Farouq Adepetu's Shapes

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 FAShapes Namespace Reference

Has classes that are used for creating 3D shapes.

Classes

- class Box

  This is class is used to create a box.
- class Cone

  This is class is used to create a cone.
- class Cylinder
- struct DrawArguments

  Data that are used as parameters to draw an object.
- class Pyramid

  This is class is used to create a pyramid.
- class Sphere

  This is class is used to create a sphere.
- struct ThreeDimensionalShapeAbstract

  An abstract class for 3D shapes.
- class Triangle

  The class stores a pointer to a vertex list and indices to the vertices of the triangle.
- struct Vertex

  Data that describes a vertex.

5.1.1 Detailed Description

Has classes that are used for creating 3D shapes.

Chapter 6

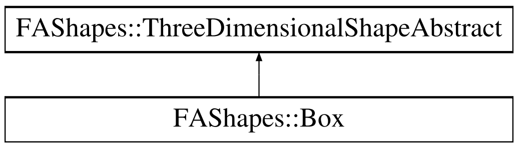
Class Documentation

6.1 FAShapes::Box Class Reference

This is class is used to create a box.

```
#include "FABox.h"
```

Inheritance diagram for FAShapes::Box:



Public Member Functions

- Box (float width=1.0f, float height=1.0f, float depth=1.0f, const FAColor::Color &color=FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f))

  Creates a Box with the specified width, height, depth and color centered around the origin.
- float GetWidth () const

  Returns the width of the box.
- float GetHeight () const

  Returns the height of the box.
- float GetDepth () const

  Returns the depth of the box.
- void SetWidth (float width)

  Sets the width of the box to the specified width.
- void SetHeight (float height)

  Sets the height of the box to the specified height.
- void SetDepth (float depth)

  Sets the depth of the box to the specified depth.
- void UpdateLocalToWorldMatrix () override final

  Updates the boxs local to world transformation matrix.
- float Volume () override final

  Returns the volume of the box.

Additional Inherited Members

6.1.1 Detailed Description

This is class is used to create a box.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Box()

```
FAShapes::Box::Box (
             float width = 1.0f,
             float height = 1.0f,
             float depth = 1.0f,
             const FAColor::Color & color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f) )
```

Creates a Box with the specified width, height, depth and color centered around the origin.

In a left-handed coordinate system the front face looks towards +z axis, the top face looks towards the +y axis and the right face looks towards the +x axis./n The Box is made using triangles. The vertices are ordered in clockwise order.

Parameters

in	*width*	The width of the box.
in	*height*	The height of the box.
in	*depth*	The depth of the box.
in	*color*	The color of the box.

6.1.3 Member Function Documentation

6.1.3.1 GetDepth()

```
float FAShapes::Box::GetDepth ( ) const
```

Returns the depth of the box.

6.1.3.2 GetHeight()

```
float FAShapes::Box::GetHeight ( ) const
```

Returns the height of the box.

6.1.3.3 GetWidth()

```
float FAShapes::Box::GetWidth ( ) const
```

Returns the width of the box.

6.1.3.4 SetDepth()

```
void FAShapes::Box::SetDepth (
             float depth )
```

Sets the depth of the box to the specified *depth*.

6.1.3.5 SetHeight()

```
void FAShapes::Box::SetHeight (
             float height )
```

Sets the height of the box to the specified *height*.

6.1.3.6 SetWidth()

```
void FAShapes::Box::SetWidth (
             float width )
```

Sets the width of the box to the specified width.

6.1.3.7 UpdateLocalToWorldMatrix()

```
void FAShapes::Box::UpdateLocalToWorldMatrix ( )  [final], [override], [virtual]
```

Updates the boxs local to world transformation matrix.

Implements FAShapes::ThreeDimensionalShapeAbstract.

6.1.3.8 Volume()

```
float FAShapes::Box::Volume ( )  [final], [override], [virtual]
```

Returns the volume of the box.

Implements FAShapes::ThreeDimensionalShapeAbstract.

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

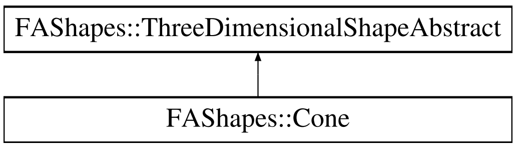
- FABox.h

6.2 FAShapes::Cone Class Reference

This is class is used to create a cone.

```
#include "FACone.h"
```

Inheritance diagram for FAShapes::Cone:



Public Member Functions

- Cone (float radius=1.0f, float height=1.0f, const FAColor::Color &color=FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f), bool fillBottom=false, unsigned int numCircles=20, unsigned int numVerticesPerCircle=20)

  Creates a cone with the specified radius, height and color and it is centered around the origin.
- float GetRadius () const

  Returns the radius of the base of the cone.
- float GetHeight () const

  Returns the height of the base of the cone.
- void SetRadius (float r)

  Sets the radius of the base of the cone to the specified value.
- void SetHeight (float h)

  Sets the height of the base of the cone to the specified value.
- void UpdateLocalToWorldMatrix () override final

  Updates the cones local to world transformation matrix.
- float Volume () override final

  Returns the volume of the cone.

Additional Inherited Members

6.2.1 Detailed Description

This is class is used to create a cone.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Cone()

```
FAShapes::Cone::Cone (
             float radius = 1.0f,
             float height = 1.0f,
             const FAColor::Color & color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f),
             bool fillBottom = false,
             unsigned int numCircles = 20,
             unsigned int numVerticesPerCircle = 20 )
```

Creates a cone with the specified radius, height and color and it is centered around the origin.

Uses the UV method to create the cone.
The more circles and vertices per cirlce, the more circular the cone looks.

Parameters

in	*radius*	The radius of the cone.
in	*height*	The height of the cone.
in	*color*	The color of the cone.
in	*fillBottom*	Pass in true to fill in the bottom of the cone.
in	*numCircles*	The number of circles the cone has.
in	*numVerticesPerCircle*	The number of vertices each circle has.

