Library Documentation

# Type Definitions

* typedef Edge Vector
* typedef Edge Normal
* typedef double Angle

# Enumerations

* enum AlignmentType
  + HORIZONTAL
  + VERTICAL
* enum MetricType
  + SCALED\_JACOBIAN
  + DISTORTION
  + WARPAGE
  + ASPECT\_RATIO

# Vertex

## Members

* float x: x-coordinate value
* float y: x-coordinate value
* float z: x-coordinate value

## Constructors/Destructors

* (public) Vertex
  + float x: x-coordinate value
  + float y: x-coordinate value
  + float z: x-coordinate value
* (public) Vertex
  + const Vertex &vertex: Vertex to be copied

## Functions/Operators

* (public) Vertex operator+: Adds x, y, z values of two vertices and returns a new vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator-: Subtracts x, y, z values of two vertices and returns a new vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator\*: Multiplies x, y, z values of the vertex with a constant and returns a new vertex
  + const float value: Constant multiplication factor
* (public) Vertex operator\*: Multiplies x, y, z values of two vertices and returns a new vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator/: Divides x, y, z values of the vertex by a constant and returns a new vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator+=: Adds x, y, z values of an input vertex to the vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator-=: Subtracts x, y, z values of an input vertex from the vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator\*=: Multiplies x, y, z values of the vertex
  + const float value: Constant multiplication factor
* (public) Vertex operator\*=: Multiplies x, y, z values of an input vertex with the vertex
  + const Vertex &vertex: Input vertex
* (public) Vertex operator/=: Divides x, y, z values of the vertex
  + const Vertex &vertex: Input vertex

# Edge

In progress…

# Triangle

In progress…

# Quadrilateral

In progress…

# MeshReader

## Members

* static MeshReader\* instance: Singleton instance of the class

## Constructors/Destructors

* (private) MeshReader
* (private) ~MeshReader

## Functions/Operators

* (public) static MeshReader\* getInstance: Returns the singleton instance of the class
* (public) TriangleMesh readTriangularMesh: Reads a triangle mesh .off file with the given file path
  + string filepath: File path of the mesh
* (public) QuadrilateralMesh readQuadrilateralMesh: Reads a quadrilateral mesh .off file with the given file path
  + string filepath: File path of the mesh

# GeometricFunctions

## Members

None

## Constructors/Destructors

None

## Functions/Operators

* (public) const static double dotProduct: Calculates the dot product of two vectors
  + const Vector &a: First vector
  + const Vector &b: Second vector
* (public) const static double crossProduct: Calculates the cross product of two vectors
  + const Vector &a: First vector
  + const Vector &b: Second vector
* (public) const static Vertex findIntersection: Finds the intersection point of two line segments
  + const Edge &a: First line segment
  + const Edge &b: Second line segment
* (public) const static Normal findNormal: Calculates the normal vector of two vectors using the cross product
  + const Vector &a: First vector
  + const Vector &b: Second vector
* (public) const static Angle calculateAngle: Calculates the angle between two vectors
  + const Vector &a: First vector
  + const Vector &b: Second vector
* (public) const static Angle calculateAngle: Converts degrees to radians
  + const Angle angle: Angle in degrees
* (public) const static Angle calculateAngle: Converts radians to degrees
  + const Angle angle: Angle in radians
* (public) const static Vector normalizeVector: Normalizes the given vector to a unit vector
  + const Vector &vector: The vector that will be normalized

# EvaluationFunctions

## Members

None

## Constructors/Destructors

None

## Functions/Operators

* (public) const static double calculateScaledJacobian: Calculates the minimum scaled jacobian metric value for a given quadrilateral
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (public) const static double calculateDistortion: Calculates the quadrilateral element distortion value for a given quadrilateral (Canaan)
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (public) const static double calculateDistortion2: Calculates the quadrilateral element distortion value for a given quadrilateral (T-Base)
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (public) const static double calculateMaximumWarpage: Calculates the maximum warpage amount for a given quadrilateral
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (public) const static double calculateAspectRatio: Calculates the aspect-ratio for a given quadrilateral
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (private) const static double calculateJacobianDeterminant: Calculates the determinant of the jacobian matrix for the given two edges with the same origin
  + const Edge &left: The edge that is on the left hand-side on counter-clockwise rotation
  + const Edge &right: The edge that is on the right hand-side on clockwise rotation
* (private) const static double calculateWarpage: Calculates the warpage amount for a given quadrilateral on given diagonal
  + const Quadrilateral &quadrilateral: Input quadrilateral element
  + const AlignmentType alignment: Chosen diagonal of quadrilateral (vertical/horizontal)
* (private) const static double calculateTriangleDistortion: Calculates the distortion of a triangular element (Lee & Lo)
  + const Vertex &a: First vertex of the triangle
  + const Vertex &b: Second vertex of the triangle
  + const Vertex &c: Third vertex of the triangle
* (private) const static Vertex calculateCornerAverage: Calculates the average point of the corner points of a quadrilateral
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (private) const static Normal calculateNormalAverage: Calculates the average normal of the normals of corner points in a quadrilateral
  + const Quadrilateral &quadrilateral: Input quadrilateral element
* (private) const static void projectTriangle: Projects a triangle to the z=0 plane
  + const Edge &left: The edge that is on the left hand-side on counter-clockwise rotation
  + const Edge &right: The edge that is on the right hand-side on clockwise rotation
* (private) const static Quadrilateral projectQuadrilateral: Projects a non-planar quadrilateral onto an average plane
  + const Quadrilateral &quadrilateral: Input quadrilateral element

# Statistics

## Members

* double minimumValue: Minimum metric value for the mesh
* double maximumValue: Maximum metric value for the mesh
* vector<double> values: List of the metric calculations for the mesh

## Constructors/Destructors

None

## Functions/Operators

* (public) virtual void outputStatistics: Outputs the statistics to the given file
  + Const string filename: Output file name
* (public) virtual void createHistogram: Creates and outputs a histogram of the metric values
  + Const string filename: Output file name

# MeshStatistics : public Statistics

## Members

* MetricType metricType: Evaluation function used in the statistics
* double totalValue: Total of the metric values for all quadrilaterals in the mesh
* unsigned int quadrilateralCount: Quadrilateral count of the mesh

## Constructors/Destructors

* (public) MeshStatistics
  + const MetricType metricType: Evaluation function used in the statistics

## Functions/Operators

* (public) void updateStatistics: Updates the statistics with the metric value of the new quad
  + const double metric: Metric result of a single quadrilateral
* (public) void outputStatistics: Outputs the statistics to the given file
  + Const string filename: Output file name
* (public) void createHistogram: Creates and outputs a histogram of the metric values
  + Const string filename: Output file name
* (public) double getMinimumValue: Returns the minimum metric value
* (public) double getMaximumValue: Returns the maximum metric value
* (public) double getAverageValue: Returns the average metric value
* (private) string getMetricName: Returns the string identifier of the metric type

# AngleStatistics : public Statistics

## Members

None

## Constructors/Destructors

* (public) AngleStatistics
  + const vector<Quadrilateral> &mesh: Quadrilateral list of a mesh

## Functions/Operators

* (public) void outputStatistics: Outputs the statistics to the given file
  + Const string filename: Output file name
* (public) void createHistogram: Creates and outputs a histogram of the metric values
  + Const string filename: Output file name