

Simulating Guidewires in Blood Vessels using Cosserat Rod Theory

Generated by Doxygen 1.9.5

1 Requirements Traceability	1
2 Namespace Index	5
2.1 Package List	5
3 Hierarchical Index	7
3.1 Class Hierarchy	7
4 Class Index	9
4.1 Class List	9
5 File Index	11
5.1 File List	11
6 Namespace Documentation	13
6.1 GuidewireSim Namespace Reference	13
7 Class Documentation	15
7.1 GuidewireSim.CollisionDetectionPrimitive Class Reference	15
7.1.1 Detailed Description	15
7.1.2 Member Function Documentation	16
7.1.2.1 AssignSphereID()	16
7.1.2.2 Awake()	16
7.1.2.3 OnCollisionEnter()	16
7.1.2.4 OnCollisionStay()	16
7.1.2.5 Start()	16
7.1.3 Member Data Documentation	16
7.1.3.1 collisionHandler	16
7.1.3.2 simulationLoop	17
7.1.3.3 sphereID	17
7.2 GuidewireSim.CollisionDetectionVessel Class Reference	17
7.2.1 Detailed Description	18
7.2.2 Member Function Documentation	18
7.2.2.1 Awake()	18
7.2.2.2 YieldSphereID()	18
7.2.3 Member Data Documentation	18
7.2.3.1 collisionHandler	18
7.2.3.2 simulationLoop	18
7.3 GuidewireSim.CollisionHandler Class Reference	19
7.3.1 Detailed Description	19
7.3.2 Member Function Documentation	19
7.3.2.1 RegisterCollision()	19
7.3.2.2 ResetRegisteredCollisions()	20
7.3.2.3 SetCollidersToPredictions()	20

7.3.2.4 Start()	20
7.3.3 Member Data Documentation	20
7.3.3.1 registeredCollisions	21
7.3.3.2 sphereColliders	21
7.3.3.3 sphereRadius	21
7.4 GuidewireSim.CollisionPair Struct Reference	21
7.4.1 Detailed Description	22
7.4.2 Constructor & Destructor Documentation	22
7.4.2.1 CollisionPair()	22
7.4.3 Member Data Documentation	22
7.4.3.1 collisionNormal	22
7.4.3.2 contactPoint	22
7.4.3.3 sphere	22
7.4.3.4 sphereID	23
7.5 GuidewireSim.CollisionSolvingStep Class Reference	23
7.5.1 Detailed Description	24
7.5.2 Member Function Documentation	24
7.5.2.1 Awake()	24
7.5.2.2 CalculateDeltaPosition()	24
7.5.2.3 CorrectCollisionPredictions()	24
7.5.2.4 DrawCollisionInformation()	25
7.5.2.5 SolveCollisionConstraint()	25
7.5.2.6 SolveCollisionConstraints()	26
7.5.3 Member Data Documentation	26
7.5.3.1 collisionHandler	26
7.5.3.2 collisionMargin	26
7.5.3.3 collisionStiffness	27
7.5.3.4 deltaPosition	27
7.5.3.5 initialPositionPrediction	27
7.5.3.6 mathHelper	27
7.5.3.7 sphereRadius	27
7.6 GuidewireSim.CollisionTestPerformer Class Reference	27
7.6.1 Member Function Documentation	28
7.6.1.1 Awake()	28
7.6.1.2 PerformCollisionTestFour()	28
7.6.1.3 PerformCollisionTestOne()	28
7.6.1.4 PerformCollisionTests()	29
7.6.1.5 PerformCollisionTestThree()	29
7.6.1.6 PerformCollisionTestTwo()	29
7.6.1.7 Start()	29
7.6.2 Member Data Documentation	29
7.6.2.1 doCollisionTestFour	29

7.6.2.2 doCollisionTestOne	29
7.6.2.3 doCollisionTestThree	29
7.6.2.4 doCollisionTestTwo	30
7.6.2.5 pullForce	30
7.6.2.6 simulationLoop	30
7.6.2.7 startTime	30
7.7 GuidewireSim.ConstraintSolvingStep Class Reference	30
7.7.1 Detailed Description	31
7.7.2 Member Function Documentation	31
7.7.2.1 Awake()	31
7.7.2.2 CorrectBendTwistPredictions()	32
7.7.2.3 CorrectStretchPredictions()	32
7.7.2.4 SolveBendTwistConstraint()	33
7.7.2.5 SolveBendTwistConstraints()	34
7.7.2.6 SolveBendTwistConstraintsInBilateralOrder()	34
7.7.2.7 SolveBendTwistConstraintsInNaiveOrder()	35
7.7.2.8 SolveStretchConstraint()	36
7.7.2.9 SolveStretchConstraints()	37
7.7.2.10 SolveStretchConstraintsInBilateralOrder()	37
7.7.2.11 SolveStretchConstraintsInNaiveOrder()	38
7.7.3 Member Data Documentation	39
7.7.3.1 bendStiffness	39
7.7.3.2 deltaOrientation	39
7.7.3.3 deltaOrientationOne	39
7.7.3.4 deltaOrientationTwo	39
7.7.3.5 deltaPositionOne	39
7.7.3.6 deltaPositionTwo	40
7.7.3.7 executeInBilateralOrder	40
7.7.3.8 mathHelper	40
7.7.3.9 stretchStiffness	40
7.8 DebugExtension Class Reference	40
7.8.1 Detailed Description	42
7.8.2 Member Function Documentation	42
7.8.2.1 DebugArrow() [1/2]	42
7.8.2.2 DebugArrow() [2/2]	43
7.8.2.3 DebugBounds() [1/2]	43
7.8.2.4 DebugBounds() [2/2]	44
7.8.2.5 DebugCapsule() [1/2]	44
7.8.2.6 DebugCapsule() [2/2]	45
7.8.2.7 DebugCircle() [1/4]	46
7.8.2.8 DebugCircle() [2/4]	47
7.8.2.9 DebugCircle() [3/4]	47

7.8.2.10 DebugCircle()	[4 / 4]	48
7.8.2.11 DebugCone()	[1 / 4]	49
7.8.2.12 DebugCone()	[2 / 4]	49
7.8.2.13 DebugCone()	[3 / 4]	50
7.8.2.14 DebugCone()	[4 / 4]	51
7.8.2.15 DebugCylinder()	[1 / 2]	51
7.8.2.16 DebugCylinder()	[2 / 2]	52
7.8.2.17 DebugLocalCube()	[1 / 4]	53
7.8.2.18 DebugLocalCube()	[2 / 4]	54
7.8.2.19 DebugLocalCube()	[3 / 4]	54
7.8.2.20 DebugLocalCube()	[4 / 4]	55
7.8.2.21 DebugPoint()	[1 / 2]	56
7.8.2.22 DebugPoint()	[2 / 2]	56
7.8.2.23 DebugWireSphere()	[1 / 2]	58
7.8.2.24 DebugWireSphere()	[2 / 2]	59
7.8.2.25 DrawArrow()	[1 / 2]	59
7.8.2.26 DrawArrow()	[2 / 2]	60
7.8.2.27 DrawBounds()	[1 / 2]	60
7.8.2.28 DrawBounds()	[2 / 2]	61
7.8.2.29 DrawCapsule()	[1 / 2]	61
7.8.2.30 DrawCapsule()	[2 / 2]	62
7.8.2.31 DrawCircle()	[1 / 4]	62
7.8.2.32 DrawCircle()	[2 / 4]	63
7.8.2.33 DrawCircle()	[3 / 4]	63
7.8.2.34 DrawCircle()	[4 / 4]	64
7.8.2.35 DrawCone()	[1 / 4]	64
7.8.2.36 DrawCone()	[2 / 4]	65
7.8.2.37 DrawCone()	[3 / 4]	65
7.8.2.38 DrawCone()	[4 / 4]	66
7.8.2.39 DrawCylinder()	[1 / 2]	66
7.8.2.40 DrawCylinder()	[2 / 2]	67
7.8.2.41 DrawLocalCube()	[1 / 4]	68
7.8.2.42 DrawLocalCube()	[2 / 4]	68
7.8.2.43 DrawLocalCube()	[3 / 4]	69
7.8.2.44 DrawLocalCube()	[4 / 4]	69
7.8.2.45 DrawPoint()	[1 / 2]	70
7.8.2.46 DrawPoint()	[2 / 2]	70
7.8.2.47 MethodsOfObject()		71
7.8.2.48 MethodsOfType()		71
7.9 GuidewireSim.DirectorsDrawer Class Reference		72
7.9.1 Detailed Description		73
7.9.2 Member Function Documentation		73

7.9.2.1 Awake()	73
7.9.2.2 CalculateArrowHeadPositions()	73
7.9.2.3 DrawArrowHeadConnectionLines()	74
7.9.2.4 DrawArrowHeadLines()	74
7.9.2.5 DrawDirectors()	74
7.9.2.6 Update()	75
7.9.3 Member Data Documentation	75
7.9.3.1 arrowHeadAngle	75
7.9.3.2 arrowHeadPercentage	75
7.9.3.3 directorColors	75
7.9.3.4 directorOneColor	75
7.9.3.5 directorThreeColor	76
7.9.3.6 directorTwoColor	76
7.9.3.7 scaleFactor	76
7.9.3.8 simulationLoop	76
7.10 GuidewireSim.ForceTestPerformer Class Reference	76
7.10.1 Detailed Description	77
7.10.2 Member Function Documentation	77
7.10.2.1 Awake()	77
7.10.2.2 PerformForceTestFour()	77
7.10.2.3 PerformForceTestOne()	78
7.10.2.4 PerformForceTests()	78
7.10.2.5 PerformForceTestThree()	78
7.10.2.6 PerformForceTestTwo()	78
7.10.2.7 PerformSingleLoopTest()	78
7.10.2.8 Start()	79
7.10.3 Member Data Documentation	79
7.10.3.1 doForceTestFour	79
7.10.3.2 doForceTestOne	79
7.10.3.3 doForceTestThree	79
7.10.3.4 doForceTestTwo	79
7.10.3.5 doSingleLoopTest	79
7.10.3.6 pullForceTestThree	80
7.10.3.7 simulationLoop	80
7.11 GuidewireSim.InitializationStep Class Reference	80
7.11.1 Detailed Description	81
7.11.2 Member Function Documentation	81
7.11.2.1 Awake()	81
7.11.2.2 InitCylinderAngularVelocities()	81
7.11.2.3 InitCylinderExternalTorques()	82
7.11.2.4 InitCylinderOrientationPredictions()	82
7.11.2.5 InitCylinderOrientations()	82

7.11.2.6 InitCylinderPositions()	83
7.11.2.7 InitCylinderScalarWeights()	83
7.11.2.8 InitDirectors()	83
7.11.2.9 InitDiscreteRestDarbouxVectors()	84
7.11.2.10 InitInertiaTensor()	84
7.11.2.11 InitInverseInertiaTensor()	85
7.11.2.12 InitSphereColliders()	85
7.11.2.13 InitSphereExternalForces()	85
7.11.2.14 InitSphereInverseMasses()	85
7.11.2.15 InitSpherePositionPredictions()	86
7.11.2.16 InitSpherePositions()	86
7.11.2.17 InitSphereVelocities()	87
7.11.2.18 InitWorldSpaceBasis()	87
7.11.3 Member Data Documentation	87
7.11.3.1 collisionHandler	87
7.11.3.2 materialDensity	87
7.11.3.3 materialRadius	88
7.11.3.4 mathHelper	88
7.12 GuidewireSim.MathHelper Class Reference	88
7.12.1 Detailed Description	89
7.12.2 Member Function Documentation	89
7.12.2.1 BendTwistConstraintDeviation()	89
7.12.2.2 CalculateCylinderPositions()	89
7.12.2.3 ColumnVectorMatrixMultiplication()	91
7.12.2.4 DarbouxSignFactor()	91
7.12.2.5 DiscreteDarbouxVector()	92
7.12.2.6 EmbeddedVector()	92
7.12.2.7 GetGaussianRandomNumber()	93
7.12.2.8 ImaginaryPart()	93
7.12.2.9 MatrixInverse()	94
7.12.2.10 MatrixVectorMultiplication()	94
7.12.2.11 QuaternionConversionFromBSM()	94
7.12.2.12 QuaternionConversionToBSM()	95
7.12.2.13 QuaternionLength()	95
7.12.2.14 RandomUnitQuaternion()	95
7.12.2.15 RodElementLength()	96
7.12.2.16 RodElementLengthDeviation()	96
7.12.2.17 ScalarMatrixMultiplication()	97
7.12.2.18 SquaredNorm()	97
7.12.2.19 StretchConstraintDeviation()	97
7.12.2.20 UpdateDirectors()	98
7.12.2.21 VectorColumnVectorMultiplication()	99

