

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 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 ShapesEngine Namespace Reference	9
5.1.1 Detailed Description	10
5.1.2 Function Documentation	10
5.1.2.1 ComputeCenter()	10
5.1.2.2 ComputeNormal()	10
5.1.2.3 CreateBox()	10
5.1.2.4 CreateCone()	11
5.1.2.5 CreateCylinder()	11
5.1.2.6 CreatePyramid()	11
5.1.2.7 CreateSphere()	11
5.1.2.8 Quad()	11
<b>6 Class Documentation</b>	<b>13</b>
6.1 ShapesEngine::Box Class Reference	13
6.1.1 Detailed Description	14
6.1.2 Constructor & Destructor Documentation	14
6.1.2.1 Box()	14
6.1.3 Member Function Documentation	14
6.1.3.1 GetDimensions()	14
6.1.3.2 InitializeBox()	15
6.1.3.3 SetDimensions()	15
6.1.3.4 UpdateModelMatrix()	15
6.1.3.5 Volume()	15
6.2 ShapesEngine::Cone Class Reference	16
6.2.1 Detailed Description	16
6.2.2 Constructor & Destructor Documentation	16
6.2.2.1 Cone()	16
6.2.3 Member Function Documentation	17
6.2.3.1 GetDimensions()	17
6.2.3.2 InitializeCone()	17
6.2.3.3 SetDimensions()	17

6.2.3.4 UpdateModelMatrix()	18
6.2.3.5 Volume()	18
6.3 ShapesEngine::Cylinder Class Reference	18
6.3.1 Detailed Description	19
6.3.2 Constructor & Destructor Documentation	19
6.3.2.1 Cylinder()	19
6.3.3 Member Function Documentation	20
6.3.3.1 GetDimensions()	20
6.3.3.2 InitializeCylinder()	20
6.3.3.3 SetDimensions()	20
6.3.3.4 UpdateModelMatrix()	21
6.3.3.5 Volume()	21
6.4 ShapesEngine::Pyramid Class Reference	21
6.4.1 Detailed Description	22
6.4.2 Constructor & Destructor Documentation	22
6.4.2.1 Pyramid()	22
6.4.3 Member Function Documentation	22
6.4.3.1 GetDimensions()	22
6.4.3.2 InitializePyramid()	23
6.4.3.3 SetDimensions()	23
6.4.3.4 UpdateModelMatrix()	23
6.4.3.5 Volume()	23
6.5 ShapesEngine::Sphere Class Reference	24
6.5.1 Detailed Description	24
6.5.2 Constructor & Destructor Documentation	24
6.5.2.1 Sphere()	24
6.5.3 Member Function Documentation	25
6.5.3.1 GetDimensions()	25
6.5.3.2 InitializeSphere()	25
6.5.3.3 SetDimensions()	25
6.5.3.4 UpdateModelMatrix()	26
6.5.3.5 Volume()	26
6.6 ShapesEngine::ThreeDimensionalShapeAbstract Class Reference	26
6.6.1 Detailed Description	27
6.6.2 Member Function Documentation	27
6.6.2.1 GetColor()	27
6.6.2.2 GetDimensions()	27
6.6.2.3 GetDrawArguments()	28
6.6.2.4 GetModelMatrix()	28
6.6.2.5 GetOrientation()	28
6.6.2.6 GetPosition()	28
6.6.2.7 SetColor()	28

6.6.2.8 SetDimensions()	29
6.6.2.9 SetDrawArguments()	29
6.6.2.10 SetOrientation()	29
6.6.2.11 SetPosition()	29
6.6.2.12 UpdateModelMatrix()	29
6.6.2.13 Volume()	30
6.7 ShapesEngine::Triangle Struct Reference	30
6.7.1 Detailed Description	30
6.8 ShapesEngine::Vertex Struct Reference	30
6.8.1 Detailed Description	30
<b>7 File Documentation</b>	<b>31</b>
7.1 Box.h	31
7.2 Cone.h	31
7.3 CreateShapes.h	32
7.4 Cylinder.h	32
7.5 Pyramid.h	32
7.6 Sphere.h	33
7.7 ThreeDimensionalShape.h	33
7.8 Triangle.h	34
7.9 Vertex.h	34
<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 . . . . .	9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

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

ShapesEngine::ThreeDimensionalShapeAbstract . . . . .	26
ShapesEngine::Box . . . . .	13
ShapesEngine::Cone . . . . .	16
ShapesEngine::Cylinder . . . . .	18
ShapesEngine::Pyramid . . . . .	21
ShapesEngine::Sphere . . . . .	24
ShapesEngine::Triangle . . . . .	30
ShapesEngine::Vertex . . . . .	30



## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ShapesEngine::Box</a>	
This class is used to render a box . . . . .	13
<a href="#">ShapesEngine::Cone</a>	
This class is used to render a cone . . . . .	16
<a href="#">ShapesEngine::Cylinder</a>	
This class is used to render a cylinder . . . . .	18
<a href="#">ShapesEngine::Pyramid</a>	
This class is used to render a pyramid . . . . .	21
<a href="#">ShapesEngine::Sphere</a>	
This class is used to render a sphere . . . . .	24
<a href="#">ShapesEngine::ThreeDimensionalShapeAbstract</a>	
An abstract class for 3D shapes . . . . .	26
<a href="#">ShapesEngine::Triangle</a>	
The struct stores a pointer to a vertex list and indices to the vertices of the triangle . . . . .	30
<a href="#">ShapesEngine::Vertex</a>	
Data that describes a vertex . . . . .	30



## Chapter 4

# File Index

### 4.1 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 5

# Namespace Documentation

### 5.1 ShapesEngine Namespace Reference

An engine for rendering 3D shapes.

#### Classes

- class [Box](#)  
*This class is used to render a box.*
- class [Cone](#)  
*This class is used to render a cone.*
- class [Cylinder](#)  
*This class is used to render a cylinder.*
- class [Pyramid](#)  
*This class is used to render a pyramid.*
- class [Sphere](#)  
*This class is used to render a sphere.*
- class [ThreeDimensionalShapeAbstract](#)  
*An abstract class for 3D shapes.*
- 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.*
- 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.*

### 5.1.1 Detailed Description

An engine for rendering 3D shapes.

### 5.1.2 Function Documentation

#### 5.1.2.1 `ComputeCenter()`

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

Returns the center of the triangle.

#### 5.1.2.2 `ComputeNormal()`

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

Returns the normal of the triangle.

#### 5.1.2.3 `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.



#### 5.1.2.4 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.

#### 5.1.2.5 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.

#### 5.1.2.6 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.

#### 5.1.2.7 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.

#### 5.1.2.8 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.



## Chapter 6

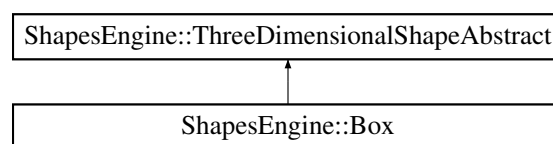
# Class Documentation

### 6.1 ShapesEngine::Box Class Reference

This class is used to render a box.

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

Inheritance diagram for ShapesEngine::Box:



#### Public Member Functions

- **Box** (float width, float height, float depth, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Creates a **Box** object.*
- 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.*
- vec3 **GetDimensions** () const override  
*Returns the dimensions of the box. The x component is the width, the y component is the height and the z component is the depth.*
- void **SetDimensions** (const vec3 &dimensions) override  
*Sets the dimensions of the box. The x component should be the width, the y component should be the height and the z component should be the depth.*
- void **UpdateModelMatrix** () override  
*Updates the boxes model matrix.*
- float **Volume** () const override  
*Returns the volume of the box.*

## Additional Inherited Members

### 6.1.1 Detailed Description

This class is used to render a box.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 Box()

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

Creates a [Box](#) object.

#### 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.

### 6.1.3 Member Function Documentation

#### 6.1.3.1 GetDimensions()

```
vec3 ShapesEngine::Box::GetDimensions ( ) const [override], [virtual]
```

Returns the dimensions of the box. The x component is the width, the y component is the height and the z component is the depth.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.1.3.2 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.

### 6.1.3.3 SetDimensions()

```
void ShapesEngine::Box::SetDimensions (
    const vec3 & dimensions ) [override], [virtual]
```

Sets the dimensions of the box. The x component should be the width, the y component should be the height and the z component should be the depth.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.1.3.4 UpdateModelMatrix()

```
void ShapesEngine::Box::UpdateModelMatrix ( ) [override], [virtual]
```

Updates the boxs model matrix.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.1.3.5 Volume()

```
float ShapesEngine::Box::Volume ( ) const [override], [virtual]
```

Returns the volume of the box.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

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

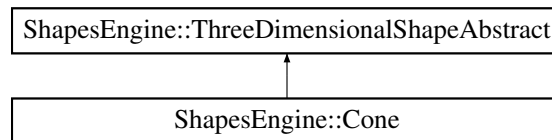
- Box.h

## 6.2 ShapesEngine::Cone Class Reference

This class is used to render a cone.

