Farouq Adepetu's Physics Engine

Generated by Doxygen 1.9.4

1 Namespace Index 1
1.1 Namespace List . . . 1

2 Hierarchical Index 3
2.1 Class Hierarchy . . . 3

3 Class Index 5
3.1 Class List . . . 5

4 File Index 7
4.1 File List . . . 7

5 Namespace Documentation 9
5.1 PhysicsEngine Namespace Reference . . . 9
5.1.1 Detailed Description . . . 10
5.1.2 Function Documentation . . . 10
5.1.2.1 ApplyForce() . . . 10
5.1.2.2 ComputeAABB() . . . 10
5.1.2.3 ComputeMassProperties() [1/2] . . . 10
5.1.2.4 ComputeMassProperties() [2/2] . . . 10
5.1.2.5 ComputeSphere() . . . 11
5.1.2.6 DragForce() . . . 11
5.1.2.7 GravitationalForce() . . . 11
5.1.2.8 InitalizeSphere() . . . 11
5.1.2.9 InitializeAABB() . . . 11
5.1.2.10 Interpolate() [1/2] . . . 12
5.1.2.11 Interpolate() [2/2] . . . 12
5.1.2.12 SimulateRigidShape() . . . 12
5.1.2.13 SubExpressions() . . . 12
5.1.2.14 TestIntersection() [1/2] . . . 13
5.1.2.15 TestIntersection() [2/2] . . . 13
5.1.2.16 TransformAABB() . . . 13
5.1.2.17 TransformSphere() . . . 13

6 Class Documentation 15
6.1 PhysicsEngine::AABB Struct Reference . . . 15
6.1.1 Detailed Description . . . 15
6.2 PhysicsEngine::BoundingBox Class Reference . . . 15
6.2.1 Detailed Description . . . 16
6.2.2 Constructor & Destructor Documentation . . . 16
6.2.2.1 BoundingBox() [1/2] . . . 16
6.2.2.2 BoundingBox() [2/2] . . . 16
6.2.3 Member Function Documentation . . . 16
6.2.3.1 InitializeBoundingBox() . . . 16

6.2.3.2 TransformBoundingVolume() . . . 17
6.2.3.3 UpdateModelMatrix() . . . 17
6.3 PhysicsEngine::BoundingSphere Class Reference . . . 17
6.3.1 Detailed Description . . . 18
6.3.2 Constructor & Destructor Documentation . . . 18
6.3.2.1 BoundingSphere() [1/2] . . . 18
6.3.2.2 BoundingSphere() [2/2] . . . 18
6.3.3 Member Function Documentation . . . 18
6.3.3.1 InitializeBoundingSphere() . . . 18
6.3.3.2 TransformBoundingVolume() . . . 18
6.3.3.3 UpdateModelMatrix() . . . 19
6.4 PhysicsEngine::BoundingVolumeAbstract Class Reference . . . 19
6.4.1 Member Function Documentation . . . 20
6.4.1.1 GetColor() . . . 20
6.4.1.2 GetDrawArguments() . . . 20
6.4.1.3 GetModelMatrix() . . . 20
6.4.1.4 GetOrientation() . . . 20
6.4.1.5 GetPosition() . . . 20
6.4.1.6 SetColor() . . . 21
6.4.1.7 SetDrawArguments() . . . 21
6.4.1.8 SetOrientation() . . . 21
6.4.1.9 SetPosition() . . . 21
6.4.1.10 TransformBoundingVolume() . . . 21
6.4.1.11 UpdateModelMatrix() . . . 22
6.5 PhysicsEngine::RigidBody Class Reference . . . 22
6.5.1 Constructor & Destructor Documentation . . . 22
6.5.1.1 RigidBody() . . . 23
6.5.2 Member Function Documentation . . . 23
6.5.2.1 AddForce() . . . 23
6.5.2.2 AddTorque() . . . 23
6.5.2.3 GetAngularMomentum() . . . 23
6.5.2.4 GetAngularVelocity() . . . 23
6.5.2.5 GetBodyInertiaTensor() . . . 23
6.5.2.6 GetCenterOfMass() . . . 24
6.5.2.7 GetInverseBodyInertiaTensor() . . . 24
6.5.2.8 GetInverseMass() . . . 24
6.5.2.9 GetLinearMomentum() . . . 24
6.5.2.10 GetLinearVelocity() . . . 24
6.5.2.11 GetMass() . . . 24
6.5.2.12 GetNetForce() . . . 24
6.5.2.13 GetNetTorque() . . . 25
6.5.2.14 GetOrientation() . . . 25

6.5.2.15 InitializeRigidBody() . . . 25
6.5.2.16 Integrate() [1/2] . . . 25
6.5.2.17 Integrate() [2/2] . . . 25
6.5.2.18 ResetForce() . . . 26
6.5.2.19 ResetTorque() . . . 26
6.5.2.20 SetAngularMomentum() . . . 26
6.5.2.21 SetAngularVelocity() . . . 26
6.5.2.22 SetBodyInertiaTensor() . . . 26
6.5.2.23 SetCenterOfMass() . . . 26
6.5.2.24 SetLinearMomentum() . . . 26
6.5.2.25 SetLinearVelocity() . . . 27
6.5.2.26 SetMass() . . . 27
6.5.2.27 SetOrientation() . . . 27
6.6 PhysicsEngine::RigidShape Class Reference . . . 27
6.6.1 Constructor & Destructor Documentation . . . 28
6.6.1.1 RigidShape() [1/2] . . . 28
6.6.1.2 RigidShape() [2/2] . . . 29
6.6.2 Member Function Documentation . . . 29
6.6.2.1 GetAngularMomentum() . . . 29
6.6.2.2 GetAngularVelocity() . . . 29
6.6.2.3 GetBodyInertiaTensor() . . . 29
6.6.2.4 GetBoundingVolumeColor() . . . 29
6.6.2.5 GetBoundingVolumeDrawArguments() . . . 29
6.6.2.6 GetBoundingVolumeModelMatrix() . . . 30
6.6.2.7 GetCenterOfMass() . . . 30
6.6.2.8 GetColor() . . . 30
6.6.2.9 GetDimensions() . . . 30
6.6.2.10 GetDrawArguments() . . . 30
6.6.2.11 GetInverseBodyInertiaTensor() . . . 30
6.6.2.12 GetInverseMass() . . . 30
6.6.2.13 GetLinearMomentum() . . . 31
6.6.2.14 GetLinearVelocity() . . . 31
6.6.2.15 GetMass() . . . 31
6.6.2.16 GetModelMatrix() . . . 31
6.6.2.17 GetOrientation() . . . 31
6.6.2.18 GetPosition() . . . 31
6.6.2.19 InitializeRigidShape() . . . 31
6.6.2.20 Integrate() . . . 32
6.6.2.21 SetAngularMomentum() . . . 32
6.6.2.22 SetAngularVelocity() . . . 32
6.6.2.23 SetBodyInertiaTensor() . . . 32
6.6.2.24 SetBoundingVolumeColor() . . . 32

6.6.2.25 SetBoundingVolumeDrawArguments() . . . 32
6.6.2.26 SetCenterOfMass() . . . 33
6.6.2.27 SetColor() . . . 33
6.6.2.28 SetDimensions() . . . 33
6.6.2.29 SetDrawArguments() . . . 33
6.6.2.30 SetLinearMomentum() . . . 33
6.6.2.31 SetLinearVelocity() . . . 33
6.6.2.32 SetMass() . . . 34
6.6.2.33 SetOrientation() . . . 34
6.6.2.34 SetPosition() . . . 34
6.6.2.35 UpdateModelMatrix() . . . 34
6.6.2.36 Volume() . . . 34
6.7 PhysicsEngine::Sphere Struct Reference . . . 34

7 File Documentation **35**

7.1 BoundingBox.h . . . 35
7.2 BoundingSphere.h . . . 35
7.3 BoundingVolume.h . . . 36
7.4 ForceFunctions.h . . . 36
7.5 PolyhedralMassProperties.h . . . 37
7.6 RigidBody.h . . . 37
7.7 RigidShape.h . . . 38

Index **41**

Chapter 1

Namespace Index

1.1 Namespace List

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

PhysicsEngine
An engine for physics simulations 9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

PhysicsEngine::AABB 15
PhysicsEngine::BoundingVolumeAbstract 19
PhysicsEngine::BoundingBox 15
PhysicsEngine::BoundingSphere 17
PhysicsEngine::RigidBody 22
PhysicsEngine::RigidShape 27
PhysicsEngine::Sphere 34

Chapter 3

Class Index

3.1 Class List

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

PhysicsEngine::AABB 15
PhysicsEngine::BoundingBox
This class is used to bound an object using an axis-aligned bounding box and also for rendering it 15
PhysicsEngine::BoundingSphere
This class is used to bound an object using a sphere and also for rendering it 17
PhysicsEngine::BoundingVolumeAbstract 19
PhysicsEngine::RigidBody 22
PhysicsEngine::RigidShape 27
PhysicsEngine::Sphere 34

Chapter 4

File Index

4.1 File List

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

BoundingBox.h . . . . . ??
BoundingSphere.h . . . . . ??
BoundingVolume.h . . . . . ??
ForceFunctions.h . . . . . ??
PolyhedralMassProperties.h . . . . . ??
RigidBody.h . . . . . ??
RigidShape.h . . . . . ??