6.2.3 Member Function Documentation

6.2.3.1 GetHeight()

```
float FAShapes::Cone::GetHeight ( ) const
```

Returns the height of the base of the cone.

6.2.3.2 GetRadius()

```
float FAShapes::Cone::GetRadius ( ) const
```

Returns the radius of the base of the cone.

6.2.3.3 SetHeight()

```
void FAShapes::Cone::SetHeight (
             float h )
```

Sets the height of the base of the cone to the specified value.

6.2.3.4 SetRadius()

```
void FAShapes::Cone::SetRadius (
             float r )
```

Sets the radius of the base of the cone to the specified value.

6.2.3.5 UpdateLocalToWorldMatrix()

```
void FAShapes::Cone::UpdateLocalToWorldMatrix ( )  [final], [override], [virtual]
```

Updates the cones local to world transformation matrix.

Implements FAShapes::ThreeDimensionalShapeAbstract.

6.2.3.6 Volume()

```
float FAShapes::Cone::Volume ( )  [final], [override], [virtual]
```

Returns the volume of the cone.

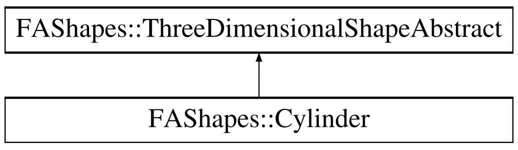
Implements FAShapes::ThreeDimensionalShapeAbstract.

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

- FACone.h

6.3 FAShapes::Cylinder Class Reference

Inheritance diagram for FAShapes::Cylinder:



Public Member Functions

- Cylinder (float radius=1.0f, float height=1.0f, const FAColor::Color &color=FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f), bool fillTopAndBottom=false, unsigned int numCircles=20, unsigned int numVerticesPerCircle=20)

    Creates a cylinder with the specified radius, height and color and it is centered around the origin.
- float GetRadius () const

    Returns the radius of the cylinder.
- float GetHeight () const

    Returns the height of the cylinder.
- void SetRadius (float r)

    Sets the radius of the cylinder to the specified value.
- void SetHeight (float h)

    Sets the height of the cylinder to the specified value.
- void UpdateLocalToWorldMatrix () override final

    Updates the cylinders local to world transformation matrix.
- float Volume () override final

    Returns the volume of the cylinder.

Additional Inherited Members

6.3.1 Constructor & Destructor Documentation

6.3.1.1 Cylinder()

```
FAShapes::Cylinder::Cylinder (
             float radius = 1.0f,
             float height = 1.0f,
             const FAColor::Color & color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f),
             bool fillTopAndBottom = false,
             unsigned int numCircles = 20,
             unsigned int numVerticesPerCircle = 20 )
```

Creates a cylinder with the specified radius, height and color and it is centered around the origin.

Uses the UV method to create the cylinder.
The more circles and vertices per cirlce, the more circular it looks.

Parameters

in	*radius*	The radius of the cylinder.
in	*height*	The height of the cylinder.
in	*color*	The color of the cylinder.
in	*fillTopAndBottom*	Pass in true to fill in the top and bottom of the cylinder.
in	*numCircles*	The number of circles the cylinder has.
in	*numVerticesPerCircle*	The number of vertices each circle has.

6.3.2 Member Function Documentation

6.3.2.1 GetHeight()

```
float FAShapes::Cylinder::GetHeight ( ) const
```

Returns the height of the cylinder.

6.3.2.2 GetRadius()

```
float FAShapes::Cylinder::GetRadius ( ) const
```

Returns the radius of the cylinder.

6.3.2.3 SetHeight()

```
void FAShapes::Cylinder::SetHeight (
             float h )
```

Sets the height of the cylinder to the specified value.

6.3.2.4 SetRadius()

```
void FAShapes::Cylinder::SetRadius (
             float r )
```

Sets the radius of the cylinder to the specified value.

6.3.2.5 UpdateLocalToWorldMatrix()

```
void FAShapes::Cylinder::UpdateLocalToWorldMatrix ( )  [final], [override], [virtual]
```

Updates the cylinders local to world transformation matrix.

Implements FAShapes::ThreeDimensionalShapeAbstract.

6.3.2.6 Volume()

```
float FAShapes::Cylinder::Volume ( )  [final], [override], [virtual]
```

Returns the volume of the cylinder.

Implements FAShapes::ThreeDimensionalShapeAbstract.

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

- FACylinder.h

6.4 FAShapes::DrawArguments Struct Reference

Data that are used as parameters to draw an object.

```
#include "FAShapesUtility.h"
```

Public Attributes

- unsigned int **indexCount**
- unsigned int **locationOfFirstIndex**
- int **indexOfFirstVertex**
- int **indexOfConstantData**

6.4.1 Detailed Description

Data that are used as parameters to draw an object.

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

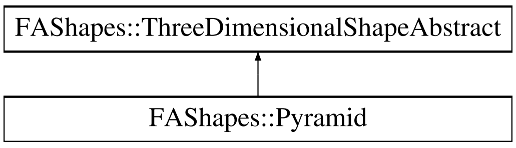
- FAShapesUtility.h

6.5 FAShapes::Pyramid Class Reference

This is class is used to create a pyramid.

```
#include "FAPyramid.h"
```

Inheritance diagram for FAShapes::Pyramid:



Public Member Functions

- Pyramid (float width=1.0f, float height=1.0f, float depth=1.0f, const FAColor::Color &color=FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f))

  Creates a pyramid with the specified width, height, depth and color centered around the origin.
- float GetWidth () const

  Returns the width of the pyramid.
- float GetHeight () const

  Returns the height of the pyramid.
- float GetDepth () const

  Returns the depth of the pyramid.
- void SetWidth (float width)

  Sets the width of the pyramid to the specified width.
- void SetHeight (float height)

  Sets the height of the pyramid to the specified height.
- void SetDepth (float depth)

  Sets the depth of the pyramid to the specified depth.
- void UpdateLocalToWorldMatrix () override final

  Updates the pyramids local to world transformation matrix.
- float Volume () override final

  Returns the volume of the pyramid.

