

# Farouq Adepetu's Shapes

Generated by Doxygen 1.9.4



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Hierarchical Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Class Index</b>	<b>5</b>
3.1 Class List	5
<b>4 File Index</b>	<b>7</b>
4.1 File List	7
<b>5 Namespace Documentation</b>	<b>9</b>
5.1 FAShapes Namespace Reference	9
5.1.1 Detailed Description	9
<b>6 Class Documentation</b>	<b>11</b>
6.1 FAShapes::Box Class Reference	11
6.1.1 Detailed Description	12
6.1.2 Constructor & Destructor Documentation	12
6.1.2.1 Box()	12
6.1.3 Member Function Documentation	12
6.1.3.1 GetDepth()	12
6.1.3.2 GetHeight()	12
6.1.3.3 GetWidth()	13
6.1.3.4 SetDepth()	13
6.1.3.5 SetHeight()	13
6.1.3.6 SetWidth()	13
6.1.3.7 UpdateLocalToWorldMatrix()	13
6.1.3.8 Volume()	14
6.2 FAShapes::Cone Class Reference	14
6.2.1 Detailed Description	14
6.2.2 Constructor & Destructor Documentation	15
6.2.2.1 Cone()	15
6.2.3 Member Function Documentation	15
6.2.3.1 GetHeight()	15
6.2.3.2 GetRadius()	15
6.2.3.3 SetHeight()	16
6.2.3.4 SetRadius()	16
6.2.3.5 UpdateLocalToWorldMatrix()	16
6.2.3.6 Volume()	16
6.3 FAShapes::Cylinder Class Reference	16
6.3.1 Constructor & Destructor Documentation	17
6.3.1.1 Cylinder()	17

6.3.2 Member Function Documentation	18
6.3.2.1 GetHeight()	18
6.3.2.2 GetRadius()	18
6.3.2.3 SetHeight()	18
6.3.2.4 SetRadius()	18
6.3.2.5 UpdateLocalToWorldMatrix()	18
6.3.2.6 Volume()	19
6.4 FAShapes::DrawArguments Struct Reference	19
6.4.1 Detailed Description	19
6.5 FAShapes::Pyramid Class Reference	19
6.5.1 Detailed Description	20
6.5.2 Constructor & Destructor Documentation	20
6.5.2.1 Pyramid()	20
6.5.3 Member Function Documentation	21
6.5.3.1 GetDepth()	21
6.5.3.2 GetHeight()	21
6.5.3.3 GetWidth()	21
6.5.3.4 SetDepth()	21
6.5.3.5 SetHeight()	21
6.5.3.6 SetWidth()	22
6.5.3.7 UpdateLocalToWorldMatrix()	22
6.5.3.8 Volume()	22
6.6 FAShapes::Sphere Class Reference	22
6.6.1 Detailed Description	23
6.6.2 Constructor & Destructor Documentation	23
6.6.2.1 Sphere()	23
6.6.3 Member Function Documentation	23
6.6.3.1 GetRadius()	24
6.6.3.2 SetRadius()	24
6.6.3.3 UpdateLocalToWorldMatrix()	24
6.6.3.4 Volume()	24
6.7 FAShapes::ThreeDimensionalShapeAbstract Struct Reference	24
6.7.1 Detailed Description	26
6.7.2 Constructor & Destructor Documentation	26
6.7.2.1 ThreeDimensionalShapeAbstract()	26
6.7.3 Member Function Documentation	27
6.7.3.1 CreateNormals()	27
6.7.3.2 CreateTriangles()	27
6.7.3.3 CreateVertices()	27
6.7.3.4 GetCenter()	27
6.7.3.5 GetColor()	28
6.7.3.6 GetDrawArguments()	28