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

Inheritance diagram for ShapesEngine::Cone:



### Public Member Functions

- [Cone](#) (float radius, float height, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Creates a [Cone](#) object.*
- void [InitializeCone](#) (float radius, float height, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the cone.*
- vec3 [GetDimensions](#) () const override  
*Returns the dimensions of the cone. The x component is the radius, the y component is the height and the z component is the radius.*
- void [SetDimensions](#) (const vec3 &dimensions) override  
*Sets the dimensions of the cone. The x component should be the radius, the y component should be the height and the z component should be the radius.*
- void [UpdateModelMatrix](#) () override  
*Updates the cones model matrix.*
- float [Volume](#) () const override  
*Returns the volume of the cone.*

### Additional Inherited Members

#### 6.2.1 Detailed Description

This class is used to render a cone.

#### 6.2.2 Constructor & Destructor Documentation

##### 6.2.2.1 Cone()

```

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

Creates a [Cone](#) object.

## Parameters

in	<i>radius</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.

## 6.2.3 Member Function Documentation

### 6.2.3.1 GetDimensions()

```
vec3 ShapesEngine::Cone::GetDimensions ( ) const [override], [virtual]
```

Returns the dimensions of the cone. The x component is the radius, the y component is the height and the z component is the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.2.3.2 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.

### 6.2.3.3 SetDimensions()

```
void ShapesEngine::Cone::SetDimensions (
    const vec3 & dimensions ) [override], [virtual]
```

Sets the dimensions of the cone. The x component should be the radius, the y component should be the height and the z component should be the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

#### 6.2.3.4 UpdateModelMatrix()

```
void ShapesEngine::Cone::UpdateModelMatrix ( ) [override], [virtual]
```

Updates the cones model matrix.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

#### 6.2.3.5 Volume()

```
float ShapesEngine::Cone::Volume ( ) const [override], [virtual]
```

Returns the volume of the cone.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

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

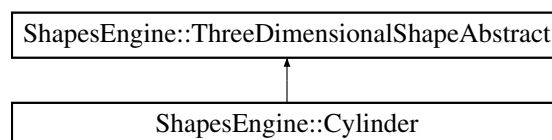
- Cone.h

## 6.3 ShapesEngine::Cylinder Class Reference

This class is used to render a cylinder.

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

Inheritance diagram for ShapesEngine::Cylinder:





## Public Member Functions

- [Cylinder](#) (float radius, float height, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Creates a [Cylinder](#) object.*
- void [InitializeCylinder](#) (float radius, float height, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Initializes the properties of the cylinder.*
- vec3 [GetDimensions](#) () const override  
*Returns the dimensions of the cylinder. The x component is the radius, the y component is the height and the z component is the radius.*
- void [SetDimensions](#) (const vec3 &dimensions) override  
*Sets the dimensions of the cylinder. The x component should be the radius, the y component should be the height and the z component should be the radius.*
- void [UpdateModelMatrix](#) () override  
*Updates the cylinders model matrix.*
- float [Volume](#) () const override  
*Returns the volume of the cylinder.*

## Additional Inherited Members

### 6.3.1 Detailed Description

This class is used to render a cylinder.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 Cylinder()

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

Creates a [Cylinder](#) object.

#### Parameters

in	<i>radius</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.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 GetDimensions()

```
vec3 ShapesEngine::Cylinder::GetDimensions ( ) const [override], [virtual]
```

Returns the dimensions of the cylinder. The x component is the radius, the y component is the height and the z component is the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

#### 6.3.3.2 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.

#### 6.3.3.3 SetDimensions()

```
void ShapesEngine::Cylinder::SetDimensions (
    const vec3 & dimensions ) [override], [virtual]
```

Sets the dimensions of the cylinder. The x component should be the radius, the y component should be the height and the z component should be the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.3.3.4 UpdateModelMatrix()

```
void ShapesEngine::Cylinder::UpdateModelMatrix ( ) [override], [virtual]
```

Updates the cylinders model matrix.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.3.3.5 Volume()

