

Farouq Adepetu's Shapes Engine

Generated by Doxygen 1.9.4



<b>1 Namespace Index</b>	<b>1</b>
1.1 Namespace List	1
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Namespace Documentation</b>	<b>7</b>
4.1 ShapesEngine Namespace Reference	7
4.1.1 Detailed Description	8
4.1.2 Function Documentation	8
4.1.2.1 ComputeCenter()	8
4.1.2.2 ComputeNormal()	8
4.1.2.3 CreateBox()	9
4.1.2.4 CreateCone()	9
4.1.2.5 CreateCylinder()	9
4.1.2.6 CreatePyramid()	9
4.1.2.7 CreateSphere()	9
4.1.2.8 Quad()	10
4.1.2.9 RenderShape()	10
4.1.2.10 SetDrawArguments()	10
4.1.2.11 UpdateShape()	10
<b>5 Class Documentation</b>	<b>11</b>
5.1 ShapesEngine::Box Class Reference	11
5.1.1 Detailed Description	12
5.1.2 Constructor & Destructor Documentation	12
5.1.2.1 Box()	12
5.1.3 Member Function Documentation	12
5.1.3.1 GetDepth()	12
5.1.3.2 GetHeight()	12
5.1.3.3 GetShape() [1/2]	12
5.1.3.4 GetShape() [2/2]	13
5.1.3.5 GetWidth()	13
5.1.3.6 InitializeBox()	13
5.1.3.7 SetDepth()	13
5.1.3.8 SetHeight()	14
5.1.3.9 SetWidth()	14
5.1.3.10 UpdateModelMatrix()	14
5.1.3.11 Volume()	14
5.2 ShapesEngine::Cone Class Reference	14
5.2.1 Detailed Description	15

5.2.2 Constructor & Destructor Documentation	15
5.2.2.1 Cone()	15
5.2.3 Member Function Documentation	15
5.2.3.1 GetHeight()	15
5.2.3.2 GetRadius()	16
5.2.3.3 GetShape() [1/2]	16
5.2.3.4 GetShape() [2/2]	16
5.2.3.5 InitializeCone()	16
5.2.3.6 SetHeight()	16
5.2.3.7 SetRadius()	17
5.2.3.8 UpdateModelMatrix()	17
5.2.3.9 Volume()	17
5.3 ShapesEngine::Cylinder Class Reference	17
5.3.1 Detailed Description	18
5.3.2 Constructor & Destructor Documentation	18
5.3.2.1 Cylinder()	18
5.3.3 Member Function Documentation	18
5.3.3.1 GetHeight()	18
5.3.3.2 GetRadius()	19
5.3.3.3 GetShape() [1/2]	19
5.3.3.4 GetShape() [2/2]	19
5.3.3.5 InitializeCylinder()	19
5.3.3.6 SetHeight()	19
5.3.3.7 SetRadius()	20
5.3.3.8 UpdateModelMatrix()	20
5.3.3.9 Volume()	20
5.4 ShapesEngine::Pyramid Class Reference	20
5.4.1 Detailed Description	21
5.4.2 Constructor & Destructor Documentation	21
5.4.2.1 Pyramid()	21
5.4.3 Member Function Documentation	21
5.4.3.1 GetDepth()	22
5.4.3.2 GetHeight()	22
5.4.3.3 GetShape() [1/2]	22
5.4.3.4 GetShape() [2/2]	22
5.4.3.5 GetWidth()	22
5.4.3.6 InitializePyramid()	22
5.4.3.7 SetDepth()	23
5.4.3.8 SetHeight()	23
5.4.3.9 SetWidth()	23
5.4.3.10 UpdateModelMatrix()	23
5.4.3.11 Volume()	24

5.5 ShapesEngine::Sphere Class Reference . . . . .	24
5.5.1 Detailed Description . . . . .	24
5.5.2 Constructor & Destructor Documentation . . . . .	24
5.5.2.1 Sphere() . . . . .	25
5.5.3 Member Function Documentation . . . . .	25
5.5.3.1 GetRadius() . . . . .	25
5.5.3.2 GetShape() [1/2] . . . . .	25
5.5.3.3 GetShape() [2/2] . . . . .	25
5.5.3.4 InitializeSphere() . . . . .	25
5.5.3.5 SetRadius() . . . . .	26
5.5.3.6 UpdateModelMatrix() . . . . .	26
5.5.3.7 Volume() . . . . .	26
5.6 ShapesEngine::ThreeDimensionalShape Struct Reference . . . . .	26
5.6.1 Detailed Description . . . . .	27
5.7 ShapesEngine::Triangle Struct Reference . . . . .	27
5.7.1 Detailed Description . . . . .	27
5.8 ShapesEngine::Vertex Struct Reference . . . . .	27
5.8.1 Detailed Description . . . . .	27
<b>6 File Documentation</b>	<b>29</b>
6.1 Box.h . . . . .	29
6.2 Cone.h . . . . .	29
6.3 CreateShapes.h . . . . .	30
6.4 Cylinder.h . . . . .	30
6.5 Pyramid.h . . . . .	31
6.6 Sphere.h . . . . .	31
6.7 ThreeDimensionalShape.h . . . . .	32
6.8 Triangle.h . . . . .	32
6.9 Vertex.h . . . . .	33
<b>Index</b>	<b>35</b>



