Farouq Adepetu's Shapes Engine

Generated by Doxygen 1.9.4

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Namespace Documentation

5.1 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 num↩ VerticesPerCircle=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

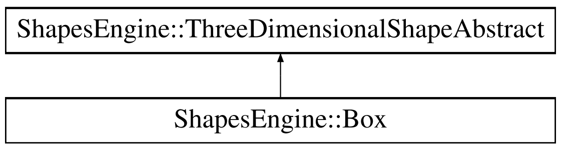
Class Documentation

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 boxs 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	*width*	The width of the box.
in	*height*	The height of the box.
in	*depth*	The depth of the box.
in	*position*	The position of the box.
in	*orientation*	The orientation of the box.
in	*color*	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	*width*	The width of the box.
in	*height*	The height of the box.
in	*depth*	The depth of the box.
in	*position*	The position of the box.
in	*orientation*	The orientation of the box.
in	*color*	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:

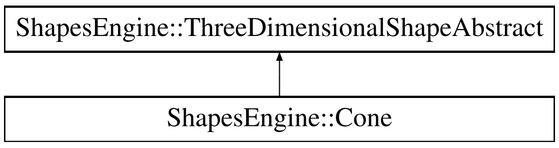
- Box.h

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	*radius*	The radius of the cone.
in	*height*	The height of the cone.
in	*position*	The position of the cone.
in	*orientation*	The orientation of the cone.
in	*color*	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	*width*	The radius of the cone.
in	*height*	The height of the cone.
in	*position*	The position of the cone.
in	*orientation*	The orientation of the cone.
in	*color*	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:

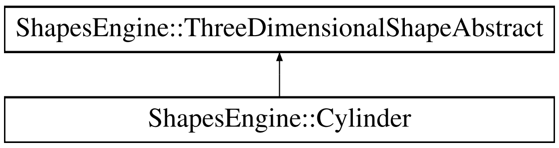
- Cone.h

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	*radius*	The radius of the cylinder.
in	*height*	The height of the cylinder.
in	*position*	The position of the cylinder.
in	*orientation*	The orientation of the cylinder.
in	*color*	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	*width*	The radius of the cylinder.
in	*height*	The height of the cylinder.
in	*position*	The position of the cylinder.
in	*orientation*	The orientation of the cylinder.
in	*color*	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:

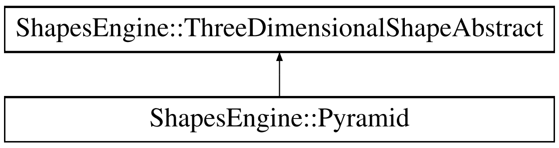
- Cylinder.h

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	*width*	The width of the pyramid.
in	*height*	The height of the pyramid.
in	*depth*	The depth of the pyramid.
in	*position*	The position of the pyramid.
in	*orientation*	The orientation of the pyramid.
in	*color*	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	*width*	The width of the pyramid.
in	*height*	The height of the pyramid.
in	*depth*	The depth of the pyramid.
in	*position*	The position of the pyramid.
in	*orientation*	The orientation of the pyramid.
in	*color*	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:

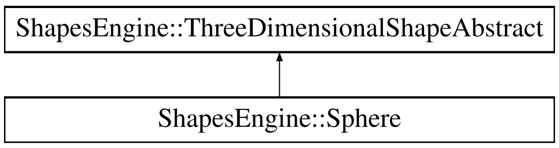
- Pyramid.h

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 Rendering↩ Engine::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	*radius*	The radius of the sphere.
in	*position*	The position of the sphere.
in	*orientation*	The orientation of the sphere.
in	*color*	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:

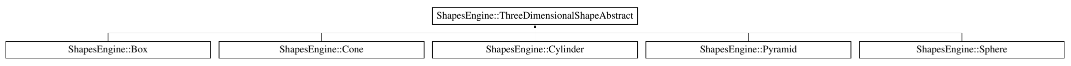
- Sphere.h

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::Get↩
Orientation ( ) 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

```
#pragma once

#include "ThreeDimensionalShape.h"
#include "RenderingEngineUtility.h"

namespace ShapesEngine
{
    class Box :  public ThreeDimensionalShapeAbstract
    {
    public:

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

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

        vec3 GetDimensions() const override;

        void SetDimensions(const vec3& dimensions) override;

        void UpdateModelMatrix() override;

        float Volume() const override;

    private:
        float mWidth;
        float mHeight;
        float mDepth;
    };
}
```