```
float ShapesEngine::Cylinder::Volume ( ) const [override], [virtual]
```

Returns the volume of the cylinder.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

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

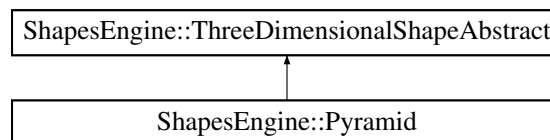
- Cylinder.h

## 6.4 ShapesEngine::Pyramid Class Reference

This class is used to render a pyramid.

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

Inheritance diagram for ShapesEngine::Pyramid:



### Public Member Functions

- [Pyramid](#) (float width, float height, float depth, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*Creates a [Pyramid](#) object.*
- 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.*
- vec3 [GetDimensions](#) () const override  
*Returns the dimensions of the pyramid. The x component is the width, the y component is the height and the z component is the depth.*
- void [SetDimensions](#) (const vec3 &dimensions) override  
*Sets the dimensions of the pyramid. The x component should be the width, the y component should be the height and the z component should be the depth.*
- void [UpdateModelMatrix](#) () override  
*Updates the pyramids model matrix.*
- float [Volume](#) () const override  
*Returns the volume of the pyramid.*

## Additional Inherited Members

### 6.4.1 Detailed Description

This class is used to render a pyramid.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 Pyramid()

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

Creates a [Pyramid](#) object.

#### 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.

### 6.4.3 Member Function Documentation

#### 6.4.3.1 GetDimensions()

```
vec3 ShapesEngine::Pyramid::GetDimensions ( ) const [override], [virtual]
```

Returns the dimensions of the pyramid. The x component is the width, the y component is the height and the z component is the depth.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.4.3.2 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.

### 6.4.3.3 SetDimensions()

```
void ShapesEngine::Pyramid::SetDimensions (
    const vec3 & dimensions ) [override], [virtual]
```

Sets the dimensions of the pyramid. The x component should be the width, the y component should be the height and the z component should be the depth.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.4.3.4 UpdateModelMatrix()

```
void ShapesEngine::Pyramid::UpdateModelMatrix ( ) [override], [virtual]
```

Updates the pyramids model matrix.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.4.3.5 Volume()

```
float ShapesEngine::Pyramid::Volume ( ) const [override], [virtual]
```

Returns the volume of the pyramid.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

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

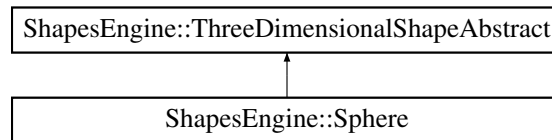
- Pyramid.h

## 6.5 ShapesEngine::Sphere Class Reference

This class is used to render a sphere.

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

Inheritance diagram for ShapesEngine::Sphere:



### Public Member Functions

- [Sphere](#) (float radius, const vec3 &position, const MathEngine::Quaternion &orientation, const RenderingEngine::Color &color)  
*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.*
- vec3 [GetDimensions](#) () const override  
*Returns the dimensions of the sphere. The x component is the radius, the y component is the radius and the z component is the radius.*
- void [SetDimensions](#) (const vec3 &dimensions) override  
*Sets the dimensions of the sphere. The x component should be the radius, the y component should be the radius and the z component should be the radius.*
- void [UpdateModelMatrix](#) () override  
*Updates the spheres model matrix.*
- float [Volume](#) () const override  
*Returns the volume of the sphere.*

### Additional Inherited Members

#### 6.5.1 Detailed Description

This class is used to render a sphere.

#### 6.5.2 Constructor & Destructor Documentation

##### 6.5.2.1 Sphere()

```

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

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

## 6.5.3 Member Function Documentation

### 6.5.3.1 GetDimensions()

```
vec3 ShapesEngine::Sphere::GetDimensions ( ) const [override], [virtual]
```

Returns the dimensions of the sphere. The x component is the radius, the y component is the radius and the z component is the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

### 6.5.3.2 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.

### 6.5.3.3 SetDimensions()

```
void ShapesEngine::Sphere::SetDimensions (
    const vec3 & dimensions ) [override], [virtual]
```

Sets the dimensions of the sphere. The x component should be the radius, the y component should be the radius and the z component should be the radius.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

#### 6.5.3.4 UpdateModelMatrix()

```
void ShapesEngine::Sphere::UpdateModelMatrix ( ) [override], [virtual]
```

Updates the spheres model matrix.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

#### 6.5.3.5 Volume()

```
float ShapesEngine::Sphere::Volume ( ) const [override], [virtual]
```

Returns the volume of the sphere.

Implements [ShapesEngine::ThreeDimensionalShapeAbstract](#).

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

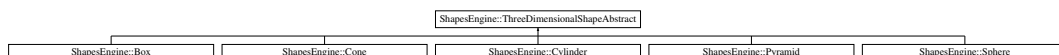
- Sphere.h

## 6.6 ShapesEngine::ThreeDimensionalShapeAbstract Class Reference

An abstract class for 3D shapes.

