Farouq Adepetu's Physics 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 PhysicsEngine Namespace Reference

An engine for physics simulations.

Classes

- struct AABB
- class RigidBody
- class RigidBox
- class RigidCone
- class RigidCylinder
- class RigidPyramid
- class RigidSphere

Functions

- AABB CreateAABB (const std::vector< ShapesEngine::Vertex > &vertexList)
- vec3 GravitationalForce (float mass, float gravityAcceleration, const vec3 &direction)
- vec3 DragForce (float dragCoefficent, 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 Interpolate (const RigidBody &r1, const RigidBody &r2, RigidBody &r3, float t)

4.1.1 Detailed Description

An engine for physics simulations.

4.1.2 Function Documentation

4.1.2.1 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

4.1.2.2 ComputeMassProperties()

```
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.

4.1.2.3 CreateAABB()

```
AABB PhysicsEngine::CreateAABB (
            const std::vector< ShapesEngine::Vertex > & vertexList )
```

brief Returns an AABB to bound the specified object.

Parameters

in	*vertexList*	The vertex list of the object to bound by the AABB.

4.1.2.4 DragForce()

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

brief Returns the force due to drag (e.g. air resistance, friction)

Formula used is F = -cv, where v is the velocity of the object and c is the drag coefficient.

4.1.2.5 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.

4.1.2.6 Interpolate()

```
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.

4.1.2.7 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.

Chapter 5

Class Documentation

5.1 PhysicsEngine::AABB Struct Reference

`#include <AABB.h>`

Public Attributes

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

5.1.1 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:

- AABB.h

5.2 PhysicsEngine::RigidBody Class Reference

Public Member Functions

- RigidBody ()
- void InitializeRigidBody (float massDensity, const MathEngine::Quaternion &initialOrientation, const std↩ ::vector< ShapesEngine::Triangle > &triangles)
- 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)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 RigidBody()

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

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

5.2.2 Member Function Documentation

5.2.2.1 AddForce()

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

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

5.2.2.2 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).

5.2.2.3 GetAngularMomentum()

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

brief Returns the angular momentum of the rigid body.

5.2.2.4 GetAngularVelocity()

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

brief Returns the angular velocity of the rigid body.

5.2.2.5 GetBodyInertiaTensor()

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

brief Returns the inertia tensor in body coordinates.

5.2.2.6 GetCenterOfMass()

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

brief Returns the center of mass of the rigid body.

5.2.2.7 GetInverseBodyInertiaTensor()

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

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

5.2.2.8 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.

5.2.2.9 GetLinearMomentum()

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

brief Returns the linear momentum of the rigid body.

5.2.2.10 GetLinearVelocity()

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

brief Returns the linear velocity of the rigid body.

5.2.2.11 GetMass()

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

brief Returns the mass of the rigid body.

5.2.2.12 GetNetForce()

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

brief Returns the net force acting on the rigid body.

5.2.2.13 GetNetTorque()

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

brief Returns the net torque acting on the rigid body.

5.2.2.14 GetOrientation()

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

brief Returns the orientaiton quaternion of the rigid body.

5.2.2.15 InitializeRigidBody()

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

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.

5.2.2.16 Integrate()

```
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.

5.2.2.17 ResetForce()

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

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

5.2.2.18 ResetTorque()

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

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

5.2.2.19 SetAngularMomentum()

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

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

5.2.2.20 SetAngularVelocity()

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

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

5.2.2.21 SetBodyInertiaTensor()

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

brief Sets the body inertia tensor to the specified matrix.

5.2.2.22 SetCenterOfMass()

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

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

5.2.2.23 SetLinearMomentum()

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

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

5.2.2.24 SetLinearVelocity()

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

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

5.2.2.25 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.

5.2.2.26 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

5.3 PhysicsEngine::RigidBox Class Reference

Public Member Functions

- RigidBox ()

  Default Constructor. Constructs a RigidBox object.
- void InitializeRigidBox (float width, float height, float depth, float massDensity, const vec3 &initialPosition, const MathEngine::Quaternion &initialOrientation, const RenderingEngine::Color &color, const std::vector< ShapesEngine::Vertex > &vertices, const std::vector< ShapesEngine::Triangle > &triangles)

  Iniitalizes a rigid box that can be used to do physics simulations.
- 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.
- const RigidBody & GetRigidBody () const

  Returns the RigidBody object.
- RigidBody & GetRigidBody ()

  Returns the RigidBody object,.
- const ShapesEngine::ThreeDimensionalShape & GetShape () const

  Returns the ThreeDimensionalShape object.
- ShapesEngine::ThreeDimensionalShape & GetShape ()

  Returns the ThreeDimensionalShape object.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidBox.
- void UpdateModelMatrix ()

  Updates the model matrix of the RigidBox.
- float Volume ()

  Returns the volume of the box.

5.3.1 Constructor & Destructor Documentation

5.3.1.1 RigidBox()

