file resource.h.

Referenced by __declspec().

11.17.1.21 #define IDS_PARAMS 4

Definition at line 8 of file resource.h.

Referenced by pcggui::BeginEditParams().

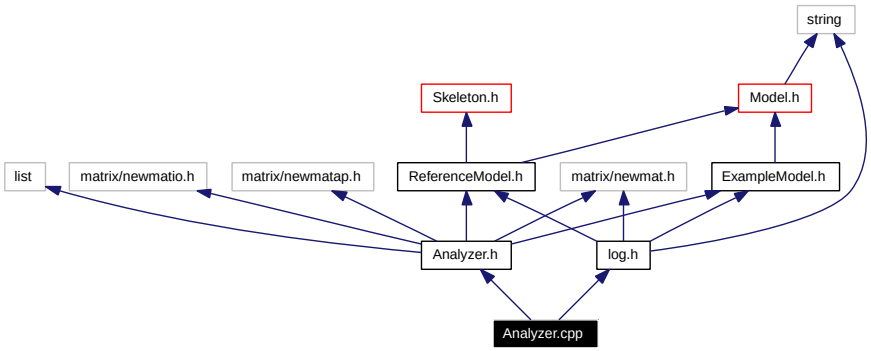
11.17.1.22 #define IDS_SPIN 5

Definition at line 9 of file resource.h.

11.18 model/Analyzer.cpp File Reference

```
#include "Analyzer.h"
#include <log.h>
```

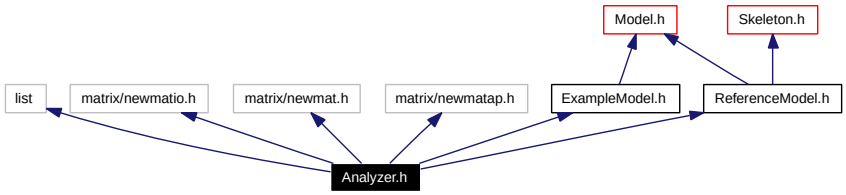
Include dependency graph for Analyzer.cpp:



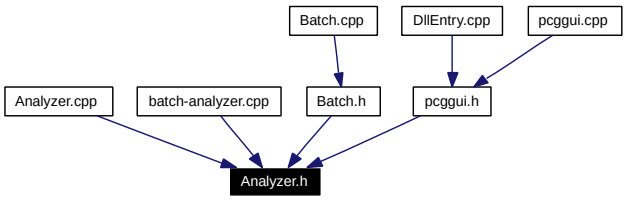
11.19 model/Analyzer.h File Reference

```
#include <list>
#include <matrix/newmatio.h>
#include <matrix/newmat.h>
#include <matrix/newmatap.h>
#include "ExampleModel.h"
#include "ReferenceModel.h"
```

Include dependency graph for Analyzer.h:



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



Namespaces

- namespace model
- namespace std

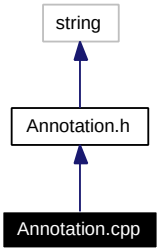
Classes

- class model::Analyzer

11.20 model/Annotation.cpp File Reference

```
#include "Annotation.h"
```

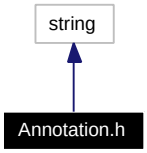
Include dependency graph for Annotation.cpp:



