GeometronLib

1.00

Generated by Doxygen 1.8.11

Contents

Chapter 1

GeometronLib 1.00 Alpha Documentation

The GeometronLib provides basic functionality for 2D and 3D geometrical objects, such as mesh generation, basic collision detection, and respective data structures for lines, rays, spheres etc.

Prerequisites:

- `GaussianLib` header files

Features:

- **AABB** (Axis-Aligned Bounding-Box)
- **OBB** (Oriented Bounding-Box)
- **Line**
- **Ray**
- **Transform2** (3x3 Matrix Manager for 2D Transformations)
- **Transform3** (4x4 Matrix Manager for 3D Transformations)
- **Frustum** (Frustum of Pyramid)
- **Projection** (4x4 Projection Matrix Manager)
- **Sphere**
- **Spline**
- **TriangleMesh**
- **MeshGenerator**
- **BezierCurve**
- **BezierTriangle**

Chapter 2

Todo List

Class Gm::BezierTriangleT< T >

This is incomplete

Class Gm::Spline< P, T >

!!!OPTIMIZE THIS!!!

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Namespace Documentation

6.1 Gm::MeshGenerator Namespace Reference

Namespace with all mesh generation functions.

Classes

- struct CapsuleDescription
- struct ConeDescription
- struct CuboidDescription
- struct CylinderDescription
- struct EllipsoidDescription
- struct IcoSphereDescription
- struct SpiralDescription
- struct TorusDescription
- struct TubeDescription

Functions

- TriangleMesh **Cuboid** (const CuboidDescription &desc)

6.1.1 Detailed Description

Namespace with all mesh generation functions.

Chapter 7

Class Documentation

7.1 Gm::AABB< Vec, T > Class Template Reference

Base AABB (Axis-Aligned Bounding-Box) class.

`#include <AABB.h>`

Public Types

- using **ThisType** = AABB< Vec, T >

Public Member Functions

- AABB ()
- **AABB** (const Vec< T > &min, const Vec< T > &max)
- void Reset (const Vec< T > &point)

  Sets the minimum and maximum to the specified point.
- void **Insert** (const Vec< T > &point)
- void **Insert** (const ThisType &aabb)
- void **Repair** ()
- Vec< T > **Size** () const
- Vec< T > **Center** () const
- std::vector< Line< Vec< T > > > Edges () const

  Returns the list of all edges of this AABB.

Public Attributes

- Vec< T > **min**
- Vec< T > **max**

7.1.1 Detailed Description

template<template< typename > class Vec, typename T>
class Gm::AABB< Vec, T >

Base AABB (Axis-Aligned Bounding-Box) class.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 template<template< typename > class Vec, typename T> Gm::AABB< Vec, T >::AABB ( ) [inline]

Constructs a maximal invald bounding-box, i.e. min has the maximal values possible, and max has the minimal values possible.

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

- AABB.h

7.2 Gm::BezierCurve< T > Class Template Reference

Curve in BB-Form (Bernstein Bezier).

`#include <BezierCurve.h>`

7.2.1 Detailed Description

template<typename T>
class Gm::BezierCurve< T >

Curve in BB-Form (Bernstein Bezier).

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

- BezierCurve.h

7.3 Gm::BezierTriangleT< T > Class Template Reference

Curved triangle patch in BB-Form (Bernstein Bezier).

`#include <BezierTriangle.h>`

Classes

- struct Evaluation

Public Member Functions

- Evaluation **operator()** (const Gs::Vector2T< T > &uv) const
- Evaluation **Evaluate** (const Gs::Vector2T< T > &uv) const
- const Gs::Vector3T< T > & **GetControlPoint** (std::size_t i, std::size_t j) const
- Gs::Vector3T< T > & **GetControlPoint** (std::size_t i, std::size_t j)
- const std::vector< Gs::Vector3T< T > > & **GetControlPoints** () const
- void **SetOrder** (std::size_t order)
- std::size_t **GetOrder** () const

7.3.1 Detailed Description

template<typename T>
class Gm::BezierTriangleT< T >

Curved triangle patch in BB-Form (Bernstein Bezier).

Todo This is incomplete

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

- BezierTriangle.h

7.4 Gm::MeshGenerator::CapsuleDescription Struct Reference

Public Attributes

- Gs::Vector3 radius = Gs::Vector3(Gs::Real(0.5))

  Radius of the top- and bottom half-ellipsoids in X, Y, and Z direction. By default (0.5, 0.5, 0.5).
- Gs::Real height = Gs::Real(1)

  Capsule height (without top- and bottom half-sphere). By default 1.
- Gs::Vector2ui mantleSegments = Gs::Vector2ui(20, 1)

  Segmentation around the cylinder (x component), and height (y component). By default (20, 1).
- unsigned int ellipsoidSegments = 10

7.4.1 Member Data Documentation

7.4.1.1 unsigned int Gm::MeshGenerator::CapsuleDescription::ellipsoidSegments = 10

Segmentation of the top- and bottom half-ellipsoids. Each component will be clamped to [3, +inf). By default 10.