```
PhysicsEngine::RigidBox::RigidBox ( )
```

Default Constructor. Constructs a RigidBox object.

5.3.2 Member Function Documentation

5.3.2.1 GetDepth()

```
float PhysicsEngine::RigidBox::GetDepth ( ) const
```

Returns the depth of the box.

5.3.2.2 GetHeight()

```
float PhysicsEngine::RigidBox::GetHeight ( ) const
```

Returns the height of the box.

5.3.2.3 GetRigidBody() [1/2]

```
RigidBody & PhysicsEngine::RigidBox::GetRigidBody ( )
```

Returns the RigidBody object,.

5.3.2.4 GetRigidBody() [2/2]

```
const RigidBody & PhysicsEngine::RigidBox::GetRigidBody ( ) const
```

Returns the RigidBody object.

5.3.2.5 GetShape() [1/2]

```
ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidBox::GetShape ( )
```

Returns the ThreeDimensionalShape object.

5.3.2.6 GetShape() [2/2]

```
const ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidBox::GetShape ( ) const
```

Returns the ThreeDimensionalShape object.

5.3.2.7 GetWidth()

```
float PhysicsEngine::RigidBox::GetWidth ( ) const
```

Returns the width of the box.

5.3.2.8 InitializeRigidBox()

```
void PhysicsEngine::RigidBox::InitializeRigidBox (
            float width,
            float height,
            float depth,
            float massDensity,
            const vec3 & initialPosition,
            const MathEngine::Quaternion & initialOrientation,
            const RenderingEngine::Color & color,
            const std::vector< ShapesEngine::Vertex > & vertices,
            const std::vector< ShapesEngine::Triangle > & triangles )
```

Iniitalizes a rigid box that can be used to do physics simulations.

Parameters

in	*width*	The width of the box.
in	*height*	The height of the box.
in	*depth*	The depth of the box.
in	*color*	The color of the box.
in	*massDensity*	The mass density of the box.
in	*initialPosition*	The initial position of the box.
in	*initialOrientation*	The initial orientation of the box.
in	*color*	The color of the box.
in	*vertices*	The vertex list of a unit box.
in	*triangles*	The triangle list of a unit box.

5.3.2.9 SetDepth()

```
void PhysicsEngine::RigidBox::SetDepth (
            float depth )
```

Sets the depth of the box.

5.3.2.10 SetHeight()

```
void PhysicsEngine::RigidBox::SetHeight (
            float height )
```

Sets the height of the box.

5.3.2.11 SetPosition()

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

Sets the position of the RigidBox.

5.3.2.12 SetWidth()

```
void PhysicsEngine::RigidBox::SetWidth (
            float width )
```

Sets the width of the box.

5.3.2.13 UpdateModelMatrix()

```
void PhysicsEngine::RigidBox::UpdateModelMatrix ( )
```

Updates the model matrix of the RigidBox.

5.3.2.14 Volume()

```
float PhysicsEngine::RigidBox::Volume ( )
```

Returns the volume of the box.

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

- RigidBox.h

5.4 PhysicsEngine::RigidCone Class Reference

Public Member Functions

- RigidCone ()

  Default Constructor. Constructs a RigidCone object.
- void InitializeRigidCone (float radius, float height, float massDensity, const vec3 &initialPosition, const Math↩ Engine::Quaternion &initialOrientation, const RenderingEngine::Color &color, const std::vector< Shapes↩ Engine::Vertex > &vertices, const std::vector< ShapesEngine::Triangle > &triangles)

  Iniitalizes a rigid cone that can be used to do physics simulations.
- 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.
- const RigidBody & GetRigidBody () const

  Returns the RigidBody object of the RigidCone.
- RigidBody & GetRigidBody ()

  Returns the RigidBody object of the RigidCone.
- const ShapesEngine::ThreeDimensionalShape & GetShape () const

  Returns the ThreeDimensionalShape object.
- ShapesEngine::ThreeDimensionalShape & GetShape ()

  Returns the ThreeDimensionalShape.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidCone.
- void UpdateModelMatrix ()

  Updates the model matrix of the RigidCone.
- float Volume ()

  Returns the volume of the cone.

5.4.1 Constructor & Destructor Documentation

5.4.1.1 RigidCone()