Additional Inherited Members

6.5.1 Detailed Description

This is class is used to create a pyramid.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Pyramid()

```
FAShapes::Pyramid::Pyramid (
             float width = 1.0f,
             float height = 1.0f,
             float depth = 1.0f,
             const FAColor::Color & color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f) )
```

Creates a pyramid with the specified width, height, depth and color centered around the origin.

In a left-handed coordinate system the front of the pyramid looks towards +z axis, the base of the pyramid looks towards the -y axis and the right face looks towards the +x axis. /n The vertices are ordered in clockwise order.

Parameters

in	*width*	The width of the pyramid.
in	*height*	The height of the pyramid.
in	*depth*	The depth of the pyramid.
in	*color*	The color of the pyramid.

6.5.3 Member Function Documentation

6.5.3.1 GetDepth()

```
float FAShapes::Pyramid::GetDepth ( ) const
```

Returns the depth of the pyramid.

6.5.3.2 GetHeight()

```
float FAShapes::Pyramid::GetHeight ( ) const
```

Returns the height of the pyramid.

6.5.3.3 GetWidth()

```
float FAShapes::Pyramid::GetWidth ( ) const
```

Returns the width of the pyramid.

6.5.3.4 SetDepth()

```
void FAShapes::Pyramid::SetDepth (
             float depth )
```

Sets the depth of the pyramid to the specified *depth*.

6.5.3.5 SetHeight()

```
void FAShapes::Pyramid::SetHeight (
             float height )
```

Sets the height of the pyramid to the specified *height*.

6.5.3.6 SetWidth()

```
void FAShapes::Pyramid::SetWidth (
             float width )
```

Sets the width of the pyramid to the specified *width*.

6.5.3.7 UpdateLocalToWorldMatrix()

```
void FAShapes::Pyramid::UpdateLocalToWorldMatrix ( )  [final], [override], [virtual]
```

Updates the pyramids local to world transformation matrix.

Implements FAShapes::ThreeDimensionalShapeAbstract.

6.5.3.8 Volume()

```
float FAShapes::Pyramid::Volume ( )  [final], [override], [virtual]
```

Returns the volume of the pyramid.

Implements FAShapes::ThreeDimensionalShapeAbstract.

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

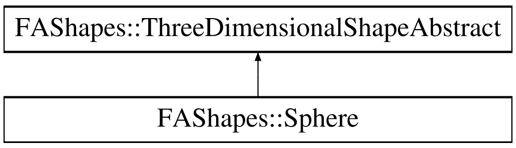
- FAPyramid.h

6.6 FAShapes::Sphere Class Reference

This is class is used to create a sphere.

```
#include "FASphere.h"
```

Inheritance diagram for FAShapes::Sphere:



Public Member Functions

- Sphere (float radius=1.0f, const FAColor::Color &color=FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f), unsigned int numCircles=20, unsigned int numVerticesPerCircle=20)

  Creates a sphere with the specified radius and color and it is centered around the origin.
- float GetRadius () const

  Returns the radius of the sphere.
- void SetRadius (float r)

  Set the radius of the sphere to the specified value.
- void UpdateLocalToWorldMatrix () override final

  Updates the spheres local to world transformation matrix.
- float Volume () override final

  Returns the volume of the sphere.

Additional Inherited Members

6.6.1 Detailed Description

This is class is used to create a sphere.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Sphere()

```
FAShapes::Sphere::Sphere (
            float radius = 1.0f,
            const FAColor::Color & color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f),
            unsigned int numCircles = 20,
            unsigned int numVerticesPerCircle = 20 )
```

Creates a sphere with the specified radius and color and it is centered around the origin.

Uses the UV method to create the sphere.
The more circles and vertices per cirlce, the more circular the sphere looks.

Parameters

in	*radius*	The radius of the cone.
in	*color*	The color of the cone.
in	*numCircles*	The number of circles the cone has.
in	*numVerticesPerCircle*	The number of vertices each circle has.

6.6.3 Member Function Documentation

6.6.3.1 GetRadius()

```
float FAShapes::Sphere::GetRadius ( ) const
```

Returns the radius of the sphere.

6.6.3.2 SetRadius()

```
void FAShapes::Sphere::SetRadius (
             float r )
```

Set the radius of the sphere to the specified value.

6.6.3.3 UpdateLocalToWorldMatrix()

```
void FAShapes::Sphere::UpdateLocalToWorldMatrix ( )  [final], [override], [virtual]
```

Updates the spheres local to world transformation matrix.

Implements FAShapes::ThreeDimensionalShapeAbstract.

6.6.3.4 Volume()

```
float FAShapes::Sphere::Volume ( )  [final], [override], [virtual]
```

Returns the volume of the sphere.

Implements FAShapes::ThreeDimensionalShapeAbstract.

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

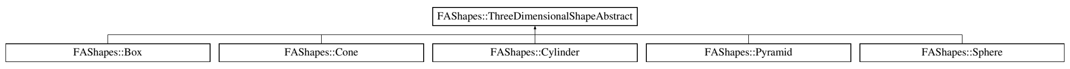
- FASphere.h

6.7 FAShapes::ThreeDimensionalShapeAbstract Struct Reference

An abstract class for 3D shapes.

```
#include "FAThreeDimensional.h"
```

Inheritance diagram for FAShapes::ThreeDimensionalShapeAbstract:



Public Member Functions

- ThreeDimensionalShapeAbstract (const FAColor::Color &color)

  Constructs a 3D shape.
- const FAMath::Vector3D & GetCenter () const

  Returns a constant reference to the center of the 3D shape.
- const FAMath::Vector3D & GetXAxis () const

  Returns a constant reference to the x axis of the 3D shape.
- const FAMath::Vector3D & GetYAxis () const

  Returns a constant reference to the y axis of the 3D shape.
- const FAMath::Vector3D & GetZAxis () const