7.12.2.22 VectorLength()	99
7.13 GuidewireSim.ObjectSetter Class Reference	99
7.13.1 Detailed Description	100
7.13.2 Member Function Documentation	100
7.13.2.1 Awake()	100
7.13.2.2 SetCylinderOrientations()	100
7.13.2.3 SetCylinderPositions()	101
7.13.2.4 SetSpherePositions()	101
7.13.3 Member Data Documentation	102
7.13.3.1 mathHelper	102
7.14 GuidewireSim.PredictionStep Class Reference	102
7.14.1 Detailed Description	102
7.14.2 Member Function Documentation	103
7.14.2.1 Awake()	103
7.14.2.2 PredictAngularVelocities()	103
7.14.2.3 PredictCylinderOrientations()	103
7.14.2.4 PredictSpherePositions()	104
7.14.2.5 PredictSphereVelocities()	104
7.14.3 Member Data Documentation	105
7.14.3.1 mathHelper	105
7.15 GuidewireSim.SimulationLoop Class Reference	105
7.15.1 Detailed Description	107
7.15.2 Member Function Documentation	107
7.15.2.1 AdaptCalculations()	108
7.15.2.2 Awake()	108
7.15.2.3 FixedUpdate()	108
7.15.2.4 PerformConstraintSolvingStep()	108
7.15.2.5 PerformInitializationStep()	109
7.15.2.6 PerformPredictionStep()	109
7.15.2.7 PerformSimulationLoop()	109
7.15.2.8 PerformUpdateStep()	109
7.15.2.9 SetCollidersStep()	110
7.15.2.10 Start()	110
7.15.3 Member Data Documentation	110
7.15.3.1 collisionHandler	110
7.15.3.2 collisionSolvingStep	110
7.15.3.3 constraintSolverSteps	110
7.15.3.4 constraintSolvingStep	110
7.15.3.5 cylinderAngularVelocities	111
7.15.3.6 cylinderExternalTorques	111
7.15.3.7 cylinderOrientationPredictions	111
7.15.3.8 cylinderOrientations	111

7.15.3.9 cylinderPositions	111
7.15.3.10 cylinders	111
7.15.3.11 cylinderScalarWeights	112
7.15.3.12 directors	112
7.15.3.13 discreteRestDarbouxVectors	112
7.15.3.14 inertiaTensor	112
7.15.3.15 initializationStep	113
7.15.3.16 inverseInertiaTensor	113
7.15.3.17 mathHelper	113
7.15.3.18 objectSetter	113
7.15.3.19 predictionStep	113
7.15.3.20 rodElementLength	113
7.15.3.21 solveBendTwistConstraints	114
7.15.3.22 solveCollisionConstraints	114
7.15.3.23 solveStretchConstraints	114
7.15.3.24 sphereExternalForces	114
7.15.3.25 sphereInverseMasses	114
7.15.3.26 spherePositionPredictions	114
7.15.3.27 spherePositions	115
7.15.3.28 spheres	115
7.15.3.29 sphereVelocities	115
7.15.3.30 timeStep	115
7.15.3.31 updateStep	115
7.15.3.32 worldSpaceBasis	116
7.15.4 Property Documentation	116
7.15.4.1 ConstraintSolverSteps	116
7.15.4.2 CylinderCount	116
7.15.4.3 ExecuteSingleLoopTest	116
7.15.4.4 SpheresCount	116
7.15.4.5 TimeStep	116
7.16 GuidewireSim.StressTestPerformer Class Reference	117
7.16.1 Detailed Description	117
7.16.2 Member Function Documentation	117
7.16.2.1 Awake()	117
7.16.2.2 PerformStressTestOne()	117
7.16.2.3 PerformStressTests()	118
7.16.2.4 Start()	118
7.16.3 Member Data Documentation	118
7.16.3.1 doStressTestOne	118
7.16.3.2 simulationLoop	118
7.17 GuidewireSim.TorqueTestPerformer Class Reference	119
7.17.1 Detailed Description	119

7.17.2 Member Function Documentation	119
7.17.2.1 Awake()	119
7.17.2.2 PerformTorqueTestOne()	119
7.17.2.3 PerformTorqueTests()	120
7.17.2.4 PerformTorqueTestThree()	120
7.17.2.5 PerformTorqueTestTwo()	120
7.17.2.6 Start()	120
7.17.3 Member Data Documentation	121
7.17.3.1 doTorqueTestOne	121
7.17.3.2 doTorqueTestThree	121
7.17.3.3 doTorqueTestTwo	121
7.17.3.4 pullTorque	121
7.17.3.5 simulationLoop	121
7.18 UnitTest_SolveBendTwistConstraint Class Reference	121
7.18.1 Detailed Description	122
7.18.2 Member Function Documentation	122
7.18.2.1 PerformUnitTests()	122
7.18.2.2 Test_SolveBendTwistConstraint()	122
7.18.3 Member Data Documentation	123
7.18.3.1 constraintSolverSteps	123
7.18.3.2 sampleSize	123
7.19 UnitTest_SolveStretchConstraint Class Reference	123
7.19.1 Detailed Description	124
7.19.2 Member Function Documentation	124
7.19.2.1 PerformUnitTests()	124
7.19.2.2 PickRandomPositions()	124
7.19.2.3 Test_SolveStretchConstraint()	125
7.19.3 Member Data Documentation	126
7.19.3.1 constraintSolverSteps	126
7.19.3.2 maximalDistanceOffset	126
7.19.3.3 rodElementLength	126
7.19.3.4 sampleSize	126
7.20 GuidewireSim.UpdateStep Class Reference	127
7.20.1 Detailed Description	127
7.20.2 Member Function Documentation	127
7.20.2.1 Awake()	127
7.20.2.2 UpdateCylinderAngularVelocities()	127
7.20.2.3 UpdateCylinderOrientations()	128
7.20.2.4 UpdateSpherePositions()	128
7.20.2.5 UpdateSphereVelocities()	129
7.20.3 Member Data Documentation	129
7.20.3.1 mathHelper	129

8 File Documentation	131
8.1 CollisionDetectionPrimitive.cs File Reference	131
8.2 CollisionDetectionVessel.cs File Reference	131
8.3 CollisionHandler.cs File Reference	131
8.4 CollisionPair.cs File Reference	132
8.5 CollisionSolvingStep.cs File Reference	132
8.6 CollisionTestPerformer.cs File Reference	132
8.7 ConstraintSolvingStep.cs File Reference	132
8.7.1 Typedef Documentation	133
8.7.1.1 BSM	133
8.8 DebugExtension.cs File Reference	133
8.9 DirectorsDrawer.cs File Reference	133
8.10 ForceTestPerformer.cs File Reference	133
8.11 InitializationStep.cs File Reference	134
8.11.1 Typedef Documentation	134
8.11.1.1 BSM	134
8.12 MathHelper.cs File Reference	134
8.12.1 Typedef Documentation	134
8.12.1.1 BSM	135
8.13 ObjectSetter.cs File Reference	135
8.13.1 Typedef Documentation	135
8.13.1.1 BSM	135
8.14 PredictionStep.cs File Reference	135
8.14.1 Typedef Documentation	136
8.14.1.1 BSM	136
8.15 SimulationLoop.cs File Reference	136
8.15.1 Typedef Documentation	136
8.15.1.1 BSM	136
8.16 StressTestPerformer.cs File Reference	136
8.17 TorqueTestPerformer.cs File Reference	137
8.18 UpdateStep.cs File Reference	137
8.18.1 Typedef Documentation	137
8.18.1.1 BSM	137
8.19 UnitTest_SolveBendTwistConstraint.cs File Reference	137
8.19.1 Typedef Documentation	138
8.19.1.1 BSM	138
8.20 UnitTest_SolveStretchConstraint.cs File Reference	138
8.20.1 Typedef Documentation	138
8.20.1.1 BSM	138
Index	139

Chapter 1

Requirements Traceability

Member [GuidewireSim.ConstraintSolvingStep.CorrectBendTwistPredictions](#) (int cylinderIndex, BSM.Quaternion[] cylinderOrientationPredictions)

The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion predictions got corrected, they should again be unit quaternions, i.e. have length approximately equal to one.

Member [GuidewireSim.ConstraintSolvingStep.CorrectStretchPredictions](#) (int sphereIndex, Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions)

The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion prediction got corrected, it should again be a unit quaternions, i.e. have length approximately equal to one.

Member [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraint](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, Vector3 discreteRestDarbouxVector, float rodElementLength, out BSM.Quaternion deltaOrientationOne, out BSM.Quaternion deltaOrientationTwo, float inertiaWeightOne=1f, float inertiaWeightTwo=1f)

`orientationOne` and `orientationTwo` should be unit quaternions, i.e. have length approximately equal to one.

`rodElementLength` should be positive.

`inertiaWeightOne` and `inertiaWeightTwo` should be values between 0 and 1.

Member [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraints](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

Member [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInBilateralOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

Member [GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInNaiveOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)

`cylinderCount` should be at least one.

`rodElementLength` should be positive.

Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraint](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e_3, float rodElementLength, out Vector3 deltaPositionOne, out Vector3 deltaPositionTwo, out BSM.Quaternion deltaOrientation, float inverseMassOne=1f, float inverseMassTwo=1f, float inertiaWeight=1f)

orientation should be a unit quaternions, i.e. have length approximately equal to one.

e_3 should be a unit quaternions, i.e. have length approximately equal to one.

rodElementLength should be positive.

inverseMassOne, inverseMassTwo and inertiaWeight should be values between 0 and 1.

Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraints](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, BSM.Quaternion[] worldSpaceBasis, float rodElementLength)

spheresCount should be at least one.

rodElementLength should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInBilateralOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e_3)

spheresCount should be at least one.

rodElementLength should be positive.

Member [GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInNaiveOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e_3)

spheresCount should be at least one.

rodElementLength should be positive.

Member [GuidewireSim.DirectorsDrawer.CalculateArrowHeadPositions](#) (Vector3 startPosition, Vector3 endPosition)

arrowHeadPositions has a length of 4.

Member [GuidewireSim.DirectorsDrawer.DrawArrowHeadLines](#) (int directorIndex, Vector3 endPosition, Vector3[] arrowHeadPositions)

arrowHeadPositions has a length of 4.

Member [GuidewireSim.InitializationStep.InitDiscreteRestDarbouxVectors](#) (int cylinderCount, BSM.Quaternion[] cylinderOrientations, out Vector3[] discreteRestDarbouxVectors, float rodElementLength)

cylinderCount should be at least one.

rodElementLength should be positive.

Member [GuidewireSim.InitializationStep.InitSpherePositions](#) (GameObject[] spheres, int spheresCount, out Vector3[] spherePositions)

spheresCount should be at least one.

Member [GuidewireSim.MathHelper.ColumnVectorMatrixMultiplication](#) (Vector3 columnVector, float[,] matrix)

matrix must be a 3×3 matrix.

Member [GuidewireSim.MathHelper.MatrixVectorMultiplication](#) (float[,] matrix, Vector3 vector)

matrix must be a 3×3 matrix.

Member [GuidewireSim.MathHelper.RandomUnitQuaternion](#) ()

The length of the drawn quaternion is approximately equal to one.

Member [GuidewireSim.SimulationLoop.AdaptCalculations](#) ()

Sets the positions of the GameObjects [spheres](#) to [spherePositions](#).

Calculates [cylinderPositions](#) based on [spherePositions](#).

Sets the positions of the GameObjects [cylinders](#) to [cylinderPositions](#).

Sets the rotations of the GameObjects [cylinders](#) to [cylinderOrientations](#).