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

- MeshGenerator.h

7.5 Gm::ClippedPolygon< T > Struct Template Reference

Public Member Functions

- void **AddVertex** (const Gs::Vector3T< T > &vertex)

Public Attributes

- unsigned char count = 0

  Number of vertices, used for this clipped triangle. This is either 3 or 4.
- std::array< Gs::Vector3T< T >, 4 > **vertices**

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

- TriangleCollision.h

7.6 Gm::MeshGenerator::ConeDescription Struct Reference

Public Attributes

- Gs::Vector2 radius = Gs::Vector2(Gs::Real(0.5))

  Cone radius in U (x component), and V (y component) direction. By default (0.5, 0.5).
- Gs::Real height = Gs::Real(1)

  Cone height. By default 1.
- Gs::Vector2ui mantleSegments = Gs::Vector2ui(20, 1)

  Segmentation around the cone (x component), and height (y component). By default (20, 1).
- unsigned int capSegments = 1

  Segmentation of the bottom cap. By default 1.

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

- MeshGenerator.h

7.7 Gm::Spline< P, T >::ControlPoint Struct Reference

Public Attributes

- P **point**
- T **interval**

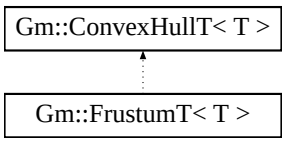
The documentation for this struct was generated from the following file:

- Spline.h

7.8 Gm::ConvexHullT< T > Class Template Reference

`#include <ConvexHull.h>`

Inheritance diagram for Gm::ConvexHullT< T >:



Public Member Functions

- **ConvexHullT** (std::size_t planeCount)
- void Normalize ()

  Normalizes all planes of this convex hull.
- bool IsPointInside (const Gs::Vector3T< T > &point)

  Returns true if the specified point is inside the convex hull.
- bool IsSphereInside (const SphereT< T > &sphere)

  Returns true if the specified sphere is inside the convex hull (or just intersecting one of its planes).

Public Attributes

- std::vector< PlaneT< T > > planes

7.8.1 Detailed Description

template<typename T>
class Gm::ConvexHullT< T >

Convex hull base class. Here a convex hull is constructed so that all plane normals point out of the hull.

7.8.2 Member Function Documentation

7.8.2.1 template<typename T > void Gm::ConvexHullT< T >::Normalize ( ) [inline]

Normalizes all planes of this convex hull.

See also

PlaneT::Normalize

7.8.3 Member Data Documentation

7.8.3.1 template<typename T > std::vector< PlaneT<T> > Gm::ConvexHullT< T >::planes

List of all planes which form the convex hull. This must be at least 3 planes to form a valid convex hull.

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

- ConvexHull.h

7.9 Gm::MeshGenerator::CuboidDescription Struct Reference

Public Attributes

- Gs::Vector3 size = Gs::Vector3(Gs::Real(1.0))

  Cuboid size. By default (1, 1, 1).
- Gs::Vector3 uvScale = Gs::Vector3(Gs::Real(1.0))

  Cuboid texture UV scaling (X, Y), (Z, Y), (X, Z).
- Gs::Vector3ui segments = Gs::Vector3ui(1, 1, 1)

  Cuboid segmentation. Each component will be clamped to [1, +inf). By default (1, 1, 1).
- bool alternateGrid = true

  Specifies whether the face grids are to be alternating or uniform. By default true.

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

- MeshGenerator.h

7.10 Gm::MeshGenerator::CylinderDescription Struct Reference

Public Attributes

- Gs::Vector2 radius = Gs::Vector2(Gs::Real(0.5))

  Cylinder radius in U (x component), and V (y component) direction. By default (0.5, 0.5).
- Gs::Real height = Gs::Real(1)

  Cylinder height. By default 1.
- Gs::Vector2ui mantleSegments = Gs::Vector2ui(20, 1)

  Segmentation around the cylinder (x component), and height (y component). By default (20, 1).
- unsigned int capSegments = 1

  Segmentation of the top- and bottom cap. By default 1.

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

- MeshGenerator.h

7.11 Gm::MeshGenerator::EllipsoidDescription Struct Reference

Public Attributes

- Gs::Vector3 radius = Gs::Vector3(Gs::Real(0.5))

  Radius in X, Y, and Z direction. By default (0.5, 0.5, 0.5).
- Gs::Vector2ui segments = Gs::Vector2ui(20, 10)

7.11.1 Member Data Documentation

7.11.1.1 Gs::Vector2ui Gm::MeshGenerator::EllipsoidDescription::segments = Gs::Vector2ui(20, 10)

Segmentation in U (x component), and V (y component) direction. Each component will be clamped to [3, +inf). By default (20, 10).

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

- MeshGenerator.h

7.12 Gm::BezierTriangleT< T >::Evaluation Struct Reference

Public Attributes

- Gs::Vector3T< T > **point**
- Gs::Vector3T< T > **normal**

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

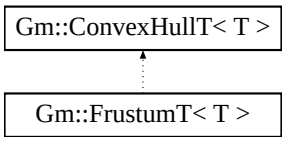
- BezierTriangle.h

7.13 Gm::FrustumT< T > Class Template Reference

Base frustum class.