```
PhysicsEngine::RigidCone::RigidCone ( )
```

Default Constructor. Constructs a RigidCone object.

5.4.2 Member Function Documentation

5.4.2.1 GetHeight()

```
float PhysicsEngine::RigidCone::GetHeight ( ) const
```

Returns the height of the cone.

5.4.2.2 GetRadius()

```
float PhysicsEngine::RigidCone::GetRadius ( ) const
```

Returns the radius of the cone.

5.4.2.3 GetRigidBody() [1/2]

```
RigidBody & PhysicsEngine::RigidCone::GetRigidBody ( )
```

Returns the RigidBody object of the RigidCone.

5.4.2.4 GetRigidBody() [2/2]

```
const RigidBody & PhysicsEngine::RigidCone::GetRigidBody ( ) const
```

Returns the RigidBody object of the RigidCone.

5.4.2.5 GetShape() [1/2]

```
ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidCone::GetShape ( )
```

Returns the ThreeDimensionalShape.

5.4.2.6 GetShape() [2/2]

```
const ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidCone::GetShape ( ) const
```

Returns the ThreeDimensionalShape object.

5.4.2.7 InitializeRigidCone()

```
void PhysicsEngine::RigidCone::InitializeRigidCone (
            float radius,
            float height,
            float massDensity,
            const vec3 & initialPosition,
            const MathEngine::Quaternion & initialOrientation,
            const RenderingEngine::Color & color,
            const std::vector< ShapesEngine::Vertex > & vertices,
            const std::vector< ShapesEngine::Triangle > & triangles )
```

Iniitalizes a rigid cone that can be used to do physics simulations.

Parameters

in	*radius*	The radius of the cone.
in	*height*	The height of the cone.
in	*color*	The color of the cone.
in	*massDensity*	The mass density of the cone.
in	*initialPosition*	The initial position of the cone.
in	*initialOrientation*	The initial orientation of the cone.
in	*color*	The color of the cone.
in	*vertices*	The vertex list of a unit cone.
in	*triangles*	The triangle list of a unit cone.

5.4.2.8 SetHeight()

```
void PhysicsEngine::RigidCone::SetHeight (
            float height )
```

Sets the height of the cone.

5.4.2.9 SetPosition()

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

Sets the position of the RigidCone.

5.4.2.10 SetRadius()

```
void PhysicsEngine::RigidCone::SetRadius (
            float radius )
```

Sets the radius of the cone.

5.4.2.11 UpdateModelMatrix()

```
void PhysicsEngine::RigidCone::UpdateModelMatrix ( )
```

Updates the model matrix of the RigidCone.

5.4.2.12 Volume()

```
float PhysicsEngine::RigidCone::Volume ( )
```

Returns the volume of the cone.

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

- RigidCone.h

5.5 PhysicsEngine::RigidCylinder Class Reference

Public Member Functions

- RigidCylinder ()

  Default Constructor. Constructs a RigidCylinder object.
- void InitializeRigidCylinder (float radius, float height, float massDensity, const vec3 &initialPosition, const MathEngine::Quaternion &initialOrientation, const RenderingEngine::Color &color, const std::vector< ShapesEngine::Vertex > &vertices, const std::vector< ShapesEngine::Triangle > &triangles)

  Iniitalizes a rigid cylinder that can be used to do physics simulations.
- 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.
- const RigidBody & GetRigidBody () const

  Returns the RigidBody object.
- RigidBody & GetRigidBody ()

  Returns the RigidBody object.
- const ShapesEngine::ThreeDimensionalShape & GetShape () const

  Returns the ThreeDimensionalShape object.
- ShapesEngine::ThreeDimensionalShape & GetShape ()

  Returns the ThreeDimensionalShape object.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidCylinder.
- void UpdateModelMatrix ()

  Updates the model matrix of the RigidCylinder.
- float Volume ()

  Returns the volume of the cylinder.

5.5.1 Constructor & Destructor Documentation

5.5.1.1 RigidCylinder()