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

Inheritance diagram for ShapesEngine::ThreeDimensionalShapeAbstract:



### Public Member Functions

- virtual void [UpdateModelMatrix](#) ()=0  
*Updates a 3D shapes model matrix.*
- virtual float [Volume](#) () const =0  
*Returns a 3D shapes volume.*
- virtual vec3 [GetDimensions](#) () const =0  
*Returns the dimensions of a 3D shape.*
- virtual void [SetDimensions](#) (const vec3 &dimensions)=0  
*Sets the dimensions of a 3D shape.*
- virtual const RenderingEngine::Color & [GetColor](#) () const  
*Returns the color of a 3D shape.*
- virtual const RenderingEngine::DrawArguments & [GetDrawArguments](#) () const  
*Returns the draw arguments of a 3D shape.*
- virtual const mat4 & [GetModelMatrix](#) () const  
*Returns the model matrix of a 3D shape.*



- virtual const vec3 & [GetPosition](#) () const  
*Returns the position of a 3D shape.*
- virtual const MathEngine::Quaternion & [GetOrientation](#) () const  
*Returns the orientation of a 3D shape.*
- virtual void [SetPosition](#) (const vec3 &position)  
*Sets the position of a 3D shape.*
- virtual void [SetOrientation](#) (const MathEngine::Quaternion &orientation)  
*Sets the orientation of a 3D shape.*
- virtual void [SetColor](#) (const RenderingEngine::Color &color)  
*Sets the color of a 3D shape.*
- virtual void [SetDrawArguments](#) (const RenderingEngine::DrawArguments &drawArgs)  
*Sets the draw arguments of a 3D shape.*

## Protected Attributes

- RenderingEngine::RenderObject **mRenderObject**

### 6.6.1 Detailed Description

An abstract class for 3D shapes.

### 6.6.2 Member Function Documentation

#### 6.6.2.1 GetColor()

```
virtual const RenderingEngine::Color & ShapesEngine::ThreeDimensionalShapeAbstract::GetColor (
) const [virtual]
```

Returns the color of a 3D shape.

#### 6.6.2.2 GetDimensions()

```
virtual vec3 ShapesEngine::ThreeDimensionalShapeAbstract::GetDimensions ( ) const [pure virtual]
```

Returns the dimensions of a 3D shape.

Implemented in [ShapesEngine::Box](#), [ShapesEngine::Cone](#), [ShapesEngine::Cylinder](#), [ShapesEngine::Pyramid](#), and [ShapesEngine::Sphere](#).

### 6.6.2.3 GetDrawArguments()

```
virtual const RenderingEngine::DrawArguments & ShapesEngine::ThreeDimensionalShapeAbstract::GetDrawArguments ( ) const [virtual]
```

Returns the draw arguments of a 3D shape.

### 6.6.2.4 GetModelMatrix()

```
virtual const mat4 & ShapesEngine::ThreeDimensionalShapeAbstract::GetModelMatrix ( ) const [virtual]
```

Returns the model matrix of a 3D shape.

### 6.6.2.5 GetOrientation()

```
virtual const MathEngine::Quaternion & ShapesEngine::ThreeDimensionalShapeAbstract::GetOrientation ( ) const [virtual]
```

Returns the orientation of a 3D shape.

### 6.6.2.6 GetPosition()

```
virtual const vec3 & ShapesEngine::ThreeDimensionalShapeAbstract::GetPosition ( ) const [virtual]
```

Returns the position of a 3D shape.

### 6.6.2.7 SetColor()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::SetColor (
    const RenderingEngine::Color & color ) [virtual]
```

Sets the color of a 3D shape.

### 6.6.2.8 SetDimensions()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::SetDimensions (
    const vec3 & dimensions ) [pure virtual]
```

Sets the dimensions of a 3D shape.

Implemented in [ShapesEngine::Box](#), [ShapesEngine::Cone](#), [ShapesEngine::Cylinder](#), [ShapesEngine::Pyramid](#), and [ShapesEngine::Sphere](#).

### 6.6.2.9 SetDrawArguments()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::SetDrawArguments (
    const RenderingEngine::DrawArguments & drawArgs ) [virtual]
```

Sets the draw arguments of a 3D shape.

### 6.6.2.10 SetOrientation()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::SetOrientation (
    const MathEngine::Quaternion & orientation ) [virtual]
```

Sets the orientation of a 3D shape.