6.7.3.7 GetLocalToWorldMatrix()	28
6.7.3.8 GetLocalVertices()	28
6.7.3.9 GetNumTriangles()	28
6.7.3.10 GetNumVertices()	28
6.7.3.11 GetTriangle()	29
6.7.3.12 GetTriangleList()	29
6.7.3.13 GetXAxis()	29
6.7.3.14 GetYAxis()	29
6.7.3.15 GetZAxis()	29
6.7.3.16 Quad()	29
6.7.3.17 RotateAxes() [1/3]	30
6.7.3.18 RotateAxes() [2/3]	30
6.7.3.19 RotateAxes() [3/3]	30
6.7.3.20 RotateCenter() [1/3]	30
6.7.3.21 RotateCenter() [2/3]	30
6.7.3.22 RotateCenter() [3/3]	31
6.7.3.23 SetCenter() [1/2]	31
6.7.3.24 SetCenter() [2/2]	31
6.7.3.25 SetColor() [1/2]	31
6.7.3.26 SetColor() [2/2]	31
6.7.3.27 SetDrawArguments() [1/2]	32
6.7.3.28 SetDrawArguments() [2/2]	32
6.7.3.29 TranslateCenter() [1/2]	32
6.7.3.30 TranslateCenter() [2/2]	32
6.7.3.31 UpdateLocalToWorldMatrix()	32
6.7.3.32 Volume()	33
6.8 FAShapes::Triangle Class Reference	33
6.8.1 Detailed Description	34
6.8.2 Constructor & Destructor Documentation	34
6.8.2.1 Triangle()	34
6.8.3 Member Function Documentation	34
6.8.3.1 GetCenter()	34
6.8.3.2 GetNormal()	34
6.8.3.3 GetP0()	35
6.8.3.4 GetP0Index()	35
6.8.3.5 GetP1()	35
6.8.3.6 GetP1Index()	35
6.8.3.7 GetP2()	35
6.8.3.8 GetP2Index()	35
6.8.3.9 SetP0Index()	36
6.8.3.10 SetP1Index()	36
6.8.3.11 SetP2Index()	36

---

6.8.3.12 SetTriangle()	36
6.8.3.13 SetTriangleIndices()	36
6.8.3.14 SetVertexList()	37
6.9 FAShapes::Vertex Struct Reference	37
6.9.1 Detailed Description	37
<b>7 File Documentation</b>	<b>39</b>
7.1 FABox.h File Reference	39
7.1.1 Detailed Description	39
7.2 FABox.h	40
7.3 FACone.h File Reference	40
7.3.1 Detailed Description	41
7.4 FACone.h	41
7.5 FACylinder.h File Reference	41
7.5.1 Detailed Description	42
7.6 FACylinder.h	42
7.7 FAPyramid.h File Reference	43
7.7.1 Detailed Description	43
7.8 FAPyramid.h	43
7.9 FAShapesUtility.h File Reference	44
7.9.1 Detailed Description	44
7.10 FAShapesUtility.h	44
7.11 FASphere.h File Reference	44
7.11.1 Detailed Description	45
7.12 FASphere.h	45
7.13 FAThreeDimensional.h File Reference	45
7.13.1 Detailed Description	46
7.14 FAThreeDimensional.h	46
7.15 FATriangle.h File Reference	47
7.15.1 Detailed Description	48
7.16 FATriangle.h	48
<b>Index</b>	<b>49</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">FAShapes</a>	Has classes that are used for creating 3D shapes . . . . .	<a href="#">9</a>
--------------------------	--	-------------------





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

FAShapes::DrawArguments . . . . .	19
FAShapes::ThreeDimensionalShapeAbstract . . . . .	24
FAShapes::Box . . . . .	11
FAShapes::Cone . . . . .	14
FAShapes::Cylinder . . . . .	16
FAShapes::Pyramid . . . . .	19
FAShapes::Sphere . . . . .	22
FAShapes::Triangle . . . . .	33
FAShapes::Vertex . . . . .	37



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">FAShapes::Box</a>	
This is class is used to create a box . . . . .	11
<a href="#">FAShapes::Cone</a>	
This is class is used to create a cone . . . . .	14
<a href="#">FAShapes::Cylinder</a> . . . . .	16
<a href="#">FAShapes::DrawArguments</a>	
Data that are used as parameters to draw an object . . . . .	19
<a href="#">FAShapes::Pyramid</a>	
This is class is used to create a pyramid . . . . .	19
<a href="#">FAShapes::Sphere</a>	
This is class is used to create a sphere . . . . .	22
<a href="#">FAShapes::ThreeDimensionalShapeAbstract</a>	
An abstract class for 3D shapes . . . . .	24
<a href="#">FAShapes::Triangle</a>	
The class stores a pointer to a vertex list and indices to the vertices of the triangle . . . . .	33
<a href="#">FAShapes::Vertex</a>	
Data that describes a vertex . . . . .	37