# Chapter 1

## Namespace Index

### 1.1 Namespace List

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

<a href="#">ShapesEngine</a>	
An engine for rendering 3D shapes . . . . .	<a href="#">7</a>





## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">ShapesEngine::Box</a>	
This class is used to create a box . . . . .	11
<a href="#">ShapesEngine::Cone</a>	
This class is used to create a cone . . . . .	14
<a href="#">ShapesEngine::Cylinder</a>	
This class is used to create a cylinder . . . . .	17
<a href="#">ShapesEngine::Pyramid</a>	
This class is used to create a pyramid . . . . .	20
<a href="#">ShapesEngine::Sphere</a>	
This class is used to create a sphere . . . . .	24
<a href="#">ShapesEngine::ThreeDimensionalShape</a>	
The struct stores the properties needed to render a 3D shape . . . . .	26
<a href="#">ShapesEngine::Triangle</a>	
The struct stores a pointer to a vertex list and indices to the vertices of the triangle . . . . .	27
<a href="#">ShapesEngine::Vertex</a>	
Data that describes a vertex . . . . .	27



## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">Box.h</a>	??
<a href="#">Cone.h</a>	??
<a href="#">CreateShapes.h</a>	??
<a href="#">Cylinder.h</a>	??
<a href="#">Pyramid.h</a>	??
<a href="#">Sphere.h</a>	??
<a href="#">ThreeDimensionalShape.h</a>	??
<a href="#">Triangle.h</a>	??
<a href="#">Vertex.h</a>	??



## Chapter 4

# Namespace Documentation

### 4.1 ShapesEngine Namespace Reference

An engine for rendering 3D shapes.

#### Classes

- class [Box](#)  
*This class is used to create a box.*
- class [Cone](#)  
*This class is used to create a cone.*
- class [Cylinder](#)  
*This class is used to create a cylinder.*
- class [Pyramid](#)  
*This class is used to create a pyramid.*
- class [Sphere](#)  
*This class is used to create a sphere.*
- struct [ThreeDimensionalShape](#)  
*The struct stores the properties needed to render a 3D shape.*
- struct [Triangle](#)  
*The struct stores a pointer to a vertex list and indices to the vertices of the triangle.*
- struct [Vertex](#)  
*Data that describes a vertex.*

#### Functions

- void [CreateBox](#) (std::vector< [Vertex](#) > &vertices, std::vector< [Triangle](#) > &triangles)  
*Creates the vertices of a unit box and connects them using triangles.*
- void [CreateCone](#) (std::vector< [Vertex](#) > &vertices, std::vector< [Triangle](#) > &triangles, unsigned int num↵  
VerticesPerCircle=20, unsigned int numCircles=20)  
*Creates the vertices of a unit cone and connects them using triangles.*
- void [CreateCylinder](#) (std::vector< [Vertex](#) > &vertices, std::vector< [Triangle](#) > &triangles, unsigned int num↵  
VerticesPerCircle=20, unsigned int numCircles=20)  
*Creates the vertices of a unit cone and connects them using triangles.*

- void [CreateSphere](#) (std::vector< [Vertex](#) > &vertices, std::vector< [Triangle](#) > &triangles, unsigned int numVerticesPerCircle=20, unsigned int numCircles=20)  
*Creates the vertices of a unit sphere and connects them using triangles.*
- void [CreatePyramid](#) (std::vector< [Vertex](#) > &vertices, std::vector< [Triangle](#) > &triangles)  
*Creates the vertices of a unit pyramid and connects them using triangles.*
- void [SetDrawArguments](#) ([ThreeDimensionalShape](#) &shape, unsigned int indexCount, unsigned int locationFirstIndex, int indexFirstVertex, unsigned int indexConstantData, const std::wstring &constantBufferKey, unsigned int rootParameterIndex, D3D\_PRIMITIVE\_TOPOLOGY primitive)  
*Sets the draw arguments used to render the 3D shape.*
- void [UpdateShape](#) (const [ThreeDimensionalShape](#) &shape, RenderingEngine::RenderScene \*scene, const void \*data, unsigned int size)  
*Updates the 3D shapes constant data.*
- void [RenderShape](#) (const [ThreeDimensionalShape](#) &shape, RenderingEngine::RenderScene \*scene)  
*Renders the 3D shape.*
- vec3 [ComputeNormal](#) (const [Triangle](#) &triangle)  
*Returns the normal of the triangle.*
- vec3 [ComputeCenter](#) (const [Triangle](#) &triangle)  
*Returns the center of the triangle.*
- void [Quad](#) (unsigned int a, unsigned int b, unsigned int c, unsigned int d, std::vector< [Triangle](#) > &triangles, [Vertex](#) \*vertices)  
*Stores the indices of the vertices of the triangles that make up a shape.*

### 4.1.1 Detailed Description

An engine for rendering 3D shapes.

### 4.1.2 Function Documentation

#### 4.1.2.1 ComputeCenter()

```
vec3 ShapesEngine::ComputeCenter (
    const Triangle & triangle )
```

Returns the center of the triangle.

#### 4.1.2.2 ComputeNormal()

```
vec3 ShapesEngine::ComputeNormal (
    const Triangle & triangle )
```

Returns the normal of the triangle.