`#include <Frustum.h>`

Inheritance diagram for Gm::FrustumT< T >:



Public Member Functions

- **FrustumT** (const Gs::Matrix4T< T > &m)
- void **SetFromMatrix** (const Gs::Matrix4T< T > &m)
- Gs::Vector3T< T > LeftTop () const

  Returns the left-top corner on the far plane.
- Gs::Vector3T< T > LeftBottom () const

  Returns the left-bottom corner on the far plane.
- Gs::Vector3T< T > RightTop () const

  Returns the right-top corner on the far plane.
- Gs::Vector3T< T > RightBottom () const

  Returns the right-bottom corner on the far plane.
- const PlaneT< T > & GetPlane (const FrustumPlane plane) const

  Returns the specified plane of this frustum.
- PlaneT< T > & GetPlane (const FrustumPlane plane)

  Returns the specified plane of this frustum.
- AABB3T< T > GetBoundingBox (const Gs::Vector3T< T > &origin) const

  Computes the bounding box of this frustum with the specified origin of the frustum.

7.13.1 Detailed Description

template<typename T>
class Gm::FrustumT< T >

Base frustum class.

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

- Frustum.h

7.14 Gm::MeshGenerator::IcoSphereDescription Struct Reference

Public Attributes

- Gs::Real radius = Gs::Real(0.5)

  Radius of the ico-sphere. By default 0.5.
- unsigned char segments = 3

  Segmentation as tessellation factor, which will be clamped to [0, 255]. By default 3.

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

- MeshGenerator.h

7.15 Gm::Line< T > Class Template Reference

Base line class.

`#include <Line.h>`

Public Member Functions

- **Line** (const Line< T > &)=default
- **Line** (const T &a, const T &b)
- **Line** (Gs::UninitializeTag)

Public Attributes

- T **a**
- T **b**

7.15.1 Detailed Description

template<typename T>
class Gm::Line< T >

Base line class.

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

- Line.h

7.16 Gm::Line< Gs::Vector2T< T > > Class Template Reference

Specialized line class with 2D vectors.

`#include <Line.h>`

Public Member Functions

- **Line** (const Line< Gs::Vector2T< T > > &)=default
- **Line** (const Gs::Vector2T< T > &a, const Gs::Vector2T< T > &b)
- **Line** (Gs::UninitializeTag)
- Gs::Vector2T< T > **Direction** () const
- Gs::Vector2T< T > **Lerp** (const T &t) const
- T **LengthSq** () const
- T **Length** () const

Public Attributes

- Gs::Vector2T< T > **a**
- Gs::Vector2T< T > **b**

7.16.1 Detailed Description

template<typename T>
class Gm::Line< Gs::Vector2T< T > >

Specialized line class with 2D vectors.

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

- Line.h

7.17 Gm::Line< Gs::Vector3T< T > > Class Template Reference

Specialized line class with 3D vectors.

`#include <Line.h>`

Public Member Functions

- **Line** (const Line< Gs::Vector3T< T > > &)=default
- **Line** (const Gs::Vector3T< T > &a, const Gs::Vector3T< T > &b)
- **Line** (Gs::UninitializeTag)
- Gs::Vector3T< T > **Direction** () const
- Gs::Vector3T< T > **Lerp** (const T &t) const
- T **LengthSq** () const
- T **Length** () const

Public Attributes

- Gs::Vector3T< T > **a**
- Gs::Vector3T< T > **b**

7.17.1 Detailed Description

template<typename T>
class Gm::Line< Gs::Vector3T< T > >

Specialized line class with 3D vectors.

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

- Line.h

7.18 Gm::OBB< Vec, T > Class Template Reference

Base OBB (Oriented Bounding-Box) class.

`#include <OBB.h>`

Public Member Functions

- **OBB** (Gs::UninitializeTag)
- **OBB** (const Vec< T > &min, const Vec< T > &max)
- **OBB** (const Vec< T > &center, const Vec< T > &xAxis, const Vec< T > &yAxis, const Vec< T > &zAxis)
- void **UpdateHalfSize** ()

Public Attributes

- Vec< T > **center**
- Vec< T > **halfSize**
- Vec< Vec< T > > **axes**

7.18.1 Detailed Description

template<template< typename > class Vec, typename T>
class Gm::OBB< Vec, T >

Base OBB (Oriented Bounding-Box) class.

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

- OBB.h

7.19 Gm::PlaneT< T > Class Template Reference

Plane base class with components: 'normal' and 'distance'.