Chapter 5

Namespace Documentation

5.1 PhysicsEngine Namespace Reference

An engine for physics simulations.

Classes

- struct AABB
- class BoundingBox

  This class is used to bound an object using an axis-aligned bounding box and also for rendering it.
- class BoundingSphere

  This class is used to bound an object using a sphere and also for rendering it.
- class BoundingVolumeAbstract
- class RigidBody
- class RigidShape
- struct Sphere

Functions

- void InitializeAABB (AABB &a, vec3 min, vec3 max)
- void ComputeAABB (AABB &aabb, const std::vector< ShapesEngine::Vertex > &vertices)
- void TransformAABB (AABB &worldAABB, const AABB &localAABB, const mat4 &model)
- bool TestIntersection (const AABB &a, const AABB &b)
- void InitalizeSphere (Sphere &sphere, const vec3 &center, float radius)
- void ComputeSphere (Sphere &sphere, const std::vector< ShapesEngine::Vertex > &vertices)
- void TransformSphere (Sphere &worldSphere, const Sphere &localSphere, const mat4 &model)
- bool TestIntersection (const Sphere &a, const Sphere &b)
- vec3 GravitationalForce (float mass, float gravityAcceleration, const vec3 &direction)
- vec3 DragForce (float k1, float k2, const vec3 &velocity)
- vec3 ApplyForce (float magnitdue, const vec3 &direction)
- void SubExpressions (double w0, double w1, double w2, double &f1, double &f2, double &f3, double &g0, double &g1, double &g2)
- void ComputeMassProperties (const std::vector< ShapesEngine::Triangle > &triangles, double &mass, vec3 &cm, mat3 &bodyInertia)
- void ComputeMassProperties (const std::vector< ShapesEngine::Triangle > &triangles, double &mass, vec3 &cm, mat3 &bodyInertia, const mat3 &scale)
- void Interpolate (const RigidBody &r1, const RigidBody &r2, RigidBody &r3, float t)
- void SimulateRigidShape (RigidShape &previousRigidShape, RigidShape &currentRigidShape, const vec3 &netForce, const vec3 &netTorque, float simulationTime)

  Simulates the current RigidShape.
- void Interpolate (const RigidShape &r1, const RigidShape &r2, RigidShape &r3, float t)

5.1.1 Detailed Description

An engine for physics simulations.

5.1.2 Function Documentation

5.1.2.1 ApplyForce()

```
vec3 PhysicsEngine::ApplyForce (
            float magnitdue,
            const vec3 & direction )
```

brief Returns a force that is being applied to an object.

Formula used is F = magnitude ∗ direction

5.1.2.2 ComputeAABB()

```
void PhysicsEngine::ComputeAABB (
            AABB & aabb,
            const std::vector< ShapesEngine::Vertex > & vertices )
```

brief Computes the properties of a AABB from the vertices of an object.

5.1.2.3 ComputeMassProperties() [1/2]

```
void PhysicsEngine::ComputeMassProperties (
            const std::vector< ShapesEngine::Triangle > & triangles,
            double & mass,
            vec3 & cm,
            mat3 & bodyInertia )
```

brief Computes the mass, center of mass and inertia tensor reltaive to the center of mass in body coordinates for a soild polyhedron.

Uses the triangles that make up the solid polyhedron to compute the values.
Assumes the mass density is 1, so if it is not you need to multiple the mass and body intertia by the mass density to get the correct values.

5.1.2.4 ComputeMassProperties() [2/2]

```
void PhysicsEngine::ComputeMassProperties (
            const std::vector< ShapesEngine::Triangle > & triangles,
            double & mass,
            vec3 & cm,
            mat3 & bodyInertia,
            const mat3 & scale )
```

brief Computes the mass, center of mass and inertia tensor reltaive to the center of mass in body coordinates for a soild polyhedron.

Uses the triangles that make up the solid polyhedron to compute the values.
Assumes the mass density is 1, so if it is not you need to multiple the mass and body intertia by the mass density to get the correct values.

Use this function if the object passed in is not scaled to the desired size.

5.1.2.5 ComputeSphere()

```
void PhysicsEngine::ComputeSphere (
            Sphere & sphere,
            const std::vector< ShapesEngine::Vertex > & vertices )
```

brief Computes the properties of a sphere from the vertices of an object using Ritter's method.

5.1.2.6 DragForce()

```
vec3 PhysicsEngine::DragForce (
            float k1,
            float k2,
            const vec3 & velocity )
```

brief Returns the force due to drag.

Formula used is F = -v(k1|v| + k2|v|^2), where v is the velocity of the object and k1 and k2 are the drag coefficients.

5.1.2.7 GravitationalForce()

```
vec3 PhysicsEngine::GravitationalForce (
            float mass,
            float gravityAcceleration,
            const vec3 & direction )
```

brief Returns the force due to gravity based off the specified parameters.

Formula used its F = mgU, where m is the mass of the object, g is acceleration due to gravity and U is the gravity direction.

5.1.2.8 InitalizeSphere()

```
void PhysicsEngine::InitalizeSphere (
            Sphere & sphere,
            const vec3 & center,
            float radius )
```

brief Initializes the properties of a sphere.

5.1.2.9 InitializeAABB()

```
void PhysicsEngine::InitializeAABB (
            AABB & a,
            vec3 min,
            vec3 max )
```

brief Initializes the specified AABB with the specified min and max values;

5.1.2.10 Interpolate() [1/2]

```
void PhysicsEngine::Interpolate (
            const RigidBody & r1,
            const RigidBody & r2,
            RigidBody & r3,
            float t )
```

brief Interpolates the center of mass and orientation between r1 and r2 and stores the interpolated rigid body in r3.

5.1.2.11 Interpolate() [2/2]

```
void PhysicsEngine::Interpolate (
            const RigidShape & r1,
            const RigidShape & r2,
            RigidShape & r3,
            float t )
```

brief Interpolates the center of mass and orientation between r1 and r2 and stores the interpolated rigid shape in r3.

5.1.2.12 SimulateRigidShape()

```
void PhysicsEngine::SimulateRigidShape (
            RigidShape & previousRigidShape,
            RigidShape & currentRigidShape,
            const vec3 & netForce,
            const vec3 & netTorque,
            float simulationTime )
```

Simulates the current RigidShape.

5.1.2.13 SubExpressions()

```
void PhysicsEngine::SubExpressions (
            double w0,
            double w1,
            double w2,
            double & f1,
            double & f2,
            double & f3,
            double & g0,
            double & g1,
            double & g2 )
```

brief These are the expressions used in computing the mass properties.

5.1.2.14 TestIntersection() [1/2]

```
bool PhysicsEngine::TestIntersection (
            const AABB & a,
            const AABB & b )
```

brief Returns true if the specified AABBs are intersecting, false otherwise.

5.1.2.15 TestIntersection() [2/2]

```
bool PhysicsEngine::TestIntersection (
            const Sphere & a,
            const Sphere & b )
```

brief Returns true if the two spheres are intersecting, false otherwise.

5.1.2.16 TransformAABB()

```
void PhysicsEngine::TransformAABB (
            AABB & worldAABB,
            const AABB & localAABB,
            const mat4 & model )
```

brief Transforms the localAABB to world space by finding the extents and returning the resultant AABB. The transformation is done using vector-matrix multiplcation. The localAABB is treated as a row vector.

5.1.2.17 TransformSphere()

```
void PhysicsEngine::TransformSphere (
            Sphere & worldSphere,
            const Sphere & localSphere,
            const mat4 & model )
```

brief Transforms the sphere from local space to world space using a row-major transformation matrix.