#### 4.1.2.3 CreateBox()

```
void ShapesEngine::CreateBox (
    std::vector< Vertex > & vertices,
    std::vector< Triangle > & triangles )
```

Creates the vertices of a unit box and connects them using triangles.

Also computes the normal for each vertex.

#### 4.1.2.4 CreateCone()

```
void ShapesEngine::CreateCone (
    std::vector< Vertex > & vertices,
    std::vector< Triangle > & triangles,
    unsigned int numVerticesPerCircle = 20,
    unsigned int numCircles = 20 )
```

Creates the vertices of a unit cone and connects them using triangles.

Also computes the normal for each vertex. Uses the UV-method to create the vertices of the cone.

#### 4.1.2.5 CreateCylinder()

```
void ShapesEngine::CreateCylinder (
    std::vector< Vertex > & vertices,
    std::vector< Triangle > & triangles,
    unsigned int numVerticesPerCircle = 20,
    unsigned int numCircles = 20 )
```

Creates the vertices of a unit cone and connects them using triangles.

Also computes the normal for each vertex./n Uses the UV-method to create the vertices of the cylinder.

#### 4.1.2.6 CreatePyramid()

```
void ShapesEngine::CreatePyramid (
    std::vector< Vertex > & vertices,
    std::vector< Triangle > & triangles )
```

Creates the vertices of a unit pyramid and connects them using triangles.

Also computes the normal for each vertex.

#### 4.1.2.7 CreateSphere()

```
void ShapesEngine::CreateSphere (
    std::vector< Vertex > & vertices,
    std::vector< Triangle > & triangles,
    unsigned int numVerticesPerCircle = 20,
    unsigned int numCircles = 20 )
```

Creates the vertices of a unit sphere and connects them using triangles.

Also computes the normal for each vertex./n Uses the UV-method to create the vertices of the sphere.

#### 4.1.2.8 Quad()

```
void ShapesEngine::Quad (
    unsigned int a,
    unsigned int b,
    unsigned int c,
    unsigned int d,
    std::vector< Triangle > & triangles,
    Vertex * vertices )
```

Stores the indices of the vertices of the triangles that make up a shape.

#### 4.1.2.9 RenderShape()

```
void ShapesEngine::RenderShape (
    const ThreeDimensionalShape & shape,
    RenderingEngine::RenderScene * scene )
```

Renders the 3D shape.

#### 4.1.2.10 SetDrawArguments()

```
void ShapesEngine::SetDrawArguments (
    ThreeDimensionalShape & shape,
    unsigned int indexCount,
    unsigned int locationFirstIndex,
    int indexFirstVertex,
    unsigned int indexConstantData,
    const std::wstring & constantBufferKey,
    unsigned int rootParameterIndex,
    D3D_PRIMITIVE_TOPOLOGY primitive )
```

Sets the draw arguments used to render the 3D shape.

#### 4.1.2.11 UpdateShape()

```
void ShapesEngine::UpdateShape (
    const ThreeDimensionalShape & shape,
    RenderingEngine::RenderScene * scene,
    const void * data,
    unsigned int size )
```

Updates the 3D shapes constant data.



## Chapter 5

# Class Documentation

### 5.1 ShapesEngine::Box Class Reference

This class is used to create a box.

```
#include "Box.h"
```

#### Public Member Functions

- [Box](#) ()  
*Creates a [Box](#) object. Call [InitializeBox](#) to initialize the box.*
- void [InitializeBox](#) (float width, float height, float depth, const vec3 position, const MathEngine::Quaternion orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the box.*
- const [ThreeDimensionalShape](#) & [GetShape](#) () const  
*Returns the [ThreeDimensionalShape](#) object.*
- [ThreeDimensionalShape](#) & [GetShape](#) ()  
*Returns the [ThreeDimensionalShape](#) object.*
- 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.*
- void [SetHeight](#) (float height)  
*Sets the height of the box.*
- void [SetDepth](#) (float depth)  
*Sets the depth of the box.*
- void [UpdateModelMatrix](#) ()  
*Updates the boxes model matrix.*
- float [Volume](#) ()  
*Returns the volume of the box.*

### 5.1.1 Detailed Description

This class is used to create a box.

### 5.1.2 Constructor & Destructor Documentation

#### 5.1.2.1 Box()

```
ShapesEngine::Box::Box ( )
```

Creates a [Box](#) object. Call InitializeBox to initialize the box.

### 5.1.3 Member Function Documentation

#### 5.1.3.1 GetDepth()

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

Returns the depth of the box.

#### 5.1.3.2 GetHeight()

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

Returns the height of the box.

#### 5.1.3.3 GetShape() [1/2]

```
ThreeDimensionalShape & ShapesEngine::Box::GetShape ( )
```

Returns the [ThreeDimensionalShape](#) object.

#### 5.1.3.4 GetShape() [2/2]

```
const ThreeDimensionalShape & ShapesEngine::Box::GetShape ( ) const
```

Returns the [ThreeDimensionalShape](#) object.

#### 5.1.3.5 GetWidth()

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

Returns the width of the box.

#### 5.1.3.6 InitializeBox()

```
void ShapesEngine::Box::InitializeBox (
    float width,
    float height,
    float depth,
    const vec3 position,
    const MathEngine::Quaternion orientation,
    const RenderingEngine::Color & color )
```

Initializes the properties of the box.