`#include <Plane.h>`

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("PlaneT")
- **PlaneT** (const Gs::Vector3T< T > &a, const Gs::Vector3T< T > &b, const Gs::Vector3T< T > &c)
- **PlaneT** (const T &x, const T &y, const T &z, const T &d)
- **PlaneT** (const Triangle3T< T > &triangle)
- **PlaneT** (const Gs::Vector3T< T > &normal, const T &distance)
- **PlaneT** (Gs::UninitializeTag)
- void Build (const Gs::Vector3T< T > &a, const Gs::Vector3T< T > &b, const Gs::Vector3T< T > &c)

  Builds this plane with the three specified points.
- void Build (const Gs::Vector3T< T > &normal, const Gs::Vector3T< T > &memberPoint)

  Builds this plane with the specified normal and member point (which lies onto the plane).
- void UpdateDistance (const Gs::Vector3T< T > &memberPoint)

  Updates the (signed) distance for the new specified member point.
- void Normalize ()

  Normalizes the normal vector and distance of this plane.
- Gs::Vector3T< T > MemberPoint () const

  Returns a point which lies onto this plane: normal ∗ distance;.
- void Flip ()

  Flips this plane.
- PlaneT< T > Flipped () const

  Returns a flipped instance of this plane.
- template<typename C >
  PlaneT< C > **Cast** () const

Public Attributes

- Gs::Vector3T< T > normal

  Normal vector of the plane.
- T distance

  Signed distance to the origin of the coordinate system.

7.19.1 Detailed Description

template<typename T>
class Gm::PlaneT< T >

Plane base class with components: 'normal' and 'distance'.

Template Parameters

T	Specifies the data type of the vector components. This should be a primitive data type such as float or double.

Remarks

The plane equation is: ax + by + cz + d = 0, where (a, b, c) is a point on the plane, (x, y, z) is the normal vector and d is the (signed) distance to the origin.

7.19.2 Member Function Documentation

7.19.2.1 template<typename T> Gs::Vector3T<T> Gm::PlaneT< T >::MemberPoint ( ) const [inline]

Returns a point which lies onto this plane: normal ∗ distance;.

Remarks

This point is the closest point from the plane to the origin of the coordinate system.

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

- Plane.h

7.20 Gm::UniformSpline< P, T >::Polynomial Struct Reference

Polynomial structure with four coefficients.

`#include <UniformSpline.h>`

Public Member Functions

- const P & **operator[ ]** (std::size_t idx) const
- P & **operator[ ]** (std::size_t idx)
- P **Evaluate** (const T &t) const

Public Attributes

- std::array< P, 4 > **coeff**

7.20.1 Detailed Description

template<typename P, typename T>
struct Gm::UniformSpline< P, T >::Polynomial

Polynomial structure with four coefficients.

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

- UniformSpline.h

7.21 Gm::ProjectionT< T > Class Template Reference

projection class.

`#include <Projection.h>`

Public Types

- using **MatrixType** = Gs::ProjectionMatrix4T< T >

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("ProjectionT")
- void SetNear (const T &near)

  Sets the near clipping plane.
- const T & GetNear () const

  Returns the near clipping plane.
- void SetFar (const T &far)

  Sets the far clipping plane.
- const T & GetFar () const

  Returns the far clipping plane.
- void SetFOV (const T &fov)

  Sets the field-of-view (FOV) in radians. By default (74∗pi/180).
- const T & GetFOV () const

  Returns the field-of-view (FOV) in radians.
- void SetAspect (const T &aspect)

Sets the aspect ratio.

- const T & GetAspect () const

  Returns the aspect ratio.

- void SetOrtho (bool isOrtho)

  Specifies whether the projection is orthogonal or perspective. By default perspective.

- bool GetOrtho () const

  Returns true if this projection is orthogonal.

- void SetOrthoSize (const Gs::Vector2T< T > &orthoSize)

  Sets the size of the orthogonal projection.

- const Gs::Vector2T< T > & GetOrthoSize () const

  Returns the size of the orthogonal projection.

- void SetFlags (int flags)

  Sets the projection matrix flags. By default 0.

- int GetFlags () const

  Returns the projection matrix flags.

- const MatrixType & GetMatrix () const

  Returns the projection matrix.

- void GetMatrix (MatrixType &matrix, int flags) const

  Returns the projection matrix with the specified flags.

7.21.1 Detailed Description

template<typename T>
class Gm::ProjectionT< T >

projection class.

Note

This class can not be used with multi-threading!

7.21.2 Member Function Documentation

7.21.2.1 template<typename T > int Gm::ProjectionT< T >::GetFlags ( ) const [inline]

Returns the projection matrix flags.

See also

Gs::ProjectionFlags

7.21.2.2 template<typename T > void Gm::ProjectionT< T >::GetMatrix (MatrixType & *matrix,* int *flags*) const [inline]

Returns the projection matrix with the specified flags.

See also

Gs::ProjectionFlags

7.21.2.3 template<typename T > void Gm::ProjectionT< T >::SetFlags (int *flags*) [inline]

Sets the projection matrix flags. By default 0.

See also

Gs::ProjectionFlags

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

- Projection.h

7.22 Gm::Ray< Vec, T > Class Template Reference

Ray base class.

`#include <Ray.h>`

Public Member Functions

- **Ray** (const Vec< T > &origin, const Vec< T > &direction)
- **Ray** (Gs::UninitializeTag)
- Vec< T > **Lerp** (const T &t) const

Public Attributes

- Vec< T > **origin**
- Vec< T > **direction**

7.22.1 Detailed Description

template<template< typename > class Vec, typename T>
class Gm::Ray< Vec, T >

Ray base class.