### 6.6.2.11 SetPosition()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::SetPosition (
    const vec3 & position ) [virtual]
```

Sets the position of a 3D shape.

### 6.6.2.12 UpdateModelMatrix()

```
virtual void ShapesEngine::ThreeDimensionalShapeAbstract::UpdateModelMatrix ( ) [pure virtual]
```

Updates a 3D shapes model matrix.

Implemented in [ShapesEngine::Box](#), [ShapesEngine::Cone](#), [ShapesEngine::Cylinder](#), [ShapesEngine::Pyramid](#), and [ShapesEngine::Sphere](#).

### 6.6.2.13 Volume()

```
virtual float ShapesEngine::ThreeDimensionalShapeAbstract::Volume ( ) const [pure virtual]
```

Returns a 3D shapes volume.

Implemented in [ShapesEngine::Box](#), [ShapesEngine::Cone](#), [ShapesEngine::Cylinder](#), [ShapesEngine::Pyramid](#), and [ShapesEngine::Sphere](#).

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

- ThreeDimensionalShape.h

## 6.7 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**

#### 6.7.1 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

## 6.8 ShapesEngine::Vertex Struct Reference

Data that describes a vertex.

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

### Public Attributes

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

#### 6.8.1 Detailed Description

Data that describes a vertex.

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

- Vertex.h

## Chapter 7

# File Documentation

### 7.1 Box.h

```
1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4 #include "RenderingEngineUtility.h"
5
6 namespace ShapesEngine
7 {
11     class Box : public ThreeDimensionalShapeAbstract
12     {
13     public:
14
23         Box(float width, float height, float depth, const vec3& position, const MathEngine::Quaternion&
orientation,
24             const RenderingEngine::Color& color);
25
35         void InitializeBox(float width, float height, float depth, const vec3& position, const
MathEngine::Quaternion& orientation,
36             const RenderingEngine::Color& color);
37
41         vec3 GetDimensions() const override;
42
46         void SetDimensions(const vec3& dimensions) override;
47
50         void UpdateModelMatrix() override;
51
54         float Volume() const override;
55
56     private:
57         float mWidth;
58         float mHeight;
59         float mDepth;
60     };
61 }
```

### 7.2 Cone.h

```
1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4 #include "RenderingEngineUtility.h"
5
6 namespace ShapesEngine
7 {
11     class Cone : public ThreeDimensionalShapeAbstract
12     {
13     public:
14
22         Cone(float radius, float height, const vec3& position, const MathEngine::Quaternion& orientation,
23             const RenderingEngine::Color& color);
24
33         void InitializeCone(float radius, float height, const vec3& position, const
MathEngine::Quaternion& orientation,
34             const RenderingEngine::Color& color);
35 }
```

```

39         vec3 GetDimensions() const override;
40
41         void SetDimensions(const vec3& dimensions) override;
42
43         void UpdateModelMatrix() override;
44
45         float Volume() const override;
46
47     private:
48         float mRadius;
49         float mHeight;
50     };
51 }

```

## 7.3 CreateShapes.h

```

1 #pragma once
2
3 #include "Triangle.h"
4 #include <vector>
5
6 namespace ShapesEngine
7 {
8     void CreateBox(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles);
9
10    void CreateCone(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
11        unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
12
13    void CreateCylinder(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
14        unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
15
16    void CreateSphere(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles,
17        unsigned int numVerticesPerCircle = 20, unsigned int numCircles = 20);
18
19    void CreatePyramid(std::vector<Vertex>& vertices, std::vector<Triangle>& triangles);
20 }

```

## 7.4 Cylinder.h

```

1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4 #include "RenderingEngineUtility.h"
5
6 namespace ShapesEngine
7 {
8     class Cylinder : public ThreeDimensionalShapeAbstract
9     {
10     public:
11         Cylinder(float radius, float height, const vec3& position, const MathEngine::Quaternion&
12             orientation,
13             const RenderingEngine::Color& color);
14
15         void InitializeCylinder(float radius, float height, const vec3& position, const
16             MathEngine::Quaternion& orientation,
17             const RenderingEngine::Color& color);
18
19         vec3 GetDimensions() const override;
20
21         void SetDimensions(const vec3& dimensions) override;
22
23         void UpdateModelMatrix() override;
24
25         float Volume() const override;
26
27     private:
28         float mRadius;
29         float mHeight;
30     };
31 }

```

## 7.5 Pyramid.h