Chapter 6

Class Documentation

6.1 PhysicsEngine::AABB Struct Reference

`#include <BoundingBox.h>`

Public Attributes

- vec3 **min**
- vec3 **max**

6.1.1 Detailed Description

brief Structure for an axis-aligned bounding box. Uses the min-max representation.

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

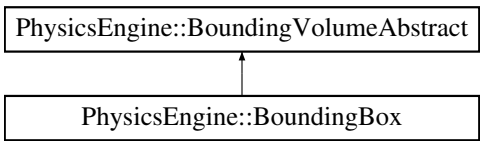
- BoundingBox.h

6.2 PhysicsEngine::BoundingBox Class Reference

This class is used to bound an object using an axis-aligned bounding box and also for rendering it.

`#include "BoundingBox.h"`

Inheritance diagram for PhysicsEngine::BoundingBox:



Public Member Functions

- BoundingBox ()
- BoundingBox (const std::vector< ShapesEngine::Vertex > &vertices, const RenderingEngine::Color &color)
- void InitializeBoundingBox (const std::vector< ShapesEngine::Vertex > &vertices, const RenderingEngine::Color &color)
- void UpdateModelMatrix () override

  Updates the BoundingBox model matrix.
- void TransformBoundingVolume (const mat4 &model) override

  Transforms the BoundingBox from local space to world space.

Additional Inherited Members

6.2.1 Detailed Description

This class is used to bound an object using an axis-aligned bounding box and also for rendering it.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 BoundingBox() [1/2]

```
PhysicsEngine::BoundingBox::BoundingBox ( )
```

brief Default constructor.

6.2.2.2 BoundingBox() [2/2]

```
PhysicsEngine::BoundingBox::BoundingBox (
            const std::vector< ShapesEngine::Vertex > & vertices,
            const RenderingEngine::Color & color )
```

brief Initializes the properties of the BoundingBox.

6.2.3 Member Function Documentation

6.2.3.1 InitializeBoundingBox()

```
void PhysicsEngine::BoundingBox::InitializeBoundingBox (
            const std::vector< ShapesEngine::Vertex > & vertices,
            const RenderingEngine::Color & color )
```

brief Initializes the properties of the BoundingBox.

6.2.3.2 TransformBoundingVolume()

```
void PhysicsEngine::BoundingBox::TransformBoundingVolume (
            const mat4 & model )  [override], [virtual]
```

Transforms the BoundingBox from local space to world space.

Implements PhysicsEngine::BoundingVolumeAbstract.

6.2.3.3 UpdateModelMatrix()

```
void PhysicsEngine::BoundingBox::UpdateModelMatrix ( )  [override], [virtual]
```

Updates the BoundingBox model matrix.

Implements PhysicsEngine::BoundingVolumeAbstract.

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

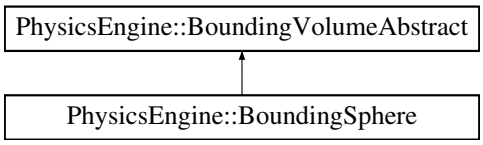
- BoundingBox.h

6.3 PhysicsEngine::BoundingSphere Class Reference

This class is used to bound an object using a sphere and also for rendering it.

```
#include "BoundingSphere.h"
```

Inheritance diagram for PhysicsEngine::BoundingSphere:



Public Member Functions

- BoundingSphere ()
- BoundingSphere (const std::vector< ShapesEngine::Vertex > &vertices, const RenderingEngine::Color &color)
- void InitializeBoundingSphere (const std::vector< ShapesEngine::Vertex > &vertices, const Rendering↩ Engine::Color &color)
- void UpdateModelMatrix () override

  Updates the BoundingSpheres model matrix.
- void TransformBoundingVolume (const mat4 &model) override

  Transforms the BoundingSphere from local space to world space.

Additional Inherited Members

6.3.1 Detailed Description

This class is used to bound an object using a sphere and also for rendering it.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 BoundingSphere() [1/2]

```
PhysicsEngine::BoundingSphere::BoundingSphere ( )
```

brief Default constructor.

6.3.2.2 BoundingSphere() [2/2]

```
PhysicsEngine::BoundingSphere::BoundingSphere (
            const std::vector< ShapesEngine::Vertex > & vertices,
            const RenderingEngine::Color & color )
```

brief Initializes the properties of the BoundingSphere.

6.3.3 Member Function Documentation

6.3.3.1 InitializeBoundingSphere()

```
void PhysicsEngine::BoundingSphere::InitializeBoundingSphere (
            const std::vector< ShapesEngine::Vertex > & vertices,
            const RenderingEngine::Color & color )
```

brief Initializes the properties of the BoundingSphere.

6.3.3.2 TransformBoundingVolume()

```
void PhysicsEngine::BoundingSphere::TransformBoundingVolume (
            const mat4 & model )  [override], [virtual]
```

Transforms the BoundingSphere from local space to world space.

Implements PhysicsEngine::BoundingVolumeAbstract.

6.3.3.3 UpdateModelMatrix()

```
void PhysicsEngine::BoundingSphere::UpdateModelMatrix ( )  [override], [virtual]
```

Updates the BoundingSpheres model matrix.

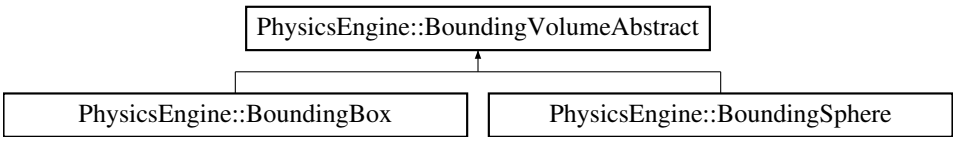
Implements PhysicsEngine::BoundingVolumeAbstract.

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

- BoundingSphere.h

6.4 PhysicsEngine::BoundingVolumeAbstract Class Reference

Inheritance diagram for PhysicsEngine::BoundingVolumeAbstract:



Public Member Functions

- virtual void UpdateModelMatrix ()=0

  Updates a bounding volumes model matrix.
- virtual void TransformBoundingVolume (const mat4 &model)=0

  Transforms the bounding volume from local space to world space.
- virtual const RenderingEngine::Color & GetColor () const

  Returns the color of a bounding volume.
- virtual const RenderingEngine::DrawArguments & GetDrawArguments () const

  Returns the draw arguments of a bounding volume.
- virtual const mat4 & GetModelMatrix () const

  Returns the model matrix of a bounding volume.
- virtual const vec3 & GetPosition () const

  Returns the position of a bounding volume.
- virtual const MathEngine::Quaternion & GetOrientation () const

  Returns the orientation of a bounding volume.
- virtual void SetPosition (const vec3 &position)

  Sets the position of a bounding volume.
- virtual void SetOrientation (const MathEngine::Quaternion &orientation)

  Sets the orientation of a bounding volume.
- virtual void SetColor (const RenderingEngine::Color &color)

  Sets the color of a bounding volume.
- virtual void SetDrawArguments (const RenderingEngine::DrawArguments &drawArgs)

  Sets the draw arguments of a bounding volume.

Protected Attributes

- RenderingEngine::RenderObject **mRenderObject**

6.4.1 Member Function Documentation

6.4.1.1 GetColor()

```
virtual const RenderingEngine::Color & PhysicsEngine::BoundingVolumeAbstract::GetColor ( )
const  [virtual]
```

Returns the color of a bounding volume.

6.4.1.2 GetDrawArguments()

```
virtual const RenderingEngine::DrawArguments & PhysicsEngine::BoundingVolumeAbstract::Get↩
DrawArguments ( ) const  [virtual]
```

Returns the draw arguments of a bounding volume.

6.4.1.3 GetModelMatrix()

```
virtual const mat4 & PhysicsEngine::BoundingVolumeAbstract::GetModelMatrix ( ) const  [virtual]
```

Returns the model matrix of a bounding volume.

6.4.1.4 GetOrientation()

```
virtual const MathEngine::Quaternion & PhysicsEngine::BoundingVolumeAbstract::GetOrientation (
) const  [virtual]
```

Returns the orientation of a bounding volume.

6.4.1.5 GetPosition()

```
virtual const vec3 & PhysicsEngine::BoundingVolumeAbstract::GetPosition ( ) const  [virtual]
```

Returns the position of a bounding volume.

