MDTL Aplication Programming Interface User Reference Manual

Pere Pau Vázquez

Report LSI–99–3–T

MDTL Aplication Programming Interface - User Reference Manual

Pere Pau Vázquez

January 14, 1999

Abstract

MDTL is a textual language created by the Computer Graphics Section of the Language and Information Systems of the Universitat Politècnica de Catalunya.

The Computer Graphics Section has several research groups which use their own format to store the information of the generated scenes. Some times it is needed by someone out of one of the groups to access to the information of these files. That's what MDTL is supposed to do.

MDTL Aplication Programming Interface is a set of classes written in C++ that will allow you to access (i. e. read and write) files written in MDTL textual language.

This documentation is a reference to the use of the classes.

Contents

1

MDTL Aplication Programming Interface

MDTL API Reference Manual

MDTL Aplication Programming Interface Reference Manual

MDTL Aplication Programming Interface is a set of classes written in C++ that will allow you to access (i. e. read and write) files written in MDTL textual language.

This documentation is a reference to the use of the classes. A couple of examples of the use is also provided.

Author: Pere Pau Vázquez
Version: 1.0

2

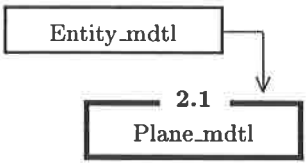
Names

2.1

class **Plane_mdtl** : public Entity_mdtl

Stores data needed by a plane

Inheritance



Public Members

2.1.1

Plane_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

2.1.2

Plane_mdtl (PPString name, double a, double b, double c, double d)

Constructor

Constructor.
Pre: True
Post: ma = a && mb = b && mc = c && md = d && mName = name

2.1.3

Plane_mdtl (double a, double b, double c, double d)

Constructor

Constructor.
Pre: True
Post: ma = a && mb = b && mc = c && md = d

2.1.4

~Plane_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

2.1.5

Plane_mdtl (const Plane_mdtl& PlaneToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = PlaneToCopy

2.1.6

Plane_mdtl& **operator=** (const Plane_mdtl& PlaneToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = PlaneToCopy

2.1.7

void **data** (double a, double b, double c, double d)

Sets values of members

Sets values of members.
Pre: True
Post: ma = a && mb = b && mc = c && md = d

2.1.8

void **data** (PPString name, double a, double b, double c, double d)

Sets values of members

Sets values of members.
Pre: True
Post: ma = a && mb = b && mc = c && md = d && mName = name

2.1.9

double **GetA** ()

Returns value of member A

Returns value of member A.
Pre: True
Post: Returns A

2.1.10

double **GetB** ()

Returns value of member B

Returns value of member B.
Pre: True
Post: Returns B

2.1.11

double **GetC** ()

Returns value of member C

Returns value of member C.
Pre: True
Post: Returns C

2.1.12

double **GetD** ()

Returns value of member D

Returns value of member D.
Pre: True
Post: Returns D

2.1.13

friend ostream& **operator<<** (ostream&, const Plane_mdtl& plane)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Plane_mdtl object in the output file.

2.1.14

friend ostream& **operator<<=** (ostream&, const Plane_mdtl& plane)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Plane_mdtl object in the output file.

3

Names

3.1

class Entity_mdtl

This class contains the names of the entities

Public Members

3.1.1 Entity_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

3.1.2 Entity_mdtl (PPString name)

Constructor

Constructor.
Pre: True Post: mname = name

3.1.3 Entity_mdtl (const Entity_mdtl& EntityToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = EntityToCopy

3.1.4 ~Entity_mdtl ()

Destructor

Destructor. Pre: True Post: *this = ?

3.1.5

void **SetName** (PPString name)

Sets value of member

Sets value of member.
Pre: True
Post: mname = name

3.1.6

PPString **GetName** ()

Returns value of member mname

Returns value of member mname.
Pre: True
Post: Returns mname

3.1.7

Entity_mdtl& **operator=** (const Entity_mdtl& EntityToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = EntityToCopy

3.1.8

friend ostream& **operator<<** (ostream&, const Entity_mdtl&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Writes mname to output file.

4

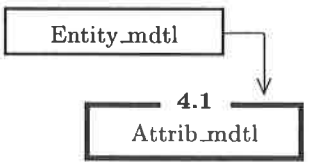
Names

4.1

class **Attrib_mdtl** : public Entity_mdtl

Stores the data referred to attributes that an entity in a MDTL file may have

Inheritance



Public Members

4.1.1

Attrib_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

4.1.2

Attrib_mdtl (PPString n)

Constructor

Constructor.
Pre: True Post: mName = n

4.1.3

Attrib_mdtl (const Attrib_mdtl& AtToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = AtToCopy

4.1.4

~Attrib_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

4.1.5

list_mdtl <MaterialS_mdtl> & **GetMatsS** ()

Returns the list of surface materials

Returns the list of surface materials.
Pre: True
Post: Returns a list of MaterialS_mdtl objects.

4.1.6

list_mdtl <TextAttribMdtl> & **GetAttrTexts** ()

Returns the list of textual attributes

Returns the list of textual attributes.
Pre: True
Post: Returns a list of TextAttribMdtl objects.

4.1.7

void **AddMatS** (const MaterialS_mdtl& m)

Adds a new material to the list of surface materials

Adds a new material to the list of surface materials.
Pre: True
Post: this->mMatsS.add(m).

4.1.8

void **AddAttrText** (const TextAttribMdtl& m)

Adds a new textual attribute to the list of textual attributes

Adds a new textual attribute to the list of textual attributes.
Pre: True
Post: this->mAttrTexts.add(m).

4.1.9

void **SetName** (PPString name)

Sets the name of the entity

Sets the name of the entity.
Pre: True
Post: mname = name

4.1.10

Attrib_mdtl& **operator=** (const Attrib_mdtl& AtToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = AtToCopy

4.1.11

friend ostream& **operator<<** (ostream&, const Attrib_mdtl& at)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Attrib_mdtl object in the output file.

4.1.12

friend ostream& **operator<<=** (ostream&, const Attrib_mdtl& at)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Attrib_mdtl object in the output file.

5

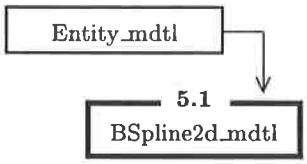
Names

5.1

class **BSpline2d_mdtl** : public Entity_mdtl

This class stores the data needed by a BSpline2d

Inheritance



Public Members

5.1.1

BSpline2d_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

5.1.2

BSpline2d_mdtl (const PPString& Name)

Constructor

Constructor.
Pre: True
Post: mName = name

5.1.3

BSpline2d_mdtl (const PPString& Name, const Knot_mdtl& Knots, const CPoints2d_mdtl& CPoints)

Constructor

Constructor.
Pre: True
Post: mKnots = Knots && mCPoints2d = CPoints && mName = name

5.1.4

~BSpline2d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

5.1.5

BSpline2d_mdtl (const BSpline2d_mdtl& BSplineToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = BSplineToCopy

5.1.6

void **SetValue** (double Value)

Sets the value of the BSpline

Sets the value of the BSpline.
Pre: True
Post: mname = name

5.1.7

void **SetKnots** (const Knot_mdtl& Knots)

Sets the knots list of the BSpline

Sets the knots list of the BSpline.
Pre: True
Post: mKnots = Knots

5.1.8

void **SetCPoints** (const CPoints2d_mdtl& CPoints)

Sets the control points of the BSpline

Sets the control points of the BSpline.
Pre: True
Post: mCPoints2d = CPoints

5.1.9

double **GetValue** ()

Returns the value of the BSpline

Returns the value of the BSpline.
Pre: True
Post: Returns a value.

5.1.10

const Knot_mdtl& **GetKnots** ()

Returns the knots list of the BSpline

Returns the knots list of the BSpline.
Pre: True
Post: Returns mKnots.

5.1.11

const CPoints2d_mdtl& **GetCPoints** ()

Returns the control points of the BSpline

Returns the control points of the BSpline.
Pre: True
Post: Returns mCPoints2d.

5.1.12

BSpline2d_mdtl& **operator=** (const BSpline2d_mdtl& BSplineToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = BSplineToCopy

5.1.13

friend ostream& **operator<<** (ostream& output, const BSpline2d_mdtl& BSpline)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new BSpline2d_mdtl object in the output file.

5.1.14

friend ostream& **operator<<=** (ostream& output, const BSpline2d_mdtl& BSpline)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new BSpline2d_mdtl object in the output file.

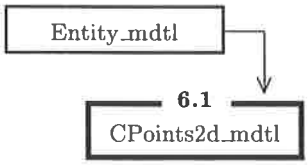
6

Names

6.1 class CPoints2d_mdtl : public Entity_mdtl

Control points of needed by a BSpline2d

Inheritance



Public Members

6.1.1

CPoints2d_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

6.1.2

CPoints2d_mdtl (list_mdtl< list_mdtl <Point2d_mdtl> > Mat)

Constructor

Constructor.
Pre: True
Post: mMat = Mat

6.1.3

CPoints2d_mdtl (PPString name, list_mdtl< list_mdtl <Point2d_mdtl> > Mat)

Constructor

Constructor.
Pre: True
Post: mMat = Mat && mname = name

6.1.4

~CPoints2d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

6.1.5

CPoints2d_mdtl (const CPoints2d_mdtl& CPointsToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = CPointsToCopy

6.1.6

void **AddRow** (list_mdtl <Point2d_mdtl> row)

Adds a new row of Point2d_mdtl to the list of rows of Point2d_mdtl objects

Adds a new row of Point2d_mdtl to the list of rows of Point2d_mdtl objects.
Pre: True
Post: this->mMat.add(row).

6.1.7

list_mdtl <Point2d_mdtl> **GetRow** (int i)

Gets the ith row of Point2d_mdtl of the list of rows

Gets the ith row of Point2d_mdtl of the list of rows.
Pre: True
Post: Returns the ith row of Point2d_mdtl objects.

6.1.8

list_mdtl < list_mdtl <Point2d_mdtl> > **GetMat** ()

Gets the list of rows of Point2d_mdtl

Gets the list of rows of Point2d_mdtl.
Pre: True
Post: Returns the list of rows.

6.1.9

CPoints2d_mdtl& **operator=** (const CPoints2d_mdtl& CPointsToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = CPointsToCopy

6.1.10

friend ostream& **operator<<** (ostream& output, const CPoints2d_mdtl& CPoints)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a newl CPoints2d_mdtl object in the output file.

6.1.11

friend ostream& **operator<<=** (ostream& output, const CPoints2d_mdtl& CPoints)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new CPoints2d_mdtl object in the output file.

7

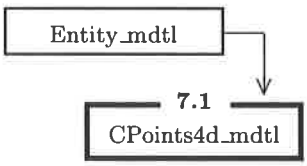
Names

7.1

class **CPoints4d_mdtl** : public Entity_mdtl

Control points of needed by a Nurb

Inheritance



Public Members

7.1.1

CPoints4d_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

7.1.2

CPoints4d_mdtl (list_mdtl < list_mdtl <Point4d_mdtl> > Mat)

Constructor

Constructor.
Pre: True
Post: mMat = Mat

7.1.3

CPoints4d_mdtl (PPString name, list_mdtl < list_mdtl <Point4d_mdtl> > Mat)

Constructor

Constructor.
Pre: True
Post: mMat = Mat && mname = name

7.1.4

~CPoints4d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

7.1.5

CPoints4d_mdtl (const CPoints4d_mdtl& CPointsToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = CPointsToCopy

7.1.6

void **AddRow** (list_mdtl < Point4d_mdtl > row)

Adds a new row of Point4d_mdtl to the list of rows of Point4d_mdtl objects

Adds a new row of Point4d_mdtl to the list of rows of Point4d_mdtl objects.
Pre: True
Post: this->mMat.add(row).

7.1.7

list_mdtl <Point4d_mdtl> **GetRow** (int i)

Gets the ith row of Point4d_mdtl of the list of rows

Gets the ith row of Point4d_mdtl of the list of rows.
Pre: True
Post: Returns the ith row of Point4d_mdtl objects.

7.1.8

list_mdtl < list_mdtl <Point4d_mdtl> > **GetMat** ()

Gets the list of rows of Point4d_mdtl

Gets the list of rows of Point4d_mdtl.
Pre: True
Post: Returns the list of rows.