##### 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>position</i>	The position of the box.
in	<i>orientation</i>	The orientation of the box.
in	<i>color</i>	The color of the box.

#### 5.1.3.7 SetDepth()

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

Sets the depth of the box.

#### 5.1.3.8 SetHeight()

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

Sets the height of the box.

#### 5.1.3.9 SetWidth()

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

Sets the width of the box.

#### 5.1.3.10 UpdateModelMatrix()

```
void ShapesEngine::Box::UpdateModelMatrix ( )
```

Updates the boxs model matrix.

#### 5.1.3.11 Volume()

```
float ShapesEngine::Box::Volume ( )
```

Returns the volume of the box.

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

- Box.h

## 5.2 ShapesEngine::Cone Class Reference

This class is used to create a cone.

```
#include "Cone.h"
```

## Public Member Functions

- [Cone](#) ()  
*Creates a [Cone](#) object. Call [InitializeCone](#) to initialize the cone.*
- void [InitializeCone](#) (float radius, float height, const vec3 position, const MathEngine::Quaternion orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the cone.*
- const [ThreeDimensionalShape](#) & [GetShape](#) () const  
*Returns the [ThreeDimensionalShape](#) object.*
- [ThreeDimensionalShape](#) & [GetShape](#) ()  
*Returns the [ThreeDimensionalShape](#) object.*
- float [GetRadius](#) () const  
*Returns the radius of the cone.*
- float [GetHeight](#) () const  
*Returns the height of the cone.*
- void [SetRadius](#) (float radius)  
*Sets the radius of the cone.*
- void [SetHeight](#) (float height)  
*Sets the height of the cone.*
- void [UpdateModelMatrix](#) ()  
*Updates the cones model matrix.*
- float [Volume](#) ()  
*Returns the volume of the cone.*

### 5.2.1 Detailed Description

This class is used to create a cone.

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 Cone()

```
ShapesEngine::Cone::Cone ( )
```

Creates a [Cone](#) object. Call [InitializeCone](#) to initialize the cone.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 GetHeight()

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

Returns the height of the cone.

### 5.2.3.2 GetRadius()

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

Returns the radius of the cone.

### 5.2.3.3 GetShape() [1/2]

```
ThreeDimensionalShape & ShapesEngine::Cone::GetShape ( )
```

Returns the [ThreeDimensionalShape](#) object.

### 5.2.3.4 GetShape() [2/2]

```
const ThreeDimensionalShape & ShapesEngine::Cone::GetShape ( ) const
```

Returns the [ThreeDimensionalShape](#) object.

### 5.2.3.5 InitializeCone()

```
void ShapesEngine::Cone::InitializeCone (
    float radius,
    float height,
    const vec3 position,
    const MathEngine::Quaternion orientation,
    const RenderingEngine::Color & color )
```

Initializes the properties of the cone.

#### Parameters

in	<i>width</i>	The radius of the cone.
in	<i>height</i>	The height of the cone.
in	<i>position</i>	The position of the cone.
in	<i>orientation</i>	The orientation of the cone.
in	<i>color</i>	The color of the cone.

### 5.2.3.6 SetHeight()

```
void ShapesEngine::Cone::SetHeight (
    float height )
```

Sets the height of the cone.

#### 5.2.3.7 SetRadius()

```
void ShapesEngine::Cone::SetRadius (
    float radius )
```

Sets the radius of the cone.

#### 5.2.3.8 UpdateModelMatrix()

```
void ShapesEngine::Cone::UpdateModelMatrix ( )
```

Updates the cones model matrix.

#### 5.2.3.9 Volume()

```
float ShapesEngine::Cone::Volume ( )
```

Returns the volume of the cone.

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

- Cone.h

## 5.3 ShapesEngine::Cylinder Class Reference

This class is used to create a cylinder.

```
#include "Cylinder.h"
```

## Public Member Functions

- [Cylinder](#) ()  
*Creates a [Cylinder](#) object. Call [InitializeCylinder](#) to initialize the cylinder.*
- void [InitializeCylinder](#) (float radius, float height, const vec3 position, const MathEngine::Quaternion orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the cylinder.*
- const [ThreeDimensionalShape](#) & [GetShape](#) () const  
*Returns the [ThreeDimensionalShape](#) object.*
- [ThreeDimensionalShape](#) & [GetShape](#) ()  
*Returns the [ThreeDimensionalShape](#) object.*
- float [GetRadius](#) () const  
*Returns the radius of the cylinder.*
- float [GetHeight](#) () const  
*Returns the height of the cylinder.*
- void [SetRadius](#) (float radius)  
*Sets the radius of the cylinder.*
- void [SetHeight](#) (float height)  
*Sets the height of the cylinder.*
- void [UpdateModelMatrix](#) ()  
*Updates the cylinders model matrix.*
- float [Volume](#) ()  
*Returns the volume of the cylinder.*

### 5.3.1 Detailed Description

This class is used to create a cylinder.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 Cylinder()

```
ShapesEngine::Cylinder::Cylinder ( )
```

Creates a [Cylinder](#) object. Call [InitializeCylinder](#) to initialize the cylinder.

### 5.3.3 Member Function Documentation

#### 5.3.3.1 GetHeight()

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

Returns the height of the cylinder.



### 5.3.3.2 GetRadius()

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

Returns the radius of the cylinder.