6.4.1.6 SetColor()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::SetColor (
            const RenderingEngine::Color & color )  [virtual]
```

Sets the color of a bounding volume.

6.4.1.7 SetDrawArguments()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::SetDrawArguments (
            const RenderingEngine::DrawArguments & drawArgs )  [virtual]
```

Sets the draw arguments of a bounding volume.

6.4.1.8 SetOrientation()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::SetOrientation (
            const MathEngine::Quaternion & orientation )  [virtual]
```

Sets the orientation of a bounding volume.

6.4.1.9 SetPosition()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::SetPosition (
            const vec3 & position )  [virtual]
```

Sets the position of a bounding volume.

6.4.1.10 TransformBoundingVolume()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::TransformBoundingVolume (
            const mat4 & model )  [pure virtual]
```

Transforms the bounding volume from local space to world space.

Implemented in PhysicsEngine::BoundingBox, and PhysicsEngine::BoundingSphere.

6.4.1.11 UpdateModelMatrix()

```
virtual void PhysicsEngine::BoundingVolumeAbstract::UpdateModelMatrix ( )  [pure virtual]
```

Updates a bounding volumes model matrix.

Implemented in PhysicsEngine::BoundingBox, and PhysicsEngine::BoundingSphere.

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

- BoundingVolume.h

6.5 PhysicsEngine::RigidBody Class Reference

Public Member Functions

- RigidBody ()
- void InitializeRigidBody (float massDensity, const MathEngine::Quaternion &initialOrientation, const std↩ ::vector< ShapesEngine::Triangle > &triangles, const mat3 &scale)
- float GetMass () const
- float GetInverseMass () const
- const mat3 & GetBodyInertiaTensor () const
- const mat3 & GetInverseBodyInertiaTensor () const
- const vec3 & GetCenterOfMass () const
- const vec3 & GetLinearVelocity () const
- const vec3 & GetLinearMomentum () const
- const MathEngine::Quaternion & GetOrientation () const
- const vec3 & GetAngularVelocity () const
- const vec3 & GetAngularMomentum () const
- const vec3 & GetNetForce () const
- const vec3 & GetNetTorque () const
- void SetMass (float mass)
- void SetCenterOfMass (const vec3 &centerOfMass)
- void SetLinearVelocity (const vec3 &velocity)
- void SetLinearMomentum (const vec3 &linearMomentum)
- void SetBodyInertiaTensor (const mat3 &bodyInertia)
- void SetAngularVelocity (const vec3 &angularVelocity)
- void SetAngularMomentum (const vec3 &angularMomentum)
- void SetOrientation (const MathEngine::Quaternion &orientation)
- void ResetForce ()
- void ResetTorque ()
- void AddForce (const vec3 &force)
- void AddTorque (const vec3 &force, const vec3 &point)
- void Integrate (float dt)
- void Integrate (const vec3 &netForce, const vec3 &netTorque, float dt)

6.5.1 Constructor & Destructor Documentation

6.5.1.1 RigidBody()

```
PhysicsEngine::RigidBody::RigidBody ( )
```

brief Default Constructor. Initializes all scalar member variables to 1.0f and all vectors to the zero vector.

6.5.2 Member Function Documentation

6.5.2.1 AddForce()

```
void PhysicsEngine::RigidBody::AddForce (
            const vec3 & force )
```

brief Adds the specified force to the net force of a rigid body.

6.5.2.2 AddTorque()

```
void PhysicsEngine::RigidBody::AddTorque (
            const vec3 & force,
            const vec3 & point )
```

brief Adds the computed torque to the net torque. Computes the torque being applied to the point using the equation torque = force x (point - center of mass).

6.5.2.3 GetAngularMomentum()

```
const vec3 & PhysicsEngine::RigidBody::GetAngularMomentum ( ) const
```

brief Returns the angular momentum of the rigid body.

6.5.2.4 GetAngularVelocity()

```
const vec3 & PhysicsEngine::RigidBody::GetAngularVelocity ( ) const
```

brief Returns the angular velocity of the rigid body.

6.5.2.5 GetBodyInertiaTensor()

```
const mat3 & PhysicsEngine::RigidBody::GetBodyInertiaTensor ( ) const
```

brief Returns the inertia tensor in body coordinates.

6.5.2.6 GetCenterOfMass()

```
const vec3 & PhysicsEngine::RigidBody::GetCenterOfMass ( ) const
```

brief Returns the center of mass of the rigid body.

6.5.2.7 GetInverseBodyInertiaTensor()

```
const mat3 & PhysicsEngine::RigidBody::GetInverseBodyInertiaTensor ( ) const
```

brief Returns the inverse of the inertia tensor in body coordinates.

6.5.2.8 GetInverseMass()

```
float PhysicsEngine::RigidBody::GetInverseMass ( ) const
```

brief Returns the inverse mass of the rigid body.

If the inverse mass equals to 0 that means the mass is infinity.

6.5.2.9 GetLinearMomentum()

```
const vec3 & PhysicsEngine::RigidBody::GetLinearMomentum ( ) const
```

brief Returns the linear momentum of the rigid body.

6.5.2.10 GetLinearVelocity()

```
const vec3 & PhysicsEngine::RigidBody::GetLinearVelocity ( ) const
```

brief Returns the linear velocity of the rigid body.

6.5.2.11 GetMass()

```
float PhysicsEngine::RigidBody::GetMass ( ) const
```

brief Returns the mass of the rigid body.

6.5.2.12 GetNetForce()

```
const vec3 & PhysicsEngine::RigidBody::GetNetForce ( ) const
```

brief Returns the net force acting on the rigid body.

6.5.2.13 GetNetTorque()

```
const vec3 & PhysicsEngine::RigidBody::GetNetTorque ( ) const
```

brief Returns the net torque acting on the rigid body.

6.5.2.14 GetOrientation()

```
const MathEngine::Quaternion & PhysicsEngine::RigidBody::GetOrientation ( ) const
```

brief Returns the orientaiton quaternion of the rigid body.

6.5.2.15 InitializeRigidBody()

```
void PhysicsEngine::RigidBody::InitializeRigidBody (
            float massDensity,
            const MathEngine::Quaternion & initialOrientation,
            const std::vector< ShapesEngine::Triangle > & triangles,
            const mat3 & scale )
```

brief Initializes the properties of a rigid body.

If you want the rigid body to have infinite mass, so it can't be moved, pass in 0.0f for the mass density and the inverse mass will be set to 0.0f to indicate infinite mass.
If the specified mass density is negative, the mass and inverse mass will be set to 0.0f.
Computes the center of mass and inertia tensors from the given triangles that make up a solid polyhedron if the object does not have infinite mass.
Stores the local space bounding volumes used for collision detection.

6.5.2.16 Integrate() [1/2]

```
void PhysicsEngine::RigidBody::Integrate (
            const vec3 & netForce,
            const vec3 & netTorque,
            float dt )
```

brief A numerical integrator using semi-implicit Euler method. Uses semi-implicit Euler method to compute the new position and orientation of a rigid body.

6.5.2.17 Integrate() [2/2]

```
void PhysicsEngine::RigidBody::Integrate (
            float dt )
```

brief A numerical integrator using semi-implicit Euler method. Uses semi-implicit Euler method to compute the new position and orientation of a rigid body.

6.5.2.18 ResetForce()

```
void PhysicsEngine::RigidBody::ResetForce ( )
```

brief Sets the net force of a rigid body to the zero vector.

6.5.2.19 ResetTorque()

```
void PhysicsEngine::RigidBody::ResetTorque ( )
```

brief Sets the net torque of a rigid body to the zero vector.

6.5.2.20 SetAngularMomentum()

```
void PhysicsEngine::RigidBody::SetAngularMomentum (
            const vec3 & angularMomentum )
```

brief Sets the angular momentum of the rigid body to the specified vector.

6.5.2.21 SetAngularVelocity()

```
void PhysicsEngine::RigidBody::SetAngularVelocity (
            const vec3 & angularVelocity )
```

brief Sets the angular velocity of the rigid body to the specified vector.

6.5.2.22 SetBodyInertiaTensor()

```
void PhysicsEngine::RigidBody::SetBodyInertiaTensor (
            const mat3 & bodyInertia )
```

brief Sets the body inertia tensor to the specified matrix.

6.5.2.23 SetCenterOfMass()