  Returns a constant reference to the z axis of the 3D shape.
- const FAMath::Matrix4x4 & GetLocalToWorldMatrix () const

  Returns a constant reference to the local to world matrix of the 3D shape.
- const Vertex ∗ GetLocalVertices () const

  Returns a constant pointer to the local vertices of the 3D shape.
- const Triangle ∗ GetTriangleList () const

  Returns a constant pointer to the triangles of the 3D shape.
- const Triangle & GetTriangle (unsigned int index) const

  Returns a constant reference to the specified triangle.
- const DrawArguments & GetDrawArguments () const

  Returns a constant reference to the draw arguments of the 3D shape.
- const FAColor::Color & GetColor () const

  Returns a constant reference to the color of the 3D shape.
- size_t GetNumTriangles () const

  Returns the number of triangles the 3D shape has.
- size_t GetNumVertices () const

  Returns the number of vertices the 3D shape has.
- void SetCenter (const FAMath::Vector3D &center)

  Sets the center of the 3D shape to the specified vector center.
- void SetCenter (float x, float y, float z)

  Sets the center of the 3D shape to the specified values.
- void SetColor (const FAColor::Color &color)

  Sets the color of the sphere to the specified color.
- void SetColor (float r, float g, float b, float a)

  Sets the color of the 3D shape to the specified RGBA values.
- void SetDrawArguments (const DrawArguments &drawArgs)

  Sets the draw arguments of the 3D shape to the specifed draw arguments sphereDrawArgs.
- void SetDrawArguments (unsigned int indexCount, unsigned int locationOfFirstIndex, int indexOfFirstVertex, int indexOfConstantData)

  Sets the draw arguments of the 3D shape to the specifed draw arguments.
- void RotateAxes (const FAMath::Matrix4x4 &rot)

  Rotates the local axis of the 3D shape by the specified rotation matrix rot.
- void RotateAxes (const FAMath::Quaternion &rotQuaternion)

  Rotates the local axis of the 3D shape by the specified rotation quaternion rotQuaternion.
- void RotateAxes (float angle, const FAMath::Vector3D axis)

  Rotates the local axis of the 3D shape by the specified angle around the specified axis.
- void RotateCenter (const FAMath::Matrix4x4 &rot)

  Rotates the center of the 3D shape by the specified rotation matrix rot.
- void RotateCenter (const FAMath::Quaternion &rotQuaternion)

  Rotates the center of the 3D shape by the specified rotation quaternion rotQuaternion.

- void RotateCenter (float angle, const FAMath::Vector3D axis)

    Rotates the center of the 3D shape by the specified angle around the specified axis.
- void TranslateCenter (float x, float y, float z)

    Translates the center by the specified values.
- void TranslateCenter (const FAMath::Vector3D &v)

    Translates the center by the specified vector v.
- virtual void UpdateLocalToWorldMatrix ()=0

    Updates the local to world matrix for the 3D shape.
- virtual float Volume ()=0

    Returns the volume of the 3D shape.

Protected Member Functions

- void Quad (unsigned int a, unsigned int b, unsigned int c, unsigned int d)

    Stores the indices of the vertices of the triangles that make up the 3D shape.
- virtual void CreateVertices ()=0

    Creates the local vertices of the 3D shape.
- virtual void CreateTriangles ()=0

    Creates the triangles that make up the 3D shape.
- virtual void CreateNormals ()=0

    Creates the normals of each vertex.

Protected Attributes

- FAMath::Vector3D **mCenter**
- FAMath::Vector3D **mX**
- FAMath::Vector3D **mY**
- FAMath::Vector3D **mZ**
- FAColor::Color **mColor**
- bool **mUpdateLocalToWorldlMatrix**
- FAMath::Matrix4x4 **mLocalToWorld**
- std::vector< Vertex > **mLocalVertices**
- std::vector< Triangle > **mTriangles**
- DrawArguments **mSphereDrawArguments** {}

6.7.1 Detailed Description

An abstract class for 3D shapes.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 ThreeDimensionalShapeAbstract()

```
FAShapes::ThreeDimensionalShapeAbstract::ThreeDimensionalShapeAbstract (
                const FAColor::Color & color )
```

Constructs a 3D shape.

Parameters

in	*color*	The color if the 3D shape.

6.7.3 Member Function Documentation

6.7.3.1 CreateNormals()

```
virtual void FAShapes::ThreeDimensionalShapeAbstract::CreateNormals ( )  [protected], [pure
virtual]
```

Creates the normals of each vertex.

6.7.3.2 CreateTriangles()

```
virtual void FAShapes::ThreeDimensionalShapeAbstract::CreateTriangles ( )  [protected], [pure
virtual]
```

Creates the triangles that make up the 3D shape.

6.7.3.3 CreateVertices()

```
virtual void FAShapes::ThreeDimensionalShapeAbstract::CreateVertices ( )  [protected], [pure
virtual]
```

Creates the local vertices of the 3D shape.

6.7.3.4 GetCenter()

```
const FAMath::Vector3D & FAShapes::ThreeDimensionalShapeAbstract::GetCenter ( ) const
```

Returns a constant reference to the center of the 3D shape.

6.7.3.5 GetColor()

```
const FAColor::Color & FAShapes::ThreeDimensionalShapeAbstract::GetColor ( ) const
```

Returns a constant reference to the color of the 3D shape.

6.7.3.6 GetDrawArguments()

```
const DrawArguments & FAShapes::ThreeDimensionalShapeAbstract::GetDrawArguments ( ) const
```

Returns a constant reference to the draw arguments of the 3D shape.

6.7.3.7 GetLocalToWorldMatrix()

```
const FAMath::Matrix4x4 & FAShapes::ThreeDimensionalShapeAbstract::GetLocalToWorldMatrix ( )
const
```

Returns a constant reference to the local to world matrix of the 3D shape.

6.7.3.8 GetLocalVertices()

```
const Vertex * FAShapes::ThreeDimensionalShapeAbstract::GetLocalVertices ( ) const
```

Returns a constant pointer to the local vertices of the 3D shape.