7.1.9

CPoints4d_mdtl& **operator=** (const CPoints4d_mdtl& CPointsToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = CPointsToCopy

7.1.10

friend ostream& **operator<<** (ostream& output, const CPoints4d_mdtl& CPoints)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a newl CPoints4d_mdtl object in the output file.

7.1.11

friend ostream& **operator<<=** (ostream& output, const CPoints4d_mdtl& CPoints)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new CPoints4d_mdtl object in the output file.

8

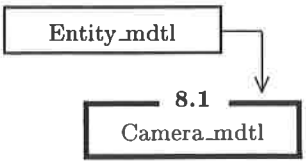
Names

8.1

class **Camera_mdtl** : public Entity_mdtl

Stores data needed for a camera definition

Inheritance



Public Members

8.1.1

Camera_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

8.1.2

Camera_mdtl (const Point3d_mdtl& Obs, const Point3d_mdtl& Vis, const Vector_mdtl& Up, double HAngl, double VAngl)

Constructor

Constructor.
Pre: True
Post: mObs = Obs && mVis = Vis && mUp = Up && mHAngle = HAngl && mVAngle = VAngl

8.1.3

Camera_mdtl (const Camera_mdtl& CameraToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = CameraToCopy

8.1.4

~Camera_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

8.1.5

void **SetObs** (const Point3d_mdtl& Obs)

Sets value of observer

Sets value of observer.
Pre: True
Post: mObs = Obs

8.1.6

void **SetVis** (const Point3d_mdtl& Vis)

Sets value of view reference point

Sets value of view reference point.
Pre: True
Post: mVis = Vis

8.1.7

void **SetUp** (const Vector_mdtl& Vec)

Sets value of up vector

Sets value of up vector.
Pre: True
Post: mUp = Up

8.1.8

void **SetHAngle** (double HAngl)

Sets value of horizontal angle

Sets value of horizontal angle.
Pre: True
Post: mHAngle = HAngl

8.1.9

void **SetVAngle** (double VAngl)

Sets value of vertical angle

Sets value of vertical angle.
Pre: True
Post: mVAngle = VAngl

8.1.10

Point3d_mdtl **GetObs** ()

Returns value of observer

Returns value of observer.
Pre: True
Post: Returns mObs

8.1.11

Point3d_mdtl **GetVis** ()

Returns value of view reference point

Returns value of view reference point.
Pre: True
Post: Returns mVis

8.1.12

Vector_mdtl **GetUp** ()

Returns value of up vector

Returns value of up vector.
Pre: True
Post: Returns mUp

8.1.13

double **GetHAngle** ()

Returns value of horizontal angle

Returns value of horizontal angle.
Pre: True
Post: Returns mHAngle

8.1.14

double **GetVAngle** ()

Returns value of vertical angle

Returns value of vertical angle.
Pre: True
Post: Returns mVAngle

8.1.15

Camera_mdtl& **operator=** (const Camera_mdtl& CamToAssign)

Copy operator

Copy operator.
Pre: True
Post: (*this) = CamToAssign

8.1.16

friend ostream& **operator<<** (ostream& output, const Camera_mdtl& cam)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Camera_mdtl object in the output file.

8.1.17

friend ostream& **operator<<=** (ostream& output, const Camera_mdtl& cam)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Camera_mdtl object in the output file.

9

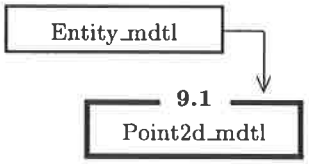
Names

9.1

class **Point2d_mdtl** : public Entity_mdtl

Stores a two-dimension point

Inheritance



Public Members

9.1.1 Point2d_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

9.1.2 Point2d_mdtl (PPString name)

Constructor

Constructor.

Pre: True
Post: mName = name

9.1.3

Point2d_mdtl (PPString name, double X, double Y)

Constructor

Constructor.
Pre: True
Post: mX = X && mY = Y && mName = name

9.1.4

Point2d_mdtl (double X, double Y)

Constructor

Constructor.
Pre: True
Post: mX = X && mY*= Y

9.1.5

~Point2d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

9.1.6

Point2d_mdtl (const Point2d_mdtl& PointToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = PointToCopy

9.1.7

void **data** (double X, double Y)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y

9.1.8

void **data** (PPString name, double X, double Y)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y && mName = name

9.1.9

double **GetX** ()

Returns value of member X

Returns value of member X.
Pre: True
Post: Returns mX

9.1.10

double **GetY** ()

Returns value of member Y

Returns value of member Y.
Pre: True
Post: Returns mY

9.1.11

Point2d_mdtl& **operator=** (const Point2d_mdtl& PointToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = PointToCopy

9.1.12

friend ostream& **operator<<** (ostream&, const Point2d_mdtl& PointToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Point2d_mdtl object in the output file.

9.1.13

friend ostream& **operator<<=** (ostream& output, const Point2d_mdtl& p)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Point2d_mdtl object in the output file.

10

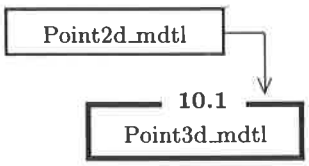
Names

10.1

class **Point3d_mdtl** : public Point2d_mdtl

Stores a three-dimension point

Inheritance



Public Members

10.1.1

Point3d_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

10.1.2

Point3d_mdtl (PPString name)

Constructor

Constructor.
Pre: True
Post: mName = name

10.1.3

Point3d_mdtl (PPString name, double X, double Y, double Z)

Constructor

Constructor.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mName = name

10.1.4

Point3d_mdtl (double X, double Y, double Z)

Constructor

Constructor.
Pre: True
Post: mX = X && mY = Y && mZ = Z

10.1.5

Point3d_mdtl (const Point3d_mdtl& PointToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = PointToCopy

10.1.6

~Point3d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

10.1.7

void **data** (PPString name)

Sets the name of the object

Sets the name of the object.
Pre: True
Post: mName = name

10.1.8

void **data** (double X, double Y, double Z)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y && mZ = Z

10.1.9

void **data** (PPString name, double X, double Y, double Z)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mName = name

10.1.10

double **GetZ** ()

Returns value of member Z

Returns value of member Z.
Pre: True
Post: Returns mZ

10.1.11

Point3d_mdtl& **operator=** (const Point3d_mdtl& PointToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = PointToCopy

10.1.12

friend ostream& **operator<<** (ostream&, const Point3d_mdtl& PointToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Point3d_mdtl object in the output file.

10.1.13

friend ostream& **operator<<=** (ostream&, const Point3d_mdtl& PointToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Point3d_mdtl object in the output file.

11

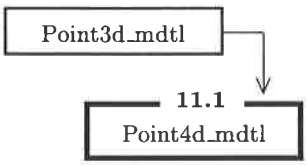
Names

11.1

class **Point4d_mdtl** : public Point3d_mdtl

Stores a four-dimension point

Inheritance



Public Members

11.1.1

Point4d_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

11.1.2

Point4d_mdtl (PPString name)

Constructor

Constructor.
Pre: True
Post: mName = name

11.1.3

Point4d_mdtl (PPString name, double X, double Y, double Z, double T)

Constructor

Constructor.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mT = T && mName = name

11.1.4

Point4d_mdtl (double X, double Y, double Z, double T)

Constructor

Constructor.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mT = T

11.1.5

Point4d_mdtl (const Point4d_mdtl& PointToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = PointToCopy

11.1.6

~Point4d_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

11.1.7

void **data** (PPString name)

Sets the name of the object

Sets the name of the object.
Pre: True
Post: mName = name

11.1.8

void **data** (double X, double Y, double Z, double T)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mT = T

11.1.9

void **data** (PPString name, double X, double Y, double Z, double T)

Sets the values of members

Sets the values of members.
Pre: True
Post: mX = X && mY = Y && mZ = Z && mT = T && mName = name

11.1.10

double **GetT** ()

Returns value of member T

Returns value of member T.
Pre: True
Post: Returns mT

11.1.11

Point4d_mdtl& **operator=** (const Point4d_mdtl& PointToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = PointToCopy

11.1.12

friend ostream& **operator<<** (ostream&, const Point4d_mdtl& PointToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Point4d_mdtl object in the output file.

11.1.13

friend ostream& **operator<<=** (ostream&, const Point4d_mdtl& PointToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Point4d_mdtl object in the output file.

12

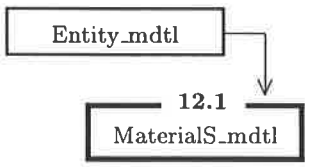
Names

12.1

class **MaterialS_mdtl** : public Entity_mdtl

Stores the data referred to a surface material

Inheritance



Public Members

12.1.1

MaterialS_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

12.1.2

MaterialS_mdtl (const MaterialS_mdtl& MatToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) =

12.1.3

~MaterialS_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

12.1.4

void **AddColor** (const Colorgen_mdtl& ColorToAdd)

Adds a new color propperty to the list of surface materials

Adds a new color propperty to the list of surface materials.
Pre: True
Post: this->mListCol.add(ColorToAdd).

12.1.5

void **AddTexture** (const Texture_mdtl& TextureToAdd)

Adds a new texture propperty to the list of surface materials

Adds a new texture propperty to the list of surface materials.
Pre: True
Post: this->mListText.add(TextureToAdd).

12.1.6

list_mdtl <Colorgen_mdtl> & **GetColors** ()

Returns the list of color attributes of the surface

Returns the list of color attributes of the surface.
Pre: True
Post: Returns mListCol.

12.1.7

list_mdtl <Texture_mdtl> & **GetTextures** ()

Returns the list of textures

Returns the list of textures.
Pre: True
Post: Returns mListText.

12.1.8

MaterialS_mdtl& **operator=** (const MaterialS_mdtl& MatToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = MatToCopy

12.1.9

friend ostream& **operator<<** (ostream&, const MaterialS_mdtl& MatToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new MaterialS_mdtl object in the output file.

12.1.10

friend ostream& **operator<<=** (ostream&, const MaterialS_mdtl& MatToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new MaterialS_mdtl object in the output file.

13

Names

13.1 class NodeEO_mdtl

Stores a node that belongs to an extended octree entity

Public Members

13.1.1 NodeEO_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

13.1.2 NodeEO_mdtl (NodeType t)

Constructor

Constructor.
Pre: True
Post: mType = t

13.1.3 NodeEO_mdtl (NodeType t, Stringlist l, EdgeConflist ec)

Constructor

Constructor.
Pre: True
Post: mType = t && mPlanes = l && mConfs = ec

13.1.4

NodeEO_mdtl (PPString name)

Constructor

Constructor.
Pre: True
Post: mName = name

13.1.5

NodeEO_mdtl (PPString name1, PPString name2, EdgeConf ec)

Constructor

Constructor.
Pre: True
Post: mType = EDGE_MDTL && mPlanes[0] = name1 && mPlanes[1] = name2 && mConfs = ec

13.1.6

~NodeEO_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

13.1.7

NodeEO_mdtl (const NodeEO_mdtl& NodeToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = NodeToCopy

13.1.8

void **Clear** ()

Clears the values of the members

Clears the values of the members.
Pre: True
Post: *this = ?

13.1.9

void **SetType** (NodeType t)

Sets the type of the node

Sets the type of the node.
Pre: True
Post: mType = t

13.1.10

void **SetPlanes** (Stringlist sl)

Sets the list of planes of the node

Sets the list of planes of the node.
Pre: True
Post: mPlanes = sl

13.1.11

void **SetConfs** (EdgeConflist ecl)

Sets the list of edge configurations of the node

Sets the list of edge configurations of the node.
Pre: True
Post: mConfs = ecl

13.1.12

void **AddPlane** (PPString nam)

Adds a plane to the list of planes of the node

Adds a plane to the list of planes of the node.
Pre: True
Post: this->mPlanes.add(nam)

13.1.13

void **AddConfig** (EdgeConf ec)

Adds a configuration to the list of configurations of the node

Adds a configuration to the list of configurations of the node.
Pre: True
Post: this->mConfs.add(ec)

13.1.14

void **AddConfig** (PPString nam)

Adds a configuration to the list of configurations of the node

Adds a configuration to the list of configurations of the node.
Pre: True
Post: this->mConfs.add(nam)

13.1.15

NodeType **GetType** ()

Returns the type of the node

Returns the type of the node.
Pre: True
Post: Returns mType.