## Chapter 4

# File Index

### 4.1 File List

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

<a href="#">FABox.h</a>	File has a Box class under the namespace <a href="#">FAShapes</a> . . . . .	39
<a href="#">FACone.h</a>	File has a Cone class under the namespace <a href="#">FAShapes</a> . . . . .	40
<a href="#">FACylinder.h</a>	File has a Cylinder class under the namespace <a href="#">FAShapes</a> . . . . .	41
<a href="#">FAPyramid.h</a>	File has a Pyramid class under the namespace <a href="#">FAShapes</a> . . . . .	43
<a href="#">FAShapesUtility.h</a>	File has structures DrawArguments and Vertex under the namespace <a href="#">FAShapes</a> . . . . .	44
<a href="#">FASphere.h</a>	File has a Sphere class under the namespace <a href="#">FAShapes</a> . . . . .	44
<a href="#">FAThreeDimensional.h</a>	File has the abstract class ThreeDimensionalShapeAbstract under the namespace <a href="#">FAShapes</a> .	45
<a href="#">FATriangle.h</a>	File has a Triangle class under the namespace <a href="#">FAShapes</a> . . . . .	47



## 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

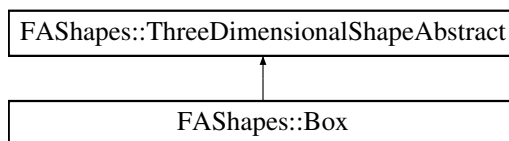
# Class Documentation

### 6.1 FAShapes::Box Class Reference

This 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 FAShapes::Color &color=FAShapes::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 box's local to world transformation matrix.*
- float **Volume** () override final  
*Returns the volume of the box.*

## Additional Inherited Members

### 6.1.1 Detailed Description

This 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 FAShapes::Color & color = FAShapes::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. The [Box](#) is made using triangles. The vertices are ordered in clockwise order.

#### Parameters

in	<i>width</i>	The width of the box.
in	<i>height</i>	The height of the box.
in	<i>depth</i>	The depth of the box.
in	<i>color</i>	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:

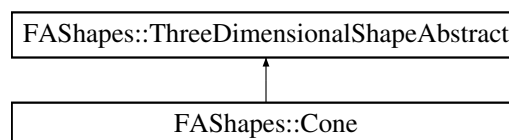
- [FABox.h](#)

## 6.2 FAShapes::Cone Class Reference

This 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 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 FAShapes::Color & color = FAShapes::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 circle, the more circular the cone looks.

#### Parameters

in	<i>radius</i>	The radius of the cone.
in	<i>height</i>	The height of the cone.
in	<i>color</i>	The color of the cone.
in	<i>fillBottom</i>	Pass in true to fill in the bottom of the cone.
in	<i>numCircles</i>	The number of circles the cone has.
in	<i>numVerticesPerCircle</i>	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.

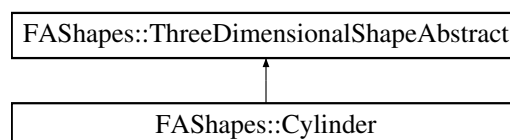
Implements [FAShapes::ThreeDimensionalShapeAbstract](#).

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 FAShapes::Color &color=FAShapes::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 FAShapes::Color & color = FAShapes::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 circle, the more circular it looks.

#### Parameters

in	<i>radius</i>	The radius of the cylinder.
in	<i>height</i>	The height of the cylinder.
in	<i>color</i>	The color of the cylinder.
in	<i>fillTopAndBottom</i>	Pass in true to fill in the top and bottom of the cylinder.
in	<i>numCircles</i>	The number of circles the cylinder has.
in	<i>numVerticesPerCircle</i>	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:

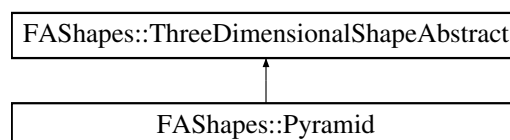
- [FAShapesUtility.h](#)

## 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 FColor::Color &color=FColor::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 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 FColor::Color & color = FColor::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	<i>width</i>	The width of the pyramid.
in	<i>height</i>	The height of the pyramid.
in	<i>depth</i>	The depth of the pyramid.
in	<i>color</i>	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:

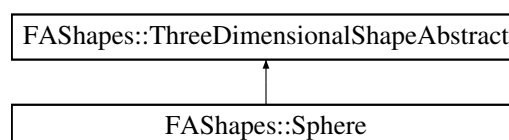
- [FAPyramid.h](#)

## 6.6 FAShapes::Sphere Class Reference

This 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 circle, the more circular the sphere looks.