6.7.3.9 GetNumTriangles()

```
size_t FAShapes::ThreeDimensionalShapeAbstract::GetNumTriangles ( ) const
```

Returns the number of triangles the 3D shape has.

6.7.3.10 GetNumVertices()

```
size_t FAShapes::ThreeDimensionalShapeAbstract::GetNumVertices ( ) const
```

Returns the number of vertices the 3D shape has.

6.7.3.11 GetTriangle()

```
const Triangle & FAShapes::ThreeDimensionalShapeAbstract::GetTriangle (
             unsigned int index ) const
```

Returns a constant reference to the specified triangle.

6.7.3.12 GetTriangleList()

```
const Triangle * FAShapes::ThreeDimensionalShapeAbstract::GetTriangleList ( ) const
```

Returns a constant pointer to the triangles of the 3D shape.

6.7.3.13 GetXAxis()

```
const FAMath::Vector3D & FAShapes::ThreeDimensionalShapeAbstract::GetXAxis ( ) const
```

Returns a constant reference to the x axis of the 3D shape.

6.7.3.14 GetYAxis()

```
const FAMath::Vector3D & FAShapes::ThreeDimensionalShapeAbstract::GetYAxis ( ) const
```

Returns a constant reference to the y axis of the 3D shape.

6.7.3.15 GetZAxis()

```
const FAMath::Vector3D & FAShapes::ThreeDimensionalShapeAbstract::GetZAxis ( ) const
```

Returns a constant reference to the z axis of the 3D shape.

6.7.3.16 Quad()

```
void FAShapes::ThreeDimensionalShapeAbstract::Quad (
             unsigned int a,
             unsigned int b,
             unsigned int c,
             unsigned int d )  [protected]
```

Stores the indices of the vertices of the triangles that make up the 3D shape.

6.7.3.17 RotateAxes() [1/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateAxes (
            const FAMath::Matrix4x4 & rot )
```

Rotates the local axis of the 3D shape by the specified rotation matrix *rot*.

6.7.3.18 RotateAxes() [2/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateAxes (
            const FAMath::Quaternion & rotQuaternion )
```

Rotates the local axis of the 3D shape by the specified rotation quaternion *rotQuaternion*.

6.7.3.19 RotateAxes() [3/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateAxes (
            float angle,
            const FAMath::Vector3D axis )
```

Rotates the local axis of the 3D shape by the specified *angle* around the specified *axis*.

Uses a quaternion to rotate.

6.7.3.20 RotateCenter() [1/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateCenter (
            const FAMath::Matrix4x4 & rot )
```

Rotates the center of the 3D shape by the specified rotation matrix *rot*.

6.7.3.21 RotateCenter() [2/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateCenter (
            const FAMath::Quaternion & rotQuaternion )
```

Rotates the center of the 3D shape by the specified rotation quaternion *rotQuaternion*.

6.7.3.22 RotateCenter() [3/3]

```
void FAShapes::ThreeDimensionalShapeAbstract::RotateCenter (
             float angle,
             const FAMath::Vector3D axis )
```

Rotates the center of the 3D shape by the specified *angle* around the specified *axis*.

Uses a quaternion to rotate.

6.7.3.23 SetCenter() [1/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetCenter (
             const FAMath::Vector3D & center )
```

Sets the center of the 3D shape to the specified vector *center*.

6.7.3.24 SetCenter() [2/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetCenter (
             float x,
             float y,
             float z )
```

Sets the center of the 3D shape to the specified values.

6.7.3.25 SetColor() [1/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetColor (
             const FAColor::Color & color )
```

Sets the color of the sphere to the specified *color*.

6.7.3.26 SetColor() [2/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetColor (
             float r,
             float g,
             float b,
             float a )
```

Sets the color of the 3D shape to the specified RGBA values.

6.7.3.27 SetDrawArguments() [1/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetDrawArguments (
             const DrawArguments & drawArgs )
```

Sets the draw arguments of the 3D shape to the specifed draw arguments *sphereDrawArgs*.

6.7.3.28 SetDrawArguments() [2/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::SetDrawArguments (
             unsigned int indexCount,
             unsigned int locationOfFirstIndex,
             int indexOfFirstVertex,
             int indexOfConstantData )
```

Sets the draw arguments of the 3D shape to the specifed draw arguments.

6.7.3.29 TranslateCenter() [1/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::TranslateCenter (
             const FAMath::Vector3D & v )
```

Translates the center by the specified vector *v*.

6.7.3.30 TranslateCenter() [2/2]

```
void FAShapes::ThreeDimensionalShapeAbstract::TranslateCenter (
             float x,
             float y,
             float z )
```

Translates the center by the specified values.

6.7.3.31 UpdateLocalToWorldMatrix()

```
virtual void FAShapes::ThreeDimensionalShapeAbstract::UpdateLocalToWorldMatrix ( )  [pure
virtual]
```

Updates the local to world matrix for the 3D shape.

Implemented in FAShapes::Box, FAShapes::Cone, FAShapes::Cylinder, FAShapes::Pyramid, and FAShapes::Sphere.

6.7.3.32 Volume()

```
virtual float FAShapes::ThreeDimensionalShapeAbstract::Volume ( )  [pure virtual]
```

Returns the volume of the 3D shape.

Implemented in FAShapes::Box, FAShapes::Cone, FAShapes::Cylinder, FAShapes::Pyramid, and FAShapes::Sphere.