13.1.16

int **GetNumPlanes** ()

Returns the number of planes contained in the list of planes

Returns the number of planes contained in the list of planes.
Pre: True
Post: Returns this->mPlanes.size().

13.1.17

int **GetNumConfs** ()

Returns the number of configurations contained in the list of edge configurations

Returns the number of configurations contained in the list of edge configurations.
Pre: True
Post: Returns this->mConfs.size().

13.1.18

Stringlist **GetPlanes** ()

Returns the list of planes, given by their names

Returns the list of planes, given by their names.
Pre: True
Post: Returns mPlanes.

13.1.19

EdgeConflist **GetConfs** ()

Returns the list of configurations

Returns the list of configurations.
Pre: True
Post: Returns mConfs.

13.1.20

NodeEO_mdtl& **operator=** (const NodeEO_mdtl& NodeToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = AtToCopy

13.1.21

friend ostream& **operator<<** (ostream&, const NodeEO_mdtl& NodeToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new node of an extended octree in the output file.

13.1.22

friend ostream& **operator<<=** (ostream&, const NodeEO_mdtl& NodeToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Writes a new NodeEO_mdtl of an extended octree
in the output file.

14

Names

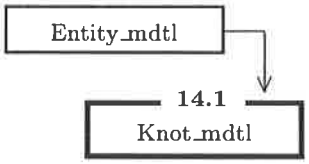
14.1 class **Knot_mdtl** : public Entity_mdtl
Knot of a two-dimension BSpline
82

14.1

class **Knot_mdtl** : public Entity_mdtl

Knot of a two-dimension BSpline

Inheritance



Public Members

14.1.1

Knot_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

14.1.2

Knot_mdtl (list_mdtl <double> Values)

Constructor

Constructor.
Pre: True
Post: mValues = Values

14.1.3

Knot_mdtl (PPString name, list_mdtl <double> Values)

Constructor

Constructor.
Pre: True
Post: mValues = Values && mName = name

14.1.4

~Knot_mdtl ()

Destructor

Destructor. Pre: True Post: *this = ?

14.1.5

Knot_mdtl (const Knot_mdtl& KnotToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = KnotToCopy

14.1.6

void **SetValues** (list_mdtl <double> Values)

Sets the list of knots

Sets the list of knots.
Pre: True
Post: mValues = Values

14.1.7

void **AddValue** (double val)

Adds a value to the list of knots

Adds a value to the list of knots.
Pre: True
Post: this->mValues.add(val)

14.1.8

list_mdtl <double> **GetValues** ()

Returns the list of values

Returns the list of values.
Pre: True
Post: Returns mValues.

14.1.9

Knot_mdtl& **operator=** (const Knot_mdtl& KnotToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = KnotToCopy

14.1.10

friend ostream& **operator<<** (ostream& output, const Knot_mdtl& k)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Knot_mdtl object in the output file.

14.1.11

friend ostream& **operator<<=** (ostream& output, const Knot_mdtl& k)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Knot_mdtl object in the output file.

15

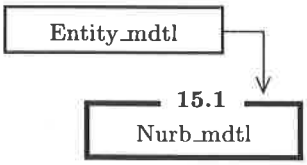
Names

15.1

class **Nurb_mdtl** : public Entity_mdtl

Stores data of a NURBS

Inheritance



Public Members

15.1.1

Nurb_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

15.1.2

Nurb_mdtl (const Nurb_mdtl& NurbToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = NurbToCopy

15.1.3

void **SetDegreeU** (double u)

Sets the degree of U knots

Sets the degree of U knots.
Pre: True
Post: mDegreeU = u

15.1.4

void **SetDegreeV** (double v)

Sets the degree of V knots

Sets the degree of V knots.
Pre: True
Post: mDegreeV = v

15.1.5

void **SetKnotsU** (const Knot_mdtl& KnotsU)

Sets the U knots

Sets the U knots.
Pre: True
Post: mKnotsU = KnotsU

15.1.6

void **SetKnotsV** (const Knot_mdtl& KnotsV)

Sets the V knots

Sets the V knots.
Pre: True
Post: mKnotsV = KnotsV

15.1.7

void **SetCPoints4d** (const CPoints4d_mdtl& CPoints4d)

Sets the control points of the NURBS

Sets the control points of the NURBS.
Pre: True
Post: mCPoints4d = CPoints4d

15.1.8

double **GetDegreeU** ()

Returns value U degree

Returns value U degree.
Pre: True
Post: Returns mDegreeU

15.1.9

double **GetDegreeV** ()

Returns value V degree

Returns value V degree.
Pre: True
Post: Returns mDegreeV

15.1.10

Knot_mdtl& **GetKnotsU** ()

Returns U knots

Returns U knots.
Pre: True
Post: Returns mDegreeV

15.1.11

Knot_mdtl& **GetKnotsV** ()

Returns V knots

Returns V knots.
Pre: True
Post: Returns mKnotsU

15.1.12

CPoints4d_mdtl& **GetCPoints4d** ()

Returns V knots

Returns V knots.
Pre: True
Post: Returns mKnotsV

15.1.13

Nurb_mdtl& **operator=** (const Nurb_mdtl& NurbToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = NurbToCopy

15.1.14

friend ostream& **operator<<** (ostream& output, const Nurb_mdtl& nu)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Nurb_mdtl object in the output file.

15.1.15

friend ostream& **operator<<=** (ostream& output, const Nurb_mdtl& nu)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Nurb_mdtl object in the output file.

16

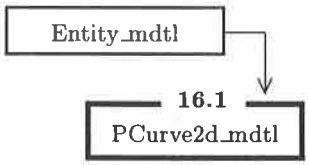
Names

16.1

class **PCurve2d_mdtl** : public Entity_mdtl

Stores data needed by a plane

Inheritance



Public Members

16.1.1

PCurve2d_mdtl ()

Constructor

Constructor.
Pre: True Post: *this = ?

16.1.2

PCurve2d_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: mName = Name

16.1.3

PCurve2d_mdtl (PPString Name, const list_mdtl <Point2d_mdtl>& Points)

Constructor

Constructor.
Pre: True
Post: mPoints = Points && mName = Name

16.1.4

~PCurve2d_mdtl ()

Destructor

Destructor. Pre: True Post: *this = ?

16.1.5

PCurve2d_mdtl (const PCurve2d_mdtl& CurveToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = CurveToCopy

16.1.6

void **SetPoints** (const list_mdtl <Point2d_mdtl>& Points)

Sets the value of the list of Points2d

Sets the value of the list of Points2d.
Pre: True
Post: mPoints = Points

16.1.7

const list_mdtl <Point2d_mdtl> & **GetPoints** ()

Returns the list of points

Returns the list of points.
Pre: True
Post: Returns mPoints

16.1.8

PCurve2d_mdtl& **operator=** (const PCurve2d_mdtl& CurveToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = CurveToCopy

16.1.9

friend ostream& **operator<<** (ostream& output, const PCurve2d_mdtl& Curve)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new PCurve2d_mdtl object in the output file.

16.1.10

friend ostream& **operator<<=** (ostream& output, const PCurve2d_mdtl& Curve)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new PCurve2d_mdtl object in the output file.

17

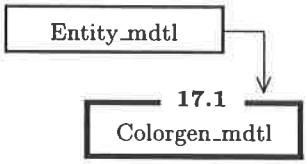
Names

17.1

class **Colorgen_mdtl** : public Entity_mdtl

Stores a color, a color spectrum or a file of textures

Inheritance



Public Members

17.1.1 Colorgen_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

17.1.2 Colorgen_mdtl (PPString name)

Constructor

Constructor.
Pre: True
Post: ma = a && mb = b && mc = c && md = d

17.1.3 ~Colorgen_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

17.1.4

Colorgen_mdtl (const Colorgen_mdtl& ColorToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = ColorToCopy

17.1.5

Colorgen_mdtl& **operator=** (const Colorgen_mdtl& ColorToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = ColorToCopy

17.1.6

friend ostream& **operator<<** (ostream&, const Colorgen_mdtl ColorToCopy&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Colorgen_mdtl object in the output file.

17.1.7

friend ostream& **operator<<=** (ostream&, const Colorgen_mdtl& ColorToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Colorgen_mdtl object in the output file.

17.1.8

PPString **GetFileName** ()

Returns the name of a file of textures

Returns the name of a file of textures.
Pre: mValueType == VALFILE
Post: Returns mType.

17.1.9

Color_mdtl **GetColor** ()

Returns the base color

Returns the base color.
Pre: mValueType != NOVAL && mValueType != VALFILE
Post: Returns mFileName.

17.1.10

list_mdtl <double> **GetSpec** ()

Returns the spectrum

Returns the spectrum.
Pre: mValueType == VALSPECTRUM
Post: Returns mCol.

17.1.11

ColorGenType **GetType** ()

Returns the spectrum

Returns the spectrum.
Pre: mValueType == VALSPECTRUM
Post: Returns mSpec.

17.1.12

double **GetRoughness** ()

Returns the Roughness

Returns the Roughness.
Pre: True.
Post: Returns mRough.

17.1.13

double **GetFactor** ()

Returns the factor

Returns the factor.
Pre: True.
Post: Returns mFactor.

17.1.14

ColValType **GetValueType** ()

Returns the type of color

Returns the type of color.
Pre: True.
Post: Returns mValueType.

17.1.15

void **SetFile** (PPString name)

Sets the name of a file of textures

Sets the name of a file of textures.
Pre: mValueType == VALFILE
Post: mValueType = VALFILE && mFileName = name

17.1.16

void **AddSpec** (double val)

Adds a value for the spectrum

Adds a value for the spectrum.
Pre: True
Post: mValueType == VALSPECTRUM && this->mSpec.add(val)

17.1.17

void **SetColor** (Color_mdtl col)

Sets the base color

Sets the base color.
Pre: True
Post: mCol = col

17.1.18

void **SetSpec** (list_mdtl <double> spec)

Sets the spectrum list values

Sets the spectrum list values.
Pre: True
Post: mValueType == VALSPECTRUM && mSpec = spec

17.1.19

void **SetRoughness** (double ro)

Sets the value of roughness

Sets the value of roughness.
Pre: True
Post: mRough = ro

17.1.20

void **SetFactor** (double fa)

Sets the value of factor

Sets the value of factor.
Pre: True
Post: mFactor = fa

17.1.21

void **SetType** (ColorGenType t)

Sets the color type

Sets the color type.
Pre: True
Post: mCType = t

18

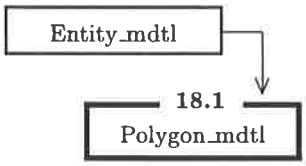
Names

18.1

class **Polygon_mdtl** : public Entity_mdtl

Stores data for a polygon

Inheritance



Public Members

18.1.1

Polygon_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

18.1.2

Polygon_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: mName = Name

18.1.3

Polygon_mdtl (const Polygon_mdtl& polToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = polToCopy

18.1.4

Polygon_mdtl **operator=** (const Polygon_mdtl& polToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = polToCopy

18.1.5

~Polygon_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

18.1.6

void **addvertex** (const Vertex_mdtl& v)

Adds a new vertex to the list of vertices

Adds a new vertex to the list of vertices.
Pre: True
Post: this->mvertlist.add(v).

18.1.7

void **addMatS** (const MaterialS_mdtl& m)

Adds a new material to the list of surface materials

Adds a new material to the list of surface materials.
Pre: True
Post: mmatS = m.

18.1.8

void **addplane** (const Plane_mdtl& p)

Sets the plane

Sets the plane.
Pre: True
Post: mplane = p.

18.1.9

void **SetType** (const Type_mdtl& t)

Sets the type of polygon, open or closed

Sets the type of polygon, open or closed.
Pre: True
Post: mty = t.

18.1.10

void **clear** ()

Clears the list of vertices

Clears the list of vertices.
Pre: True
Post: ?.

18.1.11

Vertexlist **GetVertexs** ()

Returns the list of color attributes of the surface

Returns the list of color attributes of the surface.
Pre: True
Post: Returns a list of Colorgen_mdtl objects.