```
void PhysicsEngine::RigidBody::SetCenterOfMass (
            const vec3 & centerOfMass )
```

brief Sets the center of mass the rigid body to the specified vector.

6.5.2.24 SetLinearMomentum()

```
void PhysicsEngine::RigidBody::SetLinearMomentum (
            const vec3 & linearMomentum )
```

brief Sets the linear momentum of the rigid body to the specified vector.

6.5.2.25 SetLinearVelocity()

```
void PhysicsEngine::RigidBody::SetLinearVelocity (
            const vec3 & velocity )
```

brief Sets the linear velocity of the rigid body to the specified vector.

6.5.2.26 SetMass()

```
void PhysicsEngine::RigidBody::SetMass (
            float mass )
```

brief Sets the mass of the rigid body to the specified float.

If you want the rigid body to have infinite mass, so it can't be moved, pass in 0.0f for the mass and the inverse mass will be set to 0.0f to indicate infinite mass.

If the specified mass is negative, the mass and inverse mass will be set to 0.0f.

6.5.2.27 SetOrientation()

```
void PhysicsEngine::RigidBody::SetOrientation (
            const MathEngine::Quaternion & orientation )
```

brief Sets the orientation of the rigid body to the specified quaternion.

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

- RigidBody.h

6.6 PhysicsEngine::RigidShape Class Reference

Public Member Functions

- RigidShape ()
- RigidShape (float massDensity, const std::vector< ShapesEngine::Triangle > &triangles, std::unique_ptr< ShapesEngine::ThreeDimensionalShapeAbstract > shape, std::unique_ptr< PhysicsEngine::BoundingVolumeAbstract > boundingVolume)
- void InitializeRigidShape (float massDensity, const std::vector< ShapesEngine::Triangle > &triangles, std::unique_ptr< ShapesEngine::ThreeDimensionalShapeAbstract > shape, std::unique_ptr< PhysicsEngine::BoundingVolumeAbstract > boundingVolume)
- float GetMass () const
- float GetInverseMass () const
- const mat3 & GetBodyInertiaTensor () const
- const mat3 & GetInverseBodyInertiaTensor () const
- const vec3 & GetCenterOfMass () const
- const vec3 & GetLinearVelocity () const
- const vec3 & GetLinearMomentum () const
- const MathEngine::Quaternion & GetOrientation () const
- const vec3 & GetAngularVelocity () const

- const vec3 & GetAngularMomentum () const
- void SetMass (float mass)
- void SetCenterOfMass (const vec3 &centerOfMass)
- void SetLinearVelocity (const vec3 &velocity)
- void SetLinearMomentum (const vec3 &linearMomentum)
- void SetBodyInertiaTensor (const mat3 &bodyInertia)
- void SetAngularVelocity (const vec3 &angularVelocity)
- void SetAngularMomentum (const vec3 &angularMomentum)
- void SetOrientation (const MathEngine::Quaternion &orientation)
- void Integrate (const vec3 &netForce, const vec3 &netTorque, float dt)
- void UpdateModelMatrix ()

  Updates the RigidShapes model matrix.
- float Volume ()

  Returns the RigidShapes volume.
- const RenderingEngine::Color & GetColor () const

  Returns the color of the RigidShape.
- const RenderingEngine::DrawArguments & GetDrawArguments () const

  Returns the draw arguments of the RigidShape for rendering.
- const mat4 & GetModelMatrix () const

  Returns the model matrix of the RigidShape.
- const vec3 & GetPosition () const

  Returns the position of the RigidShape.
- vec3 GetDimensions () const

  Returns the dimensions of the RigidShape.
- void SetDimensions (const vec3 &dimensions)

  Sets the dimensions of the RigidShape.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidShape.
- void SetColor (const RenderingEngine::Color &color)

  Sets the color of the RigidShape.
- void SetDrawArguments (const RenderingEngine::DrawArguments &drawArgs)

  Sets the draw arguments of the RigidShape.
- const mat4 & GetBoundingVolumeModelMatrix () const
- const RenderingEngine::DrawArguments & GetBoundingVolumeDrawArguments () const
- const RenderingEngine::Color & GetBoundingVolumeColor () const
- void SetBoundingVolumeDrawArguments (const RenderingEngine::DrawArguments &drawArgs)
- void SetBoundingVolumeColor (const RenderingEngine::Color &color)

6.6.1 Constructor & Destructor Documentation

6.6.1.1 RigidShape() [1/2]

```
PhysicsEngine::RigidShape::RigidShape ( )
```

brief Default Constructor.

6.6.1.2 RigidShape() [2/2]

```
PhysicsEngine::RigidShape::RigidShape (
            float massDensity,
            const std::vector< ShapesEngine::Triangle > & triangles,
            std::unique_ptr< ShapesEngine::ThreeDimensionalShapeAbstract > shape,
            std::unique_ptr< PhysicsEngine::BoundingVolumeAbstract > boundingVolume )
```

brief Creates a RigidShape object.

6.6.2 Member Function Documentation

6.6.2.1 GetAngularMomentum()

```
const vec3 & PhysicsEngine::RigidShape::GetAngularMomentum ( ) const
```

brief Returns the angular momentum of the RigidShape.

6.6.2.2 GetAngularVelocity()

```
const vec3 & PhysicsEngine::RigidShape::GetAngularVelocity ( ) const
```

brief Returns the angular velocity of the RigidShape.

6.6.2.3 GetBodyInertiaTensor()

```
const mat3 & PhysicsEngine::RigidShape::GetBodyInertiaTensor ( ) const
```

brief Returns the inertia tensor in body coordinates.

6.6.2.4 GetBoundingVolumeColor()

```
const RenderingEngine::Color & PhysicsEngine::RigidShape::GetBoundingVolumeColor ( ) const
```

brief Returns the color of the RigidShapes bounding volume.

6.6.2.5 GetBoundingVolumeDrawArguments()

```
const RenderingEngine::DrawArguments & PhysicsEngine::RigidShape::GetBoundingVolumeDraw↩
Arguments ( ) const
```

brief Returns the draw arguments of the RigidShapes bounding volume.

6.6.2.6 GetBoundingVolumeModelMatrix()

```
const mat4 & PhysicsEngine::RigidShape::GetBoundingVolumeModelMatrix ( ) const
```

brief Returns the model matrix of the RigidShapes bounding volume.

6.6.2.7 GetCenterOfMass()

```
const vec3 & PhysicsEngine::RigidShape::GetCenterOfMass ( ) const
```

brief Returns the center of mass of the RigidShape.

6.6.2.8 GetColor()

```
const RenderingEngine::Color & PhysicsEngine::RigidShape::GetColor ( ) const
```

Returns the color of the RigidShape.

6.6.2.9 GetDimensions()

```
vec3 PhysicsEngine::RigidShape::GetDimensions ( ) const
```

Returns the dimensions of the RigidShape.

6.6.2.10 GetDrawArguments()

```
const RenderingEngine::DrawArguments & PhysicsEngine::RigidShape::GetDrawArguments ( ) const
```

Returns the draw arguments of the RigidShape for rendering.

6.6.2.11 GetInverseBodyInertiaTensor()

```
const mat3 & PhysicsEngine::RigidShape::GetInverseBodyInertiaTensor ( ) const
```

brief Returns the inverse of the inertia tensor in body coordinates.

6.6.2.12 GetInverseMass()

```
float PhysicsEngine::RigidShape::GetInverseMass ( ) const
```

brief Returns the inverse mass of the RigidShape.

If the inverse mass equals to 0 that means the mass is infinity.

6.6.2.13 GetLinearMomentum()

```
const vec3 & PhysicsEngine::RigidShape::GetLinearMomentum ( ) const
```

brief Returns the linear momentum of the RigidShape.

6.6.2.14 GetLinearVelocity()

```
const vec3 & PhysicsEngine::RigidShape::GetLinearVelocity ( ) const
```

brief Returns the linear velocity of the RigidShape.

6.6.2.15 GetMass()

```
float PhysicsEngine::RigidShape::GetMass ( ) const
```

brief Returns the mass of the RigidShape.

6.6.2.16 GetModelMatrix()

```
const mat4 & PhysicsEngine::RigidShape::GetModelMatrix ( ) const
```

Returns the model matrix of the RigidShape.

6.6.2.17 GetOrientation()

```
const MathEngine::Quaternion & PhysicsEngine::RigidShape::GetOrientation ( ) const
```

brief Returns the orientaiton quaternion of the RigidShape.

6.6.2.18 GetPosition()