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

- Ray.h

7.23 Gm::SphereT< T > Class Template Reference

Base sphere class.

`#include <Sphere.h>`

Public Member Functions

- T **GetVolume** () const
- void **SetVolume** (const T &volume)
- T **GetArea** () const
- void **SetArea** (const T &area)

Public Attributes

- Gs::Vector3T< T > **origin**
- T **radius**

7.23.1 Detailed Description

template<typename T>
class Gm::SphereT< T >

Base sphere class.

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

- Sphere.h

7.24 Gm::MeshGenerator::SpiralDescription Struct Reference

Public Attributes

- Gs::Vector2 tubeRadius = Gs::Vector2(Gs::Real(0.25))

  Radius of the tube in U (x component), and V (y component) direction. By default (0.25, 0.25).
- Gs::Real twistDisplacement = Gs::Real(1)

  The displacement for each (360 degree) twist. By default 1.
- Gs::Real twistCount = Gs::Real(1)
- Gs::Vector2ui segments = Gs::Vector2ui(20, 10)

7.24.1 Member Data Documentation

7.24.1.1 Gs::Vector2ui Gm::MeshGenerator::SpiralDescription::segments = Gs::Vector2ui(20, 10)

Segmentation in U (x component), and V (y component) direction. Each component will be clamped to [3, +inf). By default (20, 10).

7.24.1.2 Gs::Real Gm::MeshGenerator::SpiralDescription::twistCount = Gs::Real(1)

Count of twists (in percent, i.e. 1.0 is a single twist, 2.5 are two and a half twist). By default 1.

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

- MeshGenerator.h

7.25 Gm::Spline< P, T > Class Template Reference

Spline base class.

`#include <Spline.h>`

Classes

- struct ControlPoint

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("Spline")
- P **operator()** (const T &t) const
- const ControlPoint & **operator[ ]** (std::size_t idx) const
- ControlPoint & **operator[ ]** (std::size_t idx)
- P **Evaluate** (const T &t) const
- int **GetOrder** () const
- void **SetOrder** (int order)
- void AddPoint (const P &point, const T &t)

  Adds a new control point.
- const std::vector< ControlPoint > & GetPoints () const

  Returns the list of all control points.

7.25.1 Detailed Description

template<typename P, typename T>
class Gm::Spline< P, T >

Spline base class.

Template Parameters

P	Specifies the type of the spline control points.
T	Specifies the base data type. This should be float or double.

Todo !!!OPTIMIZE THIS!!!

7.25.2 Member Function Documentation

7.25.2.1 template<typename P, typename T> void Gm::Spline< P, T >::AddPoint (const P & *point*, const T & *t*) [inline]

Adds a new control point.

Parameters

in	*point*	Specifies the point position.
in	*t*	Specifies the interpolation factor (or interval value).

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

- Spline.h

7.26 Gm::MeshGenerator::TorusDescription Struct Reference

Public Attributes

- Gs::Vector2 holeRadius = Gs::Vector2(Gs::Real(0.5))

  Radius of the hole in X, and Y direction. By default (0.5, 0.5).
- Gs::Vector2 ringRadius = Gs::Vector2(Gs::Real(0.25))

  Radius of the ring in U (x component), and V (y component) direction. By default (0.25, 0.25).
- Gs::Vector2ui segments = Gs::Vector2ui(20, 10)

7.26.1 Member Data Documentation

7.26.1.1 Gs::Vector2ui Gm::MeshGenerator::TorusDescription::segments = Gs::Vector2ui(20, 10)

Segmentation in U (x component), and V (y component) direction. Each component will be clamped to [3, +inf). By default (20, 10).

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

- MeshGenerator.h

7.27 Gm::Transform2T< T > Class Template Reference

2D transformation class.

`#include <Transform2.h>`

Public Types

- using **MatrixType** = Gs::AffineMatrix3T< T >

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("Transform2T")
- void **SetPosition** (const Gs::Vector2T< T > &position)
- const Gs::Vector2T< T > & **GetPosition** () const
- void **SetRotation** (const T &rotation)
- const T & **GetRotation** () const
- void **SetScale** (const Gs::Vector2T< T > &scale)
- const Gs::Vector2T< T > & **GetScale** () const
- const MatrixType & **GetMatrix** () const
- void **Turn** (const T &rotation, const Gs::Vector2T< T > &pivot)

7.27.1 Detailed Description

template<typename T>
class Gm::Transform2T< T >

2D transformation class.

Note

This class can not be used with multi-threading!

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

- Transform2.h

7.28 Gm::Transform3T< T > Class Template Reference

3D transformation class.