18.1.12

MaterialS_mdtl **GetMatS** ()

Returns the material of the polygon

Returns the material of the polygon.
Pre: True
Post: Returns mmatS.

18.1.13

Plane_mdtl **GetPlane** ()

Returns the plane of the polygon

Returns the plane of the polygon.
Pre: True
Post: Returns mplane.

18.1.14

Type_mdtl **GetType** ()

Returns the type of the polygon

Returns the type of the polygon.
Pre: True
Post: Returns mty.

18.1.15

friend ostream& **operator<<** (ostream&, const Polygon_mdtl&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Polygon_mdtl object in the output file.

18.1.16

friend ostream& **operator<<=** (ostream&, const Polygon_mdtl&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Polygon_mdtl object in the output file.

19

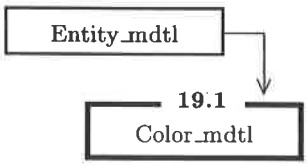
Names

19.1

class **Color_mdtl** : public Entity_mdtl

Stores a simple color in RGB or CIE format

Inheritance



Public Members

19.1.1

Color_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

19.1.2

Color_mdtl (PPString name, double X, double Y, double Z)

Constructor of a RGB color

Constructor of a RGB color.
Pre: True
Post: mx = X && my = Y && mz = Z && mName = name

19.1.3

Color_mdtl (PPString name, double X, double Y)

Constructor of a CIE color

Constructor of a CIE color.
Pre: True
Post: mx = X && my = Y && mName = name

19.1.4

Color_mdtl (double X, double Y, double Z)

Constructor of a RGB color

Constructor of a RGB color.
Pre: True
Post: mx = X && my = Y && mz = Z && mctype = COLRGB

19.1.5

Color_mdtl (double X, double Y)

Constructor of a CIE color

Constructor of a CIE color.
Pre: True
Post: mx = X && my = Y && mctype = COLCIE

19.1.6

~Color_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

19.1.7

Color_mdtl (const Color_mdtl& ColToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = ColToCopy

19.1.8

const Color_mdtl& **operator=** (const Color_mdtl& ColToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = ColToCopy

19.1.9

friend ostream& **operator<<** (ostream&, const Color_mdtl& ColToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Color_mdtl object in the output file.

19.1.10

friend ostream& **operator<<=** (ostream&, const Color_mdtl& ColToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Color_mdtl object in the output file.

19.1.11

double **GetX** ()

Returns value of member x

Returns value of member x.
Pre: True
Post: Returns mx

19.1.12

double **GetY** ()

Returns value of member y

Returns value of member y.
Pre: True
Post: Returns my

19.1.13

double **GetZ** ()

Returns value of member z

Returns value of member z.
Pre: True
Post: Returns mz

19.1.14

ColorType **GetType** ()

Returns type of the color, RGB or CIE

Returns type of the color, RGB or CIE.
Pre: mctype == COLRGB
Post: Returns mctype

19.1.15

void **data** (double X, double Y, double Z)

Sets the data of a RGB color

Sets the data of a RGB color.
Pre: True
Post: mx = X && my = Y && mz = Z && mctype = COLRGB

19.1.16

void **data** (double X, double Y)

Sets the data of a CIE color

Sets the data of a CIE color.
Pre: True
Post: mx = X && my = Y && mctype = COLCIE

20

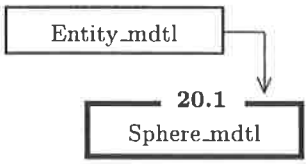
Names

20.1

class **Sphere_mdtl** : public Entity_mdtl

Stores data for a Sphere

Inheritance



Public Members

20.1.1

Sphere_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

20.1.2

Sphere_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: mName = name

20.1.3

Sphere_mdtl (PPString Name, double Radius, const Point3d_mdtl& Center)

Constructor

Constructor.
Pre: True
Post: mName = name && mRadius = Radius && mCenter = Center

20.1.4

~Sphere_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

20.1.5

Sphere_mdtl (const Sphere_mdtl& SphereToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = SphereToCopy

20.1.6

void **SetRadius** (double Radius)

Sets value of radius

Sets value of radius.
Pre: True
Post: mRadius = Radius

20.1.7

void **SetCenter** (const Point3d_mdtl& Center)

Sets value of center

Sets value of center.
Pre: True
Post: mCenter = Center

20.1.8

void **SetAttrib** (const Attrib_mdtl& Attrib)

Sets value of attributes

Sets value of attributes.
Pre: True
Post: mAttrib = Attrib

20.1.9

double **GetRadius** ()

Gets the value of radius

Gets the value of radius.
Pre: True
Post: Returns mRadius

20.1.10

const Point3d_mdtl& **GetCenter** ()

Gets the value of center

Gets the value of center.
Pre: True
Post: Returns mCenter

20.1.11

const Attrib_mdtl& **GetAttrib** ()

Gets the value of attributes

Gets the value of attributes.
Pre: True
Post: Returns mAttrib

20.1.12

Sphere_mdtl **operator=** (const Sphere_mdtl& SphereToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = SphereToCopy

20.1.13

friend ostream& **operator<<** (ostream& output, const Sphere_mdtl& Sphere)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Sphere_mdtl object in the output file.

20.1.14

friend ostream& **operator<<=** (ostream& output, const Sphere_mdtl& Sphere)

Writes the members to the output in format MDTL

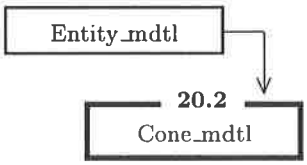
Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Sphere_mdtl object in the output file.

20.2

class **Cone_mdtl** : public Entity_mdtl

Stores data for a Cone

Inheritance



Public Members

20.2.1

Cone_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: *this = ?

20.2.2

Cone_mdtl (PPString Name, double Radius, const Point3d_mdtl& Center1, const Point3d_mdtl& Center2)

Constructor

Constructor.
Pre: True
Post: mName = name && mRadius = Radius && mCenter1 = Cente2 && mCenter2 = Center2

20.2.3

~Cone_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

20.2.4

Cone_mdtl (const Cone_mdtl& ConeToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = ConeToCopy

20.2.5

void **SetRadius** (double Radius)

Sets value of radius

Sets value of radius.
Pre: True
Post: mRadius = Radius

20.2.6

void **SetCenter1** (const Point3d_mdtl& Center1)

Sets value of center1

Sets value of center1.
Pre: True
Post: mCenter1 = Center1

20.2.7

void **SetCenter2** (const Point3d_mdtl& Center2)

Sets value of center2

Sets value of center2.
Pre: True
Post: mCenter2 = Center2

20.2.8

void **SetAttrib** (const Attrib_mdtl& Attrib)

Sets value of attributes

Sets value of attributes.
Pre: True
Post: mAttrib = Attrib

20.2.9

double **GetRadius** ()

Gets the value of radius

Gets the value of radius.
Pre: True
Post: Returns mRadius

20.2.10

const Point3d_mdtl& **GetCenter1** ()

Gets the value of center1

Gets the value of center1.
Pre: True
Post: Returns mCenter1

20.2.11

const Point3d_mdtl& **GetCenter2** ()

Gets the value of center2

Gets the value of center2.
Pre: True
Post: Returns mCenter2

20.2.12

const Attrib_mdtl& **GetAttrib** ()

Gets the value of attributes

Gets the value of attributes.
Pre: True
Post: Returns mAttrib

20.2.13

Cone_mdtl **operator=** (const Cone_mdtl& ConeToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = ConeToCopy

20.2.14

friend ostream& **operator<<** (ostream& output, const Cone_mdtl& Cone)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Cone_mdtl object in the output file.

20.2.15

friend ostream& **operator<<=** (ostream& output, const Cone_mdtl& Cone)

Writes the members to the output in format MDTL

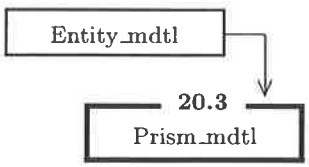
Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Cone_mdtl object in the output file.

20.3

class **Prism_mdtl** : public Entity_mdtl

Stores data for a Prism

Inheritance



Public Members

20.3.1 Prism_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

20.3.2 Prism_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: mName = name

20.3.3

Prism_mdtl (PPString Name, double Radius, long Faces, const Point3d_mdtl& Center1, const Point3d_mdtl& Center2)

Constructor

Constructor.
Pre: True
Post: mName = name && mRadius = Radius && mCenter1 = Cente2 && mCenter2 = Center2 && mFaces = Faces

20.3.4

~Prism_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

20.3.5

Prism_mdtl (const Prism_mdtl& PrismToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = PrismToCopy

20.3.6

void **SetFaces** (long Faces)

Sets value of number of faces

Sets value of number of faces.
Pre: True
Post: mFaces = Faces

20.3.7

void **SetRadius** (double Radius)

Sets value of radius

Sets value of radius.
Pre: True
Post: Radius = Radius

20.3.8

void **SetCenter1** (const Point3d_mdtl& Center1)

Sets value of center1

Sets value of center1.
Pre: True
Post: mCenter1 = Center1

20.3.9

void **SetCenter2** (const Point3d_mdtl& Center2)

Sets value of center2

Sets value of center2.
Pre: True
Post: mCenter2 = Center2

20.3.10

void **SetAttrib** (const Attrib_mdtl& Attrib)

Sets value of attributes

Sets value of attributes.
Pre: True
Post: mAttrib = Attrib

20.3.11

long **GetFaces** ()

Gets the value of number of faces

Gets the value of number of faces.
Pre: True
Post: Returns mFaces

20.3.12

double **GetRadius** ()

Gets the value of radius

Gets the value of radius.
Pre: True
Post: Returns mRadius

20.3.13

const Point3d_mdtl& **GetCenter1** ()

Gets the value of center1

Gets the value of center1.
Pre: True
Post: Returns mCenter1

20.3.14

const Point3d_mdtl& **GetCenter2** ()

Gets the value of center2

Gets the value of center2.
Pre: True
Post: Returns mCenter2

20.3.15

const Attrib_mdtl& **GetAttrib** ()

Gets the value of attributes

Gets the value of attributes.
Pre: True
Post: Returns mAttrib

20.3.16

Prism_mdtl **operator=** (const Prism_mdtl& PrismToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = ConeToCopy

20.3.17

friend ostream& **operator<<** (ostream& output, const Prism_mdtl& Prism)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Prism_mdtl object in the output file.

20.3.18

friend ostream& **operator<<=** (ostream& output, const Prism_mdtl& Prism)

Writes the members to the output in format MDTL

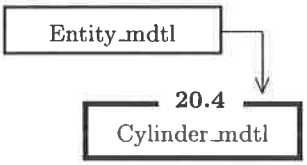
Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Prism_mdtl object in the output file.

20.4

class **Cylinder_mdtl** : public Entity_mdtl

Stores data for a Cylinder

Inheritance



Public Members

20.4.1

Cylinder_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

20.4.2

Cylinder_mdtl (PPString Name)

Constructor

Constructor.
Pre: True
Post: mName = name

20.4.3

Cylinder_mdtl (PPString Name, double Radius, const Point3d_mdtl& Center1, const Point3d_mdtl& Center2)

Constructor

Constructor.
Pre: True
Post: mName = name && mRadius = Radius && mCenter1 = Cente2 && mCenter2 = Center2

20.4.4

~Cylinder_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

20.4.5

Cylinder_mdtl (const Cylinder_mdtl& CylinderToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = CylinderToCopy

20.4.6

void **SetRadius** (double Radius)

Sets value of radius

Sets value of radius.
Pre: True
Post: mRadius = Radius

20.4.7

void **SetCenter1** (const Point3d_mdtl& Center1)

Sets value of center1

Sets value of center1.
Pre: True
Post: mCenter1 = Center1

20.4.8

void **SetCenter2** (const Point3d_mdtl& Center2)

Sets value of center2

Sets value of center2.
Pre: True
Post: mCenter2 = Center2

20.4.9

void **SetAttrib** (const Attrib_mdtl& Attrib)

Sets value of attributes

Sets value of attributes.
Pre: True
Post: mAttrib = Attrib

20.4.10

double **GetRadius** ()

Gets the value of radius

Gets the value of radius.
Pre: True
Post: Returns mRadius

20.4.11

const Point3d_mdtl& **GetCenter1** ()

Gets the value of center1

Gets the value of center1.
Pre: True
Post: Returns mCenter1

20.4.12

const Point3d_mdtl& **GetCenter2** ()

Gets the value of center2

Gets the value of center2.
Pre: True
Post: Returns mCenter2

20.4.13

const Attrib_mdtl& **GetAttrib** ()

Gets the value of attributes

Gets the value of attributes.
Pre: True
Post: Returns mAttrib

20.4.14

Cylinder_mdtl **operator=** (const Cylinder_mdtl& CylinderToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = ConeToCopy

20.4.15

friend ostream& **operator<<** (ostream& output, const Cylinder_mdtl& Cylinder)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Cylinder_mdtl object in the output file.