```
PhysicsEngine::RigidCylinder::RigidCylinder ( )
```

Default Constructor. Constructs a RigidCylinder object.

5.5.2 Member Function Documentation

5.5.2.1 GetHeight()

```
float PhysicsEngine::RigidCylinder::GetHeight ( ) const
```

Returns the height of the cylinder.

5.5.2.2 GetRadius()

```
float PhysicsEngine::RigidCylinder::GetRadius ( ) const
```

Returns the radius of the cylinder.

5.5.2.3 GetRigidBody() [1/2]

```
RigidBody & PhysicsEngine::RigidCylinder::GetRigidBody ( )
```

Returns the RigidBody object.

5.5.2.4 GetRigidBody() [2/2]

```
const RigidBody & PhysicsEngine::RigidCylinder::GetRigidBody ( ) const
```

Returns the RigidBody object.

5.5.2.5 GetShape() [1/2]

```
ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidCylinder::GetShape ( )
```

Returns the ThreeDimensionalShape object.

5.5.2.6 GetShape() [2/2]

```
const ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidCylinder::GetShape ( ) const
```

Returns the ThreeDimensionalShape object.

5.5.2.7 InitializeRigidCylinder()

```
void PhysicsEngine::RigidCylinder::InitializeRigidCylinder (
            float radius,
            float height,
            float massDensity,
            const vec3 & initialPosition,
            const MathEngine::Quaternion & initialOrientation,
            const RenderingEngine::Color & color,
            const std::vector< ShapesEngine::Vertex > & vertices,
            const std::vector< ShapesEngine::Triangle > & triangles )
```

Iniitalizes a rigid cylinder that can be used to do physics simulations.

Parameters

in	*radius*	The radius of the cylinder.
in	*height*	The height of the cylinder.
in	*color*	The color of the cylinder.
in	*massDensity*	The mass density of the cylinder.
in	*initialPosition*	The initial position of the cylinder.
in	*initialOrientation*	The initial orientation of the cylinder.
in	*color*	The color of the cylinder.
in	*vertices*	The vertex list of a unit cylinder.
in	*triangles*	The triangle list of a unit cylinder.

5.5.2.8 SetHeight()

```
void PhysicsEngine::RigidCylinder::SetHeight (
            float height )
```

Sets the height of the cylinder.

5.5.2.9 SetPosition()

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

Sets the position of the RigidCylinder.

5.5.2.10 SetRadius()

```
void PhysicsEngine::RigidCylinder::SetRadius (
            float radius )
```

Sets the radius of the cylinder.

5.5.2.11 UpdateModelMatrix()

```
void PhysicsEngine::RigidCylinder::UpdateModelMatrix ( )
```

Updates the model matrix of the RigidCylinder.

5.5.2.12 Volume()

```
float PhysicsEngine::RigidCylinder::Volume ( )
```

Returns the volume of the cylinder.

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

- RigidCylinder.h

5.6 PhysicsEngine::RigidPyramid Class Reference

Public Member Functions

- RigidPyramid ()

  Default Constructor. Constructs a RigidPyramid object.
- void InitializeRigidPyramid (float width, float height, float depth, float massDensity, const vec3 &initialPosition, const MathEngine::Quaternion &initialOrientation, const RenderingEngine::Color &color, const std::vector< ShapesEngine::Vertex > &vertices, const std::vector< ShapesEngine::Triangle > &triangles)

  Iniitalizes a rigid pyramid that can be used to do physics simulations.
- 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.

- const RigidBody & GetRigidBody () const

  Returns the RigidBody object.
- RigidBody & GetRigidBody ()

  Returns the RigidBody object.
- const ShapesEngine::ThreeDimensionalShape & GetShape () const

  Returns the ThreeDimensionalShape object.
- ShapesEngine::ThreeDimensionalShape & GetShape ()

  Returns the ThreeDimensionalShape object.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidPyramid.
- void UpdateModelMatrix ()

  Updates the model matrix of the RigidPyramid.
- float Volume ()

  Returns the volume of the pyramid.

5.6.1 Constructor & Destructor Documentation

5.6.1.1 RigidPyramid()