```
const vec3 & PhysicsEngine::RigidShape::GetPosition ( ) const
```

Returns the position of the RigidShape.

6.6.2.19 InitializeRigidShape()

```
void PhysicsEngine::RigidShape::InitializeRigidShape (
            float massDensity,
            const std::vector< ShapesEngine::Triangle > & triangles,
            std::unique_ptr< ShapesEngine::ThreeDimensionalShapeAbstract > shape,
            std::unique_ptr< PhysicsEngine::BoundingVolumeAbstract > boundingVolume )
```

brief Initializes a RigidShape object.

6.6.2.20 Integrate()

```
void PhysicsEngine::RigidShape::Integrate (
            const vec3 & netForce,
            const vec3 & netTorque,
            float dt )
```

brief A numerical integrator using semi-implicit Euler method. Uses semi-implicit Euler method to compute the new position and orientation of a rigid body.

6.6.2.21 SetAngularMomentum()

```
void PhysicsEngine::RigidShape::SetAngularMomentum (
            const vec3 & angularMomentum )
```

brief Sets the angular momentum of the RigidShape to the specified vector.

6.6.2.22 SetAngularVelocity()

```
void PhysicsEngine::RigidShape::SetAngularVelocity (
            const vec3 & angularVelocity )
```

brief Sets the angular velocity of the RigidShape to the specified vector.

6.6.2.23 SetBodyInertiaTensor()

```
void PhysicsEngine::RigidShape::SetBodyInertiaTensor (
            const mat3 & bodyInertia )
```

brief Sets the body inertia tensor to the specified matrix.

6.6.2.24 SetBoundingVolumeColor()

```
void PhysicsEngine::RigidShape::SetBoundingVolumeColor (
            const RenderingEngine::Color & color )
```

brief Sets the colr of the RigidShapes bounding volume.

6.6.2.25 SetBoundingVolumeDrawArguments()

```
void PhysicsEngine::RigidShape::SetBoundingVolumeDrawArguments (
            const RenderingEngine::DrawArguments & drawArgs )
```

brief Sets the draw arguments of the RigidShapes bounding volume for rendering.

6.6.2.26 SetCenterOfMass()

```
void PhysicsEngine::RigidShape::SetCenterOfMass (
            const vec3 & centerOfMass )
```

brief Sets the center of mass the RigidShape to the specified vector.

6.6.2.27 SetColor()

```
void PhysicsEngine::RigidShape::SetColor (
            const RenderingEngine::Color & color )
```

Sets the color of the RigidShape.

6.6.2.28 SetDimensions()

```
void PhysicsEngine::RigidShape::SetDimensions (
            const vec3 & dimensions )
```

Sets the dimensions of the RigidShape.

6.6.2.29 SetDrawArguments()

```
void PhysicsEngine::RigidShape::SetDrawArguments (
            const RenderingEngine::DrawArguments & drawArgs )
```

Sets the draw arguments of the RigidShape.

6.6.2.30 SetLinearMomentum()

```
void PhysicsEngine::RigidShape::SetLinearMomentum (
            const vec3 & linearMomentum )
```

brief Sets the linear momentum of the RigidShape to the specified vector.

6.6.2.31 SetLinearVelocity()

```
void PhysicsEngine::RigidShape::SetLinearVelocity (
            const vec3 & velocity )
```

brief Sets the linear velocity of the RigidShape to the specified vector.

6.6.2.32 SetMass()

```
void PhysicsEngine::RigidShape::SetMass (
            float mass )
```

brief Sets the mass of the RigidShape to the specified float.

If you want the RigidShape to have infinite mass, so it can't be moved, pass in 0.0f for the mass and the inverse mass will be set to 0.0f to indicate infinite mass.

If the specified mass is negative, the mass and inverse mass will be set to 0.0f.

6.6.2.33 SetOrientation()

```
void PhysicsEngine::RigidShape::SetOrientation (
            const MathEngine::Quaternion & orientation )
```

brief Sets the orientation of the RigidShape to the specified quaternion.

6.6.2.34 SetPosition()

```
void PhysicsEngine::RigidShape::SetPosition (
            const vec3 & position )
```

Sets the position of the RigidShape.

6.6.2.35 UpdateModelMatrix()

```
void PhysicsEngine::RigidShape::UpdateModelMatrix ( )
```

Updates the RigidShapes model matrix.

6.6.2.36 Volume()

```
float PhysicsEngine::RigidShape::Volume ( )
```

Returns the RigidShapes volume.

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

- RigidShape.h

6.7 PhysicsEngine::Sphere Struct Reference

Public Attributes

- vec3 **center**
- float **radius** { 1.0f }

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

- BoundingSphere.h

Chapter 7

File Documentation

7.1 BoundingBox.h

```
1 #pragma once
2
3 #include "BoundingVolume.h"
4 #include "Vertex.h"
5
6 namespace PhysicsEngine
7 {
11      struct AABB
12      {
13          vec3 min;
14          vec3 max;
15      };
16
19      void InitializeAABB(AABB& a, vec3 min, vec3 max);
20
23      void ComputeAABB(AABB& aabb, const std::vector<ShapesEngine::Vertex>& vertices);
24
28      void TransformAABB(AABB& worldAABB, const AABB& localAABB, const mat4& model);
29
32      bool TestIntersection(const AABB& a, const AABB& b);
33
34
38      class BoundingBox :  public BoundingVolumeAbstract
39      {
40      public:
43          BoundingBox();
44
47          BoundingBox(const std::vector<ShapesEngine::Vertex>& vertices, const RenderingEngine::Color&
        color);
48
51          void InitializeBoundingBox(const std::vector<ShapesEngine::Vertex>& vertices, const
        RenderingEngine::Color& color);
52
55          void UpdateModelMatrix() override;
56
59          void TransformBoundingVolume(const mat4& model) override;
60
61      private:
62          AABB mLocalAABB;
63          AABB mWorldAABB;
64      };
65 }
```

7.2 BoundingSphere.h

```
1 #pragma once
2
3 #include "BoundingVolume.h"
4 #include "Color.h"
5 #include "Vertex.h"
6 #include <vector>
7
8 namespace PhysicsEngine
9 {
```

```
10      struct Sphere
11      {
12          vec3 center;
13          float radius{ 1.0f };
14      };
15
18      void InitalizeSphere(Sphere& sphere, const vec3& center, float radius);
19
22      void ComputeSphere(Sphere& sphere, const std::vector<ShapesEngine::Vertex>& vertices);
23
26      void TransformSphere(Sphere& worldSphere, const Sphere& localSphere, const mat4& model);
27
30      bool TestIntersection(const Sphere& a, const Sphere& b);
31
35      class BoundingSphere :  public BoundingVolumeAbstract
36      {
37      public:
40          BoundingSphere();
41
44          BoundingSphere(const std::vector<ShapesEngine::Vertex>& vertices, const RenderingEngine::Color&
    color);
45
48          void InitializeBoundingSphere(const std::vector<ShapesEngine::Vertex>& vertices, const
    RenderingEngine::Color& color);
49
52          void UpdateModelMatrix() override;
53
56          void TransformBoundingVolume(const mat4& model) override;
57
58      private:
59          Sphere mLocalBoundingSphere;
60          Sphere mWorldBoundingSphere;
61      };
62  }
```

7.3 BoundingVolume.h

```
1  #pragma once
2
3  #include "Color.h"
4  #include "DrawArguments.h"
5  #include "RenderingEngineUtility.h"
6
7  namespace PhysicsEngine
8  {
9      class BoundingVolumeAbstract
10     {
11     public:
12
15         virtual void UpdateModelMatrix() = 0;
16
19         virtual void TransformBoundingVolume(const mat4& model) = 0;
20
23         virtual const RenderingEngine::Color& GetColor() const;
24
27         virtual const RenderingEngine::DrawArguments& GetDrawArguments() const;
28
31         virtual const mat4& GetModelMatrix() const;
32
35         virtual const vec3& GetPosition() const;
36
39         virtual const MathEngine::Quaternion& GetOrientation() const;
40
43         virtual void SetPosition(const vec3& position);
44
47         virtual void SetOrientation(const MathEngine::Quaternion& orientation);
48
51         virtual void SetColor(const RenderingEngine::Color& color);
52
55         virtual void SetDrawArguments(const RenderingEngine::DrawArguments& drawArgs);
56
57     protected:
58         RenderingEngine::RenderObject mRenderObject;
59     };
60 }
```

7.4 ForceFunctions.h