20.4.16

friend ostream& **operator<<=** (ostream& output, const Cylinder_mdtl& Cylinder)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Cylinder_mdtl object in the output file.

21

Names

21.1 class Texture_mdtl

Data needed for a texture of an object

Public Members

21.1.1

Texture_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

21.1.2

Texture_mdtl (const Texture_mdtl& TextureToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = TextureToCopy

21.1.3

~Texture_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

21.1.4

TextType **GetType** ()

Returns the type of the texture

Returns the type of the texture.
Pre: True
Post: Returns mtype

21.1.5

FinalColor **GetFC** ()

Returns the final color of the texture

Returns the final color of the texture.
Pre: True
Post: Returns mfc

21.1.6

Plane_mdtl **GetPlane1** ()

Returns the value of plane1

Returns the value of plane1.
Pre: True
Post: Returns mpl1

21.1.7

Plane_mdtl **GetPlane2** ()

Returns the value of plane2

Returns the value of plane2.
Pre: True
Post: Returns mpl2

21.1.8

double **GetVal** (int i)

Returns the ith value of texture

Returns the ith value of texture.
Pre: i <= 4 && i >= 0
Post: Returns mvar[i-1]

21.1.9

void **SetType** (TextType ty)

Sets the type of the texture

Sets the type of the texture.
Pre: True
Post: mtype = ty

21.1.10

void **SetFC** (FinalColor fc)

Sets the final color of the texture

Sets the final color of the texture.
Pre: True
Post: mfc = fc

21.1.11

void **SetPlanes** (const Plane_mdtl& pl1, const Plane_mdtl& pl2)

Sets values of planes

Sets values of planes.
Pre: True
Post: mpl1 = pl1 && mpl2 = pl2

21.1.12

void **SetPlane1** (const Plane_mdtl& pl1)

Sets value of plane1

Sets value of plane1.
Pre: True
Post: mpl1 = pl1

21.1.13

void **SetPlane2** (const Plane_mdtl& pl2)

Sets value of plane2

Sets value of plane2.
Pre: True
Post: mpl2 = pl2

21.1.14

void **SetVals** (double v1, double v2, double v3, double v4, double v5)

Sets values of texture

Sets values of texture.
Pre: True
Post: mval[0] = v1 && mval[1] = v2 && mval[2] = v3 && mval[3] = v4 && mval[4] = v5

21.1.15

Texture_mdtl& **operator=** (const Texture_mdtl& TextureToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = TextureToCopy

21.1.16

friend ostream& **operator<<** (ostream&, const Texture_mdtl& TextureToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Texture_mdtl object in the output file.

21.1.17

friend ostream& **operator<<=** (ostream&, const Texture_mdtl& TextureToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Texture_mdtl object in the output file.

22

Names

22.1 class TextAttribMdtl

Stores a textual attribute

Public Members

22.1.1

TextAttribMdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

22.1.2

TextAttribMdtl (const TextAttribMdtl& TextAttribToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = TextAttribToCopy

22.1.3 ~TextAttribMdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

22.1.4 PPString GetName ()

Returns value of the name of the textual attribute

Returns value of the name of the textual attribute.
Pre: True
Post: Returns mName

22.1.5 PPString GetValue ()

Returns value of the textual attribute

Returns value of the textual attribute.
Pre: True
Post: Returns mValue

22.1.6

void **SetValue** (char* val)

Sets the value

Sets the value.
Pre: True
Post: mValue = val

22.1.7

void **SetName** (char* name)

Sets the name of the textual attribute

Sets the name of the textual attribute.
Pre: True
Post: mName = name

22.1.8

void **SetValue** (PPString val)

Sets the value

Sets the value.
Pre: True
Post: mValue = val

22.1.9

void **SetName** (PPString name)

Sets the name of the textual attribute

Sets the name of the textual attribute.
Pre: True
Post: mName = name

22.1.10

TextAttribMdtl& **operator=** (const TextAttribMdtl& TextAttribToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = TextAttribToCopy

22.1.11

friend ostream& **operator<<** (ostream&, TextAttribMdtl TextAttribToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new TextAttribMdtl object in the output file.

22.1.12

friend ostream& **operator<<=** (ostream&, TextAttrib-
Mdtl TextAttribToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new TextAttribMdtl object in the output file.

23

Names

23.1

template <class T> class **list_mdtl**

Generic list used to contain objects

Public Members

Generic list used to contain objects. It uses internally a vector of the STL library.

23.1.1 list_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

23.1.2 list_mdtl (const list_mdtl& listToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = listToCopy

23.1.3

~**list_mdtl** ()

Destructor

Destructor.
Pre: True
Post: *this = ?

23.1.4

void **add** (T Tobject)

Adds a new element to the list

Adds a new element to the list.
Pre: True
Post: this->list.add(Tobject).

23.1.5

void **gofirst** ()

Makes the cursor to point to the first element of the list

Makes the cursor to point to the first element of the list.
Pre: True
Post: cur = this->list.begin().

23.1.6

vector < T > ::iterator **first** ()

Returns the first element of the list

Returns the first element of the list.
Pre: True
Post: Returns this->list.begin().

23.1.7

bool **next** ()

Makes the cursor to point to the next element of the list

Makes the cursor to point to the next element of the list.
If it does not exist, cursor does not change its value and returns false.
Pre: True
Post: Returns (cur++) != this->list.end().

23.1.8

void **golast** ()

Makes the cursor to point to the last element of the list

Makes the cursor to point to the last element of the list.
Pre: True
Post: cur = this->list.end().

23.1.9

vector < T > ::iterator **last** ()

Returns the last element of the list

Returns the last element of the list.
Pre: True
Post: Returns this->list.end().

23.1.10

void **clear** ()

Clears the values of the list

Clears the values of the list.
Pre: True
Post: this->list.erase(this->list.begin(), this->list.end()).

23.1.11

T **current** ()

Returns the current element of the list

Returns the current element of the list.
Pre: cur >= list.begin() && cur <= list.end()
Post: Returns cur.

23.1.12

int **number** ()

Returns the size of the list

Returns the size of the list.
Pre: True
Post: Returns this->list.size().

23.1.13

T **operator[]** (int pos)

Returns the posth element of the list

Returns the posth element of the list.
Pre: pos >= 0 && pos < this->list.size()
Post: Returns cur.

23.1.14

friend ostream& **operator<<** (ostream&, const list_mdtl&)

Writes the elements of the list to the output calling to their own operator<< method

Writes the elements of the list to the output calling to their own operator<< method.
Pre: True
Post: Writes the list in the output file.

24

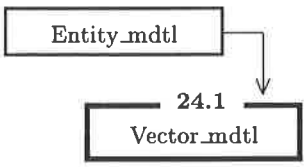
Names

24.1

class **Vector_mdtl** : public Entity_mdtl

Stores the data used for a vector

Inheritance



Public Members

24.1.1 Vector_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

24.1.2 Vector_mdtl (PPString name)

Constructor

Constructor.
Pre: True Post: mName = name

24.1.3 Vector_mdtl (const Vector_mdtl& VectorToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = VectorToCopy

24.1.4 Vector_mdtl (double v1, double v2, double v3)

Constructor

Constructor.
Pre: True Post: mCoor[0] = v1 && mCoor[1] = v2 && mCoor[2] = v3

24.1.5

Vector_mdtl (PPString name, double v1, double v2, double v3)

Constructor

Constructor.
Pre: True Post: mCoor[0] = v1 && mCoor[1] = v2 && mCoor[2] = v3 && mName = name && mCoType = COORD

24.1.6

Vector_mdtl (PPString name, const Point3d_mdtl& p1, const Point3d_mdtl& p2)

Constructor

Constructor.
Pre: True Post: mPo1 = p1 && mPo2 = p2 && mName = name && mCoType = POINTS

24.1.7

~Vector_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

24.1.8

friend ostream& **operator<<** (ostream&, const Vector_mdtl& VectorToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Vector_mdtl object in the output file.

24.1.9

friend ostream& **operator<<=** (ostream&, const Vector_mdtl& VectorToCopy)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Vector_mdtl object in the output file.

24.1.10

Vector_mdtl& **operator=** (const Vector_mdtl& VectorToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = VectorToCopy

24.1.11

void **SetVector** (double v1, double v2, double v3)

Sets the values of the vector

Sets the values of the vector.
Pre: True
Post: mCoor[0] = v1 && mCoor[1] = v2 && mCoor[2] = v3 && mCoType = COORD

24.1.12

void **SetVector** (const Point3d_mdtl& p1, const Point3d_mdtl& p2)

Sets the points of the vector

Sets the points of the vector.
Pre: True
Post: mPo1 = p1 && mPo2 = p2 && mCoType = POINTS

24.1.13

void SetCoord **(double v1, double v2, double v3)** (const Point3d_mdtl& p1)

Sets the values of the vector

Sets the values of the vector.
Pre: True
Post: mCoor[0] = v1 && mCoor[1] = v2 && mCoor[2] = v3 && mCoType = COORD

24.1.14

void **SetP2** (const Point3d_mdtl& p2)

Sets the second point of the vector

Sets the second point of the vector.
Pre: True
Post: mPo2 = p2 && mCoType = POINTS

24.1.15

Point3d_mdtl **GetP1** ()

Returns first point of the vector

Returns first point of the vector.
Pre: mCoType == POINTS
Post: Returns mPo1

24.1.16

Point3d_mdtl **GetP2** ()

Returns second point of the vector

Returns second point of the vector.
Pre: mCoType == POINTS
Post: Returns mPo2

24.1.17

coord_type **GetCoType** ()

Returns the type of coordinate this vector has

Returns the type of coordinate this vector has.
Pre: True
Post: Returns mCoType

24.1.18

double **GetCo** (int i)

Returns the ith coordinate of this vector

Returns the ith coordinate of this vector.
Pre: i<=3 && i>0 && mCoType == COORD
Post: Returns mCoType

25

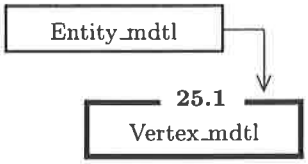
Names

25.1

class **Vertex_mdtl** : public Entity_mdtl

Stores a vertex

Inheritance



Public Members

25.1.1

Vertex_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

25.1.2

Vertex_mdtl (PPString name)

Constructor

Constructor.
Pre: True Post: mName = name

25.1.3

~Vertex_mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

25.1.4

Vertex_mdtl (const Vertex_mdtl& VertexToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = VertexToCopy

25.1.5

Vertex_mdtl& **operator=** (const Vertex_mdtl& VertexToCopy)

Copy operator

Copy operator.
Pre: True
Post: (*this) = VertexToCopy

25.1.6

void **SetPos** (const Point3d_mdtl& p1)

Sets the position of the vertex

Sets the position of the vertex.
Pre: True
Post: mPo = p1

25.1.7

void **SetVector** (const Vector_mdtl& vn)

Sets normal of the vertex

Sets normal of the vertex.
Pre: True
Post: mVec = vn

25.1.8

void **SetMatS** (const MaterialS_mdtl& mat)

Sets the material of the vertex

Sets the material of the vertex.
Pre: True
Post: mMat = mat

25.1.9

Point3d_mdtl **GetPos** ()

Returns the position of the vertex

Returns the position of the vertex.
Pre: True
Post: Returns mPo

25.1.10

Vector_mdtl **GetVector** ()

Returns the normal of the vertex

Returns the normal of the vertex.
Pre: True
Post: Returns mVec

25.1.11

MaterialS_mdtl **GetMatS** ()

Returns the material of the vertex

Returns the material of the vertex.
Pre: True
Post: Returns mMat

25.1.12

friend ostream& **operator** << (ostream&, const Vertex_mdtl&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares a new Vertex_mdtl object in the output file.

