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

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

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 boxs 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	*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.

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	*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.

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	*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.

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	*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.

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	*width*	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.

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 {
11     class Box
12     {
13     public:
14
18         Box();
19
29         void InitializeBox(float width, float height, float depth, const vec3 position, const
      MathEngine::Quaternion orientation,
30             const RenderingEngine::Color& color);
31
34         const ThreeDimensionalShape& GetShape() const;
35
38         ThreeDimensionalShape& GetShape();
39
42         float GetWidth() const;
43
46         float GetHeight() const;
47
50         float GetDepth() const;
51
54         void SetWidth(float width);
55
58         void SetHeight(float height);
59
62         void SetDepth(float depth);
63
66         void UpdateModelMatrix();
67
70         float Volume();
71
72     private:
73         ThreeDimensionalShape mShape;
74
75         float mWidth;
76         float mHeight;
77         float mDepth;
78     };
79 }
```

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
17         Sphere();
18
26         void InitializeSphere(float radius, const vec3 position, const MathEngine::Quaternion
       orientation,
27             const RenderingEngine::Color& color);
28
31         const ThreeDimensionalShape& GetShape() const;
32
35         ThreeDimensionalShape& GetShape();
36
39         float GetRadius() const;
40
43         void SetRadius(float radius);
44
47         void UpdateModelMatrix();
48
51         float Volume();
52
53     private:
54         ThreeDimensionalShape mShape;
55
56         float mRadius;
57     };
58 }
```

6.7 ThreeDimensionalShape.h

```
1 #pragma once
2
3
4 #include "Color.h"
5 #include "DrawArguments.h"
6 #include "RenderScene.h"
7
11 namespace ShapesEngine
12 {
16     struct ThreeDimensionalShape
17     {
18         vec3 position;
19
20         MathEngine::Quaternion orientation;
21
22         RenderingEngine::Color color;
23
24         mat4 modelMatrix;
25
26         RenderingEngine::DrawArguments drawArguments;
27     };
28
31     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
36     void UpdateShape(const ThreeDimensionalShape& shape, RenderingEngine::RenderScene* scene, const void*
       data, unsigned int size);
37
40     void RenderShape(const ThreeDimensionalShape& shape, RenderingEngine::RenderScene* scene);
41
42 }
```

6.8 Triangle.h

```
1 #pragma once
2
3 #include <vector>
4 #include "Vertex.h"
5
6 namespace ShapesEngine
7 {
11     struct Triangle
12     {
13         Vertex* vertexList; //pointer to a vertex list
14         unsigned int p0;
15         unsigned int p1;
16         unsigned int p2;
17     };
```

```
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