```
PhysicsEngine::RigidPyramid::RigidPyramid ( )
```

Default Constructor. Constructs a RigidPyramid object.

5.6.2 Member Function Documentation

5.6.2.1 GetDepth()

```
float PhysicsEngine::RigidPyramid::GetDepth ( ) const
```

Returns the depth of the pyramid.

5.6.2.2 GetHeight()

```
float PhysicsEngine::RigidPyramid::GetHeight ( ) const
```

Returns the height of the pyramid.

5.6.2.3 GetRigidBody() [1/2]

```
RigidBody & PhysicsEngine::RigidPyramid::GetRigidBody ( )
```

Returns the RigidBody object.

5.6.2.4 GetRigidBody() [2/2]

```
const RigidBody & PhysicsEngine::RigidPyramid::GetRigidBody ( ) const
```

Returns the RigidBody object.

5.6.2.5 GetShape() [1/2]

```
ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidPyramid::GetShape ( )
```

Returns the ThreeDimensionalShape object.

5.6.2.6 GetShape() [2/2]

```
const ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidPyramid::GetShape ( ) const
```

Returns the ThreeDimensionalShape object.

5.6.2.7 GetWidth()

```
float PhysicsEngine::RigidPyramid::GetWidth ( ) const
```

Returns the width of the pyramid.

5.6.2.8 InitializeRigidPyramid()

```
void PhysicsEngine::RigidPyramid::InitializeRigidPyramid (
            float width,
            float height,
            float depth,
            float massDensity,
            const vec3 & initialPosition,
            const MathEngine::Quaternion & initialOrientation,
            const RenderingEngine::Color & color,
            const std::vector< ShapesEngine::Vertex > & vertices,
            const std::vector< ShapesEngine::Triangle > & triangles )
```

Iniitalizes a rigid pyramid that can be used to do physics simulations.

Parameters

in	*width*	The width of the pyramid.
in	*height*	The height of the pyramid.
in	*depth*	The depth of the pyramid.
in	*color*	The color of the pyramid.
in	*massDensity*	The mass density of the pyramid.
in	*initialPosition*	The initial position of the pyramid.
in	*initialOrientation*	The initial orientation of the pyramid.
in	*color*	The color of the pyramid.
in	*vertices*	The vertex list of a unit pyramid.
in	*triangles*	The triangle list of a unit pyramid.

5.6.2.9 SetDepth()

```
void PhysicsEngine::RigidPyramid::SetDepth (
            float depth )
```

Sets the depth of the pyramid.

5.6.2.10 SetHeight()

```
void PhysicsEngine::RigidPyramid::SetHeight (
            float height )
```

Sets the height of the pyramid.

5.6.2.11 SetPosition()

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

Sets the position of the RigidPyramid.

5.6.2.12 SetWidth()

```
void PhysicsEngine::RigidPyramid::SetWidth (
            float width )
```

Sets the width of the pyramid.

5.6.2.13 UpdateModelMatrix()

```
void PhysicsEngine::RigidPyramid::UpdateModelMatrix ( )
```

Updates the model matrix of the RigidPyramid.

5.6.2.14 Volume()

```
float PhysicsEngine::RigidPyramid::Volume ( )
```

Returns the volume of the pyramid.

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

- RigidPyramid.h

5.7 PhysicsEngine::RigidSphere Class Reference

Public Member Functions

- RigidSphere ()

  Default Constructor. Constructs a RigidSphere object.
- void InitializeRigidSphere (float radius, float massDensity, const vec3 &initialPosition, const MathEngine::↩ Quaternion &initialOrientation, const RenderingEngine::Color &color, const std::vector< ShapesEngine::↩ Vertex > &vertices, const std::vector< ShapesEngine::Triangle > &triangles)

  Iniitalizes a rigid sphere that can be used to do physics simulations.
- float GetRadius () const

  Returns the radius of the sphere.
- void SetRadius (float radius)

  Sets the radius of the sphere.
- const RigidBody & GetRigidBody () const

  Returns the RigidBody object.
- RigidBody & GetRigidBody ()

  Returns the RigidBody object.
- const ShapesEngine::ThreeDimensionalShape & GetShape () const

  Returns the ThreeDimensionalShape object.
- ShapesEngine::ThreeDimensionalShape & GetShape ()

  Returns the ThreeDimensionalShape object.
- void SetPosition (const vec3 &position)

  Sets the position of the RigidSphere.
- void UpdateModelMatrix ()

  Updates the model matrix of the RigidSphere.
- float Volume ()

  Returns the volume of the sphere.

5.7.1 Constructor & Destructor Documentation

5.7.1.1 RigidSphere()