#### Parameters

in	<i>radius</i>	The radius of the cone.
in	<i>color</i>	The color of the cone.
in	<i>numCircles</i>	The number of circles the cone has.
in	<i>numVerticesPerCircle</i>	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:

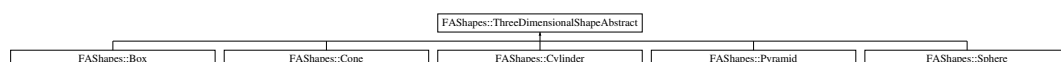
- [FASphere.h](#)

## 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 FColor::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 FColor::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 FColor::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 specified 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 specified 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 **mUpdateLocalToWorldMatrix**
- 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	<i>color</i>	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 FColor::Color & FShapes::ThreeDimensionalShapeAbstract::GetColor ( ) const
```

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

#### 6.7.3.6 GetDrawArguments()

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

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

#### 6.7.3.7 GetLocalToWorldMatrix()

```
const FMath::Matrix4x4 & FShapes::ThreeDimensionalShapeAbstract::GetLocalToWorldMatrix ( )  
const
```

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

#### 6.7.3.8 GetLocalVertices()

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

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

#### 6.7.3.9 GetNumTriangles()

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

Returns the number of triangles the 3D shape has.

#### 6.7.3.10 GetNumVertices()

```
size_t FShapes::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 specified 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 specified 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	<i>vertexList</i>	A pointer to a vertex list.
in	<i>p0Index</i>	The index of the first point of the triangle.
in	<i>p1Index</i>	The index of the second point of the triangle.
in	<i>p2Index</i>	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	<i>vertexList</i>	A pointer to a vertex list.
in	<i>p0Index</i>	The index of the first point of the triangle.
in	<i>p1Index</i>	The index of the second point of the triangle.
in	<i>p2Index</i>	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	<i>p0Index</i>	The index of the first point of the triangle.
in	<i>p1Index</i>	The index of the second point of the triangle.
in	<i>p2Index</i>	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 FBox.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 FBox.h

[Go to the documentation of this file.](#)

```

1  #pragma once
2
3  #include "FAThreeDimensional.h"
4
5  namespace FAShapes
6  {
7      class Box : public ThreeDimensionalShapeAbstract
8      {
9      public:
10
11          Box(float width = 1.0f, float height = 1.0f , float depth = 1.0f,
12              const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f));
13
14          float GetWidth() const;
15
16          float GetHeight() const;
17
18          float GetDepth() const;
19
20          void SetWidth(float width);
21
22          void SetHeight(float height);
23
24          void SetDepth(float depth);
25
26          void UpdateLocalToWorldMatrix() override final;
27
28          float Volume() override final;
29
30      private:
31          //Dimensions of the box
32          float mWidth;
33          float mHeight;
34          float mDepth;
35
36          //Creates the vertices of the box.
37          void CreateVertices() override final;
38
39          //Creates the triangles that make up box.
40          void CreateTriangles() override final;
41
42          //Creates the normals of the box.
43          void CreateNormals() override final;
44      };
45  }
```

## 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.](#)

```

1  #pragma once
2
3  #include "FAThreeDimensional.h"
4
5  namespace FAShapes
6  {
7      class Cone : public ThreeDimensionalShapeAbstract
8      {
9      public:
10         Cone(float radius = 1.0f, float height = 1.0f, const FAColor::Color& color = FAColor::Color(0.0f,
11         0.0f, 0.0f, 1.0f),
12             bool fillBottom = false,
13             unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);
14
15         float GetRadius() const;
16
17         float GetHeight() const;
18
19         void SetRadius(float r);
20
21         void SetHeight(float h);
22
23         void UpdateLocalToWorldMatrix() override final;
24
25         float Volume() override final;
26
27     private:
28         //Radius of the cone.
29         float mRadius;
30
31         //Height of the cone
32         float mHeight;
33
34         //The number of slices the cone has.
35         unsigned int mNumCircles;
36
37         //The number of vertices each slice has.
38         unsigned int mNumVerticesPerCircle;
39
40         //True to fill the bottom, false to not fill the bottom.
41         bool mFillBottom;
42
43         //Creates the vertices of the cone.
44         void CreateVertices() override final;
45
46         //Creates the triangles that make up the cone.
47         void CreateTriangles() override final;
48
49         //Creates the normals of the cone.
50         void CreateNormals() override final;
51     };
52 }