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

- FAThreeDimensional.h

6.8 FAShapes::Triangle Class Reference

The class stores a pointer to a vertex list and indices to the vertices of the triangle.

```
#include "FATriangle.h"
```

Public Member Functions

- Triangle (Vertex ∗vertexList=nullptr, unsigned int p0Index=0, unsigned int p1Index=0, unsigned int p2Index=0)

  Constructs a triangle.
- const Vertex & GetP0 () const

  Returns a constant reference to the P0 vertex of the triangle.
- const Vertex & GetP1 () const

  Returns a constant reference to the P1 vertex of the triangle.
- const Vertex & GetP2 () const

  Returns a constant reference to the P2 vertex of the triangle.
- unsigned int GetP0Index () const

  Returns the index of where P0 is in the vertex list.
- unsigned int GetP1Index () const

  Returns the index of where P1 is in the vertex list.
- unsigned int GetP2Index () const

  Returns the index of where P2 is in the vertex list.
- FAMath::Vector3D GetNormal () const

  Returns the normal of the triangle.
- FAMath::Vector3D GetCenter () const

  Returns the center of the triangle.
- void SetVertexList (Vertex ∗vertexList)

  Sets the pointer to a vertex list to the specified pointers.
- void SetP0Index (unsigned int index)

  Sets the P0 index to the specified index.
- void SetP1Index (unsigned int index)

  Sets the P1 index to the specified index.
- void SetP2Index (unsigned int index)

  Sets the P2 index to the specified index.
- void SetTriangleIndices (unsigned int p0Index, unsigned int p1Index, unsigned int p2Index)

  Sets the indices of the vertices that make up the triangle to the specified vertices.
- void SetTriangle (Vertex ∗vertexList, unsigned int p0Index, unsigned int p1Index, unsigned int p2Index)

  Sets the triangle variables.

6.8.1 Detailed Description

The class stores a pointer to a vertex list and indices to the vertices of the triangle.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Triangle()

```
FAShapes::Triangle::Triangle (
             Vertex * vertexList = nullptr,
             unsigned int p0Index = 0,
             unsigned int p1Index = 0,
             unsigned int p2Index = 0 )
```

Constructs a triangle.

Parameters

in	*vertexList*	A pointer to a vertex list.
in	*p0Index*	The index of the first point of the triangle.
in	*p1Index*	The index of the second point of the triangle.
in	*p2Index*	The index of the third point of the triangle.

6.8.3 Member Function Documentation

6.8.3.1 GetCenter()

```
FAMath::Vector3D FAShapes::Triangle::GetCenter ( ) const
```

Returns the center of the triangle.

6.8.3.2 GetNormal()

```
FAMath::Vector3D FAShapes::Triangle::GetNormal ( ) const
```

Returns the normal of the triangle.

6.8.3.3 GetP0()

```
const Vertex & FAShapes::Triangle::GetP0 ( ) const
```

Returns a constant reference to the P0 vertex of the triangle.

6.8.3.4 GetP0Index()

```
unsigned int FAShapes::Triangle::GetP0Index ( ) const
```

Returns the index of where P0 is in the vertex list.

6.8.3.5 GetP1()

```
const Vertex & FAShapes::Triangle::GetP1 ( ) const
```

Returns a constant reference to the P1 vertex of the triangle.

6.8.3.6 GetP1Index()

```
unsigned int FAShapes::Triangle::GetP1Index ( ) const
```

Returns the index of where P1 is in the vertex list.

6.8.3.7 GetP2()

```
const Vertex & FAShapes::Triangle::GetP2 ( ) const
```

Returns a constant reference to the P2 vertex of the triangle.

6.8.3.8 GetP2Index()

```
unsigned int FAShapes::Triangle::GetP2Index ( ) const
```

Returns the index of where P2 is in the vertex list.

6.8.3.9 SetP0Index()

```
void FAShapes::Triangle::SetP0Index (
            unsigned int index )
```

Sets the P0 index to the specified *index*.

6.8.3.10 SetP1Index()

```
void FAShapes::Triangle::SetP1Index (
            unsigned int index )
```

Sets the P1 index to the specified *index*.

6.8.3.11 SetP2Index()

```
void FAShapes::Triangle::SetP2Index (
            unsigned int index )
```

Sets the P2 index to the specified *index*.

6.8.3.12 SetTriangle()

```
void FAShapes::Triangle::SetTriangle (
            Vertex * vertexList,
            unsigned int p0Index,
            unsigned int p1Index,
            unsigned int p2Index )
```

Sets the triangle variables.

Parameters

in	*vertexList*	A pointer to a vertex list.
in	*p0Index*	The index of the first point of the triangle.
in	*p1Index*	The index of the second point of the triangle.
in	*p2Index*	The index of the third point of the triangle.

6.8.3.13 SetTriangleIndices()

```
void FAShapes::Triangle::SetTriangleIndices (
             unsigned int p0Index,
             unsigned int p1Index,
             unsigned int p2Index )
```

Sets the indices of the vertices that make up the triangle to the specified vertices.

Parameters

in	*p0Index*	The index of the first point of the triangle.
in	*p1Index*	The index of the second point of the triangle.
in	*p2Index*	The index of the third point of the triangle.

6.8.3.14 SetVertexList()

```
void FAShapes::Triangle::SetVertexList (
             Vertex * vertexList )
```

Sets the pointer to a vertex list to the specified pointers.

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

- FATriangle.h

6.9 FAShapes::Vertex Struct Reference

Data that describes a vertex.

```
#include "FAShapesUtility.h"
```

Public Attributes

- FAMath::Vector3D **position**
- FAColor::Color **color**
- FAMath::Vector3D **normal**

6.9.1 Detailed Description

Data that describes a vertex.

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

- FAShapesUtility.h

Chapter 7

File Documentation

7.1 FABox.h File Reference

File has a Box class under the namespace FAShapes.

```
#include "FAThreeDimensional.h"
```

Classes

- class FAShapes::Box

  This is class is used to create a box.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.1.1 Detailed Description

File has a Box class under the namespace FAShapes.