```
PhysicsEngine::RigidSphere::RigidSphere ( )
```

Default Constructor. Constructs a RigidSphere object.

5.7.2 Member Function Documentation

5.7.2.1 GetRadius()

```
float PhysicsEngine::RigidSphere::GetRadius ( ) const
```

Returns the radius of the sphere.

5.7.2.2 GetRigidBody() [1/2]

```
RigidBody & PhysicsEngine::RigidSphere::GetRigidBody ( )
```

Returns the RigidBody object.

5.7.2.3 GetRigidBody() [2/2]

```
const RigidBody & PhysicsEngine::RigidSphere::GetRigidBody ( ) const
```

Returns the RigidBody object.

5.7.2.4 GetShape() [1/2]

```
ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidSphere::GetShape ( )
```

Returns the ThreeDimensionalShape object.

5.7.2.5 GetShape() [2/2]

```
const ShapesEngine::ThreeDimensionalShape & PhysicsEngine::RigidSphere::GetShape ( ) const
```

Returns the ThreeDimensionalShape object.

5.7.2.6 InitializeRigidSphere()

```
void PhysicsEngine::RigidSphere::InitializeRigidSphere (
            float radius,
            float massDensity,
            const vec3 & initialPosition,
            const MathEngine::Quaternion & initialOrientation,
            const RenderingEngine::Color & color,
            const std::vector< ShapesEngine::Vertex > & vertices,
            const std::vector< ShapesEngine::Triangle > & triangles )
```

Iniitalizes a rigid sphere that can be used to do physics simulations.

Parameters

in	*radius*	The radius of the sphere.
in	*color*	The color of the sphere.
in	*massDensity*	The mass density of the sphere.
in	*initialPosition*	The initial position of the sphere.
in	*initialOrientation*	The initial orientation of the sphere.
in	*color*	The color of the sphere.
in	*vertices*	The vertex list of a unit sphere.
in	*triangles*	The triangle list of a unit sphere.

5.7.2.7 SetPosition()

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

Sets the position of the RigidSphere.

5.7.2.8 SetRadius()

```
void PhysicsEngine::RigidSphere::SetRadius (
            float radius )
```

Sets the radius of the sphere.

5.7.2.9 UpdateModelMatrix()

```
void PhysicsEngine::RigidSphere::UpdateModelMatrix ( )
```

Updates the model matrix of the RigidSphere.

5.7.2.10 Volume()

```
float PhysicsEngine::RigidSphere::Volume ( )
```

Returns the volume of the sphere.

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

- RigidSphere.h

Chapter 6

File Documentation

6.1 AABB.h

```
1 #pragma once
2
3 #include "MathEngine.h"
4 #include "Vertex.h"
5 #include <vector>
6
7 namespace PhysicsEngine
8 {
12      struct AABB
13      {
14          vec3 min;
15          vec3 max;
16      };
17
21      AABB CreateAABB(const std::vector<ShapesEngine::Vertex>& vertexList);
22
23 }
```

6.2 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 dragCoefficent, const vec3& velocity);
18
23      vec3 ApplyForce(float magnitdue, const vec3& direction);
24 }
```