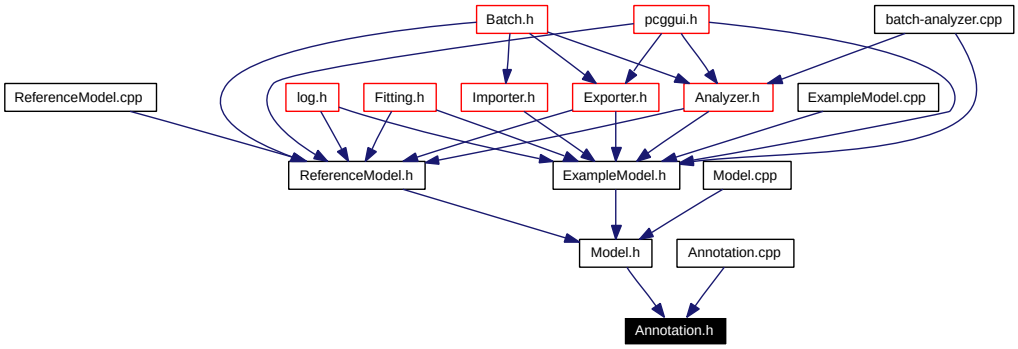
11.21 model/Annotation.h File Reference

`#include <string>`

Include dependency graph for Annotation.h:



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



Namespaces

- namespace model

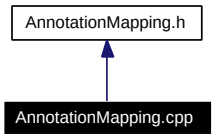
Classes

- class model::Annotation

11.22 model/AnnotationMapping.cpp File Reference

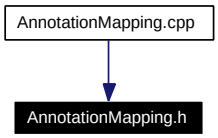
```
#include "AnnotationMapping.h"
```

Include dependency graph for AnnotationMapping.cpp:



11.23 model/AnnotationMapping.h File Reference

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



Namespaces

- namespace model

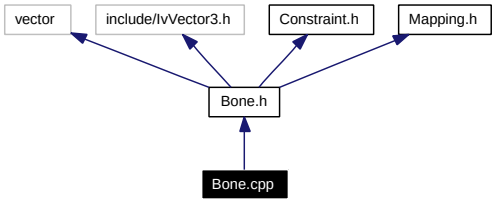
Classes

- class model::AnnotationMapping

11.24 model/Bone.cpp File Reference

```
#include "Bone.h"
```

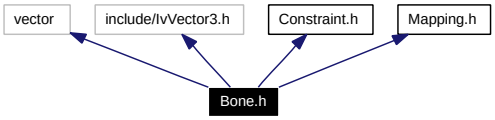
Include dependency graph for Bone.cpp:



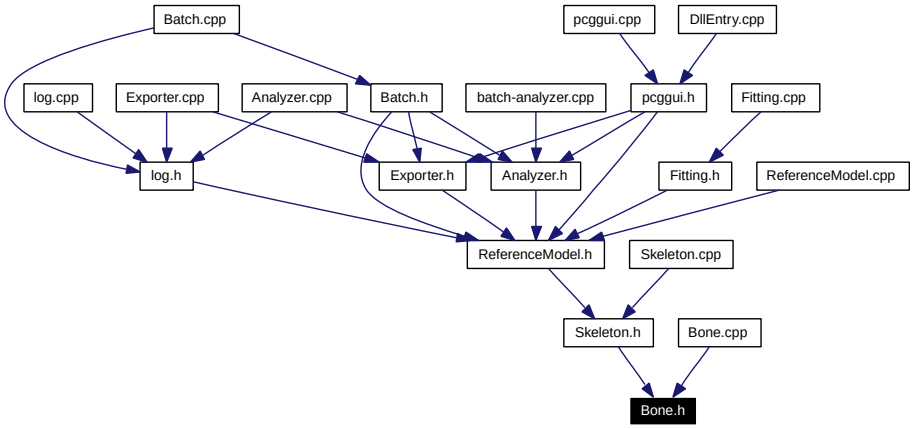
11.25 model/Bone.h File Reference

```
#include <vector>
#include "include/IvVector3.h"
#include "Constraint.h"
#include "Mapping.h"
```

Include dependency graph for Bone.h:



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



Namespaces

- namespace model

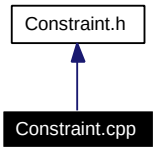
Classes

- class model::Bone

11.26 model/Constraint.cpp File Reference

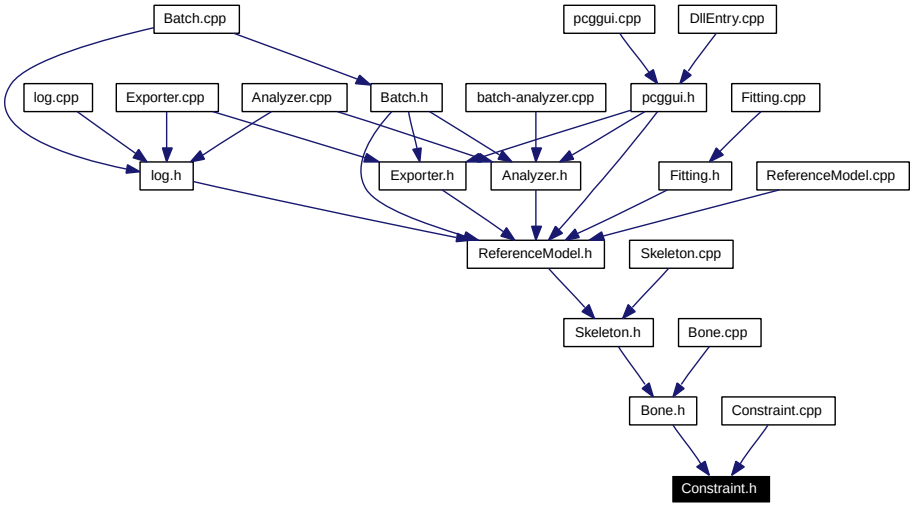
```
#include "Constraint.h"
```

Include dependency graph for Constraint.cpp:



11.27 model/Constraint.h File Reference

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



Namespaces

- namespace model

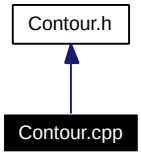
Classes

- class model::Constraint

11.28 model/Contour.cpp File Reference

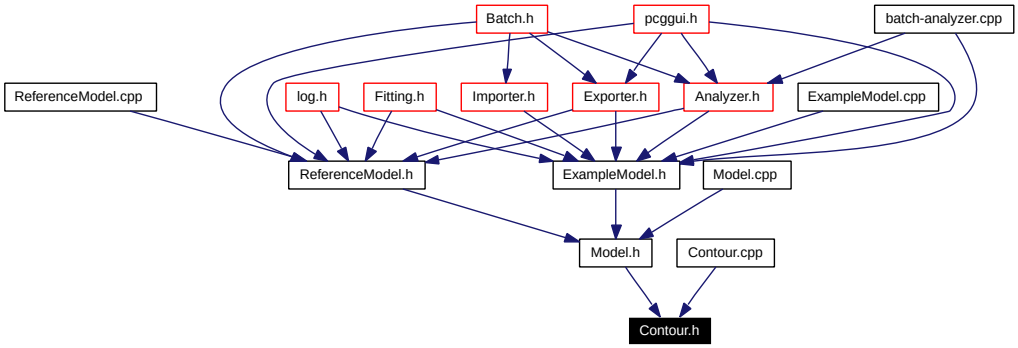
```
#include "Contour.h"
```

Include dependency graph for Contour.cpp:



11.29 model/Contour.h File Reference

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



Namespaces

- namespace model

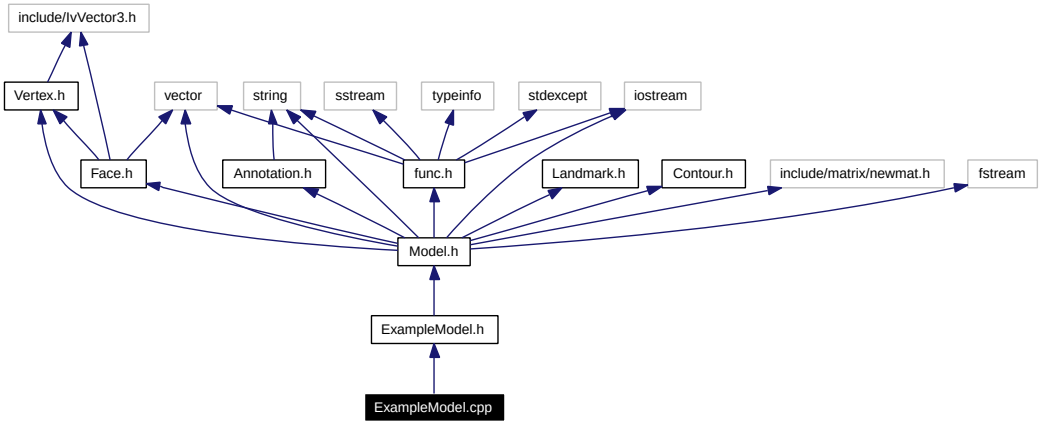
Classes

- class model::Contour

11.30 model/ExampleModel.cpp File Reference

```
#include "ExampleModel.h"
```

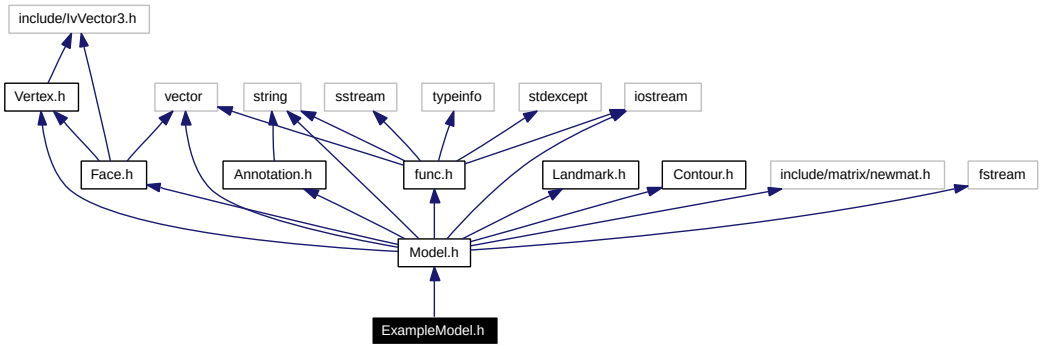
Include dependency graph for ExampleModel.cpp:



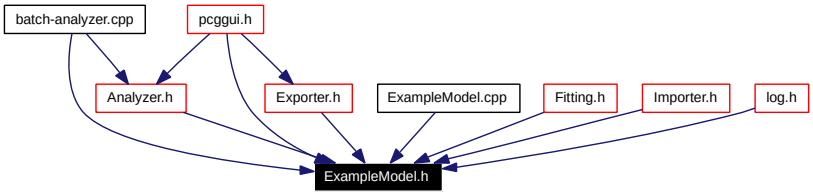
11.31 model/ExampleModel.h File Reference

```
#include "Model.h"
```

Include dependency graph for ExampleModel.h:



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



Namespaces

- namespace model

Classes

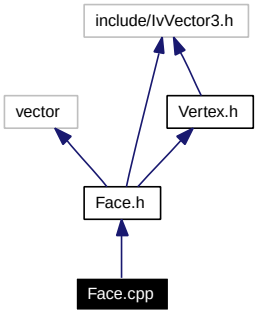
- class model::ExampleModel

  An ExampleModel in PCG.

11.32 model/Face.cpp File Reference

```
#include "Face.h"
```

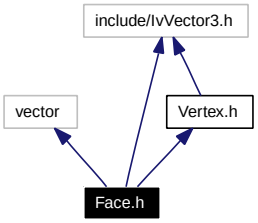
Include dependency graph for Face.cpp:



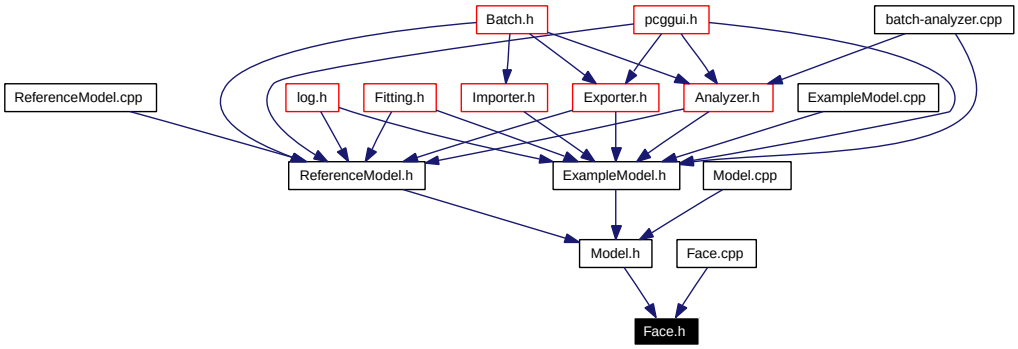
11.33 model/Face.h File Reference

```
#include <vector>
#include "include/IvVector3.h"
#include "Vertex.h"
```

Include dependency graph for Face.h:



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



Namespaces

- namespace model

Classes

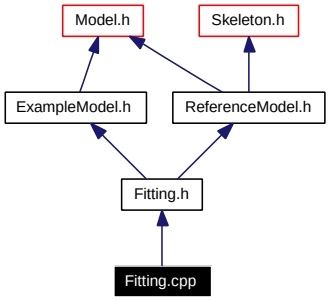
- class model::Face

  A Face class.

11.34 model/Fitting.cpp File Reference

```
#include "Fitting.h"
```

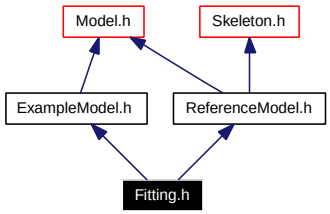
Include dependency graph for Fitting.cpp:



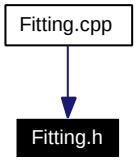
11.35 model/Fitting.h File Reference

```
#include "ExampleModel.h"
#include "ReferenceModel.h"
```

Include dependency graph for Fitting.h:



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



Namespaces

- namespace model

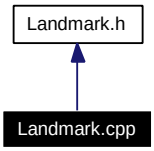
Classes

- class model::Fitting

11.36 model/Landmark.cpp File Reference

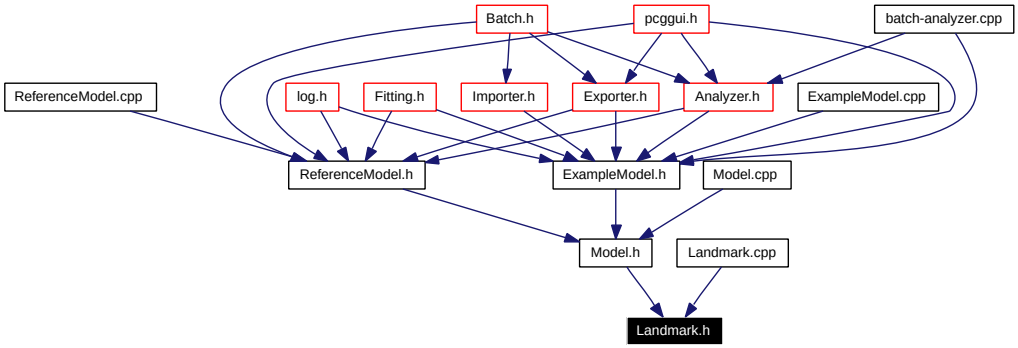
```
#include "Landmark.h"
```

Include dependency graph for Landmark.cpp:



11.37 model/Landmark.h File Reference

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



Namespaces

- namespace model

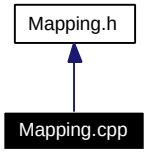
Classes

- class model::Landmark

11.38 model/Mapping.cpp File Reference

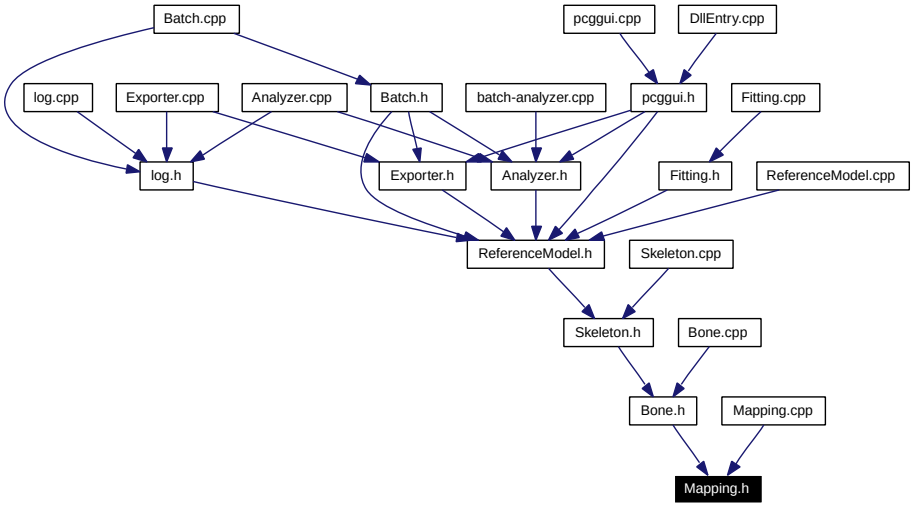
```
#include "Mapping.h"
```

Include dependency graph for Mapping.cpp:



11.39 model/Mapping.h File Reference

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



Namespaces

- namespace model

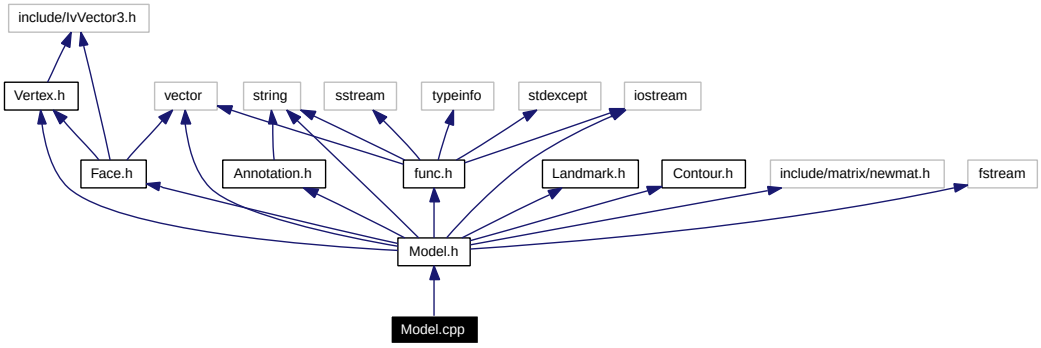
Classes

- class model::Mapping

11.40 model/Model.cpp File Reference

```
#include "Model.h"
```

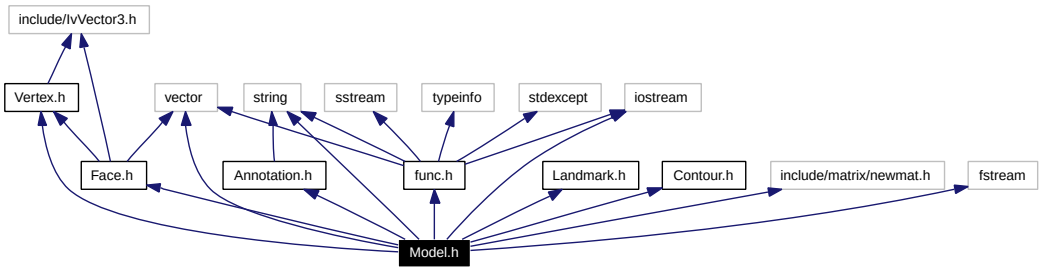
Include dependency graph for Model.cpp:



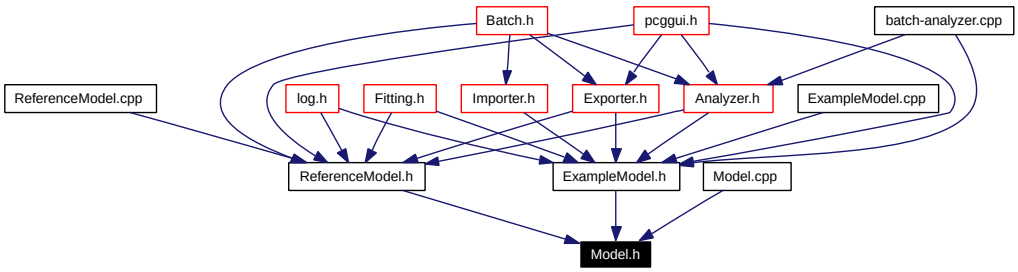
11.41 model/Model.h File Reference

```
#include "Vertex.h"
#include "Face.h"
#include "Landmark.h"
#include "Contour.h"
#include "Annotation.h"
#include <vector>
#include "include/matrix/newmat.h"
#include <iostream>
#include <fstream>
#include <string>
#include <func.h>
```

Include dependency graph for Model.h:



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



Namespaces

- namespace model

Classes

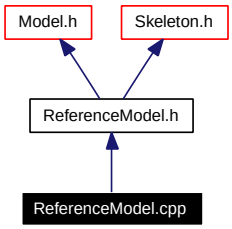
- class model::Model

  A abstract class of a Model in PCG.

11.42 model/ReferenceModel.cpp File Reference

```
#include "ReferenceModel.h"
```

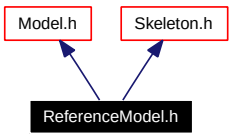
Include dependency graph for ReferenceModel.cpp:



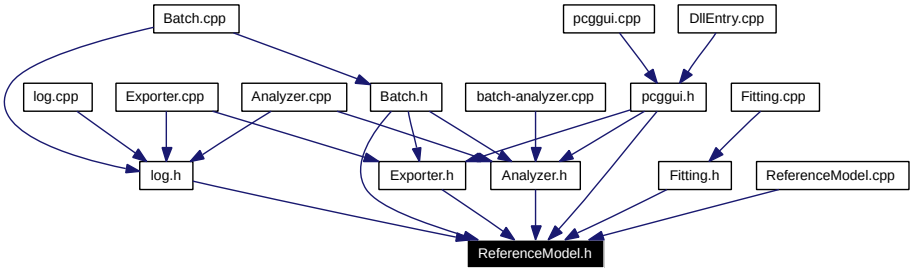
11.43 model/ReferenceModel.h File Reference

```
#include "Model.h"
#include "Skeleton.h"
```

Include dependency graph for ReferenceModel.h:



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



Namespaces

- namespace model

Classes

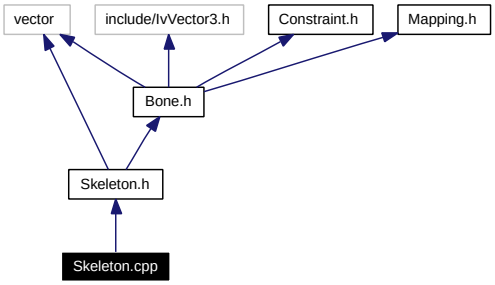
- class model::ReferenceModel

  An ReferenceModel in PCG.