```

## 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.](#)

```

1 #pragma once
2
3 #include "FAThreeDimensional.h"
4
5 namespace FAShapes
6 {
7     class Cylinder : public ThreeDimensionalShapeAbstract
8     {
9     public:
10
11         Cylinder(float radius = 1.0f, float height = 1.0f, const FAColor::Color& color =
12             FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f),
13             bool fillTopAndBottom = false,
14             unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);
15
16         float GetRadius() const;
17
18         float GetHeight() const;
19
20         void SetRadius(float r);
21
22         void SetHeight(float h);
23
24         void UpdateLocalToWorldMatrix() override final;
25
26         float Volume() override final;
27
28     private:
29
30         //radius of the cylinder
31         float mRadius;
32
33         //Height of the cylinder
34         float mHeight;
35
36         //The number of slices the cylinder has.
37         unsigned int mNumCircles;
38
39         //The number of vertices each slice has.
40         unsigned int mNumVerticesPerCircle;
41
42         //True to fill the top and bottom, false to not fill the top and bottom.
43         bool mFillTopAndBottom;
44
45         //Creates the vertices of the cylinder.
46         void CreateVertices() override final;
47
48         //Creates the triangles that make up the cylinder.
49         void CreateTriangles() override final;
50
51         //Creates the normals of the cylinder.
52         void CreateNormals() override final;
53
54     };
55 }

```



## 7.7 FAPyramid.h File Reference

File has a Pyramid class under the namespace [FAShapes](#).

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

### Classes

- class [FAShapes::Pyramid](#)  
*This 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.](#)

```
1 #pragma once
2
3 #include "FAThreeDimensional.h"
4
5 namespace FAShapes
6 {
7     class Pyramid : public ThreeDimensionalShapeAbstract
8     {
9     public:
10
11         Pyramid(float width = 1.0f, float height = 1.0f, float depth = 1.0f,
12             const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f, 1.0f));
13
14         float GetWidth() const;
15
16         float GetHeight() const;
17
18         float GetDepth() const;
19
20         void SetWidth(float width);
21
22         void SetHeight(float height);
23
24         void SetDepth(float depth);
25
26         void UpdateLocalToWorldMatrix() override final;
27
28         float Volume() override final;
29
30     private:
31         //Dimensions of the pyramid
32         float mWidth;
33         float mHeight;
34         float mDepth;
35
36         //Creates the vertices of the pyramid.
37         void CreateVertices() override final;
38
39         //Creates the triangles that make up pyramid.
40         void CreateTriangles() override final;
41
42         //Creates the normals of the pyramid.
43         void CreateNormals() override final;
44     };
45 }
```

## 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.](#)

```
1 #pragma once
2
3
4
5
6
7
8 #include "FAMathEngine.h"
9 #include "FAColor.h"
10
11
12
13
14 namespace FAShapes
15 {
16     struct DrawArguments
17     {
18         unsigned int indexCount;
19         unsigned int locationOfFirstIndex;
20         int indexOfFirstVertex;
21         int indexOfConstantData;
22     };
23
24     struct Vertex
25     {
26         FAMath::Vector3D position;
27         FAColor::Color color;
28         FAMath::Vector3D normal;
29     };
30 }
```

## 7.11 FASphere.h File Reference

File has a Sphere class under the namespace [FAShapes](#).

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

## Classes

- class [FAShapes::Sphere](#)

*This 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.](#)

```

1 #pragma once
2
3 #include "FAThreeDimensional.h"
4
5 namespace FAShapes
6 {
7     class Sphere : public ThreeDimensionalShapeAbstract
8     {
9     public:
10
11         Sphere(float radius = 1.0f, const FAColor::Color& color = FAColor::Color(0.0f, 0.0f, 0.0f ,
12         1.0f),
13             unsigned int numCircles = 20, unsigned int numVerticesPerCircle = 20);
14
15         float GetRadius() const;
16
17         void SetRadius(float r);
18
19         void UpdateLocalToWorldMatrix() override final;
20
21         float Volume() override final;
22
23     private:
24         //Radius of the sphere.
25         float mRadius;
26
27         //The number of slices the sphere has.
28         unsigned int mNumCircles;
29
30         //The number of vertices each slice has.
31         unsigned int mNumVerticesPerCircle;
32
33         //Creates the vertices of the sphere.
34         void CreateVertices() override final;
35
36         //Creates the triangles that make up the sphere.
37         void CreateTriangles() override final;
38
39         //Creates the normals of the sphere.
40         void CreateNormals() override final;
41     };
42 }
```