25.1.13

friend ostream& **operator** <<= (ostream&, const Vertex_mdtl&)

Writes the members to the output in format MDTL

Writes the members to the output in format MDTL.
Pre: True
Post: Declares implicitly (without new name) a new Vertex_mdtl object in the output file.

26

Names

26.1 class Universe_mdtl

Data needed for an universe of an octree

Public Members

26.1.1

Universe_mdtl ()

Constructor

Constructor.
Pre: True
Post: *this = ?

26.1.2

Universe_mdtl (double p11, double p12, double p13, double w, double h, double dep, double de, double t)

Constructor

Constructor.
Pre: True Post: mOrig.data(p11, p12, p13) && mWidth = w && mHeight = h && mDepth = dep && mDegree = de && mTol = t

26.1.3

Universe_mdtl (double p11, double p12, double p13, double w, double h, double dep, double de)

Constructor

Constructor.
Pre: True Post: mOrig.data(p11, p12, p13) && mWidth = w && mHeight = h && mDepth = dep && mDegree = de

26.1.4

Universe_mdtl (const Universe_mdtl& UnivToCopy)

Copy constructor

Copy constructor.
Pre: True
Post: (*this) = UnivToCopy

26.1.5

void **Addtol** (double t)

Sets the tolerance

Sets the tolerance.
Pre: True
Post: mTol = t

26.1.6

void **SetData** (double p11, double p12, double p13, double w, double h, double dep, double de)

Sets the values of all members

Sets the values of all members.
Pre: True
Post: mOrig.data(p11, p12, p13) && mWidth = w && mHeight = h && mDepth = dep && mDegree = de && mTol = t

26.1.7

void **SetData** (const Point3d_mdtl& p, double w, double h, double dep, double de)

Sets the values of all members

Sets the values of all members.
Pre: True
Post: mOrig = p && mWidth = w && mHeight = h && mDepth = dep && mDegree = de

26.1.8

Point3d_mdtl **GetOrigin** ()

Returns the origin of the universe

Returns the origin of the universe.
Pre: True
Post: Returns mOrig

26.1.9

double **GetWidth** ()

Returns the width of the universe

Returns the width of the universe.
Pre: True
Post: Returns mWidth

26.1.10

double **GetHeight** ()

Returns the height of the universe

Returns the height of the universe.
Pre: True
Post: Returns mHeight

26.1.11

double **GetDepth** ()

Returns the depth of the universe

Returns the depth of the universe.
Pre: True
Post: Returns mDepth

26.1.12

double **GetDegree** ()

Returns the degree of the universe

Returns the degree of the universe.
Pre: True
Post: Returns mDegree

26.1.13

double **GetTol** ()

Returns the tolerance of the universe

Returns the tolerance of the universe.
Pre: True
Post: Returns mTol

27

Names

27.1 class mdtl

Stores all the data needed for the management of files in MDTL language

Public Members

Allows the use of the plane entity with name "n" if the state of reading permits it .. 244

27.1.133bool **UseSurf** (PPString n)
Allows the use of the surface entity with name "n" if the state of reading permits it .. 244

27.1.134bool **UseNurb** (PPString n)
Allows the use of the NURBS entity with name "n" if the state of reading permits it .. 244

27.1.135bool **UseKnots** (PPString n)
Allows the use of the Knots entity with name "n" if the state of reading permits it .. 245

27.1.136bool **UseTrimmingC** (PPString n)
Allows the use of the trimming curve entity with name "n" if the state of reading permits it .. 245

27.1.137bool **UsePCurve2d** (PPString n)
Allows the use of the PCurve2d entity with name "n" if the state of reading permits it .. 245

27.1.138bool **UseBSpline2d** (PPString n)
Allows the use of the BSpline2d entity with name "n" if the state of reading permits it .. 246

27.1.139bool **UseCurve3d** (PPString n)
Allows the use of the Curve3d entity with name "n" if the state of reading permits it .. 246

27.1.140bool **UseSphere** (PPString n)
Allows the use of the Sphere entity with name "n" if the state of reading permits it .. 246

27.1.141bool **UseCylinder** (PPString n)
Allows the use of the cylinder entity with name "n" if the state of reading permits it .. 247

27.1.142bool **UseCone** (PPString n)

Stores all the data needed for the management of files in MDTL language. Keeps aware of the state of reading and writing and allows the use of several methods for input and output of entities in MDTL format.

27.1.1

mdtl (Openmode_mdtl op)

Constructor

Constructor. No files associated with mdtl object.
Pre: True
Post: mode = op

27.1.2

mdtl (PPString name, Openmode_mdtl op)

Constructor

Constructor. File name associated with mdtl object.

Pre: True
Post: mode = op && filename = name

27.1.3

mdtl (PPString name)

Constructor

Constructor. File name associated with mdtl object.
Pre: True
Post: filename = name && mode = READ

27.1.4

~mdtl ()

Destructor

Destructor.
Pre: True
Post: *this = ?

27.1.5

void **init_file** ()

Inits file for reading or writing

Inits file for reading or writing.
Pre: Name given
Post: File opened for reading or writing.

27.1.6

void **end_file** ()

Sets the file name

Sets the file name.
Pre: True
Post: filename = name

27.1.7

EntityType **r_next** ()

Reads next entity of the file

Reads next entity of the file.
Pre: mode == READ
Post: Returns the type corresponding to the next entity of the file.
If there are no entities left, it returns VOID_MDTL.

27.1.8

Symbol_mdtl **r_current** ()

Returns symbol of the next entity of the file

Returns symbol of the next entity of the file.
Pre: mode == READ
Post: Returns the symbol corresponding to the next entity of the file.
If there are no entities left, it returns VOID_MDTL.

27.1.9

Attrib_mdtl **GetCurAttrib** ()

Returns the current attributes to be applied to the entities being read

Returns the current attributes to be applied to the entities being read.
Pre: mode == READ
Post: Returns attr_stck.top()

27.1.10

void **w_OptPlane_name** (PPString, double, double, double, double)

Writes an optional plane of an octree to a file

Writes an optional plane of an octree to a file.
Pre: mode == WRITE && writing an octree
Post: Plane written.

27.1.11

PPString **w_OptPlane** (double, double, double, double)

Writes an optional plane with the given name of an octree to a file

Writes an optional plane with the given name of an octree to a file.
Pre: mode == WRITE && writing an octree
Post: Plane written.

27.1.12

void **w_OptPlane** (Plane_mdtl &)

Writes an optional plane an octree to a file

Writes an optional plane an octree to a file.
Pre: mode == WRITE && writing an octree
Post: Plane written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.13

void **w_Universe** (double, double, double, double, double, double, double)

Writes an universe with the values passed of an octree

Writes an universe with the values passed of an octree.
Pre: mode == WRITE && writing an octree
Post: Universe written.

27.1.14

void **w_Universe** (double, double, double, double, double, double, double, double)

Writes an universe with the values passed of an octree

Writes an universe with the values passed of an octree.
Pre: mode == WRITE && writing an octree
Post: Universe written.

27.1.15

void **w_Universe** (Universe_mdtl &)

Writes an universe of an octree

Writes an universe of an octree.
Pre: mode == WRITE && writing an octree
Post: Universe written.

27.1.16

void **DeclEO_name** (PPString)

Starts declaring an octree as an independent entity

Starts declaring an octree as an independent entity.
Pre: mode == WRITE
Post: Octree declared.

27.1.17

void **endEO** ()

Finishes declaration of an octree

Finishes declaration of an octree.
Pre: mode == WRITE
Post: Octree finished.

27.1.18

void **w_VertexNode** (Stringlist &, EdgeConflist &)

Writes a vertex node of an octree

Writes a vertex node of an octree.
Pre: mode == WRITE
Post: Node written.

27.1.19

void **w_QuasiVertexNode** (Stringlist&, EdgeConflist&)

Writes a quasivertex node of an octree

Writes a quasivertex node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.20

void **w_GreyNode** ()

Writes a grey node of an octree

Writes a grey node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.21

void **w_MinimumGreyNode** ()

Writes a minimumgrey node of an octree

Writes a minimumgrey node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.22

void **w_BlackNode** ()

Writes a black node of an octree

Writes a black node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.23

void **w_EdgeNode** (PPString, PPString, EdgeConf)

Writes an edge node of an octree

Writes an edge node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.24

void **w_FaceNode** (PPString)

Writes a face node of an octree

Writes a face node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.25

void **w_WhiteNode** ()

Writes a white node of an octree

Writes a white node of an octree.
Pre: mode == WRITE && writing an octree
Post: Node written.

27.1.26

void **r_EO_name** (PPString name)

Starts reading an extended octree with the given name

Starts reading an extended octree with the given name.
Pre: mode == READ
Post: Octree declaration read.

27.1.27

Universe_mdtl **r_Universe** ()

Reads a universe

Reads a universe.
Pre: mode == READ && reading an octree
Post: Universe declaration read.

27.1.28

NodeEO_mdtl **r_First_NodeEO** ()

Reads first node of an extended octree

Reads first node of an extended octree.
Pre: mode == READ && reading an octree
Post: Node read.

27.1.29

NodeEO_mdtl **r_Next_NodeEO** ()

Reads next node of an extended octree

Reads next node of an extended octree.
Pre: mode == READ && reading an octree
Post: Node read.

27.1.30

PPString **DeclSolidBr** ()

Starts declaring a SolidBr

Starts declaring a SolidBr.
Pre: mode == WRITE
Post: SolidBr declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.31

void **DeclSolidBr_name** (PPString)

Starts declaring a SolidBr with the given name

Starts declaring a SolidBr with the given name.
Pre: mode == WRITE
Post: SolidBr declared.

27.1.32

void **EndSolidBr** ()

Finishes declaration of a SolidBr

Finishes declaration of a SolidBr.
Pre: mode == WRITE && writing a SolidBr
Post: SolidBr finished.

27.1.33

PPString **DeclBoundary** (Type_mdtl)

Starts declaring a Boundary

Starts declaring a Boundary.
Pre: mode == WRITE
Post: Boundary declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.34

void **DeclBoundary_name** (PPString, Type_mdtl)

Starts declaring a Boundary with the given name

Starts declaring a Boundary with the given name.
Pre: mode == WRITE
Post: Boundary declared.

27.1.35

void **EndBoundary** ()

Finishes Boundary declaration of a

Finishes Boundary declaration of a .
Pre: mode == WRITE && writing a Boundary
Post: Boundary finished.

27.1.36

PPString **DeclAttrib** ()

Starts declaring an attribute

Starts declaring an attribute.
Pre: mode == WRITE
Post: Attribute declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.37

PPString **DeclAttrib** (Attrib_mdtl& at)

Writes given attribute to the file

Writes given attribute to the file.
Pre: mode == WRITE
Post: Attribute written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.38

void **DeclAttrib_name** (PPString name)

Starts declaring an attribute with the given name

Starts declaring an attribute with the given name.
Pre: mode == WRITE
Post: Attribute declared.

27.1.39

void **EndAttrib** ()

Finishes declaration of an attribute

Finishes declaration of an attribute.
Pre: mode == WRITE && writing an attribute
Post: Attribute finished.

27.1.40

void **DeclMaterialS_name** (PPString)

Starts declaring a surface material with the given name

Starts declaring a surface material with the given name.
Pre: mode == WRITE
Post: Surface material declared.

27.1.41

void **DeclMaterialS_name** (PPString, MaterialS_mdtl& mat)

Writes the given surface material

Writes the given surface material.
Pre: mode == WRITE
Post: Surface material written.

27.1.42

PPString **DeclMaterialS** ()

Starts declaring a surface material

Starts declaring a surface material.
Pre: mode == WRITE
Post: Surface material declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.43

PPString **DeclMaterialS** (MaterialS_mdtl& mat)

Writes the given surface material

Writes the given surface material.
Pre: mode == WRITE
Post: Surface material written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.44

void **DeclColor** (const Colorgen_mdtl& col)

Writes a color

Writes a color.
Pre: mode == WRITE && writing a material
Post: Color written.