Member `GuidewireSim.SimulationLoop.FixedUpdate ()`

Execute the simulation loop if and only if `ExecuteSingleLoopTest` is false.

Member `GuidewireSim.SimulationLoop.PerformConstraintSolvingStep ()`

Performs the constraint solving of every constraint `#solverStep` many times.

Solve stretch constraints, if and only if `solveStretchConstraints` is true.

Solve bend twist constraints, if and only if `solveBendTwistConstraints` is true.

If `solveStretchConstraints`, then `SpheresCount` is at least two.

If `solveStretchConstraints`, then `CylinderCount` is at least one.

If `solveBendTwistConstraints`, then `SpheresCount` is at least three.

If `solveBendTwistConstraints`, then `CylinderCount` is at least two.

If `solveStretchConstraints`, after the step is complete the deviation between the actual rod element length and the default (rest state) `rodElementLength` should be close to zero.

If `solveStretchConstraints`, after the step is complete the deviation of the stretch constraint to zero should be close to zero.

If `solveCollisionConstraints`, after this step is complete clear the list `registeredCollisions`, since these collisions are now resolved.

Member `GuidewireSim.SimulationLoop.PerformInitializationStep ()`

Set `SpheresCount` to the length of `spheres`.

Set `CylinderCount` to the length of `cylinders`.

Call every init method of `initializationStep`.

Member `GuidewireSim.SimulationLoop.PerformPredictionStep ()`

Predict the `sphereVelocities`.

Predict the `spherePositionPredictions`.

Predict the `cylinderAngularVelocities`.

Predict the `cylinderOrientationPredictions`.

Member `GuidewireSim.SimulationLoop.PerformUpdateStep ()`

Update `sphereVelocities`.

Update `spherePositions`.

Update `cylinderAngularVelocities`.

Update `cylinderOrientations`.

Update `directors`.

Member `GuidewireSim.StressTestPerformer.PerformStressTestOne (float applyForceTime=1f)`

Output a log message when no further forces are applied to the guidewire.

Member `GuidewireSim.TorqueTestPerformer.PerformTorqueTestThree (Vector3 pullTorque, float applyTorqueTime=10f)`

Output a log message when no further torques are applied to the guidewire.

Member `GuidewireSim.TorqueTestPerformer.PerformTorqueTestTwo (Vector3 pullTorque, float applyTorqueTime=1f)`

Output a log message when no further torques are applied to the guidewire.

Member `Unit_Test_SolveBendTwistConstraint.Test_SolveBendTwistConstraint (int iterations, BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, GuidewireSim.MathHelper mathHelper, GuidewireSim.ConstraintSolvingStep constraintSolvingStep)`

`orientationOne` and `orientationTwo` are still unit quaternions at the end of the test.

The deviation between the bend twist constraint and zero is lower than a reasonable tolerance, i.e. close to zero., which means that the algorithm of `SolveBendTwistConstraint()` converges towards the fulfillment of the bend twist constraint.

Member **UnitTest_SolveStretchConstraint.PickRandomPositions** (out Vector3 particlePositionOne, out Vector3 particlePositionTwo)

Picks the first particle position uniformly distributed so that $x, y, z \in [-5, 5]$.

Picks a distance between the two particles that is uniformly distributed in the interval $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$.

Picks the second particle position uniformly distributed on the surface of the sphere with center particlePositionOne and radius startDistance.

Member **UnitTest_SolveStretchConstraint.Test_SolveStretchConstraint** (int iterations, Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, GuidewireSim.MathHelper mathHelper, GuidewireSim.ConstraintSolvingStep constraintSolvingStep)

orientation is still a unit quaternion at the end of the test.

The deviation between the stretch constraint and zero is lower than the tolerance 0.1, which means that the algorithm of SolveStretchConstraint() converges towards the fulfillment of the stretch constraint.

The deviation between the actual distance of particlePositionOne and particlePositionTwo and the rest rod element length is lower than a reasonable tolerance, i.e. close to zero.

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

GuidewireSim	13
--	----

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

GuidewireSim.CollisionPair	21
DebugExtension	40
MonoBehaviour	
GuidewireSim.CollisionDetectionPrimitive	15
GuidewireSim.CollisionDetectionVessel	17
GuidewireSim.CollisionHandler	19
GuidewireSim.CollisionSolvingStep	23
GuidewireSim.CollisionTestPerformer	27
GuidewireSim.ConstraintSolvingStep	30
GuidewireSim.DirectorsDrawer	72
GuidewireSim.ForceTestPerformer	76
GuidewireSim.InitializationStep	80
GuidewireSim.MathHelper	88
GuidewireSim.ObjectSetter	99
GuidewireSim.PredictionStep	102
GuidewireSim.SimulationLoop	105
GuidewireSim.StressTestPerformer	117
GuidewireSim.TorqueTestPerformer	119
GuidewireSim.UpdateStep	127
UnitTest_SolveBendTwistConstraint	121
UnitTest_SolveStretchConstraint	123

Chapter 4

Class Index

4.1 Class List

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

GuidewireSim.CollisionDetectionPrimitive	15
GuidewireSim.CollisionDetectionVessel	17
GuidewireSim.CollisionHandler	19
GuidewireSim.CollisionPair	21
GuidewireSim.CollisionSolvingStep	23
GuidewireSim.CollisionTestPerformer	27
GuidewireSim.ConstraintSolvingStep	30
DebugExtension	
Debug Extension	40
GuidewireSim.DirectorsDrawer	72
GuidewireSim.ForceTestPerformer	76
GuidewireSim.InitializationStep	80
GuidewireSim.MathHelper	88
GuidewireSim.ObjectSetter	99
GuidewireSim.PredictionStep	102
GuidewireSim.SimulationLoop	105
GuidewireSim.StressTestPerformer	117
GuidewireSim.TorqueTestPerformer	119
UnitTest_SolveBendTwistConstraint	121
UnitTest_SolveStretchConstraint	123
GuidewireSim.UpdateStep	127

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

CollisionDetectionPrimitive.cs	131
CollisionDetectionVessel.cs	131
CollisionHandler.cs	131
CollisionPair.cs	132
CollisionSolvingStep.cs	132
CollisionTestPerformer.cs	132
ConstraintSolvingStep.cs	132
DebugExtension.cs	133
DirectorsDrawer.cs	133
ForceTestPerformer.cs	133
InitializationStep.cs	134
MathHelper.cs	134
ObjectSetter.cs	135
PredictionStep.cs	135
SimulationLoop.cs	136
StressTestPerformer.cs	136
TorqueTestPerformer.cs	137
UpdateStep.cs	137
UnitTest_SolveBendTwistConstraint.cs	137
UnitTest_SolveStretchConstraint.cs	138

Chapter 6

Namespace Documentation

6.1 GuidewireSim Namespace Reference

Classes

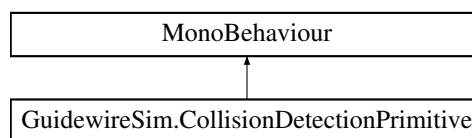
- class [CollisionDetectionPrimitive](#)
- class [CollisionDetectionVessel](#)
- class [CollisionHandler](#)
- struct [CollisionPair](#)
- class [CollisionSolvingStep](#)
- class [CollisionTestPerformer](#)
- class [ConstraintSolvingStep](#)
- class [DirectorsDrawer](#)
- class [ForceTestPerformer](#)
- class [InitializationStep](#)
- class [MathHelper](#)
- class [ObjectSetter](#)
- class [PredictionStep](#)
- class [SimulationLoop](#)
- class [StressTestPerformer](#)
- class [TorqueTestPerformer](#)
- class [UpdateStep](#)

Chapter 7

Class Documentation

7.1 GuidewireSim.CollisionDetectionPrimitive Class Reference

Inheritance diagram for GuidewireSim.CollisionDetectionPrimitive:



Public Attributes

- int [sphereID](#)

Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [AssignSphereID](#) ()
- void [OnCollisionEnter](#) (Collision other)
- void [OnCollisionStay](#) (Collision other)

Private Attributes

- [SimulationLoop](#) [simulationLoop](#)
The [SimulationLoop](#) component in the Simulation GameObject.
- [CollisionHandler](#) [collisionHandler](#)
The [CollisionHandler](#) component in the Simulation GameObject.

7.1.1 Detailed Description

This class is responsible for tracking collisions of the object it is attached to. Attach this component only to sphere objects of the guidewire.

7.1.2 Member Function Documentation

7.1.2.1 AssignSphereID()

```
void GuidewireSim.CollisionDetectionPrimitive.AssignSphereID ( ) [private]
```

Assigns the unique ID of the object sphere it is attached to to [sphereID](#).

7.1.2.2 Awake()

```
void GuidewireSim.CollisionDetectionPrimitive.Awake ( ) [private]
```

7.1.2.3 OnCollisionEnter()

```
void GuidewireSim.CollisionDetectionPrimitive.OnCollisionEnter (
    Collision other ) [private]
```

Registers a collision that Unity's collision detection detected.

7.1.2.4 OnCollisionStay()

```
void GuidewireSim.CollisionDetectionPrimitive.OnCollisionStay (
    Collision other ) [private]
```

Registers a collision that Unity's collision detection detected.

7.1.2.5 Start()

```
void GuidewireSim.CollisionDetectionPrimitive.Start ( ) [private]
```

7.1.3 Member Data Documentation

7.1.3.1 collisionHandler

```
CollisionHandler GuidewireSim.CollisionDetectionPrimitive.collisionHandler [private]
```

The [CollisionHandler](#) component in the Simulation GameObject.

7.1.3.2 simulationLoop

`SimulationLoop` GuidewireSim.CollisionDetectionPrimitive.simulationLoop [private]

The `SimulationLoop` component in the Simulation GameObject.

7.1.3.3 sphereID

`int` GuidewireSim.CollisionDetectionPrimitive.sphereID

The unique ID of the sphere that this component is attached to. Matches the position in spheres in #SimulationLoop.

Note

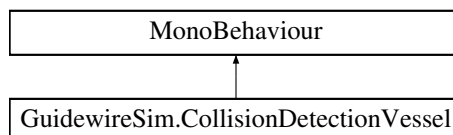
Should also match the position in spherePositions, sphereVelocities, sphereExternalForces, spherePositionPredictions in #SimulationLoop.

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

- [CollisionDetectionPrimitive.cs](#)

7.2 GuidewireSim.CollisionDetectionVessel Class Reference

Inheritance diagram for GuidewireSim.CollisionDetectionVessel:



Private Member Functions

- void [Awake](#) ()
- int [YieldSphereID](#) (Transform sphereTransform)

Private Attributes

- [SimulationLoop](#) simulationLoop
The `SimulationLoop` component in the Simulation GameObject.
- [CollisionHandler](#) collisionHandler
The `CollisionHandler` component in the Simulation GameObject.

7.2.1 Detailed Description

Similarly to [CollisionDetectionPrimitive](#), this class is responsible for tracking collisions of the object it is attached to. Attach this component only to blood vessel objects.

7.2.2 Member Function Documentation

7.2.2.1 Awake()

```
void GuidewireSim.CollisionDetectionVessel.Awake ( ) [private]
```

7.2.2.2 YieldSphereID()

```
int GuidewireSim.CollisionDetectionVessel.YieldSphereID (
    Transform sphereTransform ) [private]
```

7.2.3 Member Data Documentation

7.2.3.1 collisionHandler

```
CollisionHandler GuidewireSim.CollisionDetectionVessel.collisionHandler [private]
```

The [CollisionHandler](#) component in the Simulation GameObject.

7.2.3.2 simulationLoop

```
SimulationLoop GuidewireSim.CollisionDetectionVessel.simulationLoop [private]
```

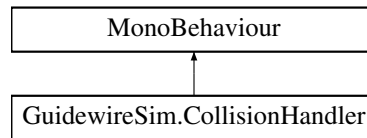
The [SimulationLoop](#) component in the Simulation GameObject.