```

1 #pragma once

```

```

2
3 #include "ThreeDimensionalShape.h"
4 #include "RenderingEngineUtility.h"
5
6 namespace ShapesEngine
7 {
11     class Pyramid : public ThreeDimensionalShapeAbstract
12     {
13     public:
14
23         Pyramid(float width, float height, float depth, const vec3& position, const
MathEngine::Quaternion& orientation,
24             const RenderingEngine::Color& color);
25
35         void InitializePyramid(float width, float height, float depth, const vec3& position, const
MathEngine::Quaternion& orientation,
36             const RenderingEngine::Color& color);
37
41         vec3 GetDimensions() const override;
42
46         void SetDimensions(const vec3& dimensions) override;
47
50         void UpdateModelMatrix() override;
51
54         float Volume() const override;
55
56     private:
57         float mWidth;
58         float mHeight;
59         float mDepth;
60     };
61 }

```

## 7.6 Sphere.h

```

1 #pragma once
2
3 #include "ThreeDimensionalShape.h"
4 #include "RenderingEngineUtility.h"
5
6 namespace ShapesEngine
7 {
11     class Sphere : public ThreeDimensionalShapeAbstract
12     {
13     public:
14
18         Sphere(float radius, const vec3& position, const MathEngine::Quaternion& orientation,
19             const RenderingEngine::Color& color);
20
28         void InitializeSphere(float radius, const vec3& position, const MathEngine::Quaternion&
orientation,
29             const RenderingEngine::Color& color);
30
34         vec3 GetDimensions() const override;
35
39         void SetDimensions(const vec3& dimensions) override;
40
43         void UpdateModelMatrix() override;
44
47         float Volume() const override;
48
49     private:
50         float mRadius;
51     };
52 }

```

## 7.7 ThreeDimensionalShape.h

```

1 #pragma once
2
3 #include "Color.h"
4 #include "DrawArguments.h"
5 #include "RenderingEngineUtility.h"
6 #include "Vertex.h"
7 #include <vector>
8
12 namespace ShapesEngine
13 {
17     class ThreeDimensionalShapeAbstract

```

```

18     {
19     public:
22         virtual void UpdateModelMatrix() = 0;
23
26         virtual float Volume() const = 0;
27
30         virtual vec3 GetDimensions() const = 0;
31
34         virtual void SetDimensions(const vec3& dimensions) = 0;
35
38         virtual const RenderingEngine::Color& GetColor() const;
39
42         virtual const RenderingEngine::DrawArguments& GetDrawArguments() const;
43
46         virtual const mat4& GetModelMatrix() const;
47
50         virtual const vec3& GetPosition() const;
51
54         virtual const MathEngine::Quaternion& GetOrientation() const;
55
58         virtual void SetPosition(const vec3& position);
59
62         virtual void SetOrientation(const MathEngine::Quaternion& orientation);
63
66         virtual void SetColor(const RenderingEngine::Color& color);
67
70         virtual void SetDrawArguments(const RenderingEngine::DrawArguments& drawArgs);
71
72     protected:
73         RenderingEngine::RenderObject mRenderObject;
74     };
75
76 }

```

## 7.8 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
16     vec3 ComputeNormal(const Triangle& triangle);
17
18     vec3 ComputeCenter(const Triangle& triangle);
19
20     void Quad(unsigned int a, unsigned int b, unsigned int c, unsigned int d, std::vector<Triangle>&
21             triangles, Vertex* vertices);
22 }

```

## 7.9 Vertex.h

```

1 #pragma once
2
3 #include "MathEngine.h"
4
5 namespace ShapesEngine
6 {
7     struct Vertex
8     {
9         vec3 position;
10        vec3 normal;
11        vec2 texCoords;
12    };
13 }