```
1  #pragma once
```

```
2
3 #include "MathEngine.h"
4
5 namespace PhysicsEngine
6 {
11      vec3 GravitationalForce(float mass, float gravityAcceleration, const vec3& direction);
12
17      vec3 DragForce(float k1, float k2, const vec3& velocity);
18
23      vec3 ApplyForce(float magnitdue, const vec3& direction);
24 }
```

7.5 PolyhedralMassProperties.h

```
1 #pragma once
2
3 #include "Triangle.h"
4 #include <vector>
5
6 namespace PhysicsEngine
7 {
10      void SubExpressions(double w0, double w1, double w2, double& f1, double& f2, double& f3, double& g0,
      double& g1, double& g2);
11
18      void ComputeMassProperties(const std::vector<ShapesEngine::Triangle>& triangles, double& mass, vec3&
      cm,
19          mat3& bodyInertia);
20
29      void ComputeMassProperties(const std::vector<ShapesEngine::Triangle>& triangles, double& mass, vec3&
      cm,
30          mat3& bodyInertia, const mat3& scale);
31 }
```

7.6 RigidBody.h

```
1 #pragma once
2
3 #include "PolyhedralMassProperties.h"
4
8 namespace PhysicsEngine
9 {
10      class RigidBody
11      {
12      public:
16          RigidBody();
17
29          void InitializeRigidBody(float massDensity, const MathEngine::Quaternion& initialOrientation,
30              const std::vector<ShapesEngine::Triangle>& triangles, const mat3& scale);
31
34          float GetMass() const;
35
40          float GetInverseMass() const;
41
44          const mat3& GetBodyInertiaTensor() const;
45
48          const mat3& GetInverseBodyInertiaTensor() const;
49
52          const vec3& GetCenterOfMass() const;
53
56          const vec3& GetLinearVelocity() const;
57
60          const vec3& GetLinearMomentum() const;
61
64          const MathEngine::Quaternion& GetOrientation() const;
65
68          const vec3& GetAngularVelocity() const;
69
72          const vec3& GetAngularMomentum() const;
73
76          const vec3& GetNetForce() const;
77
80          const vec3& GetNetTorque() const;
81
89          void SetMass(float mass);
90
93          void SetCenterOfMass(const vec3& centerOfMass);
94
97          void SetLinearVelocity(const vec3& velocity);
98
```

```
101         void SetLinearMomentum(const vec3& linearMomentum);
102
105         void SetBodyInertiaTensor(const mat3& bodyInertia);
106
109         void SetAngularVelocity(const vec3& angularVelocity);
110
113         void SetAngularMomentum(const vec3& angularMomentum);
114
117         void SetOrientation(const MathEngine::Quaternion& orientation);
118
121         void ResetForce();
122
125         void ResetTorque();
126
129         void AddForce(const vec3& force);
130
134         void AddTorque(const vec3& force, const vec3& point);
135
139         void Integrate(float dt);
140
144         void Integrate(const vec3& netForce, const vec3& netTorque, float dt);
145
146     private:
147         float mMass;
148         float mInverseMass;
149
150         mat3 mBodyInertiaTensor;
151         mat3 mInverseBodyInertiaTensor;
152         mat3 mWorldCMInertiaTensor;
153         mat3 mInverseWorldCMInertiaTensor;
154
155         vec3 mCenterOfMass;
156         vec3 mLinearVelocity;
157         vec3 mLinearMomentum;
158         vec3 mNetForce;
159
160         MathEngine::Quaternion mOrientation;
161         vec3 mAngularVelocity;
162         vec3 mAngularMomentum;
163         vec3 mNetTorque;
164     };
165
168     void Interpolate(const RigidBody& r1, const RigidBody& r2, RigidBody& r3, float t);
169 }
```

7.7 RigidShape.h

```
1 #pragma once
2
3 #include "RigidBody.h"
4 #include "ThreeDimensionalShape.h"
5 #include "BoundingVolume.h"
6 #include <memory>
7
8 namespace PhysicsEngine
9 {
10     class RigidShape
11     {
12     public:
13
16         RigidShape();
17
20         RigidShape(float massDensity, const std::vector<ShapesEngine::Triangle>& triangles,
21             std::unique_ptr<ShapesEngine::ThreeDimensionalShapeAbstract> shape,
22             std::unique_ptr<PhysicsEngine::BoundingVolumeAbstract> boundingVolume);
23
26         void InitializeRigidShape(float massDensity,const std::vector<ShapesEngine::Triangle>& triangles,
27             std::unique_ptr<ShapesEngine::ThreeDimensionalShapeAbstract> shape,
28             std::unique_ptr<PhysicsEngine::BoundingVolumeAbstract> boundingVolume);
29
30
      //-----------------------------------------------------------------------------------------------------------------------
31         //Rigid Body Delegates
32
35         float GetMass() const;
36
41         float GetInverseMass() const;
42
45         const mat3& GetBodyInertiaTensor() const;
46
49         const mat3& GetInverseBodyInertiaTensor() const;
50
53         const vec3& GetCenterOfMass() const;
```

```
54
57          const vec3& GetLinearVelocity() const;
58
61          const vec3& GetLinearMomentum() const;
62
65          const MathEngine::Quaternion& GetOrientation() const;
66
69          const vec3& GetAngularVelocity() const;
70
73          const vec3& GetAngularMomentum() const;
74
82          void SetMass(float mass);
83
86          void SetCenterOfMass(const vec3& centerOfMass);
87
90          void SetLinearVelocity(const vec3& velocity);
91
94          void SetLinearMomentum(const vec3& linearMomentum);
95
98          void SetBodyInertiaTensor(const mat3& bodyInertia);
99
102          void SetAngularVelocity(const vec3& angularVelocity);
103
106          void SetAngularMomentum(const vec3& angularMomentum);
107
110          void SetOrientation(const MathEngine::Quaternion& orientation);
111
115          void Integrate(const vec3& netForce, const vec3& netTorque, float dt);
116
117
      //-----------------------------------------------------------------------------------------------------------------------------
118
119
120
      //-----------------------------------------------------------------------------------------------------------------------------
121          //ThreeDimensionalShapeAbstract Delegates
122
125          void UpdateModelMatrix();
126
129          float Volume();
130
133          const RenderingEngine::Color& GetColor() const;
134
137          const RenderingEngine::DrawArguments& GetDrawArguments() const;
138
141          const mat4& GetModelMatrix() const;
142
145          const vec3& GetPosition() const;
146
149          vec3 GetDimensions() const;
150
153          void SetDimensions(const vec3& dimensions);
154
157          void SetPosition(const vec3& position);
158
161          void SetColor(const RenderingEngine::Color& color);
162
165          void SetDrawArguments(const RenderingEngine::DrawArguments& drawArgs);
166
167
      //-----------------------------------------------------------------------------------------------------------------------------
168
169
170
171
      //-----------------------------------------------------------------------------------------------------------------------------
172          //Bounding Volume Delegates
173
176          const mat4& GetBoundingVolumeModelMatrix() const;
177
180          const RenderingEngine::DrawArguments& GetBoundingVolumeDrawArguments() const;
181
184          const RenderingEngine::Color& GetBoundingVolumeColor() const;
185
188          void SetBoundingVolumeDrawArguments(const RenderingEngine::DrawArguments& drawArgs);
189
192          void SetBoundingVolumeColor(const RenderingEngine::Color& color);
193
194
      //-----------------------------------------------------------------------------------------------------------------------------
195
196      private:
197          std::unique_ptr<ShapesEngine::ThreeDimensionalShapeAbstract> mShape;
198          RigidBody mRigidBody;
199          vec3 mOffset;
200
201          std::unique_ptr<PhysicsEngine::BoundingVolumeAbstract> mBoundingVolume;
```