## 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.](#)

```

1 #pragma once
2
3 #include "FATriangle.h"
4 #include <vector>
5
6 namespace FAShapes
7 {
8     class ThreeDimensionalShapeAbstract
9     {
10     public:
11
12         ThreeDimensionalShapeAbstract(const FAColor::Color& color);
13
14         const FAMath::Vector3D& GetCenter() const;
15
16         const FAMath::Vector3D& GetXAxis() const;
17
18         const FAMath::Vector3D& GetYAxis() const;
19
20         const FAMath::Vector3D& GetZAxis() const;
21
22         const FAMath::Matrix4x4& GetLocalToWorldMatrix() const;
23
24         const Vertex* GetLocalVertices() const;
25
26         const Triangle* GetTriangleList() const;
27
28         const Triangle& GetTriangle(unsigned int index) const;
29
30         const DrawArguments& GetDrawArguments() const;
31
32         const FAColor::Color& GetColor() const;
33
34         size_t GetNumTriangles() const;
35
36         size_t GetNumVertices() const;
37
38         void SetCenter(const FAMath::Vector3D& center);
39
40         void SetCenter(float x, float y, float z);
41
42         void SetColor(const FAColor::Color& color);
43
44         void SetColor(float r, float g, float b, float a);
45
46         void SetDrawArguments(const DrawArguments& drawArgs);
47
48         void SetDrawArguments(unsigned int indexCount, unsigned int locationOfFirstIndex,
49                               int indexOfFirstVertex, int indexOfConstantData);
50
51         void RotateAxes(const FAMath::Matrix4x4& rot);
52
53         void RotateAxes(const FAMath::Quaternion& rotQuaternion);
54     };
55 }
```

```

105
110     void RotateAxes(float angle, const FAMath::Vector3D axis);
111
114     void RotateCenter(const FAMath::Matrix4x4& rot);
115
118     void RotateCenter(const FAMath::Quaternion& rotQuaternion);
119
124     void RotateCenter(float angle, const FAMath::Vector3D axis);
125
128     void TranslateCenter(float x, float y, float z);
129
132     void TranslateCenter(const FAMath::Vector3D& v);
133
136     virtual void UpdateLocalToWorldMatrix() = 0;
137
140     virtual float Volume() = 0;
141
142 protected:
143     //Center of the 3D shape.
144     FAMath::Vector3D mCenter;
145
146     //Local axes of the 3D shape.
147     FAMath::Vector3D mX;
148     FAMath::Vector3D mY;
149     FAMath::Vector3D mZ;
150
151     //Color of the 3D shape.
152     FAColor::Color mColor;
153
154     //If true, updates the local to world matrix.
155     bool mUpdateLocalToWorldMatrix;
156
157     //Local to world matrix of the 3D shape.
158     FAMath::Matrix4x4 mLocalToWorld;
159
160     //Local vertices of the 3D shape.
161     std::vector<Vertex> mLocalVertices;
162
163     //The triangles that make up the 3D shape.
164     std::vector<Triangle> mTriangles;
165
166     //The arguments needed to render the 3D shape.
167     DrawArguments mSphereDrawArguments{};
168
171     void Quad(unsigned int a, unsigned int b, unsigned int c, unsigned int d);
172
175     virtual void CreateVertices() = 0;
176
179     virtual void CreateTriangles() = 0;
180
183     virtual void CreateNormals() = 0;
184
185 };
186 }

```

## 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.](#)

```

1  #pragma once
2
3  #include "FAShapesUtility.h"
4
5  namespace FAShapes
6  {
7      class Triangle
8      {
9      public:
10
11          Triangle(Vertex* vertexList = nullptr, unsigned int p0Index = 0, unsigned int p1Index = 0,
12                  unsigned int p2Index = 0);
13
14          const Vertex& GetP0() const;
15
16          const Vertex& GetP1() const;
17
18          const Vertex& GetP2() const;
19
20          unsigned int GetP0Index() const;
21
22          unsigned int GetP1Index() const;
23
24          unsigned int GetP2Index() const;
25
26          FAMath::Vector3D GetNormal() const;
27
28          FAMath::Vector3D GetCenter() const;
29
30          void SetVertexList(Vertex* vertexList);
31
32          void SetP0Index(unsigned int index);
33
34          void SetP1Index(unsigned int index);
35
36          void SetP2Index(unsigned int index);
37
38          void SetTriangleIndices(unsigned int p0Index, unsigned int p1Index, unsigned int p2Index);
39
40          void SetTriangle(Vertex* vertexList, unsigned int p0Index, unsigned int p1Index, unsigned int
41                           p2Index);
42
43      private:
44          Vertex* mVertexList; //pointer to a vertex list
45          unsigned int mIndexList[3]; //indices into a vertex list
46      };
47 }