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

- [CollisionDetectionVessel.cs](#)

7.3 GuidewireSim.CollisionHandler Class Reference

Inheritance diagram for GuidewireSim.CollisionHandler:



Public Member Functions

- void [RegisterCollision](#) (Transform sphere, int sphereID, Vector3 contactPoint, Vector3 collisionNormal)
- void [ResetRegisteredCollisions](#) ()
- void [SetCollidersToPredictions](#) (int spheresCount, Vector3[] spherePositionPredictions, Vector3[] spherePositions)

Public Attributes

- List< [CollisionPair](#) > [registeredCollisions](#)
All collisions that occurred between the last and the current frame in OnTriggerEnter.
- SphereCollider[] [sphereColliders](#)

Private Member Functions

- void [Start](#) ()

Private Attributes

- float [sphereRadius](#) = 5f
The radius of the sphere elements of the guidewire.

7.3.1 Detailed Description

This class manages all collisions that should be resolved, i.e. the collisions of the last frame.

7.3.2 Member Function Documentation

7.3.2.1 RegisterCollision()

```

void GuidewireSim.CollisionHandler.RegisterCollision (
    Transform sphere,
    int sphereID,
    Vector3 contactPoint,
    Vector3 collisionNormal )
  
```

Registers a collision by adding it to [registeredCollisions](#).

Parameters

<i>sphere</i>	The sphere of the guidewire that collided.
<i>sphereID</i>	The unique ID of <i>sphere</i> .
<i>contactPoint</i>	The contact point of the collision.
<i>collisionNormal</i>	The normal of the collision.

7.3.2.2 ResetRegisteredCollisions()

```
void GuidewireSim.CollisionHandler.ResetRegisteredCollisions ( )
```

Clears the list of all registered collisions.

7.3.2.3 SetCollidersToPredictions()

```
void GuidewireSim.CollisionHandler.SetCollidersToPredictions (
    int spheresCount,
    Vector3[] spherePositionPredictions,
    Vector3[] spherePositions )
```

Sets the position of the collider of each sphere to the sphere's position prediction.

Note

The position of the collider is implicitly set by setting the colliders center argument.

Parameters

<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>spherePositions</i>	The position at the current frame of each sphere.

7.3.2.4 Start()

```
void GuidewireSim.CollisionHandler.Start ( ) [private]
```

7.3.3 Member Data Documentation

7.3.3.1 registeredCollisions

```
List<CollisionPair> GuidewireSim.CollisionHandler.registeredCollisions
```

All collisions that occurred between the last and the current frame in `OnTriggerEnter`.

7.3.3.2 sphereColliders

```
SphereCollider [] GuidewireSim.CollisionHandler.sphereColliders
```

Each element stores a reference to the `SphereCollider` of the respective element in `spheres` in [SimulationLoop](#).

Example

The second element in this list is the `SphereCollider` corresponding to the sphere `GameObject` that is referenced in the second element of `spheres` in [SimulationLoop](#).

7.3.3.3 sphereRadius

```
float GuidewireSim.CollisionHandler.sphereRadius = 5f [private]
```

The radius of the sphere elements of the guidewire.

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

- [CollisionHandler.cs](#)

7.4 GuidewireSim.CollisionPair Struct Reference

Public Member Functions

- [CollisionPair](#) (Transform [sphere](#), int [sphereID](#), Vector3 [contactPoint](#), Vector3 [collisionNormal](#))

Public Attributes

- Transform [sphere](#)
The sphere object of the guidewire that was part of the collision.
- Vector3 [contactPoint](#)
The contact point of the collision.
- Vector3 [collisionNormal](#)
The normal of the collision.
- int [sphereID](#)
The ID of the sphere object of the guidewire that was part of the collision.

7.4.1 Detailed Description

Carries all information of a collision that occurred.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 CollisionPair()

```
GuidewireSim.CollisionPair.CollisionPair (
    Transform sphere,
    int sphereID,
    Vector3 contactPoint,
    Vector3 collisionNormal )
```

7.4.3 Member Data Documentation

7.4.3.1 collisionNormal

```
Vector3 GuidewireSim.CollisionPair.collisionNormal
```

The normal of the collision.

7.4.3.2 contactPoint

```
Vector3 GuidewireSim.CollisionPair.contactPoint
```

The contact point of the collision.

7.4.3.3 sphere

```
Transform GuidewireSim.CollisionPair.sphere
```

The sphere object of the guidewire that was part of the collision.

7.4.3.4 sphereID

```
int GuidewireSim.CollisionPair.sphereID
```

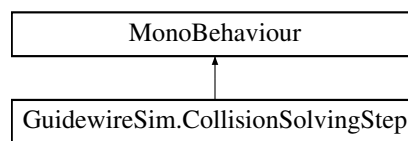
The ID of the sphere object of the guidewire that was part of the collision.

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

- [CollisionPair.cs](#)

7.5 GuidewireSim.CollisionSolvingStep Class Reference

Inheritance diagram for GuidewireSim.CollisionSolvingStep:



Public Member Functions

- void [SolveCollisionConstraints](#) (Vector3[] spherePositionPredictions, int solverStep, int constraintSolverSteps)

Private Member Functions

- void [Awake](#) ()
- void [SolveCollisionConstraint](#) (Vector3 spherePositionPrediction, Vector3 contactPoint, Vector3 collisionNormal, int solverStep, out Vector3 [deltaPosition](#))
- void [DrawCollisionInformation](#) (Vector3 spherePositionPrediction, Vector3 contactPoint, Vector3 collisionNormal)
- Vector3 [CalculateDeltaPosition](#) (Vector3 spherePositionPrediction, Vector3 closestSurfacePoint, Vector3 normalVector)
- void [CorrectCollisionPredictions](#) (int sphereIndex, Vector3[] spherePositionPredictions, int solverStep, int constraintSolverSteps)

Private Attributes

- [MathHelper](#) [mathHelper](#)
The component [MathHelper](#) that provides math related helper functions.
- [CollisionHandler](#) [collisionHandler](#)
The component [CollisionHandler](#) that tracks all collisions.
- Vector3 [deltaPosition](#) = new Vector3()
The correction of [spherePositionPrediction](#) in method [SolveCollisionConstraint\(\)](#).
- Vector3 [initialPositionPrediction](#) = new Vector3()
- float [sphereRadius](#) = 5f
The radius of a sphere of the guidewire.
- float [collisionMargin](#) = 0.1f
- float [collisionStiffness](#) = 0.001f
The collision constraint stiffness parameter.

7.5.1 Detailed Description

This class implements the collision solving that is part of the constraint solving step.

7.5.2 Member Function Documentation

7.5.2.1 Awake()

```
void GuidewireSim.CollisionSolvingStep.Awake ( ) [private]
```

7.5.2.2 CalculateDeltaPosition()

```
Vector3 GuidewireSim.CollisionSolvingStep.CalculateDeltaPosition (
    Vector3 spherePositionPrediction,
    Vector3 closestSurfacePoint,
    Vector3 normalVector ) [private]
```

Calculates the displacement of the collision constraint.

Parameters

<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>closestSurfacePoint</i>	The contact point of the collision.
<i>normalVector</i>	The collision normal.

Returns

The delta position, i.e. the calculated displacement.

7.5.2.3 CorrectCollisionPredictions()

```
void GuidewireSim.CollisionSolvingStep.CorrectCollisionPredictions (
    int sphereIndex,
    Vector3[] spherePositionPredictions,
    int solverStep,
    int constraintSolverSteps ) [private]
```

Corrects the position prediction of the sphere of *sphereIndex* with the calculated displacement.

Parameters

<i>sphereIndex</i>	The sphere ID of the colliding sphere.
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>solverStep</i>	The current iteration of the constraint solving step.
<i>constraintSolverSteps</i>	The total number of solver steps of the constraint solving step.

7.5.2.4 DrawCollisionInformation()

```
void GuidewireSim.CollisionSolvingStep.DrawCollisionInformation (
    Vector3 spherePositionPrediction,
    Vector3 contactPoint,
    Vector3 collisionNormal ) [private]
```

Draws the contact point, collision normal, and displacement corrections into the scene of the collision that occurred.

Parameters

<i>spherePositionPrediction</i>	The position prediction of the sphere that collided.
<i>contactPoint</i>	The contact point of the collision.
<i>collisionNormal</i>	The normal of the collision.

7.5.2.5 SolveCollisionConstraint()

```
void GuidewireSim.CollisionSolvingStep.SolveCollisionConstraint (
    Vector3 spherePositionPrediction,
    Vector3 contactPoint,
    Vector3 collisionNormal,
    int solverStep,
    out Vector3 deltaPosition ) [private]
```

Solves the collision constraint for one collision that occurred this frame.

Parameters

<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>contactPoint</i>	The contact point of the collision.
<i>collisionNormal</i>	The normal of the collision.
<i>solverStep</i>	The current iteration of the constraint solving step.

Attention

Current calculation of the normal only works for spheres.

7.5.2.6 SolveCollisionConstraints()

```
void GuidewireSim.CollisionSolvingStep.SolveCollisionConstraints (
    Vector3[] spherePositionPredictions,
    int solverStep,
    int constraintSolverSteps )
```

Is responsible for executing one iteration of the constraint solving step for the collision constraint of each collision of this frame.

Parameters

<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>solverStep</i>	The current iteration of the constraint solving step.
<i>constraintSolverSteps</i>	The total number of solver steps of the constraint solving step.

7.5.3 Member Data Documentation

7.5.3.1 collisionHandler

```
CollisionHandler GuidewireSim.CollisionSolvingStep.collisionHandler [private]
```

The component [CollisionHandler](#) that tracks all collisions.

7.5.3.2 collisionMargin

```
float GuidewireSim.CollisionSolvingStep.collisionMargin = 0.1f [private]
```

A margin by which a colliding element of the guidewire is set away from the object colliding with in the direction of the normal.

Note

Without this margin, the colliding element of the guidewire (e.g. a sphere) is corrected such that its surface exactly touches the object colliding with, which results in the guidewire still penetrating the object.

7.5.3.3 collisionStiffness

```
float GuidewireSim.CollisionSolvingStep.collisionStiffness = 0.001f [private]
```

The collision constraint stiffness parameter.

7.5.3.4 deltaPosition

```
Vector3 GuidewireSim.CollisionSolvingStep.deltaPosition = new Vector3() [private]
```

The correction of `spherePositionPrediction` in method [SolveCollisionConstraint\(\)](#).

7.5.3.5 initialPositionPrediction

```
Vector3 GuidewireSim.CollisionSolvingStep.initialPositionPrediction = new Vector3() [private]
```

7.5.3.6 mathHelper

```
MathHelper GuidewireSim.CollisionSolvingStep.mathHelper [private]
```

The component [MathHelper](#) that provides math related helper functions.

7.5.3.7 sphereRadius

```
float GuidewireSim.CollisionSolvingStep.sphereRadius = 5f [private]
```

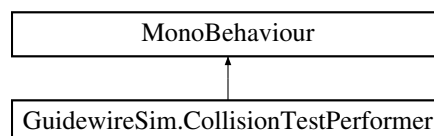
The radius of a sphere of the guidewire.

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

- [CollisionSolvingStep.cs](#)

7.6 GuidewireSim.CollisionTestPerformer Class Reference

Inheritance diagram for GuidewireSim.CollisionTestPerformer:



Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformCollisionTests](#) ()
- void [PerformCollisionTestOne](#) ()
- IEnumerator [PerformCollisionTestTwo](#) (float applyForceTime=1.5f)
- IEnumerator [PerformCollisionTestThree](#) (float exitVelocity=4f)
- IEnumerator [PerformCollisionTestFour](#) (float pullForceFactor=0.3f)

Private Attributes

- [SimulationLoop](#) `simulationLoop`
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- Vector3 [pullForce](#) = new Vector3(0f, 0f, 5f)
External force that is applied in Force Test Three.
- bool [doCollisionTestOne](#) = false
- bool [doCollisionTestTwo](#) = false
- bool [doCollisionTestThree](#) = false
- bool [doCollisionTestFour](#) = false
- float [startTime](#) = 0f

7.6.1 Member Function Documentation

7.6.1.1 Awake()

```
void GuidewireSim.CollisionTestPerformer.Awake ( ) [private]
```

7.6.1.2 PerformCollisionTestFour()

```
IEnumerator GuidewireSim.CollisionTestPerformer.PerformCollisionTestFour (
    float pullForceFactor = 0.3f ) [private]
```

7.6.1.3 PerformCollisionTestOne()

```
void GuidewireSim.CollisionTestPerformer.PerformCollisionTestOne ( ) [private]
```

Performs torque test one. This test applies an external force to one end of the guidewire.