11.44 model/Skeleton.cpp File Reference

```
#include "Skeleton.h"
```

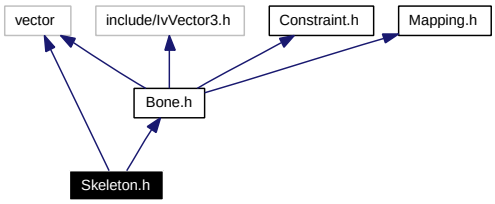
Include dependency graph for Skeleton.cpp:



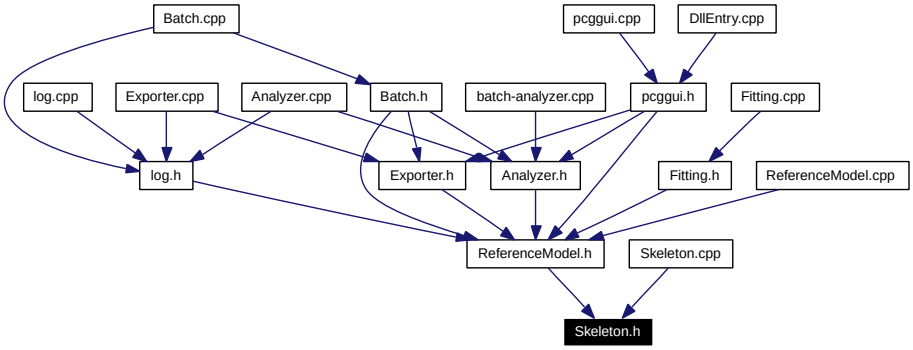
11.45 model/Skeleton.h File Reference

```
#include "Bone.h"
#include <vector>
```

Include dependency graph for Skeleton.h:



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



Namespaces

- namespace model

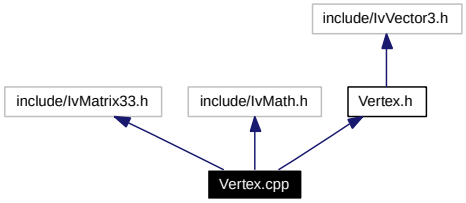
Classes

- class model::Skeleton

11.46 model/Vertex.cpp File Reference

```
#include "include/IvMatrix33.h"
#include "include/IvMath.h"
#include "Vertex.h"
```

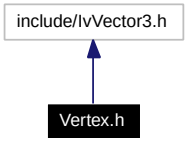
Include dependency graph for Vertex.cpp:



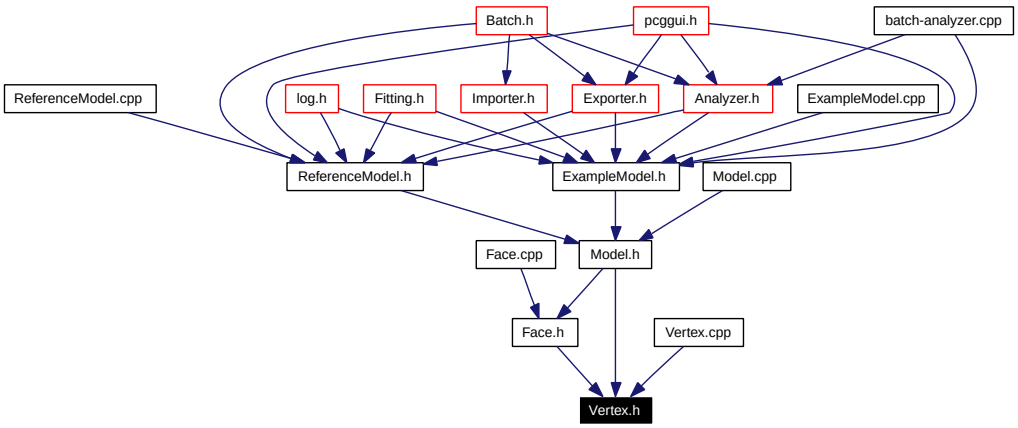
11.47 model/Vertex.h File Reference

`#include "include/IvVector3.h"`

Include dependency graph for Vertex.h:



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



Namespaces

- namespace model

Classes

- class model::Vertex

  A Vertex class.

Index