```

# Index

- Box
  - FAShapes::Box, [12](#)
- Cone
  - FAShapes::Cone, [15](#)
- CreateNormals
  - FAShapes::ThreeDimensionalShapeAbstract, [27](#)
- CreateTriangles
  - FAShapes::ThreeDimensionalShapeAbstract, [27](#)
- CreateVertices
  - FAShapes::ThreeDimensionalShapeAbstract, [27](#)
- Cylinder
  - FAShapes::Cylinder, [17](#)
- FABox.h, [39](#)
- FACone.h, [40](#)
- FACylinder.h, [41](#)
- FAPyramid.h, [43](#)
- FAShapes, [9](#)
- FAShapes::Box, [11](#)
  - Box, [12](#)
  - GetDepth, [12](#)
  - GetHeight, [12](#)
  - GetWidth, [12](#)
  - SetDepth, [13](#)
  - SetHeight, [13](#)
  - SetWidth, [13](#)
  - UpdateLocalToWorldMatrix, [13](#)
  - Volume, [13](#)
- FAShapes::Cone, [14](#)
  - Cone, [15](#)
  - GetHeight, [15](#)
  - GetRadius, [15](#)
  - SetHeight, [15](#)
  - SetRadius, [16](#)
  - UpdateLocalToWorldMatrix, [16](#)
  - Volume, [16](#)
- FAShapes::Cylinder, [16](#)
  - Cylinder, [17](#)
  - GetHeight, [18](#)
  - GetRadius, [18](#)
  - SetHeight, [18](#)
  - SetRadius, [18](#)
  - UpdateLocalToWorldMatrix, [18](#)
  - Volume, [18](#)
- FAShapes::DrawArguments, [19](#)
- FAShapes::Pyramid, [19](#)
  - GetDepth, [21](#)
  - GetHeight, [21](#)
  - GetWidth, [21](#)
  - Pyramid, [20](#)
  - SetDepth, [21](#)
  - SetHeight, [21](#)
  - SetWidth, [21](#)
  - UpdateLocalToWorldMatrix, [22](#)
  - Volume, [22](#)
- FAShapes::Sphere, [22](#)
  - GetRadius, [23](#)
  - SetRadius, [24](#)
  - Sphere, [23](#)
  - UpdateLocalToWorldMatrix, [24](#)
  - Volume, [24](#)
- FAShapes::ThreeDimensionalShapeAbstract, [24](#)
  - CreateNormals, [27](#)
  - CreateTriangles, [27](#)
  - CreateVertices, [27](#)
  - GetCenter, [27](#)
  - GetColor, [27](#)
  - GetDrawArguments, [28](#)
  - GetLocalToWorldMatrix, [28](#)
  - GetLocalVertices, [28](#)
  - GetNumTriangles, [28](#)
  - GetNumVertices, [28](#)
  - GetTriangle, [28](#)
  - GetTriangleList, [29](#)
  - GetXAxis, [29](#)
  - GetYAxis, [29](#)
  - GetZAxis, [29](#)
  - Quad, [29](#)
  - RotateAxes, [29](#), [30](#)
  - RotateCenter, [30](#)
  - SetCenter, [31](#)
  - SetColor, [31](#)
  - SetDrawArguments, [31](#), [32](#)
  - ThreeDimensionalShapeAbstract, [26](#)
  - TranslateCenter, [32](#)
  - UpdateLocalToWorldMatrix, [32](#)
  - Volume, [32](#)
- FAShapes::Triangle, [33](#)
  - GetCenter, [34](#)
  - GetNormal, [34](#)
  - GetP0, [34](#)
  - GetP0Index, [35](#)
  - GetP1, [35](#)
  - GetP1Index, [35](#)
  - GetP2, [35](#)
  - GetP2Index, [35](#)
  - SetP0Index, [35](#)
  - SetP1Index, [36](#)