7.6.1.4 PerformCollisionTests()

```
void GuidewireSim.CollisionTestPerformer.PerformCollisionTests ( ) [private]
```

Performs each Torque Test whose respective serialized boolean is set to true in the Unity inspector.

7.6.1.5 PerformCollisionTestThree()

```
IEnumerator GuidewireSim.CollisionTestPerformer.PerformCollisionTestThree (
    float exitVelocity = 4f ) [private]
```

7.6.1.6 PerformCollisionTestTwo()

```
IEnumerator GuidewireSim.CollisionTestPerformer.PerformCollisionTestTwo (
    float applyForceTime = 1.5f ) [private]
```

7.6.1.7 Start()

```
void GuidewireSim.CollisionTestPerformer.Start ( ) [private]
```

7.6.2 Member Data Documentation

7.6.2.1 doCollisionTestFour

```
bool GuidewireSim.CollisionTestPerformer.doCollisionTestFour = false [private]
```

7.6.2.2 doCollisionTestOne

```
bool GuidewireSim.CollisionTestPerformer.doCollisionTestOne = false [private]
```

7.6.2.3 doCollisionTestThree

```
bool GuidewireSim.CollisionTestPerformer.doCollisionTestThree = false [private]
```

7.6.2.4 doCollisionTestTwo

```
bool GuidewireSim.CollisionTestPerformer.doCollisionTestTwo = false [private]
```

7.6.2.5 pullForce

```
Vector3 GuidewireSim.CollisionTestPerformer.pullForce = new Vector3(0f, 0f, 5f) [private]
```

External force that is applied in Force Test Three.

7.6.2.6 simulationLoop

```
SimulationLoop GuidewireSim.CollisionTestPerformer.simulationLoop [private]
```

The [SimulationLoop](#) component that executes all steps of the simulation loop.

7.6.2.7 startTime

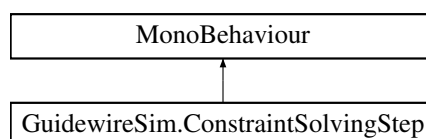
```
float GuidewireSim.CollisionTestPerformer.startTime = 0f [private]
```

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

- [CollisionTestPerformer.cs](#)

7.7 GuidewireSim.ConstraintSolvingStep Class Reference

Inheritance diagram for GuidewireSim.ConstraintSolvingStep:



Public Member Functions

- void [SolveStretchConstraints](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, BSM.Quaternion[] worldSpaceBasis, float rodElementLength)
- void [SolveBendTwistConstraints](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [SolveStretchConstraint](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e_3, float rodElementLength, out Vector3 [deltaPositionOne](#), out Vector3 [deltaPositionTwo](#), out BSM.Quaternion [deltaOrientation](#), float inverseMassOne=1f, float inverseMassTwo=1f, float inertiaWeight=1f)
- void [SolveBendTwistConstraint](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, Vector3 discreteRestDarbouxVector, float rodElementLength, out BSM.Quaternion [deltaOrientationOne](#), out BSM.Quaternion [deltaOrientationTwo](#), float inertiaWeightOne=1f, float inertiaWeightTwo=1f)

Private Member Functions

- void [Awake](#) ()
- void [SolveStretchConstraintsInBilateralOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e_3)
- void [SolveStretchConstraintsInNaiveOrder](#) (Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions, int spheresCount, float rodElementLength, BSM.Quaternion e_3)
- void [SolveBendTwistConstraintsInBilateralOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [SolveBendTwistConstraintsInNaiveOrder](#) (BSM.Quaternion[] cylinderOrientationPredictions, int cylinderCount, Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [CorrectStretchPredictions](#) (int sphereIndex, Vector3[] spherePositionPredictions, BSM.Quaternion[] cylinderOrientationPredictions)
- void [CorrectBendTwistPredictions](#) (int cylinderIndex, BSM.Quaternion[] cylinderOrientationPredictions)

Private Attributes

- [MathHelper mathHelper](#)
The component [MathHelper](#) that provides math related helper functions.
- Vector3 [deltaPositionOne](#) = new Vector3()
The correction of `particlePositionOne` in method [SolveStretchConstraint\(\)](#).
- Vector3 [deltaPositionTwo](#) = new Vector3()
The correction of `particlePositionTwo` in method [SolveStretchConstraint\(\)](#).
- BSM.Quaternion [deltaOrientation](#) = new BSM.Quaternion()
The correction of `orientation` in method [SolveStretchConstraint\(\)](#).
- BSM.Quaternion [deltaOrientationOne](#) = new BSM.Quaternion()
The correction of `orientationOne` in method [SolveBendTwistConstraint\(\)](#).
- BSM.Quaternion [deltaOrientationTwo](#) = new BSM.Quaternion()
The correction of `orientationTwo` in method [SolveBendTwistConstraint\(\)](#).
- float [stretchStiffness](#) = 0.1f
- float [bendStiffness](#) = 0.1f
- bool [executelnBilateralOrder](#) = false
Whether to solve both constraints in bilateral interleaving order. Naive order is used when false.

7.7.1 Detailed Description

This class executes and implements various algorithms of the constraint solving step of the algorithm and manages all coherent data.

7.7.2 Member Function Documentation

7.7.2.1 Awake()

```
void GuidewireSim.ConstraintSolvingStep.Awake ( ) [private]
```

7.7.2.2 CorrectBendTwistPredictions()

```
void GuidewireSim.ConstraintSolvingStep.CorrectBendTwistPredictions (
    int cylinderIndex,
    BSM.Quaternion[] cylinderOrientationPredictions ) [private]
```

Corrects the predictions of the bend twist constraint by adding `deltaOrientationOne` and `deltaOrientationTwo`.

Note

Note that `deltaOrientationOne` and `deltaOrientationTwo` may have a length unequal one by definition.

Parameters

<i>cylinderIndex</i>	The index of the first element of <code>cylinderOrientationPredictions</code> that gets corrected.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions of which two quaternions get corrected in this method.

Requirements The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion predictions got corrected, they should again be unit quaternions, i.e. have length approximately equal to one.

7.7.2.3 CorrectStretchPredictions()

```
void GuidewireSim.ConstraintSolvingStep.CorrectStretchPredictions (
    int sphereIndex,
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions ) [private]
```

Corrects the predictions of the stretch constraint by adding `deltaPositionOne`, `deltaPositionTwo` and `deltaOrientation`.

Note

Note that `deltaOrientation` may have a length unequal one by definition.

Parameters

<i>sphereIndex</i>	The index of the first element of <code>spherePositionPredictions</code> that gets corrected.
<i>spherePositionPredictions</i>	The array of position predictions of which two positions get corrected in this method.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions of which one quaternions gets corrected in this method.

Requirements The relevant entries of `cylinderOrientationPredictions` should be unit quaternions, i.e. have length approximately equal to one.

After the quaternion prediction got corrected, it should again be a unit quaternions, i.e. have length approximately equal to one.

7.7.2.4 SolveBendTwistConstraint()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraint (
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    Vector3 discreteRestDarbouxVector,
    float rodElementLength,
    out BSM.Quaternion deltaOrientationOne,
    out BSM.Quaternion deltaOrientationTwo,
    float inertiaWeightOne = 1f,
    float inertiaWeightTwo = 1f )
```

Solves the bend twist constraint by calculating the corrections `deltaOrientationOne` and `deltaOrientationTwo`.

Note

To be more precise, the bend twist constraint is not solved but minimized, i.e. the constraint will after correcting with the corrections be closer to zero.

Parameters

	<i>orientationOne</i>	The first orientation quaternion prediction of the orientation element to be corrected.
	<i>orientationTwo</i>	The second orientation quaternion prediction of the orientation element to be corrected.
	<i>discreteRestDarbouxVector</i>	The discrete Darboux Vector at the rest configuration, i.e. at frame 0.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
out	<i>deltaOrientationOne</i>	The correction of <code>orientationOne</code> .
out	<i>deltaOrientationTwo</i>	The correction of <code>orientationTwo</code> .
	<i>inertiaWeightOne</i>	The inertia weight scalar for <code>orientationOne</code> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.
	<i>inertiaWeightTwo</i>	The inertia weight scalar for <code>orientationTwo</code> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.

Requirements `orientationOne` and `orientationTwo` should be unit quaternions, i.e. have length approximately equal to one.

`rodElementLength` should be positive.

`inertiaWeightOne` and `inertiaWeightTwo` should be values between 0 and 1.

7.7.2.5 SolveBendTwistConstraints()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraints (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength )
```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint, i.e. corrects each orientation prediction one time.

Note

Can be executed in naive order or bilateral interleaving order.

Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Requirements *cylinderCount* should be at least one.

rodElementLength should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

7.7.2.6 SolveBendTwistConstraintsInBilateralOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInBilateralOrder (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength ) [private]
```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint in bilateral order, i.e. corrects each orientation prediction one time.

Note

You can read more about bilateral order in the 2016 paper "Position and Orientation Based Cosserat Rods".

Attention

The index shifting of this algorithm is not easy to understand, but got deeply tested.

Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Requirements *cylinderCount* should be at least one.
rodElementLength should be positive.

7.7.2.7 SolveBendTwistConstraintsInNaiveOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveBendTwistConstraintsInNaiveOrder (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] discreteRestDarbouxVectors,
    float rodElementLength ) [private]
```

Is responsible for executing one iteration of the constraint solving step for the bend twist constraint in naive order, i.e. corrects each orientation prediction one time.

Note

Naive order means the predictions are updated beginning from one end of the guidewire to the other end of the guidewire.

Parameters

<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Requirements *cylinderCount* should be at least one.
rodElementLength should be positive.

7.7.2.8 SolveStretchConstraint()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraint (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
    BSM.Quaternion e_3,
    float rodElementLength,
    out Vector3 deltaPositionOne,
    out Vector3 deltaPositionTwo,
    out BSM.Quaternion deltaOrientation,
    float inverseMassOne = 1f,
    float inverseMassTwo = 1f,
    float inertiaWeight = 1f )
```

Solves the stretch constraint by calculating the corrections `deltaPositionOne` and `deltaPositionTwo`, `deltaOrientation`.

Note

To be more precise, the stretch constraint is not solved but minimized, i.e. the constraint will after correcting with the corrections be closer to zero.

Parameters

	<i>particlePositionOne</i>	The first particle position prediction of the centerline element to be corrected.
	<i>particlePositionTwo</i>	The second particle position prediction of the centerline element to be corrected.
	<i>orientation</i>	The orientation quaternion prediction of the orientation element between the particle positions to be corrected.
	<i>e_3</i>	The third basis vector of the world space coordinates embedded as a quaternion with scalar part 0.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
out	<i>deltaPositionOne</i>	The correction of <code>particlePositionOne</code> .
out	<i>deltaPositionTwo</i>	The correction of <code>particlePositionTwo</code> .
out	<i>deltaOrientation</i>	The correction of <code>orientation</code> .
	<i>inverseMassOne</i>	The inverse mass scalar for <code>particlePositionOne</code> . Use a value of 1 for a moving particle and 0 for a fixed particle.
	<i>inverseMassTwo</i>	The inverse mass scalar for <code>particlePositionTwo</code> . Use a value of 1 for a moving particle and 0 for a fixed particle.
	<i>inertiaWeight</i>	The inertia weight scalar for <code>orientation</code> . Use a value of 1 for a moving orientation and 0 for a fixed orientation.

Requirements `orientation` should be a unit quaternions, i.e. have length approximately equal to one.

`e_3` should be a unit quaternions, i.e. have length approximately equal to one.

`rodElementLength` should be positive.

`inverseMassOne`, `inverseMassTwo` and `inertiaWeight` should be values between 0 and 1.

7.7.2.9 SolveStretchConstraints()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraints (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    BSM.Quaternion[] worldSpaceBasis,
    float rodElementLength )
```

Is responsible for executing one iteration of the constraint solving step for the stretch constraint, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

Note

Can be executed in naive order or bilateral interleaving order.

Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system as embedded quaternions with scalar part 0.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Requirements *spheresCount* should be at least one.

rodElementLength should be positive.