7.2 Cone.h

```
#pragma once

#include "ThreeDimensionalShape.h"
#include "RenderingEngineUtility.h"

namespace ShapesEngine
{
    class Cone :  public ThreeDimensionalShapeAbstract
    {
    public:

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

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

        vec3 GetDimensions() const override;

        void SetDimensions(const vec3& dimensions) override;

        void UpdateModelMatrix() override;

        float Volume() const override;

    private:
        float mRadius;
        float mHeight;
    };
}
```

7.3 CreateShapes.h

```
#pragma once

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

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

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

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

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

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

7.4 Cylinder.h

```
#pragma once

#include "ThreeDimensionalShape.h"
#include "RenderingEngineUtility.h"

namespace ShapesEngine
{
    class Cylinder :  public ThreeDimensionalShapeAbstract
    {
    public:

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

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

        vec3 GetDimensions() const override;

        void SetDimensions(const vec3& dimensions) override;

        void UpdateModelMatrix() override;

        float Volume() const override;

    private:
        float mRadius;
        float mHeight;
    };
}
```

7.5 Pyramid.h

```
#pragma once
```

```cpp

#include "ThreeDimensionalShape.h"
#include "RenderingEngineUtility.h"

namespace ShapesEngine
{
    class Pyramid :  public ThreeDimensionalShapeAbstract
    {
    public:

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

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

        vec3 GetDimensions() const override;

        void SetDimensions(const vec3& dimensions) override;

        void UpdateModelMatrix() override;

        float Volume() const override;

    private:
        float mWidth;
        float mHeight;
        float mDepth;
    };
}
```

7.6 Sphere.h

```cpp
#pragma once

#include "ThreeDimensionalShape.h"
#include "RenderingEngineUtility.h"

namespace ShapesEngine
{
    class Sphere :  public ThreeDimensionalShapeAbstract
    {
    public:

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

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

        vec3 GetDimensions() const override;

        void SetDimensions(const vec3& dimensions) override;

        void UpdateModelMatrix() override;

        float Volume() const override;

    private:
        float mRadius;
    };
}
```

7.7 ThreeDimensionalShape.h

```cpp
#pragma once

#include "Color.h"
#include "DrawArguments.h"
#include "RenderingEngineUtility.h"
#include "Vertex.h"
#include <vector>

namespace ShapesEngine
{
    class ThreeDimensionalShapeAbstract
    {
    public:
        virtual void UpdateModelMatrix() = 0;

        virtual float Volume() const = 0;

        virtual vec3 GetDimensions() const = 0;

        virtual void SetDimensions(const vec3& dimensions) = 0;

        virtual const RenderingEngine::Color& GetColor() const;

        virtual const RenderingEngine::DrawArguments& GetDrawArguments() const;

        virtual const mat4& GetModelMatrix() const;

        virtual const vec3& GetPosition() const;

        virtual const MathEngine::Quaternion& GetOrientation() const;

        virtual void SetPosition(const vec3& position);

        virtual void SetOrientation(const MathEngine::Quaternion& orientation);

        virtual void SetColor(const RenderingEngine::Color& color);

        virtual void SetDrawArguments(const RenderingEngine::DrawArguments& drawArgs);

    protected:
        RenderingEngine::RenderObject mRenderObject;
    };

}
```

7.8 Triangle.h

```
#pragma once

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

namespace ShapesEngine
{
    struct Triangle
    {
        Vertex* vertexList; //pointer to a vertex list
        unsigned int p0;
        unsigned int p1;
        unsigned int p2;
    };

    vec3 ComputeNormal(const Triangle& triangle);

    vec3 ComputeCenter(const Triangle& triangle);

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

7.9 Vertex.h

```
#pragma once

#include "MathEngine.h"

namespace ShapesEngine
{
    struct Vertex
    {
        vec3 position;
        vec3 normal;
        vec2 texCoords;
    };
}
```

Index