```
202     };
203
206     void SimulateRigidShape(RigidShape& previousRigidShape, RigidShape& currentRigidShape, const vec3&
      netForce, const vec3& netTorque, float simulationTime);
207
210     void Interpolate(const RigidShape& r1, const RigidShape& r2, RigidShape& r3, float t);
211 };
```

Index

AddForce
    PhysicsEngine::RigidBody, 23
AddTorque
    PhysicsEngine::RigidBody, 23
ApplyForce
    PhysicsEngine, 10

BoundingBox
    PhysicsEngine::BoundingBox, 16
BoundingSphere
    PhysicsEngine::BoundingSphere, 18

ComputeAABB
    PhysicsEngine, 10
ComputeMassProperties
    PhysicsEngine, 10
ComputeSphere
    PhysicsEngine, 10

DragForce
    PhysicsEngine, 11

GetAngularMomentum
    PhysicsEngine::RigidBody, 23
    PhysicsEngine::RigidShape, 29
GetAngularVelocity
    PhysicsEngine::RigidBody, 23
    PhysicsEngine::RigidShape, 29
GetBodyInertiaTensor
    PhysicsEngine::RigidBody, 23
    PhysicsEngine::RigidShape, 29
GetBoundingVolumeColor
    PhysicsEngine::RigidShape, 29
GetBoundingVolumeDrawArguments
    PhysicsEngine::RigidShape, 29
GetBoundingVolumeModelMatrix
    PhysicsEngine::RigidShape, 29
GetCenterOfMass
    PhysicsEngine::RigidBody, 23
    PhysicsEngine::RigidShape, 30
GetColor
    PhysicsEngine::BoundingVolumeAbstract, 20
    PhysicsEngine::RigidShape, 30
GetDimensions
    PhysicsEngine::RigidShape, 30
GetDrawArguments
    PhysicsEngine::BoundingVolumeAbstract, 20
    PhysicsEngine::RigidShape, 30
GetInverseBodyInertiaTensor
    PhysicsEngine::RigidBody, 24
    PhysicsEngine::RigidShape, 30
GetInverseMass
    PhysicsEngine::RigidBody, 24
    PhysicsEngine::RigidShape, 30
GetLinearMomentum
    PhysicsEngine::RigidBody, 24
    PhysicsEngine::RigidShape, 30
GetLinearVelocity
    PhysicsEngine::RigidBody, 24
    PhysicsEngine::RigidShape, 31
GetMass
    PhysicsEngine::RigidBody, 24
    PhysicsEngine::RigidShape, 31
GetModelMatrix
    PhysicsEngine::BoundingVolumeAbstract, 20
    PhysicsEngine::RigidShape, 31
GetNetForce
    PhysicsEngine::RigidBody, 24
GetNetTorque
    PhysicsEngine::RigidBody, 24
GetOrientation
    PhysicsEngine::BoundingVolumeAbstract, 20
    PhysicsEngine::RigidBody, 25
    PhysicsEngine::RigidShape, 31
GetPosition
    PhysicsEngine::BoundingVolumeAbstract, 20
    PhysicsEngine::RigidShape, 31
GravitationalForce
    PhysicsEngine, 11

InitalizeSphere
    PhysicsEngine, 11
InitializeAABB
    PhysicsEngine, 11
InitializeBoundingBox
    PhysicsEngine::BoundingBox, 16
InitializeBoundingSphere
    PhysicsEngine::BoundingSphere, 18
InitializeRigidBody
    PhysicsEngine::RigidBody, 25
InitializeRigidShape
    PhysicsEngine::RigidShape, 31
Integrate
    PhysicsEngine::RigidBody, 25
    PhysicsEngine::RigidShape, 31
Interpolate
    PhysicsEngine, 11, 12

PhysicsEngine, 9
    ApplyForce, 10

ComputeAABB, 10
ComputeMassProperties, 10
ComputeSphere, 10
DragForce, 11
GravitationalForce, 11
InitalizeSphere, 11
InitializeAABB, 11
Interpolate, 11, 12
SimulateRigidShape, 12
SubExpressions, 12
TestIntersection, 12, 13
TransformAABB, 13
TransformSphere, 13
PhysicsEngine::AABB, 15
PhysicsEngine::BoundingBox, 15
BoundingBox, 16
InitializeBoundingBox, 16
TransformBoundingVolume, 16
UpdateModelMatrix, 17
PhysicsEngine::BoundingSphere, 17
BoundingSphere, 18
InitializeBoundingSphere, 18
TransformBoundingVolume, 18
UpdateModelMatrix, 18
PhysicsEngine::BoundingVolumeAbstract, 19
GetColor, 20
GetDrawArguments, 20
GetModelMatrix, 20
GetOrientation, 20
GetPosition, 20
SetColor, 20
SetDrawArguments, 21
SetOrientation, 21
SetPosition, 21
TransformBoundingVolume, 21
UpdateModelMatrix, 21
PhysicsEngine::RigidBody, 22
AddForce, 23
AddTorque, 23
GetAngularMomentum, 23
GetAngularVelocity, 23
GetBodyInertiaTensor, 23
GetCenterOfMass, 23
GetInverseBodyInertiaTensor, 24
GetInverseMass, 24
GetLinearMomentum, 24
GetLinearVelocity, 24
GetMass, 24
GetNetForce, 24
GetNetTorque, 24
GetOrientation, 25
InitializeRigidBody, 25
Integrate, 25
ResetForce, 25
ResetTorque, 26
RigidBody, 22
SetAngularMomentum, 26
SetAngularVelocity, 26
SetBodyInertiaTensor, 26
SetCenterOfMass, 26
SetLinearMomentum, 26
SetLinearVelocity, 26
SetMass, 27
SetOrientation, 27
PhysicsEngine::RigidShape, 27
GetAngularMomentum, 29
GetAngularVelocity, 29
GetBodyInertiaTensor, 29
GetBoundingVolumeColor, 29
GetBoundingVolumeDrawArguments, 29
GetBoundingVolumeModelMatrix, 29
GetCenterOfMass, 30
GetColor, 30
GetDimensions, 30
GetDrawArguments, 30
GetInverseBodyInertiaTensor, 30
GetInverseMass, 30
GetLinearMomentum, 30
GetLinearVelocity, 31
GetMass, 31
GetModelMatrix, 31
GetOrientation, 31
GetPosition, 31
InitializeRigidShape, 31
Integrate, 31
RigidShape, 28
SetAngularMomentum, 32
SetAngularVelocity, 32
SetBodyInertiaTensor, 32
SetBoundingVolumeColor, 32
SetBoundingVolumeDrawArguments, 32
SetCenterOfMass, 32
SetColor, 33
SetDimensions, 33
SetDrawArguments, 33
SetLinearMomentum, 33
SetLinearVelocity, 33
SetMass, 33
SetOrientation, 34
SetPosition, 34
UpdateModelMatrix, 34
Volume, 34
PhysicsEngine::Sphere, 34

ResetForce
PhysicsEngine::RigidBody, 25
ResetTorque
PhysicsEngine::RigidBody, 26
RigidBody
PhysicsEngine::RigidBody, 22
RigidShape
PhysicsEngine::RigidShape, 28

SetAngularMomentum
PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 32
SetAngularVelocity

PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 32
SetBodyInertiaTensor
PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 32
SetBoundingVolumeColor
PhysicsEngine::RigidShape, 32
SetBoundingVolumeDrawArguments
PhysicsEngine::RigidShape, 32
SetCenterOfMass
PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 32
SetColor
PhysicsEngine::BoundingVolumeAbstract, 20
PhysicsEngine::RigidShape, 33
SetDimensions
PhysicsEngine::RigidShape, 33
SetDrawArguments
PhysicsEngine::BoundingVolumeAbstract, 21
PhysicsEngine::RigidShape, 33
SetLinearMomentum
PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 33
SetLinearVelocity
PhysicsEngine::RigidBody, 26
PhysicsEngine::RigidShape, 33
SetMass
PhysicsEngine::RigidBody, 27
PhysicsEngine::RigidShape, 33
SetOrientation
PhysicsEngine::BoundingVolumeAbstract, 21
PhysicsEngine::RigidBody, 27
PhysicsEngine::RigidShape, 34
SetPosition
PhysicsEngine::BoundingVolumeAbstract, 21
PhysicsEngine::RigidShape, 34
SimulateRigidShape
PhysicsEngine, 12
SubExpressions
PhysicsEngine, 12

TestIntersection
PhysicsEngine, 12, 13
TransformAABB
PhysicsEngine, 13
TransformBoundingVolume
PhysicsEngine::BoundingBox, 16
PhysicsEngine::BoundingSphere, 18
PhysicsEngine::BoundingVolumeAbstract, 21
TransformSphere
PhysicsEngine, 13

UpdateModelMatrix
PhysicsEngine::BoundingBox, 17
PhysicsEngine::BoundingSphere, 18
PhysicsEngine::BoundingVolumeAbstract, 21
PhysicsEngine::RigidShape, 34

Volume