`#include <Transform3.h>`

Public Types

- using **MatrixType** = Gs::AffineMatrix4T< T >

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("Transform3T")
- void **SetPosition** (const Gs::Vector3T< T > &position)
- const Gs::Vector3T< T > & **GetPosition** () const
- void **SetRotation** (const Gs::QuaternionT< T > &rotation)
- const Gs::QuaternionT< T > & **GetRotation** () const
- void **SetScale** (const Gs::Vector3T< T > &scale)
- const Gs::Vector3T< T > & **GetScale** () const
- const MatrixType & **GetMatrix** () const
- void MoveGlobal (const Gs::Vector3T< T > &direction)

  Moves this transformation into the specified direction.
- void MoveLocal (const Gs::Vector3T< T > &direction)

  Moves this transformation into the specified direction with respect to the current rotation.
- void Turn (const Gs::QuaternionT< T > &rotation, const Gs::Vector3T< T > &pivot)

  Turns this transformation with the specified (reltative) angles around the (global) pivot.

7.28.1 Detailed Description

template<typename T>
class Gm::Transform3T< T >

3D transformation class.

Note

This class can not be used with multi-threading!

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

- Transform3.h

7.29 Gm::Triangle< T > Class Template Reference

Triangle base class.

`#include <Triangle.h>`

Public Member Functions

- **Triangle** (const Triangle< T > &)=default
- **Triangle** (const T &a, const T &b, const T &c)
- **Triangle** (Gs::UninitializeTag)
- T & **operator[ ]** (std::size_t vertex)
- const T & **operator[ ]** (std::size_t vertex) const

Public Attributes

- T **a**
- T **b**
- T **c**

7.29.1 Detailed Description

template<typename T>
class Gm::Triangle< T >

Triangle base class.

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

- Triangle.h

7.30 Gm::Triangle< Gs::Vector2T< T > > Class Template Reference

Template specializationn for 2D triangles.

`#include <Triangle.h>`

Public Member Functions

- **Triangle** (const Triangle< Gs::Vector2T< T > > &)=default
- **Triangle** (const Gs::Vector2T< T > &a, const Gs::Vector2T< T > &b, const Gs::Vector2T< T > &c)
- **Triangle** (Gs::UninitializeTag)
- Gs::Vector2T< T > & **operator[ ]** (std::size_t vertex)
- const Gs::Vector2T< T > & **operator[ ]** (std::size_t vertex) const
- T **Area** () const
- Gs::Vector2T< T > Barycentric (const Gs::Vector3T< T > &barycentricCoords) const

  Computes the cartesian coordinate by the specified barycentric coordinate with respect to this triangle.
- Triangle< Gs::Vector2T< T > > Barycentric (const Triangle< Gs::Vector3T< T > > &barycentricCoords) const

  Computes the triangle with cartesian coordinates by the specified triangle with barycentric coordinates with respecti to this triangle.
- T Angle (std::size_t vertex) const

  Returns the angle (in radians) of the specified triangle vertex (0, 1, or 2).

Public Attributes

- Gs::Vector2T< T > **a**
- Gs::Vector2T< T > **b**
- Gs::Vector2T< T > **c**

7.30.1 Detailed Description

template<typename T>
class Gm::Triangle< Gs::Vector2T< T > >

Template specializationn for 2D triangles.

7.30.2 Member Function Documentation

7.30.2.1 template<typename T > Gs::Vector2T<T> Gm::Triangle< Gs::Vector2T< T > >::Barycentric (const Gs::Vector3T< T > & *barycentricCoords*) const [inline]

Computes the cartesian coordinate by the specified barycentric coordinate with respect to this triangle.

Parameters

in	*barycentricCoords*	Specifies the barycentric coordinates with respect to this triangle. The sum of all components must be one, i.e. x+y+z = 1.

7.30.2.2 template<typename T > Triangle< Gs::Vector2T<T> > Gm::Triangle< Gs::Vector2T< T > >::Barycentric (const Triangle< Gs::Vector3T< T > > & *barycentricCoords*) const [inline]

Computes the triangle with cartesian coordinates by the specified triangle with barycentric coordinates with respecti to this triangle.

Parameters

in	*barycentricCoords*	Specifies the triangle with barycentric coordinates with respect to this triangle. If this input parameter is { { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 } }, the result is equal to this triangle. The sum of all components must be one for each triangle vertex, i.e. x+y+z = 1.

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

- Triangle.h

7.31 Gm::Triangle< Gs::Vector3T< T > > Class Template Reference

Template specializationn for 3D triangles.

`#include <Triangle.h>`

Public Member Functions

- **Triangle** (const Triangle< Gs::Vector3T< T > > &)=default
- **Triangle** (const Gs::Vector3T< T > &a, const Gs::Vector3T< T > &b, const Gs::Vector3T< T > &c)
- **Triangle** (Gs::UninitializeTag)
- Gs::Vector3T< T > & **operator[ ]** (std::size_t vertex)
- const Gs::Vector3T< T > & **operator[ ]** (std::size_t vertex) const
- T Area () const

  Returns the area of this triangle.
- Gs::Vector3T< T > Normal () const

  Returns the normal vector of this triangle.
- Gs::Vector3T< T > UnitNormal () const

  Returns the normal vector of this triangle in unit length (length = 1.0).
- Gs::Vector3T< T > Barycentric (const Gs::Vector3T< T > &barycentricCoords) const

  Computes the cartesian coordinate by the specified barycentric coordinate with respect to this triangle.
- Triangle< Gs::Vector3T< T > > Barycentric (const Triangle< Gs::Vector3T< T > > &barycentricCoords) const

  Computes the triangle with cartesian coordinates by the specified triangle with barycentric coordinates with respecti to this triangle.
- T Angle (std::size_t vertex) const

  Returns the angle (in radians) of the specified triangle vertex (0, 1, or 2).