Executes the constraint solving step in bilateral interleaving order if [executeInBilateralOrder](#) and otherwise in naive order.

7.7.2.10 SolveStretchConstraintsInBilateralOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInBilateralOrder (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    float rodElementLength,
    BSM.Quaternion e_3 ) [private]
```

Executes one iteration of the constraint solving step for the stretch constraint in bilateral order, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

Note

You can read more about bilateral order in the 2016 paper "Position and Orientation Based Cosserat Rods".

Attention

The index shifting of this algorithm is not easy to understand, but got deeply tested.

Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
<i>e_3</i>	The third basis vector of the world coordinate system as embedded quaternions with scalar part 0.

Requirements *spheresCount* should be at least one.
rodElementLength should be positive.

7.7.2.11 SolveStretchConstraintsInNaiveOrder()

```
void GuidewireSim.ConstraintSolvingStep.SolveStretchConstraintsInNaiveOrder (
    Vector3[] spherePositionPredictions,
    BSM.Quaternion[] cylinderOrientationPredictions,
    int spheresCount,
    float rodElementLength,
    BSM.Quaternion e_3 ) [private]
```

Executes one iteration of the constraint solving step for the stretch constraint in naive order, i.e. corrects each particle position prediction one time and also each orientation prediction one time.

Note

Naive order means the predictions are updated beginning from one end of the guidewire to the other end of the guidewire.

Parameters

<i>spherePositionPredictions</i>	The array of position predictions that get corrected in this step.
<i>cylinderOrientationPredictions</i>	The array of orientation predictions that get corrected in this step.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.
<i>e_3</i>	The third basis vector of the world coordinate system as embedded quaternions with scalar part 0.

Requirements *spheresCount* should be at least one.
rodElementLength should be positive.

7.7.3 Member Data Documentation

7.7.3.1 bendStiffness

```
float GuidewireSim.ConstraintSolvingStep.bendStiffness = 0.1f [private]
```

7.7.3.2 deltaOrientation

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientation = new BSM.Quaternion()  
[private]
```

The correction of orientation in method [SolveStretchConstraint\(\)](#).

7.7.3.3 deltaOrientationOne

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientationOne = new BSM.Quaternion()  
[private]
```

The correction of orientationOne in method [SolveBendTwistConstraint\(\)](#).

7.7.3.4 deltaOrientationTwo

```
BSM.Quaternion GuidewireSim.ConstraintSolvingStep.deltaOrientationTwo = new BSM.Quaternion()  
[private]
```

The correction of orientationTwo in method [SolveBendTwistConstraint\(\)](#).

7.7.3.5 deltaPositionOne

```
Vector3 GuidewireSim.ConstraintSolvingStep.deltaPositionOne = new Vector3() [private]
```

The correction of particlePositionOne in method [SolveStretchConstraint\(\)](#).

7.7.3.6 `deltaPositionTwo`

```
Vector3 GuidewireSim.ConstraintSolvingStep.deltaPositionTwo = new Vector3() [private]
```

The correction of `particlePositionTwo` in method [SolveStretchConstraint\(\)](#).

7.7.3.7 `executeInBilateralOrder`

```
bool GuidewireSim.ConstraintSolvingStep.executeInBilateralOrder = false [private]
```

Whether to solve both constraints in bilateral interleaving order. Naive order is used when false.

7.7.3.8 `mathHelper`

```
MathHelper GuidewireSim.ConstraintSolvingStep.mathHelper [private]
```

The component [MathHelper](#) that provides math related helper functions.

7.7.3.9 `stretchStiffness`

```
float GuidewireSim.ConstraintSolvingStep.stretchStiffness = 0.1f [private]
```

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

- [ConstraintSolvingStep.cs](#)

7.8 DebugExtension Class Reference

Debug Extension

Static Public Member Functions

- static void [DebugPoint](#) (Vector3 position, Color color, float scale=1.0f, float duration=0, bool depthTest=true)
- static void [DebugPoint](#) (Vector3 position, float scale=1.0f, float duration=0, bool depthTest=true)
- static void [DebugBounds](#) (Bounds bounds, Color color, float duration=0, bool depthTest=true)
- static void [DebugBounds](#) (Bounds bounds, float duration=0, bool depthTest=true)
- static void [DebugLocalCube](#) (Transform transform, Vector3 size, Color color, Vector3 center=default(Vector3), float duration=0, bool depthTest=true)
- static void [DebugLocalCube](#) (Transform transform, Vector3 size, Vector3 center=default(Vector3), float duration=0, bool depthTest=true)
- static void [DebugLocalCube](#) (Matrix4x4 space, Vector3 size, Color color, Vector3 center=default(Vector3), float duration=0, bool depthTest=true)
- static void [DebugLocalCube](#) (Matrix4x4 space, Vector3 size, Vector3 center=default(Vector3), float duration=0, bool depthTest=true)
- static void [DebugCircle](#) (Vector3 position, Vector3 up, Color color, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugCircle](#) (Vector3 position, Color color, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugCircle](#) (Vector3 position, Vector3 up, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugCircle](#) (Vector3 position, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugWireSphere](#) (Vector3 position, Color color, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugWireSphere](#) (Vector3 position, float radius=1.0f, float duration=0, bool depthTest=true)
- static void [DebugCylinder](#) (Vector3 start, Vector3 end, Color color, float radius=1, float duration=0, bool depthTest=true)
- static void [DebugCylinder](#) (Vector3 start, Vector3 end, float radius=1, float duration=0, bool depthTest=true)
- static void [DebugCone](#) (Vector3 position, Vector3 direction, Color color, float angle=45, float duration=0, bool depthTest=true)
- static void [DebugCone](#) (Vector3 position, Vector3 direction, float angle=45, float duration=0, bool depthTest=true)
- static void [DebugCone](#) (Vector3 position, Color color, float angle=45, float duration=0, bool depthTest=true)
- static void [DebugCone](#) (Vector3 position, float angle=45, float duration=0, bool depthTest=true)
- static void [DebugArrow](#) (Vector3 position, Vector3 direction, Color color, float duration=0, bool depthTest=true)
- static void [DebugArrow](#) (Vector3 position, Vector3 direction, float duration=0, bool depthTest=true)
- static void [DebugCapsule](#) (Vector3 start, Vector3 end, Color color, float radius=1, float duration=0, bool depthTest=true)
- static void [DebugCapsule](#) (Vector3 start, Vector3 end, float radius=1, float duration=0, bool depthTest=true)
- static void [DrawPoint](#) (Vector3 position, Color color, float duration=2f, float scale=1.0f)
- static void [DrawPoint](#) (Vector3 position, float scale=1.0f)
- static void [DrawBounds](#) (Bounds bounds, Color color)
- static void [DrawBounds](#) (Bounds bounds)
- static void [DrawLocalCube](#) (Transform transform, Vector3 size, Color color, Vector3 center=default(Vector3))
- static void [DrawLocalCube](#) (Transform transform, Vector3 size, Vector3 center=default(Vector3))
- static void [DrawLocalCube](#) (Matrix4x4 space, Vector3 size, Color color, Vector3 center=default(Vector3))
- static void [DrawLocalCube](#) (Matrix4x4 space, Vector3 size, Vector3 center=default(Vector3))
- static void [DrawCircle](#) (Vector3 position, Vector3 up, Color color, float radius=1.0f)
- static void [DrawCircle](#) (Vector3 position, Color color, float radius=1.0f)
- static void [DrawCircle](#) (Vector3 position, Vector3 up, float radius=1.0f)
- static void [DrawCircle](#) (Vector3 position, float radius=1.0f)
- static void [DrawCylinder](#) (Vector3 start, Vector3 end, Color color, float radius=1.0f)
- static void [DrawCylinder](#) (Vector3 start, Vector3 end, float radius=1.0f)
- static void [DrawCone](#) (Vector3 position, Vector3 direction, Color color, float angle=45)
- static void [DrawCone](#) (Vector3 position, Vector3 direction, float angle=45)
- static void [DrawCone](#) (Vector3 position, Color color, float angle=45)
- static void [DrawCone](#) (Vector3 position, float angle=45)
- static void [DrawArrow](#) (Vector3 position, Vector3 direction, Color color)

- static void [DrawArrow](#) (Vector3 position, Vector3 direction)
- static void [DrawCapsule](#) (Vector3 start, Vector3 end, Color color, float radius=1)
- static void [DrawCapsule](#) (Vector3 start, Vector3 end, float radius=1)
- static string [MethodsOfObject](#) (System.Object obj, bool includeInfo=false)
- static string [MethodsOfType](#) (System.Type type, bool includeInfo=false)

7.8.1 Detailed Description

Debug Extension

- Static class that extends Unity's debugging functionality.
- Attempts to mimic Unity's existing debugging behaviour for ease-of-use.
- Includes gizmo drawing methods for less memory-intensive debug visualization.

7.8.2 Member Function Documentation

7.8.2.1 DebugArrow() [1/2]

```
static void DebugExtension.DebugArrow (
    Vector3 position,
    Vector3 direction,
    Color color,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs an arrow.

Parameters

<i>position</i>	
-----------------	--

The start position of the arrow.

Parameters

<i>direction</i>	<ul style="list-style-type: none"> • The direction the arrow will point in.
<i>color</i>	<ul style="list-style-type: none"> • The color of the arrow.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the arrow.

Parameters

<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the arrow should be faded when behind other objects.
------------------	---

7.8.2.2 DebugArrow() [2/2]

```
static void DebugExtension.DebugArrow (
    Vector3 position,
    Vector3 direction,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs an arrow.

Parameters

<i>position</i>	
-----------------	--

The start position of the arrow.

Parameters

<i>direction</i>	<ul style="list-style-type: none">• The direction the arrow will point in.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the arrow.
<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the arrow should be faded when behind other objects.

7.8.2.3 DebugBounds() [1/2]

```
static void DebugExtension.DebugBounds (
    Bounds bounds,
    Color color,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs an axis-aligned bounding box.

Parameters

<i>bounds</i>	
---------------	--

The bounds to debug.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the bounds.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the bounds.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the bounds should be faded when behind other objects.

7.8.2.4 DebugBounds() [2/2]

```
static void DebugExtension.DebugBounds (
    Bounds bounds,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs an axis-aligned bounding box.

Parameters

<i>bounds</i>	
---------------	--

The bounds to debug.

Parameters

<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the bounds.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the bounds should be faded when behind other objects.

7.8.2.5 DebugCapsule() [1/2]

```
static void DebugExtension.DebugCapsule (
    Vector3 start,
```



```

Vector3 end,
Color color,
float radius = 1,
float duration = 0,
bool depthTest = true ) [static]

```

- Debugs a capsule.

Parameters

<i>start</i>	
--------------	--

The position of one end of the capsule.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the capsule.
<i>color</i>	<ul style="list-style-type: none"> • The color of the capsule.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the capsule.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the capsule.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the capsule should be faded when behind other objects.

7.8.2.6 DebugCapsule() [2/2]

```

static void DebugExtension.DebugCapsule (
    Vector3 start,
    Vector3 end,
    float radius = 1,
    float duration = 0,
    bool depthTest = true ) [static]

```

- Debugs a capsule.

Parameters

<i>start</i>	
--------------	--

The position of one end of the capsule.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the capsule.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the capsule.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the capsule.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the capsule should be faded when behind other objects.

7.8.2.7 DebugCircle() [1/4]

```
static void DebugExtension.DebugCircle (
    Vector3 position,
    Color color,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the circle.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the circle.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the circle.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the circle should be faded when behind other objects.

7.8.2.8 DebugCircle() [2/4]

```
static void DebugExtension.DebugCircle (
    Vector3 position,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>radius</i>	<ul style="list-style-type: none"> • The radius of the circle.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the circle.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the circle should be faded when behind other objects.

7.8.2.9 DebugCircle() [3/4]

```
static void DebugExtension.DebugCircle (
    Vector3 position,
    Vector3 up,
    Color color,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>up</i>	<ul style="list-style-type: none"> The direction perpendicular to the surface of the circle.
<i>color</i>	<ul style="list-style-type: none"> The color of the circle.
<i>radius</i>	<ul style="list-style-type: none"> The radius of the circle.
<i>duration</i>	<ul style="list-style-type: none"> How long to draw the circle.
<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not the circle should be faded when behind other objects.