7.2 FABox.h

Go to the documentation of this file.

```
#pragma once

#include "FAThreeDimensional.h"


namespace FAShapes
{
    class Box :  public ThreeDimensionalShapeAbstract
    {
    public:

        Box(float width = 1.0f, float height = 1.0f , float depth = 1.0f,
            const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f));

        float GetWidth() const;

        float GetHeight() const;

        float GetDepth() const;

        void SetWidth(float width);

        void SetHeight(float height);

        void SetDepth(float depth);

        void UpdateLocalToWorldMatrix() override final;

        float Volume() override final;

    private:
        //Dimensions of the box
        float mWidth;
        float mHeight;
        float mDepth;

        //Creates the vertices of the box.
        void CreateVertices() override final;

        //Creates the triangles that make up box.
        void CreateTriangles() override final;

        //Creates the normals of the box.
        void CreateNormals() override final;
    };
}
```

7.3 FACone.h File Reference

File has a Cone class under the namespace FAShapes.

```
#include "FAThreeDimensional.h"
```

Classes

- class FAShapes::Cone

    This is class is used to create a cone.

Namespaces

- namespace FAShapes

    Has classes that are used for creating 3D shapes.

7.3.1 Detailed Description

File has a Cone class under the namespace FAShapes.

7.4 FACone.h

Go to the documentation of this file.

```
#pragma once

#include "FAThreeDimensional.h"

namespace FAShapes
{
    class Cone :  public ThreeDimensionalShapeAbstract
    {
    public:

        Cone(float radius = 1.0f, float height = 1.0f, const FAColor::Color& color = FAColor::Color(0.0f,
      0.0f, 0.0f, 1.0f),
            bool fillBottom = false,
            unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);

        float GetRadius() const;

        float GetHeight() const;

        void SetRadius(float r);

        void SetHeight(float h);

        void UpdateLocalToWorldMatrix() override final;

        float Volume() override final;

    private:

        //Radius of the cone.
        float mRadius;

        //Height of the cone
        float mHeight;

        //The number of slices the cone has.
        unsigned int mNumCircles;

        //The number of vertices each slice has.
        unsigned int mNumVerticesPerCircle;

        //True to fill the bottom, false to not fill the bottom.
        bool mFillBottom;

        //Creates the vertices of the cone.
        void CreateVertices() override final;

        //Creates the triangles that make up the cone.
        void CreateTriangles() override final;

        //Creates the normals of the cone.
        void CreateNormals() override final;
    };
}
```

7.5 FACylinder.h File Reference

File has a Cylinder class under the namespace FAShapes.

```
#include "FAThreeDimensional.h"
```

Classes

- class FAShapes::Cylinder

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.5.1 Detailed Description

File has a Cylinder class under the namespace FAShapes.

7.6 FACylinder.h

Go to the documentation of this file.

```
#pragma once

#include "FAThreeDimensional.h"

namespace FAShapes
{
    class Cylinder :  public ThreeDimensionalShapeAbstract
    {
    public:

        Cylinder(float radius = 1.0f, float height = 1.0f, const FAColor::Color& color =
      FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f),
            bool fillTopAndBottom = false,
            unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);

        float GetRadius() const;

        float GetHeight() const;

        void SetRadius(float r);

        void SetHeight(float h);

        void UpdateLocalToWorldMatrix() override final;

        float Volume() override final;

    private:

        //radius of the cylinder
        float mRadius;

        //Height of the cylinder
        float mHeight;

        //The number of slices the cylinder has.
        unsigned int mNumCircles;

        //The number of vertices each slice has.
        unsigned int mNumVerticesPerCircle;

        //True to fill the top and bottom, false to not fill the top and bottom.
        bool mFillTopAndBottom;

        //Creates the vertices of the cylinder.
        void CreateVertices() override final;

        //Creates the triangles that make up the cylinder.
        void CreateTriangles() override final;

        //Creates the normals of the cylinder.
        void CreateNormals() override final;

    };
}
```

7.7 FAPyramid.h File Reference

File has a Pyramid class under the namespace FAShapes.

```
#include "FAThreeDimensional.h"
```

Classes

- class FAShapes::Pyramid

  This is class is used to create a pyramid.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.7.1 Detailed Description

File has a Pyramid class under the namespace FAShapes.

7.8 FAPyramid.h

Go to the documentation of this file.

```
#pragma once

#include "FAThreeDimensional.h"

namespace FAShapes
{
    class Pyramid :  public ThreeDimensionalShapeAbstract
    {
    public:

        Pyramid(float width = 1.0f, float height = 1.0f, float depth = 1.0f,
            const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f));

        float GetWidth() const;

        float GetHeight() const;

        float GetDepth() const;

        void SetWidth(float width);

        void SetHeight(float height);

        void SetDepth(float depth);

        void UpdateLocalToWorldMatrix() override final;

        float Volume() override final;

    private:
        //Dimensions of the pyramid
        float mWidth;
        float mHeight;
        float mDepth;

        //Creates the vertices of the pyramid.
        void CreateVertices() override final;

        //Creates the triangles that make up pyramid.
        void CreateTriangles() override final;

        //Creates the normals of the pyramid.
        void CreateNormals() override final;
    };
}
```

7.9 FAShapesUtility.h File Reference

File has structures DrawArguments and Vertex under the namespace FAShapes.

```
#include "FAMathEngine.h"
#include "FAColor.h"
```

Classes

- struct FAShapes::DrawArguments

  Data that are used as parameters to draw an object.
- struct FAShapes::Vertex

  Data that describes a vertex.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.9.1 Detailed Description

File has structures DrawArguments and Vertex under the namespace FAShapes.

7.10 FAShapesUtility.h

Go to the documentation of this file.

```
#pragma once

#include "FAMathEngine.h"
#include "FAColor.h"

namespace FAShapes
{
    struct DrawArguments
    {
        unsigned int indexCount;
        unsigned int locationOfFirstIndex;
        int indexOfFirstVertex;
        int indexOfConstantData;
    };


    struct Vertex
    {
        FAMath::Vector3D position;
        FAColor::Color color;
        FAMath::Vector3D normal;
    };
}
```

7.11 FASphere.h File Reference

File has a Sphere class under the namespace FAShapes.

```
#include "FAThreeDimensional.h"
```

Classes

- class FAShapes::Sphere

  This is class is used to create a sphere.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.11.1 Detailed Description

File has a Sphere class under the namespace FAShapes.