6.3 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 }
```

6.4 RigidBody.h

```
1 #pragma once
2
3 #include "PolyhedralMassProperties.h"
4 #include <vector>
5
9 namespace PhysicsEngine
10 {
11     class RigidBody
12     {
13     public:
17         RigidBody();
18
28         void InitializeRigidBody(float massDensity, const MathEngine::Quaternion& initialOrientation,
      const std::vector<ShapesEngine::Triangle>& triangles);
29
32         float GetMass() const;
33
38         float GetInverseMass() const;
39
42         const mat3& GetBodyInertiaTensor() const;
43
46         const mat3& GetInverseBodyInertiaTensor() const;
47
50         const vec3& GetCenterOfMass() const;
51
54         const vec3& GetLinearVelocity() const;
55
58         const vec3& GetLinearMomentum() const;
59
62         const MathEngine::Quaternion& GetOrientation() const;
63
66         const vec3& GetAngularVelocity() const;
67
70         const vec3& GetAngularMomentum() const;
71
74         const vec3& GetNetForce() const;
75
78         const vec3& GetNetTorque() const;
79
87         void SetMass(float mass);
88
91         void SetCenterOfMass(const vec3& centerOfMass);
92
95         void SetLinearVelocity(const vec3& velocity);
96
99         void SetLinearMomentum(const vec3& linearMomentum);
100
103         void SetBodyInertiaTensor(const mat3& bodyInertia);
104
107         void SetAngularVelocity(const vec3& angularVelocity);
108
111         void SetAngularMomentum(const vec3& angularMomentum);
112
115         void SetOrientation(const MathEngine::Quaternion& orientation);
116
119         void ResetForce();
120
123         void ResetTorque();
124
127         void AddForce(const vec3& force);
128
132         void AddTorque(const vec3& force, const vec3& point);
133
137         void Integrate(float dt);
138
139     private:
140         float mMass;
141         float mInverseMass;
142
143         mat3 mBodyInertiaTensor;
144         mat3 mInverseBodyInertiaTensor;
145         mat3 mWorldCMInertiaTensor;
146         mat3 mInverseWorldCMInertiaTensor;
147
148         vec3 mCenterOfMass;
149         vec3 mLinearVelocity;
150         vec3 mLinearMomentum;
151         vec3 mNetForce;
152
153         MathEngine::Quaternion mOrientation;
154         vec3 mAngularVelocity;
155         vec3 mAngularMomentum;
156         vec3 mNetTorque;
157     };
158
```

```
161     void Interpolate(const RigidBody& r1, const RigidBody& r2, RigidBody& r3, float t);
162 }
```

6.5 RigidBox.h

```
1 #pragma once
2
3 #include "RigidBody.h"
4 #include "ThreeDimensionalShape.h"
5
6 namespace PhysicsEngine
7 {
8     class RigidBox
9
10     {
11     public:
12
16         RigidBox();
17
31         void InitializeRigidBox(float width, float height, float depth, float massDensity,
32             const vec3& initialPosition, const MathEngine::Quaternion& initialOrientation, const
      RenderingEngine::Color& color,
33             const std::vector<ShapesEngine::Vertex>& vertices, const std::vector<ShapesEngine::Triangle>&
      triangles);
34
37         float GetWidth() const;
38
41         float GetHeight() const;
42
45         float GetDepth() const;
46
49         void SetWidth(float width);
50
53         void SetHeight(float height);
54
57         void SetDepth(float depth);
58
61         const RigidBody& GetRigidBody() const;
62
65         RigidBody& GetRigidBody();
66
69         const ShapesEngine::ThreeDimensionalShape& GetShape() const;
70
73         ShapesEngine::ThreeDimensionalShape& GetShape();
74
77         void SetPosition(const vec3& position);
78
81         void UpdateModelMatrix();
82
85         float Volume();
86
87     private:
88         float mWidth;
89         float mHeight;
90         float mDepth;
91
92         RigidBody mRigidBody;
93         ShapesEngine::ThreeDimensionalShape mShape;
94         vec3 mOffset;
95     };
96 }
```

6.6 RigidCone.h

```
1 #pragma once
2
3 #include "RigidBody.h"
4 #include "ThreeDimensionalShape.h"
5
6 namespace PhysicsEngine
7 {
8     class RigidCone
9     {
10     public:
11
15         RigidCone();
16
29         void InitializeRigidCone(float radius, float height, float massDensity,
```

```
30            const vec3& initialPosition, const MathEngine::Quaternion& initialOrientation, const
      RenderingEngine::Color& color,
31            const std::vector<ShapesEngine::Vertex>& vertices, const std::vector<ShapesEngine::Triangle>&
      triangles);
32
35        float GetRadius() const;
36
39        float GetHeight() const;
40
43        void SetRadius(float radius);
44
47        void SetHeight(float height);
48
51        const RigidBody& GetRigidBody() const;
52
55        RigidBody& GetRigidBody();
56
59        const ShapesEngine::ThreeDimensionalShape& GetShape() const;
60
63        ShapesEngine::ThreeDimensionalShape& GetShape();
64
67        void SetPosition(const vec3& position);
68
71        void UpdateModelMatrix();
72
75        float Volume();
76
77    private:
78        float mRadius;
79        float mHeight;
80
81        RigidBody mRigidBody;
82        ShapesEngine::ThreeDimensionalShape mShape;
83        vec3 mOffset;
84    };
85 }
```

6.7 RigidCylinder.h