7.8.2.10 DebugCircle() [4/4]

```
static void DebugExtension.DebugCircle (
    Vector3 position,
    Vector3 up,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>up</i>	<ul style="list-style-type: none"> The direction perpendicular to the surface of the circle.
<i>radius</i>	<ul style="list-style-type: none"> The radius of the circle.
<i>duration</i>	<ul style="list-style-type: none"> How long to draw the circle.
<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not the circle should be faded when behind other objects.

7.8.2.11 DebugCone() [1/4]

```
static void DebugExtension.DebugCone (
    Vector3 position,
    Color color,
    float angle = 45,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>angle</i>	<ul style="list-style-type: none">• The angle of the cone.
<i>color</i>	<ul style="list-style-type: none">• The color of the cone.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the cone.
<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the cone should be faded when behind other objects.

7.8.2.12 DebugCone() [2/4]

```
static void DebugExtension.DebugCone (
    Vector3 position,
    float angle = 45,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>angle</i>	<ul style="list-style-type: none"> The angle of the cone.
<i>duration</i>	<ul style="list-style-type: none"> How long to draw the cone.
<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not the cone should be faded when behind other objects.

7.8.2.13 DebugCone() [3/4]

```
static void DebugExtension.DebugCone (
    Vector3 position,
    Vector3 direction,
    Color color,
    float angle = 45,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>direction</i>	<ul style="list-style-type: none"> The direction for the cone gets wider in.
<i>angle</i>	<ul style="list-style-type: none"> The angle of the cone.
<i>color</i>	<ul style="list-style-type: none"> The color of the cone.
<i>duration</i>	<ul style="list-style-type: none"> How long to draw the cone.
<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not the cone should be faded when behind other objects.

7.8.2.14 DebugCone() [4/4]

```
static void DebugExtension.DebugCone (
    Vector3 position,
    Vector3 direction,
    float angle = 45,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>direction</i>	<ul style="list-style-type: none">• The direction for the cone gets wider in.
<i>angle</i>	<ul style="list-style-type: none">• The angle of the cone.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the cone.
<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the cone should be faded when behind other objects.

7.8.2.15 DebugCylinder() [1/2]

```
static void DebugExtension.DebugCylinder (
    Vector3 start,
    Vector3 end,
    Color color,
    float radius = 1,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cylinder.

Parameters

<i>start</i>	
--------------	--

The position of one end of the cylinder.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the cylinder.
<i>color</i>	<ul style="list-style-type: none"> • The color of the cylinder.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the cylinder.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the cylinder.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the cylinder should be faded when behind other objects.

7.8.2.16 DebugCylinder() [2/2]

```
static void DebugExtension.DebugCylinder (
    Vector3 start,
    Vector3 end,
    float radius = 1,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a cylinder.

Parameters

<i>start</i>	
--------------	--

The position of one end of the cylinder.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the cylinder.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the cylinder.

Parameters

<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the cylinder.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the cylinder should be faded when behind other objects.

7.8.2.17 DebugLocalCube() [1/4]

```
static void DebugExtension.DebugLocalCube (
    Matrix4x4 space,
    Vector3 size,
    Color color,
    Vector3 center = default(Vector3),
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a local cube.

Parameters

<i>space</i>	
--------------	--

The space the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> • The size of the cube.
<i>color</i>	<ul style="list-style-type: none"> • Color of the cube.
<i>center</i>	<ul style="list-style-type: none"> • The position (relative to transform) where the cube will be debugged.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the cube.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the cube should be faded when behind other objects.

7.8.2.18 DebugLocalCube() [2/4]

```
static void DebugExtension.DebugLocalCube (
    Matrix4x4 space,
    Vector3 size,
    Vector3 center = default(Vector3),
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a local cube.

Parameters

<i>space</i>	
--------------	--

The space the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> • The size of the cube.
<i>center</i>	<ul style="list-style-type: none"> • The position (relative to transform) where the cube will be debugged.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the cube.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the cube should be faded when behind other objects.

7.8.2.19 DebugLocalCube() [3/4]

```
static void DebugExtension.DebugLocalCube (
    Transform transform,
    Vector3 size,
    Color color,
    Vector3 center = default(Vector3),
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a local cube.

Parameters

<i>transform</i>	
------------------	--

The transform that the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none">• The size of the cube.
<i>color</i>	<ul style="list-style-type: none">• Color of the cube.
<i>center</i>	<ul style="list-style-type: none">• The position (relative to transform) where the cube will be debugged.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the cube.
<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the cube should be faded when behind other objects.

7.8.2.20 DebugLocalCube() [4/4]

```
static void DebugExtension.DebugLocalCube (  
    Transform transform,  
    Vector3 size,  
    Vector3 center = default(Vector3),  
    float duration = 0,  
    bool depthTest = true ) [static]
```

- Debugs a local cube.

Parameters

<i>transform</i>	
------------------	--

The transform that the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none">• The size of the cube.
<i>center</i>	<ul style="list-style-type: none">• The position (relative to transform) where the cube will be debugged.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the cube.

Parameters

<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not the cube should be faded when behind other objects.
------------------	--

7.8.2.21 DebugPoint() [1/2]

```
static void DebugExtension.DebugPoint (
    Vector3 position,
    Color color,
    float scale = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a point.

Parameters

<i>position</i>	
-----------------	--

The point to debug.

Parameters

<i>color</i>	<ul style="list-style-type: none"> The color of the point.
<i>scale</i>	<ul style="list-style-type: none"> The size of the point.
<i>duration</i>	<ul style="list-style-type: none"> How long to draw the point.
<i>depthTest</i>	<ul style="list-style-type: none"> Whether or not this point should be faded when behind other objects.

7.8.2.22 DebugPoint() [2/2]

```
static void DebugExtension.DebugPoint (
    Vector3 position,
    float scale = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a point.

Parameters

<i>position</i>	
-----------------	--

The point to debug.

Parameters

<i>scale</i>	<ul style="list-style-type: none"> • The size of the point.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the point.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not this point should be faded when behind other objects.

7.8.2.23 DebugWireSphere() [1/2]

```
static void DebugExtension.DebugWireSphere (
    Vector3 position,
    Color color,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a wire sphere.

Parameters

<i>position</i>	
-----------------	--

The position of the center of the sphere.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the sphere.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the sphere.
<i>duration</i>	<ul style="list-style-type: none"> • How long to draw the sphere.
<i>depthTest</i>	<ul style="list-style-type: none"> • Whether or not the sphere should be faded when behind other objects.

7.8.2.24 DebugWireSphere() [2/2]

```
static void DebugExtension.DebugWireSphere (
    Vector3 position,
    float radius = 1.0f,
    float duration = 0,
    bool depthTest = true ) [static]
```

- Debugs a wire sphere.

Parameters

<i>position</i>	
-----------------	--

The position of the center of the sphere.

Parameters

<i>radius</i>	<ul style="list-style-type: none">• The radius of the sphere.
<i>duration</i>	<ul style="list-style-type: none">• How long to draw the sphere.
<i>depthTest</i>	<ul style="list-style-type: none">• Whether or not the sphere should be faded when behind other objects.

7.8.2.25 DrawArrow() [1/2]

```
static void DebugExtension.DrawArrow (
    Vector3 position,
    Vector3 direction ) [static]
```

- Draws an arrow.

Parameters

<i>position</i>	
-----------------	--

The start position of the arrow.

Parameters

<i>direction</i>	<ul style="list-style-type: none">• The direction the arrow will point in.
------------------	--

7.8.2.26 DrawArrow() [2/2]

```
static void DebugExtension.DrawArrow (
    Vector3 position,
    Vector3 direction,
    Color color ) [static]
```

- Draws an arrow.

Parameters

<i>position</i>	
-----------------	--

The start position of the arrow.

Parameters

<i>direction</i>	<ul style="list-style-type: none">• The direction the arrow will point in.
<i>color</i>	<ul style="list-style-type: none">• The color of the arrow.

7.8.2.27 DrawBounds() [1/2]

```
static void DebugExtension.DrawBounds (
    Bounds bounds ) [static]
```

- Draws an axis-aligned bounding box.

Parameters

<i>bounds</i>	
---------------	--

The bounds to draw.

7.8.2.28 DrawBounds() [2/2]

```
static void DebugExtension.DrawBounds (
    Bounds bounds,
    Color color ) [static]
```

- Draws an axis-aligned bounding box.

Parameters

<i>bounds</i>	
---------------	--

The bounds to draw.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the bounds.
--------------	--

7.8.2.29 DrawCapsule() [1/2]

```
static void DebugExtension.DrawCapsule (
    Vector3 start,
    Vector3 end,
    Color color,
    float radius = 1 ) [static]
```

- Draws a capsule.

Parameters

<i>start</i>	
--------------	--

The position of one end of the capsule.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the capsule.
<i>color</i>	<ul style="list-style-type: none"> • The color of the capsule.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the capsule.

7.8.2.30 DrawCapsule() [2/2]

```
static void DebugExtension.DrawCapsule (
    Vector3 start,
    Vector3 end,
    float radius = 1 ) [static]
```

- Draws a capsule.

Parameters

<i>start</i>	
--------------	--

The position of one end of the capsule.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the capsule.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the capsule.

7.8.2.31 DrawCircle() [1/4]

```
static void DebugExtension.DrawCircle (
    Vector3 position,
    Color color,
    float radius = 1.0f ) [static]
```

- Draws a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the circle.
--------------	--

Parameters

<i>radius</i>	<ul style="list-style-type: none">• The radius of the circle.
---------------	---

7.8.2.32 DrawCircle() [2/4]

```
static void DebugExtension.DrawCircle (  
    Vector3 position,  
    float radius = 1.0f ) [static]
```

- Draws a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>radius</i>	<ul style="list-style-type: none">• The radius of the circle.
---------------	---

7.8.2.33 DrawCircle() [3/4]

```
static void DebugExtension.DrawCircle (  
    Vector3 position,  
    Vector3 up,  
    Color color,  
    float radius = 1.0f ) [static]
```

- Draws a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>up</i>	<ul style="list-style-type: none">• The direction perpendicular to the surface of the circle.
<i>color</i>	<ul style="list-style-type: none">• The color of the circle.
<i>radius</i>	<ul style="list-style-type: none">• The radius of the circle.

7.8.2.34 DrawCircle() [4/4]

```
static void DebugExtension.DrawCircle (
    Vector3 position,
    Vector3 up,
    float radius = 1.0f ) [static]
```

- Draws a circle.

Parameters

<i>position</i>	
-----------------	--

Where the center of the circle will be positioned.

Parameters

<i>up</i>	<ul style="list-style-type: none">• The direction perpendicular to the surface of the circle.
<i>radius</i>	<ul style="list-style-type: none">• The radius of the circle.

7.8.2.35 DrawCone() [1/4]

```
static void DebugExtension.DrawCone (
    Vector3 position,
    Color color,
    float angle = 45 ) [static]
```

- Draws a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>color</i>	<ul style="list-style-type: none"> • The color of the cone.
<i>angle</i>	<ul style="list-style-type: none"> • The angle of the cone.

7.8.2.36 DrawCone() [2/4]

```
static void DebugExtension.DrawCone (
    Vector3 position,
    float angle = 45 ) [static]
```

- Draws a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>angle</i>	<ul style="list-style-type: none"> • The angle of the cone.
--------------	--

7.8.2.37 DrawCone() [3/4]

```
static void DebugExtension.DrawCone (
    Vector3 position,
    Vector3 direction,
    Color color,
    float angle = 45 ) [static]
```

- Draws a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>direction</i>	<ul style="list-style-type: none"> • The direction for the cone to get wider in.
<i>color</i>	<ul style="list-style-type: none"> • The color of the cone.
<i>angle</i>	<ul style="list-style-type: none"> • The angle of the cone.

7.8.2.38 DrawCone() [4/4]

```
static void DebugExtension.DrawCone (
    Vector3 position,
    Vector3 direction,
    float angle = 45 ) [static]
```

- Draws a cone.

Parameters

<i>position</i>	
-----------------	--

The position for the tip of the cone.

Parameters

<i>direction</i>	<ul style="list-style-type: none"> • The direction for the cone to get wider in.
<i>angle</i>	<ul style="list-style-type: none"> • The angle of the cone.

7.8.2.39 DrawCylinder() [1/2]