### 5.3.3.3 GetShape() [1/2]

```
ThreeDimensionalShape & ShapesEngine::Cylinder::GetShape ( )
```

Returns the [ThreeDimensionalShape](#) object.

### 5.3.3.4 GetShape() [2/2]

```
const ThreeDimensionalShape & ShapesEngine::Cylinder::GetShape ( ) const
```

Returns the [ThreeDimensionalShape](#) object.

### 5.3.3.5 InitializeCylinder()

```
void ShapesEngine::Cylinder::InitializeCylinder (
    float radius,
    float height,
    const vec3 position,
    const MathEngine::Quaternion orientation,
    const RenderingEngine::Color & color )
```

Initializes the properties of the cylinder.

#### Parameters

in	<i>width</i>	The radius of the cylinder.
in	<i>height</i>	The height of the cylinder.
in	<i>position</i>	The position of the cylinder.
in	<i>orientation</i>	The orientation of the cylinder.
in	<i>color</i>	The color of the cylinder.

### 5.3.3.6 SetHeight()

```
void ShapesEngine::Cylinder::SetHeight (
    float height )
```

Sets the height of the cylinder.

#### 5.3.3.7 SetRadius()

```
void ShapesEngine::Cylinder::SetRadius (
    float radius )
```

Sets the radius of the cylinder.

#### 5.3.3.8 UpdateModelMatrix()

```
void ShapesEngine::Cylinder::UpdateModelMatrix ( )
```

Updates the cylinders model matrix.

#### 5.3.3.9 Volume()

```
float ShapesEngine::Cylinder::Volume ( )
```

Returns the volume of the cylinder.

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

- Cylinder.h

## 5.4 ShapesEngine::Pyramid Class Reference

This class is used to create a pyramid.

```
#include "Pyramid.h"
```

## Public Member Functions

- [Pyramid](#) ()  
*Creates a [Pyramid](#) object. Call [InitializePyramid](#) to initialize the pyramid.*
- void [InitializePyramid](#) (float width, float height, float depth, const vec3 position, const MathEngine::Quaternion orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the pyramid.*
- const [ThreeDimensionalShape](#) & [GetShape](#) () const  
*Returns the [ThreeDimensionalShape](#) object.*
- [ThreeDimensionalShape](#) & [GetShape](#) ()  
*Returns the [ThreeDimensionalShape](#) object.*
- 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.*
- void [SetHeight](#) (float height)  
*Sets the height of the pyramid.*
- void [SetDepth](#) (float depth)  
*Sets the depth of the pyramid.*
- void [UpdateModelMatrix](#) ()  
*Updates the pyramids model matrix.*
- float [Volume](#) ()  
*Returns the volume of the pyramid.*

### 5.4.1 Detailed Description

This class is used to create a pyramid.

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 Pyramid()

```
ShapesEngine::Pyramid::Pyramid ( )
```

Creates a [Pyramid](#) object. Call [InitializePyramid](#) to initialize the pyramid.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 GetDepth()

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

Returns the depth of the pyramid.

#### 5.4.3.2 GetHeight()

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

Returns the height of the pyramid.

#### 5.4.3.3 GetShape() [1/2]

```
ThreeDimensionalShape & ShapesEngine::Pyramid::GetShape ( )
```

Returns the [ThreeDimensionalShape](#) object.

#### 5.4.3.4 GetShape() [2/2]

```
const ThreeDimensionalShape & ShapesEngine::Pyramid::GetShape ( ) const
```

Returns the [ThreeDimensionalShape](#) object.

#### 5.4.3.5 GetWidth()

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

Returns the width of the pyramid.

#### 5.4.3.6 InitializePyramid()

```
void ShapesEngine::Pyramid::InitializePyramid (
    float width,
    float height,
    float depth,
    const vec3 position,
    const MathEngine::Quaternion orientation,
    const RenderingEngine::Color & color )
```

Initializes the properties of the pyramid.

## 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>position</i>	The position of the pyramid.
in	<i>orientation</i>	The orientation of the pyramid.
in	<i>color</i>	The color of the pyramid.

**5.4.3.7 SetDepth()**

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

Sets the depth of the pyramid.

**5.4.3.8 SetHeight()**

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

Sets the height of the pyramid.

**5.4.3.9 SetWidth()**

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

Sets the width of the pyramid.

**5.4.3.10 UpdateModelMatrix()**

```
void ShapesEngine::Pyramid::UpdateModelMatrix ( )
```

Updates the pyramids model matrix.

#### 5.4.3.11 Volume()

```
float ShapesEngine::Pyramid::Volume ( )
```

Returns the volume of the pyramid.

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

- Pyramid.h

## 5.5 ShapesEngine::Sphere Class Reference

This class is used to create a sphere.

```
#include "Sphere.h"
```

### Public Member Functions

- [Sphere](#) ()  
*Creates a [Sphere](#) object. Call InitializeSphere to initialize the sphere.*
- void [InitializeSphere](#) (float radius, const vec3 position, const MathEngine::Quaternion orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the sphere.*
- const [ThreeDimensionalShape](#) & [GetShape](#) () const  
*Returns the [ThreeDimensionalShape](#) object.*
- [ThreeDimensionalShape](#) & [GetShape](#) ()  
*Returns the [ThreeDimensionalShape](#) object.*
- float [GetRadius](#) () const  
*Returns the radius of the sphere.*
- void [SetRadius](#) (float radius)  
*Sets the radius of the sphere.*
- void [UpdateModelMatrix](#) ()  
*Updates the spheres model matrix.*
- float [Volume](#) ()  
*Returns the volume of the sphere.*

### 5.5.1 Detailed Description

This class is used to create a sphere.