27.1.45

void **w_Texture_aut** (Plane_mdtl, Plane_mdtl, FinalColor)

Writes a texture

Writes a texture.
Pre: mode == WRITE && writing a material
Post: Texture written.

27.1.46

void **w_Texture_sph** (FinalColor)

Writes a texture

Writes a texture.
Pre: mode == WRITE && writing a material
Post: Texture written.

27.1.47

void **w_Texture_expl** (double, double, double, double, double, FinalColor)

Writes a texture

Writes a texture.
Pre: mode == WRITE && writing a material
Post: Texture written.

27.1.48

void **EndMaterialS** ()

Finishes declaration of a surface material

Finishes declaration of a surface material.
Pre: mode == WRITE && writing a material
Post: MaterialS finished.

27.1.49

void **w_TextAttrib** (PPString, PPString)

Writes a textual attribute

Writes a textual attribute.
Pre: mode == WRITE && writing an attribute
Post: Texture written.

27.1.50

PPString **DeclShell** (Type_mdtl)

Starts declaring a shell

Starts declaring a shell.
Pre: mode == WRITE
Post: Shell declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.51

void **DeclShell_name** (PPString, Type_mdtl)

Starts declaring a shell with the given name

Starts declaring a shell with the given name.
Pre: mode == WRITE
Post: Shell declared.

27.1.52

void **EndShell** ()

Finishes declaration of a shell

Finishes declaration of a shell.
Pre: mode == WRITE && writing a shell
Post: Shell finished.

27.1.53

void **DeclEdges** ()

Starts declaring edges of a BRep

Starts declaring edges of a BRep.
Pre: mode == WRITE && writing a SolidBr
Post: Edges declared.

27.1.54

void **DeclEdgePair** (Point3d_mdtl p1, Point3d_mdtl p2, Point3d_mdtl p3, Point3d_mdtl p4)

Writes an EdgePair

Writes an EdgePair.
Pre: mode == WRITE && writing edges
Post: EdgePair written.

27.1.55

void **EndEdges** ()

Finishes declaration of edges

Finishes declaration of edges.
Pre: mode == WRITE && writing edges
Post: Edges finished.

27.1.56

void **DeclEdgePair** ()

Starts declaring an EdgePair

Starts declaring an EdgePair.
Pre: mode == WRITE && writing a SolidBr
Post: EdgePair declared.

27.1.57

void **EndEdgePair** ()

Finishes declaration of an EdgePair

Finishes declaration of an EdgePair.
Pre: mode == WRITE && writing edges
Post: EdgePair finished.

27.1.58

EdgeType **r_next_Edge** ()

Starts reading an Edge

Starts reading an Edge.
Pre: mode == READ && reading edges
Post: Edge declaration read.Returns edge type.
Returns VOID_EDGE if last edge has been already read.

27.1.59

EdgeType **r_next_EdgePair** ()

Starts reading an EdgePair

Starts reading an EdgePair.
Pre: mode == READ && reading edge pairs
Post: EdgePair declaration read. Returns edge type.
Returns VOID_EDGE if last edge has been already read.

27.1.60

EdgeType **r_Edge** ()

Starts reading an Edge

Starts reading an Edge.
Pre: mode == READ && reading edges
Post: Edge declaration read. Returns edge type.
Returns VOID_EDGE if last edge has been already read.

27.1.61

void **r_EdgePair2Points1** (Point3d_mdtl& p1,
Point3d_mdtl&p2)

Reads an EdgePair formed by 2 points

Reads an EdgePair formed by 2 points.
Pre: mode == READ && reading edge pairs
Post: First EdgePair read.

27.1.62

void **r_EdgePair2Points2** (Point3d_mdtl& p1,
Point3d_mdtl&p2)

Reads an EdgePair formed by 2 points

Reads an EdgePair formed by 2 points.
Pre: mode == READ && reading edge pairs
Post: Second EdgePair read.

27.1.63

PPString **DeclSurf** (const Nurb_mdtl& nu)

Starts declaring a Surface which contains a NURBS

Starts declaring a Surface which contains a NURBS.
Pre: mode == WRITE
Post: Surface declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.64

void **DeclSurfFace** ()

Starts declaring a face of a surface

Starts declaring a face of a surface.
Pre: mode == WRITE
Post: Face declared.

27.1.65

void **EndSurf** ()

Finishes declaration of a surface

Finishes declaration of a surface.
Pre: mode == WRITE && declaring a surface
Post: Surface finished.

27.1.66

PPString **DeclSurf** ()

Starts declaring a Surface

Starts declaring a Surface.
Pre: mode == WRITE && writing a surface
Post: Surface declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.67

void **DeclSPolig** (Type_mdtl t)

Starts declaring a SPolig

Starts declaring a SPolig.
Pre: mode == WRITE && declaring a surface
Post: SPolig declared.

27.1.68

void **EndSPolig** ()

Finishes declaration of a SPolig

Finishes declaration of a SPolig.
Pre: mode == WRITE && declaring a SPolig
Post: SPolig finished.

27.1.69

TrimmCType **r_SPolig** (Type_mdtl &t)

Starts reading a SPolig

Starts reading a SPolig.
Pre: mode == READ && reading a surface
Post: SPolig declaration read. Returns the trimming curve type.

27.1.70

PPString **r_Surf** ()

Starts reading a surface

Starts reading a surface.
Pre: mode == READ
Post: Surface declaration read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.71

void **r_Surf_name** (PPString s)

Starts reading a surface with the given name

Starts reading a surface with the given name.
Pre: mode == READ
Post: Surface declaration read.

27.1.72

Nurb_mdtl **r_Nurb** ()

Reads NURBS

Reads NURBS.
Pre: mode == READ && reading a surface
Post: NURBS read.

27.1.73

Nurb_mdtl **r_Nurb_name** (PPString s)

Reads NURBS with the given name

Reads NURBS with the given name.
Pre: mode == READ
Post: NURBS read.

27.1.74

TrimmCType **r_TrimmC** ()

Reads type of next trimming curve

Reads type of next trimming curve.
Pre: mode == READ && reading a trimming curve
Post: Type of trimming curve read.

27.1.75

void **r_ValuesTrimmC** (double& val1, double& val2)

Reads values of current trimming curve

Reads values of current trimming curve.
Pre: mode == READ && reading a trimming curve
Post: Values of trimming curve read.

27.1.76

BSpline2d_mdtl **r_BSplineRef** ()

Reads next BSpline

Reads next BSpline.
Pre: mode == READ && current trimming curve is a BSpline
Post: BSpline2d of trimming read.

27.1.77

PCurve2d_mdtl **r_PCurve2dRef** ()

Reads values of current trimming curve

Reads values of current trimming curve.
Pre: mode == READ && current trimming curve is a BSpline
Post: PCurve2d of trimming curve read.

27.1.78

void **r_EndTrimmC** ()

Finishes reading a trimming curve

Finishes reading a trimming curve.
Pre: mode == READ && reading a trimming curve
Post: Trimming curve finished.

27.1.79

Flag_mdtl **r_FlagEdge** ()

Reads flag of the edge

Reads flag of the edge.
Pre: mode == READ && reading edges
Post: Flag read.

27.1.80

void **DeclBSpline_name** (const BSpline2d_mdtl& bsp)

Writes a BSpline

Writes a BSpline.
Pre: mode == WRITE
Post: BSpline written.

27.1.81

PPString **DeclBSpline** (BSpline2d_mdtl& bsp)

Writes a BSpline

Writes a BSpline.
Pre: mode == WRITE
Post: BSpline written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.82

PPString **DeclBSpline** (double v1, double v2, const BSpline2d_mdtl& bsp)

Writes a BSpline with values given

Writes a BSpline with values given.
Pre: mode == WRITE
Post: BSpline written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.83

void **DeclTrimmingC_name** (PPString name)

Starts declaring a trimming curve with the given name

Starts declaring a trimming curve with the given name.
Pre: mode == WRITE
Post: Trimming curve declared.

27.1.84

PPString **DeclTrimmingC** ()

Starts declaring a trimming curve

Starts declaring a trimming curve.
Pre: mode == WRITE
Post: Trimming curve declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.85

void **EndTrimmingC** ()

Finishes declaration of a trimming curve which is not part of an edge

Finishes declaration of a trimming curve which is not part of an edge.
Pre: mode == WRITE
Post: Trimming curve finished.

27.1.86

void **EndTrimmingC** (Flag_mdtl F)

Finishes declaration of a trimming curve which is part of an edge

Finishes declaration of a trimming curve which is part of an edge.
Pre: mode == WRITE && declaring edges
Post: Trimming curve finished.

27.1.87

PPString **DeclFace** ()

Starts declaring a Face

Starts declaring a Face.
Pre: mode == WRITE
Post: Face declared.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.88

void **DeclFace_name** (PPString)

Starts declaring a Face with the given name

Starts declaring a Face with the given name.
Pre: mode == WRITE
Post: Face declared.

27.1.89

void **DeclPlanarFace** (Plane_mdtl)

Starts declaring a planar Face

Starts declaring a planar Face.
Pre: mode == WRITE && declaring a face
Post: Face declared.

27.1.90

PPString **DeclPlane** (const Plane_mdtl&)

Writes a plane

Writes a plane.
Pre: mode == WRITE
Post: Plane written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.91

void **DeclPlanarFace** (double, double, double, double)

Writes Planar Face with the given values

Writes Planar Face with the given values.
Pre: mode == WRITE && declaring a face
Post: Planar Face written.

27.1.92

void **EndPlanarFace** ()

Finishes declaration of a Planar Face

Finishes declaration of a Planar Face.
Pre: mode == WRITE && declaring a planar face.
Post: Planar Face finished.

27.1.93

void **EndSurfFace** ()

Finishes declaration of a SurfFace

Finishes declaration of a SurfFace.
Pre: mode == WRITE && declaring a planar face.
Post: SurfFacefinished.

27.1.94

PPString **DeclPolig** (const Polygon_mdtl&)

Writes a polygon

Writes a polygon.
Pre: mode == WRITE
Post: Polygon written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.95

void **EndPolig** ()

Finishes declaration of a polygon

Finishes declaration of a polygon.
Pre: mode == WRITE
Post: Polygon finished.

27.1.96

void **DeclPolig** (PPString, Type_mdtl)

Starts declaring a polygon

Starts declaring a polygon.
Pre: mode == WRITE
Post: Polygon declared.

27.1.97

void **w_Vertex** (Point3d_mdtl p)

Writes a vertex given by its position

Writes a vertex given by its position.
Pre: mode == WRITE
Post: Vertex written.

27.1.98

void **w_Vertex** (Point3d_mdtl&, Point3d_mdtl&, Point3d_mdtl&)

Writes a vertex given by its position and a normal given by two points

Writes a vertex given by its position and a normal given by two points.
Pre: mode == WRITE
Post: Vertex written.

27.1.99

void **w_Vertex** (Point3d_mdtl, double, double, double)

Writes a vertex given by its position and a normal given by three coordinates

Writes a vertex given by its position and a normal given by three coordinates.
Pre: mode == WRITE
Post: Vertex written.

27.1.100

PPString **r_SolidBr** ()

Starts reading a SolidBr

Starts reading a SolidBr.
Pre: mode == READ
Post: SolidBr declaration read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.101

void **r_SolidBr_name** (PPString)

Starts reading a SolidBr with the given name

Starts reading a SolidBr with the given name.
Pre: mode == READ
Post: SolidBr declaration read.

27.1.102

PPString **r_Boundary** (Type_mdtl&)

Starts reading a Boundary

Starts reading a Boundary.
Pre: mode == READ
Post: Boundary declaration read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.103

void **r_Boundary_name** (PPString, Type_mdtl&)

Starts reading a Boundary with the given name

Starts reading a Boundary with the given name.
Pre: mode == READ
Post: Boundary declaration read.

27.1.104

PPString **r_Shell** (Type_mdtl&)

Starts reading a Shell

Starts reading a Shell.
Pre: mode == READ
Post: Shell declaration and type read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.105

void **r_Shell_name** (PPString, Type_mdtl&)

Starts reading a Shell with the given name

Starts reading a Shell with the given name.
Pre: mode == READ
Post: Shell declaration and type read.
Returns the name of the shell if it has been declared
as an independent entity. If reading shells of a boundary, when
the boundary is finished it returns "}".