7.12 FASphere.h

Go to the documentation of this file.

```
#pragma once

#include "FAThreeDimensional.h"

namespace FAShapes
{
    class Sphere :  public ThreeDimensionalShapeAbstract
    {
    public:

        Sphere(float radius = 1.0f, const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f ,
      1.0f),
            unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);

        float GetRadius() const;

        void SetRadius(float r);

        void UpdateLocalToWorldMatrix() override final;

        float Volume() override final;

    private:
        //Radius of the sphere.
        float mRadius;

        //The number of slices the sphere has.
        unsigned int mNumCircles;

        //The number of vertices each slice has.
        unsigned int mNumVerticesPerCircle;

        //Creates the vertices of the sphere.
        void CreateVertices() override final;

        //Creates the triangles that make up the sphere.
        void CreateTriangles() override final;

        //Creates the normals of the sphere.
        void CreateNormals() override final;
    };
}
```

7.13 FAThreeDimensional.h File Reference

File has the abstract class ThreeDimensionalShapeAbstract under the namespace FAShapes.

```
#include "FATriangle.h"
#include <vector>
```

Classes

- struct FAShapes::ThreeDimensionalShapeAbstract

  An abstract class for 3D shapes.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.13.1 Detailed Description

File has the abstract class ThreeDimensionalShapeAbstract under the namespace FAShapes.

7.14 FAThreeDimensional.h

Go to the documentation of this file.

```
#pragma once

#include "FATriangle.h"
#include <vector>

namespace FAShapes
{
    class ThreeDimensionalShapeAbstract
    {
    public:

        ThreeDimensionalShapeAbstract(const FAColor::Color& color);

        const FAMath::Vector3D& GetCenter() const;

        const FAMath::Vector3D& GetXAxis() const;

        const FAMath::Vector3D& GetYAxis() const;

        const FAMath::Vector3D& GetZAxis() const;

        const FAMath::Matrix4x4& GetLocalToWorldMatrix() const;

        const Vertex* GetLocalVertices() const;

        const Triangle* GetTriangleList() const;

        const Triangle& GetTriangle(unsigned int index) const;

        const DrawArguments& GetDrawArguments() const;

        const FAColor::Color& GetColor() const;

        size_t GetNumTriangles() const;

        size_t GetNumVertices() const;

        void SetCenter(const FAMath::Vector3D& center);

        void SetCenter(float x, float y, float z);

        void SetColor(const FAColor::Color& color);

        void SetColor(float r, float g, float b, float a);

        void SetDrawArguments(const DrawArguments& drawArgs);

        void SetDrawArguments(unsigned int indexCount, unsigned int locationOfFirstIndex,
            int indexOfFirstVertex, int indexOfConstantData);

        void RotateAxes(const FAMath::Matrix4x4& rot);

        void RotateAxes(const FAMath::Quaternion& rotQuaternion);

        void RotateAxes(float angle, const FAMath::Vector3D axis);

        void RotateCenter(const FAMath::Matrix4x4& rot);

        void RotateCenter(const FAMath::Quaternion& rotQuaternion);

        void RotateCenter(float angle, const FAMath::Vector3D axis);

        void TranslateCenter(float x, float y, float z);

        void TranslateCenter(const FAMath::Vector3D& v);

        virtual void UpdateLocalToWorldMatrix() = 0;

        virtual float Volume() = 0;

    protected:
        //Center of the 3D shape.
        FAMath::Vector3D mCenter;

        //Local axes of the 3D shape.
        FAMath::Vector3D mX;
        FAMath::Vector3D mY;
        FAMath::Vector3D mZ;

        //Color of the 3D shape.
        FAColor::Color mColor;

        //If true, updates the local to world matrix.
        bool mUpdateLocalToWorldlMatrix;

        //Local to world matrix of the 3D shape.
        FAMath::Matrix4x4 mLocalToWorld;

        //Local vertices of the 3D shape.
        std::vector<Vertex> mLocalVertices;

        //The triangles that make up the 3D shape.
        std::vector<Triangle> mTriangles;

        //The arguments needed to render the 3D shape.
        DrawArguments mSphereDrawArguments{};

        void Quad(unsigned int a, unsigned int b, unsigned int c, unsigned int d);

        virtual void CreateVertices() = 0;

        virtual void CreateTriangles() = 0;

        virtual void CreateNormals() = 0;

    };
}
```

7.15 FATriangle.h File Reference

File has a Triangle class under the namespace FAShapes.

`#include "FAShapesUtility.h"`

Classes

- class FAShapes::Triangle

  The class stores a pointer to a vertex list and indices to the vertices of the triangle.

Namespaces

- namespace FAShapes

  Has classes that are used for creating 3D shapes.

7.15.1 Detailed Description

File has a Triangle class under the namespace FAShapes.

7.16 FATriangle.h

Go to the documentation of this file.

```
#pragma once

#include "FAShapesUtility.h"

namespace FAShapes
{
    class Triangle
    {
    public:

        Triangle(Vertex* vertexList = nullptr, unsigned int p0Index = 0, unsigned int p1Index = 0, unsigned int p2Index = 0);

        const Vertex& GetP0() const;

        const Vertex& GetP1() const;

        const Vertex& GetP2() const;

        unsigned int GetP0Index() const;

        unsigned int GetP1Index() const;

        unsigned int GetP2Index() const;

        FAMath::Vector3D GetNormal() const;

        FAMath::Vector3D GetCenter() const;

        void SetVertexList(Vertex* vertexList);

        void SetP0Index(unsigned int index);

        void SetP1Index(unsigned int index);

        void SetP2Index(unsigned int index);

        void SetTriangleIndices(unsigned int p0Index, unsigned int p1Index, unsigned int p2Index);

        void SetTriangle(Vertex* vertexList, unsigned int p0Index, unsigned int p1Index, unsigned int p2Index);

    private:
        Vertex* mVertexList; //pointer to a vertex list
        unsigned int mIndexList[3]; //indices into a vertex list
    };
}
```

Index