### 5.5.2 Constructor & Destructor Documentation

### 5.5.2.1 Sphere()

```
ShapesEngine::Sphere::Sphere ( )
```

Creates a [Sphere](#) object. Call InitializeSphere to initialize the sphere.

## 5.5.3 Member Function Documentation

### 5.5.3.1 GetRadius()

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

Returns the radius of the sphere.

### 5.5.3.2 GetShape() [1/2]

```
ThreeDimensionalShape & ShapesEngine::Sphere::GetShape ( )
```

Returns the [ThreeDimensionalShape](#) object.

### 5.5.3.3 GetShape() [2/2]

```
const ThreeDimensionalShape & ShapesEngine::Sphere::GetShape ( ) const
```

Returns the [ThreeDimensionalShape](#) object.

### 5.5.3.4 InitializeSphere()

```
void ShapesEngine::Sphere::InitializeSphere (
    float radius,
    const vec3 position,
    const MathEngine::Quaternion orientation,
    const RenderingEngine::Color & color )
```

Initializes the properties of the sphere.

#### Parameters

in	<i>radius</i>	The radius of the sphere.
in	<i>position</i>	The position of the sphere.
in	<i>orientation</i>	The orientation of the sphere.
in	<i>color</i>	The color of the sphere.

#### 5.5.3.5 SetRadius()

```
void ShapesEngine::Sphere::SetRadius (
    float radius )
```

Sets the radius of the sphere.

#### 5.5.3.6 UpdateModelMatrix()

```
void ShapesEngine::Sphere::UpdateModelMatrix ( )
```

Updates the spheres model matrix.

#### 5.5.3.7 Volume()

```
float ShapesEngine::Sphere::Volume ( )
```

Returns the volume of the sphere.

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

- Sphere.h

## 5.6 ShapesEngine::ThreeDimensionalShape Struct Reference

The struct stores the properties needed to render a 3D shape.

```
#include "ThreeDimensionalShape.h"
```

### Public Attributes

- vec3 **position**
- MathEngine::Quaternion **orientation**
- RenderingEngine::Color **color**
- mat4 **modelMatrix**
- RenderingEngine::DrawArguments **drawArguments**



### 5.6.1 Detailed Description

The struct stores the properties needed to render a 3D shape.

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

- ThreeDimensionalShape.h

## 5.7 ShapesEngine::Triangle Struct Reference

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

```
#include "Triangle.h"
```

### Public Attributes

- [Vertex](#) \* **vertexList**
- unsigned int **p0**
- unsigned int **p1**
- unsigned int **p2**

### 5.7.1 Detailed Description

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

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

- Triangle.h

## 5.8 ShapesEngine::Vertex Struct Reference

Data that describes a vertex.

```
#include "Vertex.h"
```

### Public Attributes

- vec3 **position**
- vec3 **normal**
- vec2 **texCoords**

### 5.8.1 Detailed Description

Data that describes a vertex.

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

- Vertex.h



## Chapter 6

# File Documentation

### 6.1 Box.h

```
1 #pragma once
2
3
4 #include "ThreeDimensionalShape.h"
5
6 namespace ShapesEngine
7 {
8     class Box
9     {
10     public:
11         Box();
12
13         void InitializeBox(float width, float height, float depth, const vec3 position, const
MathEngine::Quaternion orientation,
14             const RenderingEngine::Color& color);
15
16         const ThreeDimensionalShape& GetShape() const;
17
18         ThreeDimensionalShape& GetShape();
19
20         float GetWidth() const;
21
22         float GetHeight() const;
23
24         float GetDepth() const;
25
26         void SetWidth(float width);
27
28         void SetHeight(float height);
29
30         void SetDepth(float depth);
31
32         void UpdateModelMatrix();
33
34         float Volume();
35
36     private:
37         ThreeDimensionalShape mShape;
38
39         float mWidth;
40         float mHeight;
41         float mDepth;
42     };
43 }
```

### 6.2 Cone.h

```
1 #pragma once
2
3
4 #include "ThreeDimensionalShape.h"
5
6 namespace ShapesEngine
```

```

7 {
11     class Cone
12     {
13     public:
14
18         Cone();
19
28         void InitializeCone(float radius, float height, const vec3 position, const MathEngine::Quaternion
orientation,
29             const RenderingEngine::Color& color);
30
33         const ThreeDimensionalShape& GetShape() const;
34
37         ThreeDimensionalShape& GetShape();
38
41         float GetRadius() const;
42
45         float GetHeight() const;
46
49         void SetRadius(float radius);
50
53         void SetHeight(float height);
54
57         void UpdateModelMatrix();
58
61         float Volume();
62
63     private:
64         ThreeDimensionalShape mShape;
65
66         float mRadius;
67         float mHeight;
68     };
69 }