Public Attributes

- Gs::Vector3T< T > **a**
- Gs::Vector3T< T > **b**
- Gs::Vector3T< T > **c**

7.31.1 Detailed Description

template<typename T>
class Gm::Triangle< Gs::Vector3T< T > >

Template specializationn for 3D triangles.

7.31.2 Member Function Documentation

7.31.2.1 template<typename T> Gs::Vector3T<T> Gm::Triangle< Gs::Vector3T< T > >::Barycentric (const Gs::Vector3T< T > & *barycentricCoords*) const [inline]

Computes the cartesian coordinate by the specified barycentric coordinate with respect to this triangle.

Parameters

in	*barycentricCoords*	Specifies the barycentric coordinates with respect to this triangle. The sum of all components must be one, i.e. x+y+z = 1.

7.31.2.2 template<typename T> Triangle< Gs::Vector3T<T> > Gm::Triangle< Gs::Vector3T< T > >::Barycentric (const Triangle< Gs::Vector3T< T > > & *barycentricCoords*) const [inline]

Computes the triangle with cartesian coordinates by the specified triangle with barycentric coordinates with respecti to this triangle.

Parameters

in	*barycentricCoords*	Specifies the triangle with barycentric coordinates with respect to this triangle. If this input parameter is { { 1, 0, 0 }, { 0, 1, 0 }, { 0, 0, 1 } }, the result is equal to this triangle. The sum of all components must be one for each triangle vertex, i.e. x+y+z = 1.

7.31.2.3 template<typename T> Gs::Vector3T<T> Gm::Triangle< Gs::Vector3T< T > >::Normal ( ) const [inline]

Returns the normal vector of this triangle.

Remarks

This normal vector is not guaranteed to have a unit length of 1.0! To get a normal vector of unit length use "UnitNormal".

See also

UnitNormal

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

- Triangle.h

7.32 Gm::TriangleMesh Class Reference

Triangle mesh base class.

`#include <TriangleMesh.h>`

Classes

- struct Vertex

  Base vertex structure. Contains the members: position, normal, and texCoord.

Public Types

- using **VertexIndex** = std::size_t
- using **TriangleIndex** = std::size_t
- using **Edge** = Gm::Line< VertexIndex >
- using **Triangle** = Gm::Triangle< VertexIndex >

Public Member Functions

- **TriangleMesh** (TriangleMesh &&rhs)
- TriangleMesh & **operator=** (const TriangleMesh &rhs)=default
- TriangleMesh & **operator=** (TriangleMesh &&rhs)
- void Clear ()

  Clears all vertices and triangles.
- VertexIndex AddVertex (const Gs::Vector3 &position, const Gs::Vector3 &normal, const Gs::Vector2 &texCoord)

  Adds a new vertex with the specified attributes and returns the index of the new vertex.
- TriangleIndex AddTriangle (VertexIndex v0, VertexIndex v1, VertexIndex v2)

  Adds a new triangle with the specified three indices and returns the index of the new triangle.
- Vertex Barycentric (TriangleIndex triangleIndex, const Gs::Vector3 &barycentricCoords) const

  Returns the vertex, interpolated from the triangle with the specified barycentric coordinates.
- std::vector< Edge > Edges () const

  Computes the set of all triangle edges.
- std::vector< Edge > SilhouetteEdges (Gs::Real toleranceAngle=Gs::Real(0)) const

  Computes the set of all triangle edges which are part of the silhouette.
- std::set< TriangleIndex > TriangleNeighbors (std::set< TriangleIndex > triangleIndices, std::size_t searchDepth=1, bool edgeBondOnly=false, bool searchViaPosition=false) const

  Computes the list of all neighbors of the specified triangle.
- std::vector< TriangleIndex > FindTriangles (VertexIndex vertexIndex) const

  Computes the list of all triangles that are connected to the specified vertex.
- std::vector< TriangleIndex > FindTriangles (const Edge &edge) const

  Computes the list of all triangles that are connected to the specified edge.
- Gs::Vector3 TriangleNormal (TriangleIndex triangleIndex) const

  Returns the normal vector of the specified triangle (in unit length of 1.0).
- AABB3 BoundingBox () const

  Computes the axis-aligned bounding-box of this mesh.
- AABB3 BoundingBox (const Gs::AffineMatrix4 &matrix) const

  Computes the axis-aligned bounding-box of this mesh with the specified transformation matrix.
- AABB3 BoundingBoxMultiThreaded (std::size_t threadCount) const

  Computes the axis-aligned bounding-box of this mesh with the specified number of threads.
- void Clip (const Plane &clipPlane, TriangleMesh &front, TriangleMesh &back) const

Public Attributes

- std::vector< Vertex > vertices

  Vertex array list.
- std::vector< Triangle > triangles

  Triangle array list. Make sure that all triangle indices are less than the number of vertices of this mesh!