27.1.106

PPString **r_Face** (FaceType& t)

Starts reading a Face

Starts reading a Face.
Pre: mode == READ
Post: Face declaration and type read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".
If we are reading the faces of a shell, to indicate there are no
faces left the name of the face will be "}"

27.1.107

void **r_EndSbr** ()

Finishes reading a SolidBr

Finishes reading a SolidBr.
Pre: mode == READ && reading a SolidBr
Post: SolidBr finished.

27.1.108

Plane_mdtl **r_plane** ()

Reads plane

Reads plane.
Pre: mode == READ
Post: Plane read.

27.1.109

Plane_mdtl **r_plane_name** (PPString name)

Reads plane with the given name

Reads plane with the given name.
Pre: mode == READ
Post: Plane read.

27.1.110

Polygon_mdtl **r_polygon** ()

Reads polygon

Reads polygon.
Pre: mode == READ
Post: Polygon read.
If we are reading the polygons of the face, to indicate there are no polygons left the name of the polygon will be "}".

27.1.111

Point3d_mdtl **r_Point3D** ()

Reads Point3d

Reads Point3d.
Pre: mode == READ
Post: Point3d read.

27.1.112

MaterialS_mdtl **r_matS** ()

Reads surface material

Reads surface material.
Pre: mode == READ
Post: Surface material read.

27.1.113

MaterialS_mdtl **r_matS_name** (PPString)

Reads surface material with the given name

Reads surface material with the given name.
Pre: mode == READ
Post: Surface material read.

27.1.114

AttrType **AttType** ()

Reads next attribute type

Reads next attribute type.
Pre: mode == READ && reading an attribute.
Post: Attribute type read.
Returns VOID_ATTR if finished reading attributes.

27.1.115

PropType **r_PropType** ()

Reads next phisical propperty of a surface material type

Reads next phisical propperty of a surface material type.
Pre: mode == READ && reading a material.
Post: Attribute type read.
Returns VOID_PROP if finished reading phisical propperties.

27.1.116

Color_mdtl **r_Color** ()

Reads color

Reads color.
Pre: mode == READ && reading a phisical propperty.
Post: Color read.

27.1.117

void **r_TextAttr** (PPString&, PPString&)

Reads textual attribute

Reads textual attribute.
Pre: mode == READ && reading attributes.
Post: Textual attribute read.

27.1.118

Texture_mdtl **r_Texture** ()

Reads texture

Reads texture.
Pre: mode == READ && reading attributes.
Post: Texture read.

27.1.119

bool **UseEntity** (PPString n)

Allows the use of the entity with name "n" if the state of reading permits it

Allows the use of the entity with name "n" if the state of reading permits it. This means that an entity can only be referenced this way when, if using no references, an entity of this kind could be declared here. This function serves to reference already defined entities without having to declare it again.

27.1.120

bool **UseMaterialS** (PPString n)

Allows the use of the surface material entity with name "n" if the state of reading permits it

Allows the use of the surface material entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.121

bool **UseBoundary** (PPString n)

Allows the use of the boundary entity with name "n" if the state of reading permits it

Allows the use of the boundary entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.122

bool **UseSolidBr** (PPString n)

Allows the use of the SolidBr entity with name "n" if the state of reading permits it

Allows the use of the SolidBr entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.123

bool **UseAttrib** (PPString n)

Allows the use of the attribute entity with name "n" if the state of reading permits it

Allows the use of the attribute entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.124

bool **UseCamera** (PPString n)

Allows the use of the camera entity with name "n" if the state of reading permits it

Allows the use of the camera entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.125

bool **UsePoint2d** (PPString n)

Allows the use of the Point2d entity with name "n" if the state of reading permits it

Allows the use of the Point2d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.126

bool **UsePoint3d** (PPString n)

Allows the use of the Point3d entity with name "n" if the state of reading permits it

Allows the use of the Point3d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.127

bool **UsePoint4d** (PPString n)

Allows the use of the Point4d entity with name "n" if the state of reading permits it

Allows the use of the Point4d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.128

bool **UsePolygon** (PPString n)

Allows the use of the polygon entity with name "n" if the state of reading permits it

Allows the use of the polygon entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.129

bool **UseWalkthrough** (PPString n)

Allows the use of the walkthrough entity with name "n" if the state of reading permits it

Allows the use of the walkthrough entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.130

bool **UseShell** (PPString n)

Allows the use of the shell entity with name "n" if the state of reading permits it

Allows the use of the shell entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.131

bool **UseFace** (PPString n)

Allows the use of the face entity with name "n" if the state of reading permits it

Allows the use of the face entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.132

bool **UsePlane** (PPString n)

Allows the use of the plane entity with name "n" if the state of reading permits it

Allows the use of the plane entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.133

bool **UseSurf** (PPString n)

Allows the use of the surface entity with name "n" if the state of reading permits it

Allows the use of the surface entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.134

bool **UseNurb** (PPString n)

Allows the use of the NURBS entity with name "n" if the state of reading permits it

Allows the use of the NURBS entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.135

bool **UseKnots** (PPString n)

Allows the use of the Knots entity with name "n" if the state of reading permits it

Allows the use of the Knots entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.136

bool **UseTrimmingC** (PPString n)

Allows the use of the trimming curve entity with name "n" if the state of reading permits it

Allows the use of the trimming curve entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.137

bool **UsePCurve2d** (PPString n)

Allows the use of the PCurve2d entity with name "n" if the state of reading permits it

Allows the use of the PCurve2d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.138

bool **UseBSpline2d** (PPString n)

Allows the use of the BSpline2d entity with name "n" if the state of reading permits it

Allows the use of the BSpline2d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.139

bool **UseCurve3d** (PPString n)

Allows the use of the Curve3d entity with name "n" if the state of reading permits it

Allows the use of the Curve3d entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.140

bool **UseSphere** (PPString n)

Allows the use of the Sphere entity with name "n" if the state of reading permits it

Allows the use of the Sphere entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.141

bool **UseCylinder** (PPString n)

Allows the use of the cylinder entity with name "n" if the state of reading permits it

Allows the use of the cylinder entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.142

bool **UseCone** (PPString n)

Allows the use of the cone entity with name "n" if the state of reading permits it

Allows the use of the cone entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.143

bool **UsePrism** (PPString n)

Allows the use of the prism entity with name "n" if the state of reading permits it

Allows the use of the prism entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.144

bool **UseSolidEO** (PPString n)

Allows the use of the extended octree entity with name "n" if the state of reading permits it

Allows the use of the extended octree entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.145

bool **UseSolidFO** (PPString n)

Allows the use of the face octree entity with name "n" if the state of reading permits it

Allows the use of the face octree entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.146

bool **UseSolidMDCO** (PPString n)

Allows the use of the maximum division classical octree entity with name "n" if the state of reading permits it

Allows the use of the maximum division classical octree entity
with name "n" if the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.147

bool **UseVector** (PPString n)

Allows the use of the vector entity with name "n" if the state of reading permits it

Allows the use of the vector entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.148

bool **UseRestriction** (PPString n)

Allows the use of the restriction entity with name "n" if the state of reading permits it

Allows the use of the restriction entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.149

bool **UsePipe** (PPString n)

Allows the use of the pipe entity with name "n" if the state of reading permits it

Allows the use of the pipe entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.150

bool **UseMultRes** (PPString n)

Allows the use of the multi resolution entity with name "n" if the state of reading permits it

Allows the use of the multi resolution entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.151

bool **UseGroup** (PPString n)

Allows the use of the group entity with name "n" if the state of reading permits it

Allows the use of the group entity with name "n" if
the state of reading permits it.
Returns false if the name of the entity given belongs to a different
kind of entity. If it does not exist it will be stored
as an entity pending to be declared.

27.1.152

PPString **DeclGroup** ()

Starts declaring a group

Starts declaring a group.
Pre: mode == WRITE
Post: Group declared.
When a group is declared any entity can be declared inside of it.
Returns the new name of the entity if it has been declared

as an independent entity, otherwise it returns "VOID_NAME".

27.1.153

void **DeclGroup_name** (PPString n)

Starts declaring a group with the given name

Starts declaring a group with the given name.
Pre: mode == WRITE
Post: Group declared.
When a group is declared any entity can be declared inside of it.

27.1.154

void **EndGroup** ()

Finishes declaration of a group

Finishes declaration of a group.
Pre: mode == WRITE && declaring a group.
Post: Group finished.

27.1.155

PPString **r_Group** ()

Starts reading a group

Starts reading a group.
Pre: mode == READ
Post: Group declaration read.

Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.156

void **r_Group_name** (PPString name)

Starts reading a group with the given name

Starts reading a group with the given name.
Pre: mode == READ
Post: Group declaration read.

27.1.157

EntityType **r_nextGroup** (PPString& name)

Reads the type of the next entity in the group

Reads the type of the next entity in the group.
Pre: mode == READ && reading a group
Post: Entity type read.
If there are no more entities left VOID_MDTL is returned.

27.1.158

PPString **DeclMultRes** ()

Starts declaring a multi resolution

Starts declaring a multi resolution.
Pre: mode == WRITE
Post: MultRes declared.
When a multi-resolution is declared any entity can be declared inside of it.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.159

void **DeclMultRes_name** (PPString name)

Starts declaring a multi resolution with the given name

Starts declaring a multi resolution with the given name.
Pre: mode == WRITE
Post: MultRes declared.
When a multi-resolution is declared any entity can be declared inside of it.

27.1.160

void **EndMultRes** ()

Finishes declaration of a multi resolution

Finishes declaration of a multi resolution.
Pre: mode == WRITE && declaring a multi-resolution
Post: MultRes finished.

27.1.161

void **DeclRes** (double tol, PPString entity)

Writes a new entity in the multi-resolution giving its tolerance

Writes a new entity in the multi-resolution giving its tolerance.
It uses the entity with the name given. This entity must be already declared.
Pre: mode == WRITE && declaring a multi-resolution.
Post: Resolution written.

27.1.162

void **DeclRes** (double tol)

Writes a new tolerance for a new entity

Writes a new tolerance for a new entity.
Pre: mode == WRITE
Post: Resolution declared.

27.1.163

PPString **r_MultRes** ()

Starts reading a multi-resolution

Starts reading a multi-resolution.
Pre: mode == READ
Post: Multi-resolution declaration read.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.164

void **r_MultRes_name** (PPString name)

Starts reading a multi-resolution with the given name

Starts reading a multi-resolution with the given name.
Pre: mode == READ
Post: Multi-resolution declaration read.

27.1.165

EntityType **r_nextMultRes** (double& res, PPString& name)

Reads the type of the next entity in the multi-resolution

Reads the type of the next entity in the multi-resolution.
Pre: mode == READ && reading a multi-resolution
Post: Entity type read.
If there are no more entities left VOID_MDTL is returned.

27.1.166

PPString **DeclCamera** (const Point3d_mdtl&, const Point3d_mdtl&, const Vector_mdtl&, double, double)

Writes a camera giving the observer, center, direction and angles by parameter

Writes a camera giving the observer, center, direction and angles by parameter.
Pre: mode == WRITE
Post: Camera written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.167

PPString **DeclCamera** (const Camera_mdtl&)

Writes a camera

Writes a camera.
Pre: mode == WRITE

Post: Camera written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.168

Camera_mdtl **r_Camera** ()

Reads a camera

Reads a camera.
Pre: mode == READ
Post: Camera read.

27.1.169

Camera_mdtl **r_Camera_name** (PPString name)

Reads a camera with the given name

Reads a camera with the given name.
Pre: mode == READ
Post: Camera read.

27.1.170

PPString **DeclPoint3d** (const Point3d_mdtl&)

Writes a Point3d

Writes a Point3d.
Pre: mode == WRITE
Post: Point3d written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

27.1.171

PPString **DeclPoint2d** (const Point2d_mdtl&)

Writes a Point2d

Writes a Point2d.
Pre: mode == WRITE
Post: Point2d written.
Returns the new name of the entity if it has been declared
as an independent entity, otherwise it returns "VOID_NAME".

Class Graph