```
1 #pragma once
2
3 #include "RigidBody.h"
4 #include "ThreeDimensionalShape.h"
5
6 namespace PhysicsEngine
7 {
8     class RigidCylinder
9     {
10    public:
11
15        RigidCylinder();
16
29        void InitializeRigidCylinder(float radius, float height, float massDensity,
30            const vec3& initialPosition, const MathEngine::Quaternion& initialOrientation, const
      RenderingEngine::Color& color,
31            const std::vector<ShapesEngine::Vertex>& vertices, const std::vector<ShapesEngine::Triangle>&
      triangles);
32
35        float GetRadius() const;
36
39        float GetHeight() const;
40
43        void SetRadius(float radius);
44
47        void SetHeight(float height);
48
51        const RigidBody& GetRigidBody() const;
52
55        RigidBody& GetRigidBody();
56
59        const ShapesEngine::ThreeDimensionalShape& GetShape() const;
60
63        ShapesEngine::ThreeDimensionalShape& GetShape();
64
67        void SetPosition(const vec3& position);
68
71        void UpdateModelMatrix();
72
75        float Volume();
76
77    private:
78        float mRadius;
79        float mHeight;
```

```
80
81          RigidBody mRigidBody;
82          ShapesEngine::ThreeDimensionalShape mShape;
83          vec3 mOffset;
84      };
85  }
```

6.8 RigidPyramid.h

```
1  #pragma once
2
3  #include "RigidBody.h"
4  #include "ThreeDimensionalShape.h"
5
6  namespace PhysicsEngine
7  {
8      class RigidPyramid
9
10      {
11      public:
12
16          RigidPyramid();
17
31          void InitializeRigidPyramid(float width, float height, float depth, float massDensity,
32              const vec3& initialPosition, const MathEngine::Quaternion& initialOrientation, const
       RenderingEngine::Color& color,
33              const std::vector<ShapesEngine::Vertex>& vertices, const std::vector<ShapesEngine::Triangle>&
       triangles);
34
37          float GetWidth() const;
38
41          float GetHeight() const;
42
45          float GetDepth() const;
46
49          void SetWidth(float width);
50
53          void SetHeight(float height);
54
57          void SetDepth(float depth);
58
61          const RigidBody& GetRigidBody() const;
62
65          RigidBody& GetRigidBody();
66
69          const ShapesEngine::ThreeDimensionalShape& GetShape() const;
70
73          ShapesEngine::ThreeDimensionalShape& GetShape();
74
77          void SetPosition(const vec3& position);
78
81          void UpdateModelMatrix();
82
85          float Volume();
86
87      private:
88          float mWidth;
89          float mHeight;
90          float mDepth;
91
92          RigidBody mRigidBody;
93          ShapesEngine::ThreeDimensionalShape mShape;
94          vec3 mOffset;
95      };
96  }
```

6.9 RigidSphere.h

```
1  #pragma once
2
3  #include "RigidBody.h"
4  #include "ThreeDimensionalShape.h"
5
6  namespace PhysicsEngine
7  {
8      class RigidSphere
9      {
10      public:
11
```

```
15          RigidSphere();
16
28          void InitializeRigidSphere(float radius, float massDensity,
29              const vec3& initialPosition, const MathEngine::Quaternion& initialOrientation, const
    RenderingEngine::Color& color,
30              const std::vector<ShapesEngine::Vertex>& vertices, const std::vector<ShapesEngine::Triangle>&
    triangles);
31
34          float GetRadius() const;
37          void SetRadius(float radius);
38
41          const RigidBody& GetRigidBody() const;
42
45          RigidBody& GetRigidBody();
46
49          const ShapesEngine::ThreeDimensionalShape& GetShape() const;
50
53          ShapesEngine::ThreeDimensionalShape& GetShape();
54
57          void SetPosition(const vec3& position);
58
61          void UpdateModelMatrix();
62
65          float Volume();
66
67      private:
68          float mRadius;
69
70          RigidBody mRigidBody;
71          ShapesEngine::ThreeDimensionalShape mShape;
72          vec3 mOffset;
73      };
74  }
```

Index