7.32.1 Detailed Description

Triangle mesh base class.

Remarks

This class is used for generation and modification of all triangle meshes. However, it is only meant to be used to operate with this library, but not to use it within the graphics engine of your project.

7.32.2 Member Function Documentation

7.32.2.1 AABB3 Gm::TriangleMesh::BoundingBoxMultiThreaded (std::size_t *threadCount*) const

Computes the axis-aligned bounding-box of this mesh with the specified number of threads.

Parameters

in	*threadCount*	Specifies the number of threads. This will be clamped to the range [1, vertices.size()].

Remarks

This may only increase performance with very large triangle meshes, i.e. over 1 Mio. vertices and more.

7.32.2.2 void Gm::TriangleMesh::Clip (const Plane & *clipPlane,* TriangleMesh & *front,* TriangleMesh & *back*) const

Clips this triangle mesh into a front- and back sided mesh by the specified clipping plane.

See also

ClipTriangle

7.32.2.3 std::vector<Edge> Gm::TriangleMesh::SilhouetteEdges (Gs::Real *toleranceAngle* = `Gs::Real(0)`) const

Computes the set of all triangle edges which are part of the silhouette.

Parameters

in	*toleranceAngle*	Specifies the tolerance angle (in radians) to reject edges. Must be in the range [0, pi].

See also

Edges

7.32.2.4 std::set<TriangleIndex> Gm::TriangleMesh::TriangleNeighbors (std::set< TriangleIndex > *triangleIndices*, std::size_t *searchDepth = 1*, bool *edgeBondOnly = `false`*, bool *searchViaPosition = `false`*) const

Computes the list of all neighbors of the specified triangle.

Parameters

in	*triangleIndices*	Specifies the indices of the root triangles to search for neighbors.
in	*searchDepth*	Specifies the number of iterations
in	*edgeBondOnly*	Specifies whether to only search triangles that are connected at their edges and not only at their corners. By default false.
in	*searchViaPosition*	Specifies whether to search triangles via the position of their vertices (true), or only search via the index of their vertices (false). By default false.

Returns

Set of triangle indices of the neighbor search result including the input triangle indices.

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

- TriangleMesh.h

7.33 Gm::MeshGenerator::TubeDescription Struct Reference

Public Attributes

- Gs::Vector2 innerRadius = Gs::Vector2(Gs::Real(0.25))

  Radius of the inner cylinder in U (x component), and V (y component) direction. By default (0.25, 0.25).
- Gs::Vector2 outerRadius = Gs::Vector2(Gs::Real(0.5))

  Radius of the outer cylinder in U (x component), and V (y component) direction. By default (0.5, 0.5).
- Gs::Real height = Gs::Real(1)

  Tube height. By default 1.
- Gs::Vector2ui mantleSegments = Gs::Vector2ui(20, 1)

  Segmentation around the (inner and outer) cylinder (x component), and height (y component). By default (20, 1).
- unsigned int capSegments = 1

  Segmentation of the top- and bottom cap. By default 1.

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

- MeshGenerator.h

7.34 Gm::UniformSpline< P, T > Class Template Reference

Spline class with uniform weights.

`#include <UniformSpline.h>`

Classes

- struct Polynomial

  Polynomial structure with four coefficients.

Public Member Functions

- **__GM_ASSERT_FLOAT_TYPE__** ("UniformSpline")
- P **operator()** (const T &t) const
- void Build (const std::vector< P > &points, const T &expansion=T(1))

  Builds the spline polynomials.
- void Clear ()

  Clears the spline polynoms.
- const Polynomial & **operator[ ]** (std::size_t idx) const
- Polynomial & **operator[ ]** (std::size_t idx)
- P **Evaluate** (T t) const
- const std::vector< Polynomial > & **GetPolynomials** () const

Static Public Attributes

- static const std::size_t dimension = P::components

  Spline dimension (e.g. 2 for a 2D-vector).

7.34.1 Detailed Description

template<typename P, typename T>
class Gm::UniformSpline< P, T >

Spline class with uniform weights.

See also

Spline

7.34.2 Member Function Documentation

7.34.2.1 template<typename P, typename T> void Gm::UniformSpline< P, T >::Build (const std::vector< P > & *points,* const T & *expansion = T(1)*) [inline]

Builds the spline polynomials.

Parameters

in	*points*	Specifies the control points.
in	*expansion*	Specifies the expansion of the polynomials. If the expansion is 0.0, this spline will be a linear spline. By default 1.0.

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

- UniformSpline.h

7.35 Gm::TriangleMesh::Vertex Struct Reference

Base vertex structure. Contains the members: position, normal, and texCoord.

`#include <TriangleMesh.h>`

Public Attributes

- Gs::Vector3 **position**
- Gs::Vector3 **normal**
- Gs::Vector2 **texCoord**

7.35.1 Detailed Description

Base vertex structure. Contains the members: position, normal, and texCoord.

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

- TriangleMesh.h

Index