```

## 6.3 CreateShapes.h

```

1 #pragma once
2
3 #include "Triangle.h"
4 #include <vector>
5
6 namespace ShapesEngine
7 {
12     void CreateBox(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles);
13
19     void CreateCone(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
20         unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
21
27     void CreateCylinder(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
28         unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
29
35     void CreateSphere(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
36         unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
37
42     void CreatePyramid(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles);
43 }

```

## 6.4 Cylinder.h

```

1 #pragma once
2
3
4 #include "ThreeDimensionalShape.h"
5
6 namespace ShapesEngine
7 {
11     class Cylinder
12     {
13     public:
14
18         Cylinder();
19
28         void InitializeCylinder(float radius, float height, const vec3 position, const
MathEngine::Quaternion orientation,
29             const RenderingEngine::Color& color);
30
33         const ThreeDimensionalShape& GetShape() const;
34

```

```

37     ThreeDimensionalShape& GetShape();
38
41     float GetRadius() const;
42
45     float GetHeight() const;
46
49     void SetRadius(float radius);
50
53     void SetHeight(float height);
54
57     void UpdateModelMatrix();
58
61     float Volume();
62
63 private:
64     ThreeDimensionalShape mShape;
65
66     float mRadius;
67     float mHeight;
68 };
69 }

```

## 6.5 Pyramid.h

```

1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4
5 namespace ShapesEngine
6 {
10     class Pyramid
11     {
12     public:
13
17         Pyramid();
18
28         void InitializePyramid(float width, float height, float depth, const vec3 position, const
MathEngine::Quaternion orientation,
29             const RenderingEngine::Color& color);
30
33         const ThreeDimensionalShape& GetShape() const;
34
37         ThreeDimensionalShape& GetShape();
38
41         float GetWidth() const;
42
45         float GetHeight() const;
46
49         float GetDepth() const;
50
53         void SetWidth(float width);
54
57         void SetHeight(float height);
58
61         void SetDepth(float depth);
62
65         void UpdateModelMatrix();
66
69         float Volume();
70
71     private:
72         ThreeDimensionalShape mShape;
73
74         float mWidth;
75         float mHeight;
76         float mDepth;
77     };
78 }

```

## 6.6 Sphere.h

```

1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4
5 namespace ShapesEngine
6 {
10     class Sphere
11     {

```

```

12     public:
13
14         Sphere();
15
16         void InitializeSphere(float radius, const vec3 position, const MathEngine::Quaternion
orientation,
27             const RenderingEngine::Color& color);
28
29         const ThreeDimensionalShape& GetShape() const;
30
31         ThreeDimensionalShape& GetShape();
32
33         float GetRadius() const;
34
35         void SetRadius(float radius);
36
37         void UpdateModelMatrix();
38
39         float Volume();
40
41     private:
42         ThreeDimensionalShape mShape;
43
44         float mRadius;
45     };
46 }

```

## 6.7 ThreeDimensionalShape.h

```

1 #pragma once
2
3
4 #include "Color.h"
5 #include "DrawArguments.h"
6 #include "RenderScene.h"
7
8 namespace ShapesEngine
9 {
10     struct ThreeDimensionalShape
11     {
12         vec3 position;
13
14         MathEngine::Quaternion orientation;
15
16         RenderingEngine::Color color;
17
18         mat4 modelMatrix;
19
20         RenderingEngine::DrawArguments drawArguments;
21     };
22
23     void SetDrawArguments(ThreeDimensionalShape& shape, unsigned int indexCount, unsigned int
locationFirstIndex, int indexFirstVertex,
32         unsigned int indexConstantData, const std::wstring& constantBufferKey, unsigned int
rootParameterIndex, D3D_PRIMITIVE_TOPOLOGY primitive);
33
34     void UpdateShape(const ThreeDimensionalShape& shape, RenderingEngine::RenderScene* scene, const void*
data, unsigned int size);
35
36     void RenderShape(const ThreeDimensionalShape& shape, RenderingEngine::RenderScene* scene);
37
38 }

```

## 6.8 Triangle.h

```

1 #pragma once
2
3 #include <vector>
4 #include "Vertex.h"
5
6 namespace ShapesEngine
7 {
8     struct Triangle
9     {
10         Vertex* vertexList; //pointer to a vertex list
11         unsigned int p0;
12         unsigned int p1;
13         unsigned int p2;
14     };
15 }

```

```
18
21     vec3 ComputeNormal(const Triangle& triangle);
22
25     vec3 ComputeCenter(const Triangle& triangle);
26
29     void Quad(unsigned int a, unsigned int b, unsigned int c, unsigned int d, std::vector<Triangle>&
triangles, Vertex* vertices);
30 }
```

## 6.9 Vertex.h

```
1 #pragma once
2
3 #include "MathEngine.h"
4
5 namespace ShapesEngine
6 {
10     struct Vertex
11     {
12         vec3 position;
13         vec3 normal;
14         vec2 texCoords;
15     };
16 }
```