```



# Index

- Box
  - ShapesEngine::Box, [14](#)
- ComputeCenter
  - ShapesEngine, [10](#)
- ComputeNormal
  - ShapesEngine, [10](#)
- Cone
  - ShapesEngine::Cone, [16](#)
- CreateBox
  - ShapesEngine, [10](#)
- CreateCone
  - ShapesEngine, [10](#)
- CreateCylinder
  - ShapesEngine, [11](#)
- CreatePyramid
  - ShapesEngine, [11](#)
- CreateSphere
  - ShapesEngine, [11](#)
- Cylinder
  - ShapesEngine::Cylinder, [19](#)
- GetColor
  - ShapesEngine::ThreeDimensionalShapeAbstract, [27](#)
- GetDimensions
  - ShapesEngine::Box, [14](#)
  - ShapesEngine::Cone, [17](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Pyramid, [22](#)
  - ShapesEngine::Sphere, [25](#)
  - ShapesEngine::ThreeDimensionalShapeAbstract, [27](#)
- GetDrawArguments
  - ShapesEngine::ThreeDimensionalShapeAbstract, [27](#)
- GetModelMatrix
  - ShapesEngine::ThreeDimensionalShapeAbstract, [28](#)
- GetOrientation
  - ShapesEngine::ThreeDimensionalShapeAbstract, [28](#)
- GetPosition
  - ShapesEngine::ThreeDimensionalShapeAbstract, [28](#)
- InitializeBox
  - ShapesEngine::Box, [14](#)
- InitializeCone
  - ShapesEngine::Cone, [17](#)
- InitializeCylinder
  - ShapesEngine::Cylinder, [20](#)
- InitializePyramid
  - ShapesEngine::Pyramid, [22](#)
- InitializeSphere
  - ShapesEngine::Sphere, [25](#)
- Pyramid
  - ShapesEngine::Pyramid, [22](#)
- Quad
  - ShapesEngine, [11](#)
- SetColor
  - ShapesEngine::ThreeDimensionalShapeAbstract, [28](#)
- SetDimensions
  - ShapesEngine::Box, [15](#)
  - ShapesEngine::Cone, [17](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Pyramid, [23](#)
  - ShapesEngine::Sphere, [25](#)
  - ShapesEngine::ThreeDimensionalShapeAbstract, [28](#)
- SetDrawArguments
  - ShapesEngine::ThreeDimensionalShapeAbstract, [29](#)
- SetOrientation
  - ShapesEngine::ThreeDimensionalShapeAbstract, [29](#)
- SetPosition
  - ShapesEngine::ThreeDimensionalShapeAbstract, [29](#)
- ShapesEngine, [9](#)
  - ComputeCenter, [10](#)
  - ComputeNormal, [10](#)
  - CreateBox, [10](#)
  - CreateCone, [10](#)
  - CreateCylinder, [11](#)
  - CreatePyramid, [11](#)
  - CreateSphere, [11](#)
  - Quad, [11](#)
- ShapesEngine::Box, [13](#)
  - Box, [14](#)
  - GetDimensions, [14](#)
  - InitializeBox, [14](#)
  - SetDimensions, [15](#)
  - UpdateModelMatrix, [15](#)
  - Volume, [15](#)
- ShapesEngine::Cone, [16](#)

- Cone, [16](#)
- GetDimensions, [17](#)
- InitializeCone, [17](#)
- SetDimensions, [17](#)
- UpdateModelMatrix, [18](#)
- Volume, [18](#)
- ShapesEngine::Cylinder, [18](#)
  - Cylinder, [19](#)
  - GetDimensions, [20](#)
  - InitializeCylinder, [20](#)
  - SetDimensions, [20](#)
  - UpdateModelMatrix, [20](#)
  - Volume, [21](#)
- ShapesEngine::Pyramid, [21](#)
  - GetDimensions, [22](#)
  - InitializePyramid, [22](#)
  - Pyramid, [22](#)
  - SetDimensions, [23](#)
  - UpdateModelMatrix, [23](#)
  - Volume, [23](#)
- ShapesEngine::Sphere, [24](#)
  - GetDimensions, [25](#)
  - InitializeSphere, [25](#)
  - SetDimensions, [25](#)
  - Sphere, [24](#)
  - UpdateModelMatrix, [25](#)
  - Volume, [26](#)
- ShapesEngine::ThreeDimensionalShapeAbstract, [26](#)
  - GetColor, [27](#)
  - GetDimensions, [27](#)
  - GetDrawArguments, [27](#)
  - GetModelMatrix, [28](#)
  - GetOrientation, [28](#)
  - GetPosition, [28](#)
  - SetColor, [28](#)
  - SetDimensions, [28](#)
  - SetDrawArguments, [29](#)
  - SetOrientation, [29](#)
  - SetPosition, [29](#)
  - UpdateModelMatrix, [29](#)
  - Volume, [29](#)
- ShapesEngine::Triangle, [30](#)
- ShapesEngine::Vertex, [30](#)
- Sphere
  - ShapesEngine::Sphere, [24](#)
- UpdateModelMatrix
  - ShapesEngine::Box, [15](#)
  - ShapesEngine::Cone, [18](#)
  - ShapesEngine::Cylinder, [20](#)
  - ShapesEngine::Pyramid, [23](#)
  - ShapesEngine::Sphere, [25](#)
  - ShapesEngine::ThreeDimensionalShapeAbstract, [29](#)
- Volume
  - ShapesEngine::Box, [15](#)
  - ShapesEngine::Cone, [18](#)
  - ShapesEngine::Cylinder, [21](#)
- ShapesEngine::Pyramid, [23](#)
- ShapesEngine::Sphere, [26](#)
- ShapesEngine::ThreeDimensionalShapeAbstract, [29](#)