- SetP2Index, 36
- SetTriangle, 36
- SetTriangleIndices, 36
- SetVertexList, 37
- Triangle, 34
- FAShapes::Vertex, 37
- FAShapesUtility.h, 44
- FASphere.h, 44
- FAThreeDimensional.h, 45
- FATriangle.h, 47
- GetCenter
  - FAShapes::ThreeDimensionalShapeAbstract, 27
  - FAShapes::Triangle, 34
- GetColor
  - FAShapes::ThreeDimensionalShapeAbstract, 27
- GetDepth
  - FAShapes::Box, 12
  - FAShapes::Pyramid, 21
- GetDrawArguments
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetHeight
  - FAShapes::Box, 12
  - FAShapes::Cone, 15
  - FAShapes::Cylinder, 18
  - FAShapes::Pyramid, 21
- GetLocalToWorldMatrix
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetLocalVertices
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetNormal
  - FAShapes::Triangle, 34
- GetNumTriangles
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetNumVertices
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetP0
  - FAShapes::Triangle, 34
- GetP0Index
  - FAShapes::Triangle, 35
- GetP1
  - FAShapes::Triangle, 35
- GetP1Index
  - FAShapes::Triangle, 35
- GetP2
  - FAShapes::Triangle, 35
- GetP2Index
  - FAShapes::Triangle, 35
- GetRadius
  - FAShapes::Cone, 15
  - FAShapes::Cylinder, 18
  - FAShapes::Sphere, 23
- GetTriangle
  - FAShapes::ThreeDimensionalShapeAbstract, 28
- GetTriangleList
  - FAShapes::ThreeDimensionalShapeAbstract, 29
- GetWidth
  - FAShapes::Box, 12
  - FAShapes::Pyramid, 21
- GetXAxis
  - FAShapes::ThreeDimensionalShapeAbstract, 29
- GetYAxis
  - FAShapes::ThreeDimensionalShapeAbstract, 29
- GetZAxis
  - FAShapes::ThreeDimensionalShapeAbstract, 29
- Pyramid
  - FAShapes::Pyramid, 20
- Quad
  - FAShapes::ThreeDimensionalShapeAbstract, 29
- RotateAxes
  - FAShapes::ThreeDimensionalShapeAbstract, 29, 30
- RotateCenter
  - FAShapes::ThreeDimensionalShapeAbstract, 30
- SetCenter
  - FAShapes::ThreeDimensionalShapeAbstract, 31
- SetColor
  - FAShapes::ThreeDimensionalShapeAbstract, 31
- SetDepth
  - FAShapes::Box, 13
  - FAShapes::Pyramid, 21
- SetDrawArguments
  - FAShapes::ThreeDimensionalShapeAbstract, 31, 32
- SetHeight
  - FAShapes::Box, 13
  - FAShapes::Cone, 15
  - FAShapes::Cylinder, 18
  - FAShapes::Pyramid, 21
- SetP0Index
  - FAShapes::Triangle, 35
- SetP1Index
  - FAShapes::Triangle, 36
- SetP2Index
  - FAShapes::Triangle, 36
- SetRadius
  - FAShapes::Cone, 16
  - FAShapes::Cylinder, 18
  - FAShapes::Sphere, 24
- SetTriangle
  - FAShapes::Triangle, 36
- SetTriangleIndices
  - FAShapes::Triangle, 36
- SetVertexList
  - FAShapes::Triangle, 37
- SetWidth
  - FAShapes::Box, 13
  - FAShapes::Pyramid, 21
- Sphere
  - FAShapes::Sphere, 23
- ThreeDimensionalShapeAbstract
  - FAShapes::ThreeDimensionalShapeAbstract, 26
- TranslateCenter



FAShapes::ThreeDimensionalShapeAbstract, [32](#)

Triangle

FAShapes::Triangle, [34](#)

UpdateLocalToWorldMatrix

FAShapes::Box, [13](#)

FAShapes::Cone, [16](#)

FAShapes::Cylinder, [18](#)

FAShapes::Pyramid, [22](#)

FAShapes::Sphere, [24](#)

FAShapes::ThreeDimensionalShapeAbstract, [32](#)

Volume

FAShapes::Box, [13](#)

FAShapes::Cone, [16](#)

FAShapes::Cylinder, [18](#)

FAShapes::Pyramid, [22](#)

FAShapes::Sphere, [24](#)

FAShapes::ThreeDimensionalShapeAbstract, [32](#)