# Index

- Box
  - ShapesEngine::Box, [12](#)
- ComputeCenter
  - ShapesEngine, [8](#)
- ComputeNormal
  - ShapesEngine, [8](#)
- Cone
  - ShapesEngine::Cone, [15](#)
- CreateBox
  - ShapesEngine, [8](#)
- CreateCone
  - ShapesEngine, [9](#)
- CreateCylinder
  - ShapesEngine, [9](#)
- CreatePyramid
  - ShapesEngine, [9](#)
- CreateSphere
  - ShapesEngine, [9](#)
- Cylinder
  - ShapesEngine::Cylinder, [18](#)
- GetDepth
  - ShapesEngine::Box, [12](#)
  - ShapesEngine::Pyramid, [21](#)
- GetHeight
  - ShapesEngine::Box, [12](#)
  - ShapesEngine::Cone, [15](#)
  - ShapesEngine::Cylinder, [18](#)
  - ShapesEngine::Pyramid, [22](#)
- GetRadius
  - ShapesEngine::Cone, [15](#)
  - ShapesEngine::Cylinder, [18](#)
  - ShapesEngine::Sphere, [25](#)
- GetShape
  - ShapesEngine::Box, [12](#)
  - ShapesEngine::Cone, [16](#)
  - ShapesEngine::Cylinder, [19](#)
  - ShapesEngine::Pyramid, [22](#)
  - ShapesEngine::Sphere, [25](#)
- GetWidth
  - ShapesEngine::Box, [13](#)
  - ShapesEngine::Pyramid, [22](#)
- InitializeBox
  - ShapesEngine::Box, [13](#)
- InitializeCone
  - ShapesEngine::Cone, [16](#)
- InitializeCylinder
  - ShapesEngine::Cylinder, [19](#)
- InitializePyramid
  - ShapesEngine::Pyramid, [22](#)
- InitializeSphere
  - ShapesEngine::Sphere, [25](#)
- Pyramid
  - ShapesEngine::Pyramid, [21](#)
- Quad
  - ShapesEngine, [9](#)
- RenderShape
  - ShapesEngine, [10](#)
- SetDepth
  - ShapesEngine::Box, [13](#)
  - ShapesEngine::Pyramid, [23](#)
- SetDrawArguments
  - ShapesEngine, [10](#)
- SetHeight
  - ShapesEngine::Box, [13](#)
  - ShapesEngine::Cone, [16](#)
  - ShapesEngine::Cylinder, [19](#)
  - ShapesEngine::Pyramid, [23](#)
- SetRadius
  - ShapesEngine::Cone, [17](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Sphere, [26](#)
- SetWidth
  - ShapesEngine::Box, [14](#)
  - ShapesEngine::Pyramid, [23](#)
- ShapesEngine, [7](#)
  - ComputeCenter, [8](#)
  - ComputeNormal, [8](#)
  - CreateBox, [8](#)
  - CreateCone, [9](#)
  - CreateCylinder, [9](#)
  - CreatePyramid, [9](#)
  - CreateSphere, [9](#)
  - Quad, [9](#)
  - RenderShape, [10](#)
  - SetDrawArguments, [10](#)
  - UpdateShape, [10](#)
- ShapesEngine::Box, [11](#)
  - Box, [12](#)
  - GetDepth, [12](#)
  - GetHeight, [12](#)
  - GetShape, [12](#)
  - GetWidth, [13](#)
  - InitializeBox, [13](#)

- SetDepth, [13](#)
- SetHeight, [13](#)
- SetWidth, [14](#)
- UpdateModelMatrix, [14](#)
- Volume, [14](#)
- ShapesEngine::Cone, [14](#)
  - Cone, [15](#)
  - GetHeight, [15](#)
  - GetRadius, [15](#)
  - GetShape, [16](#)
  - InitializeCone, [16](#)
  - SetHeight, [16](#)
  - SetRadius, [17](#)
  - UpdateModelMatrix, [17](#)
  - Volume, [17](#)
- ShapesEngine::Cylinder, [17](#)
  - Cylinder, [18](#)
  - GetHeight, [18](#)
  - GetRadius, [18](#)
  - GetShape, [19](#)
  - InitializeCylinder, [19](#)
  - SetHeight, [19](#)
  - SetRadius, [20](#)
  - UpdateModelMatrix, [20](#)
  - Volume, [20](#)
- ShapesEngine::Pyramid, [20](#)
  - GetDepth, [21](#)
  - GetHeight, [22](#)
  - GetShape, [22](#)
  - GetWidth, [22](#)
  - InitializePyramid, [22](#)
  - Pyramid, [21](#)
  - SetDepth, [23](#)
  - SetHeight, [23](#)
  - SetWidth, [23](#)
  - UpdateModelMatrix, [23](#)
  - Volume, [23](#)
- ShapesEngine::Sphere, [24](#)
  - GetRadius, [25](#)
  - GetShape, [25](#)
  - InitializeSphere, [25](#)
  - SetRadius, [26](#)
  - Sphere, [24](#)
  - UpdateModelMatrix, [26](#)
  - Volume, [26](#)
- ShapesEngine::ThreeDimensionalShape, [26](#)
- ShapesEngine::Triangle, [27](#)
- ShapesEngine::Vertex, [27](#)
- Sphere
  - ShapesEngine::Sphere, [24](#)
- UpdateModelMatrix
  - ShapesEngine::Box, [14](#)
  - ShapesEngine::Cone, [17](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Pyramid, [23](#)
  - ShapesEngine::Sphere, [26](#)
- UpdateShape
  - ShapesEngine, [10](#)
- Volume
  - ShapesEngine::Box, [14](#)
  - ShapesEngine::Cone, [17](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Pyramid, [23](#)
  - ShapesEngine::Sphere, [26](#)