```
static void DebugExtension.DrawCylinder (
    Vector3 start,
```

```
Vector3 end,
Color color,
float radius = 1.0f ) [static]
```

- Draws a cylinder.

Parameters

<i>start</i>	
--------------	--

The position of one end of the cylinder.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the cylinder.
<i>color</i>	<ul style="list-style-type: none"> • The color of the cylinder.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the cylinder.

7.8.2.40 DrawCylinder() [2/2]

```
static void DebugExtension.DrawCylinder (
    Vector3 start,
    Vector3 end,
    float radius = 1.0f ) [static]
```

- Draws a cylinder.

Parameters

<i>start</i>	
--------------	--

The position of one end of the cylinder.

Parameters

<i>end</i>	<ul style="list-style-type: none"> • The position of the other end of the cylinder.
<i>radius</i>	<ul style="list-style-type: none"> • The radius of the cylinder.

7.8.2.41 DrawLocalCube() [1/4]

```
static void DebugExtension.DrawLocalCube (
    Matrix4x4 space,
    Vector3 size,
    Color color,
    Vector3 center = default(Vector3) ) [static]
```

- Draws a local cube.

Parameters

<i>space</i>	
--------------	--

The space the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> • The local size of the cube.
<i>center</i>	<ul style="list-style-type: none"> • The local position of the cube.
<i>color</i>	<ul style="list-style-type: none"> • The color of the cube.

7.8.2.42 DrawLocalCube() [2/4]

```
static void DebugExtension.DrawLocalCube (
    Matrix4x4 space,
    Vector3 size,
    Vector3 center = default(Vector3) ) [static]
```

- Draws a local cube.

Parameters

<i>space</i>	
--------------	--

The space the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> The local size of the cube.
<i>center</i>	<ul style="list-style-type: none"> The local position of the cube.

7.8.2.43 DrawLocalCube() [3/4]

```
static void DebugExtension.DrawLocalCube (
    Transform transform,
    Vector3 size,
    Color color,
    Vector3 center = default(Vector3) ) [static]
```

- Draws a local cube.

Parameters

<i>transform</i>	
------------------	--

The transform the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> The local size of the cube.
<i>center</i>	<ul style="list-style-type: none"> The local position of the cube.
<i>color</i>	<ul style="list-style-type: none"> The color of the cube.

7.8.2.44 DrawLocalCube() [4/4]

```
static void DebugExtension.DrawLocalCube (
    Transform transform,
    Vector3 size,
    Vector3 center = default(Vector3) ) [static]
```

- Draws a local cube.

Parameters

<i>transform</i>	
------------------	--

The transform the cube will be local to.

Parameters

<i>size</i>	<ul style="list-style-type: none"> The local size of the cube.
<i>center</i>	<ul style="list-style-type: none"> The local position of the cube.

7.8.2.45 DrawPoint() [1/2]

```
static void DebugExtension.DrawPoint (
    Vector3 position,
    Color color,
    float duration = 2f,
    float scale = 1.0f ) [static]
```

- Draws a point.

Parameters

<i>position</i>	
-----------------	--

The point to draw.

Parameters

<i>color</i>	<ul style="list-style-type: none"> The color of the drawn point.
<i>scale</i>	<ul style="list-style-type: none"> The size of the drawn point.

7.8.2.46 DrawPoint() [2/2]

```
static void DebugExtension.DrawPoint (
    Vector3 position,
    float scale = 1.0f ) [static]
```

- Draws a point.

Parameters

<i>position</i>	
-----------------	--

The point to draw.

Parameters

<i>scale</i>	<ul style="list-style-type: none"> • The size of the drawn point.
--------------	--

7.8.2.47 MethodsOfObject()

```
static string DebugExtension.MethodsOfObject (
    System.Object obj,
    bool includeInfo = false ) [static]
```

- Gets the methods of an object.

Returns

A list of methods accessible from this object.

Parameters

<i>obj</i>	
------------	--

The object to get the methods of.

Parameters

<i>includeInfo</i>	<ul style="list-style-type: none"> • Whether or not to include each method's method info in the list.
--------------------	--

7.8.2.48 MethodsOfType()

```
static string DebugExtension.MethodsOfType (
    System.Type type,
    bool includeInfo = false ) [static]
```

- Gets the methods of a type.

Returns

A list of methods accessible from this type.

Parameters

<i>type</i>	
-------------	--

The type to get the methods of.

Parameters

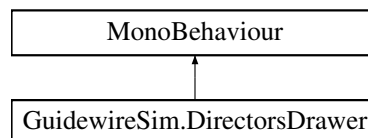
<i>includeInfo</i>	<ul style="list-style-type: none"> Whether or not to include each method's method info in the list.
--------------------	--

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

- [DebugExtension.cs](#)

7.9 GuidewireSim.DirectorsDrawer Class Reference

Inheritance diagram for GuidewireSim.DirectorsDrawer:



Private Member Functions

- void [Awake](#) ()
- void [Update](#) ()
- void [DrawDirectors](#) (Vector3[] cylinderPositions, Vector3[][] directors)
- Vector3[] [CalculateArrowHeadPositions](#) (Vector3 startPosition, Vector3 endPosition)
- void [DrawArrowHeadLines](#) (int directorIndex, Vector3 endPosition, Vector3[] arrowHeadPositions)
- void [DrawArrowHeadConnectionLines](#) (int directorIndex, Vector3[] arrowHeadPositions)

Private Attributes

- [SimulationLoop](#) *simulationLoop*
The component [SimulationLoop](#).
- float [scaleFactor](#)
The scale factor that gets multiplied to the length of the respective director.
- float [arrowHeadAngle](#)
The angle spread of the arrow head.
- float [arrowHeadPercentage](#)

- The percentage of the length of the arrow that the arrow head covers.*
- Color `directorOneColor` = `Color.green`
 - The color that the lines representing the first director are drawn with.*
- Color `directorTwoColor` = `Color.blue`
 - The color that the lines representing the second director are drawn with.*
- Color `directorThreeColor` = `Color.red`
 - The color that the lines representing the third director are drawn with.*
- `Color[] directorColors` = `new Color[3] {Color.red, Color.green, Color.blue}`

7.9.1 Detailed Description

This class represents each orientation by drawing all of its directors as arrows in each frame.

7.9.2 Member Function Documentation

7.9.2.1 Awake()

```
void GuidewireSim.DirectorsDrawer.Awake ( ) [private]
```

7.9.2.2 CalculateArrowHeadPositions()

```
Vector3[] GuidewireSim.DirectorsDrawer.CalculateArrowHeadPositions (
    Vector3 startPosition,
    Vector3 endPosition ) [private]
```

Calculates the end position of each line of each arrow head. E.g. an arrow head consists of four lines, each of them starting at `endPosition` and spreading in different directions to form the shape of an arrow tip.

Parameters

<i>startPosition</i>	The start position of the director, i.e. the position of the orientation.
<i>endPosition</i>	The position of the tip of the arrow head.

Returns

The end positions of the four lines that form the arrow head.

Requirements `arrowHeadPositions` has a length of 4.

7.9.2.3 DrawArrowHeadConnectionLines()

```
void GuidewireSim.DirectorsDrawer.DrawArrowHeadConnectionLines (
    int directorIndex,
    Vector3[] arrowHeadPositions ) [private]
```

Draws the four lines that connect the arrow head tips with each other. E.g. draws the line from `arrowHeadPositions` 0 and `arrowHeadPositions` 1.

Parameters

<i>directorIndex</i>	The index of the director under consideration.
<i>arrowHeadPositions</i>	The end positions of the four lines that form the arrow head.

7.9.2.4 DrawArrowHeadLines()

```
void GuidewireSim.DirectorsDrawer.DrawArrowHeadLines (
    int directorIndex,
    Vector3 endPosition,
    Vector3[] arrowHeadPositions ) [private]
```

Draws the four lines that form the arrow head for the director that corresponds to `directorIndex`.

Parameters

<i>directorIndex</i>	The index of the director under consideration.
<i>endPosition</i>	The position of the tip of the arrow head.
<i>arrowHeadPositions</i>	The end positions of the four lines that form the arrow head.

Requirements `arrowHeadPositions` has a length of 4.

7.9.2.5 DrawDirectors()

```
void GuidewireSim.DirectorsDrawer.DrawDirectors (
    Vector3[] cylinderPositions,
    Vector3 directors[][] ) [private]
```

Draws the director basis of each orientation element as arrows.

Parameters

<i>cylinderPositions</i>	The center of mass of each cylinder, i.e. the position of each orientation element.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

7.9.2.6 Update()

```
void GuidewireSim.DirectorsDrawer.Update ( ) [private]
```

7.9.3 Member Data Documentation

7.9.3.1 arrowHeadAngle

```
float GuidewireSim.DirectorsDrawer.arrowHeadAngle [private]
```

The angle spread of the arrow head.

7.9.3.2 arrowHeadPercentage

```
float GuidewireSim.DirectorsDrawer.arrowHeadPercentage [private]
```

The percentage of the length of the arrow that the arrow head covers.

7.9.3.3 directorColors

```
Color [ ] GuidewireSim.DirectorsDrawer.directorColors = new Color[3] {Color.red, Color.green,  
Color.blue} [private]
```

The color that the lines representing the three directors are drawn with.

Note

The i-th director is drawn in the i-th Color.

7.9.3.4 directorOneColor

```
Color GuidewireSim.DirectorsDrawer.directorOneColor = Color.green [private]
```

The color that the lines representing the first director are drawn with.

7.9.3.5 directorThreeColor

```
Color GuidewireSim.DirectorsDrawer.directorThreeColor = Color.red [private]
```

The color that the lines representing the third director are drawn with.

7.9.3.6 directorTwoColor

```
Color GuidewireSim.DirectorsDrawer.directorTwoColor = Color.blue [private]
```

The color that the lines representing the second director are drawn with.

7.9.3.7 scaleFactor

```
float GuidewireSim.DirectorsDrawer.scaleFactor [private]
```

The scale factor that gets multiplied to the length of the respective director.

7.9.3.8 simulationLoop

```
SimulationLoop GuidewireSim.DirectorsDrawer.simulationLoop [private]
```

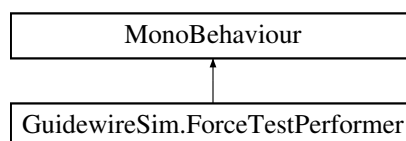
The component [SimulationLoop](#).

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

- [DirectorsDrawer.cs](#)

7.10 GuidewireSim.ForceTestPerformer Class Reference

Inheritance diagram for GuidewireSim.ForceTestPerformer:



Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformForceTests](#) ()
- void [PerformForceTestOne](#) ()
- void [PerformForceTestTwo](#) ()
- IEnumerator [PerformForceTestThree](#) (Vector3 pullForce, float applyForceTime=1f)
- void [PerformForceTestFour](#) ()
- void [PerformSingleLoopTest](#) ()

Private Attributes

- [SimulationLoop](#) `simulationLoop`
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- bool [doForceTestOne](#) = false
Whether to run Force Test One. This test applies gravity to all spheres.
- bool [doForceTestTwo](#) = false
Whether to run Force Test Two. This test applies an external force to one end of the guidewire.
- bool [doForceTestThree](#) = false
- bool [doForceTestFour](#) = false
- bool [doSingleLoopTest](#) = false
- Vector3 [pullForceTestThree](#) = new Vector3(0f, 3f, 0f)
External force that is applied in Force Test Three.

7.10.1 Detailed Description

This class enables the user to test the impact of external forces with one button within the Unity inspector.

7.10.2 Member Function Documentation

7.10.2.1 Awake()

```
void GuidewireSim.ForceTestPerformer.Awake ( ) [private]
```

7.10.2.2 PerformForceTestFour()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestFour ( ) [private]
```

Performs force test four. This test applies an external force to one end of the guidewire and the opposite force at the other end of the guidewire.

7.10.2.3 PerformForceTestOne()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestOne ( ) [private]
```

Performs force test one. This test applies gravity to all spheres.

7.10.2.4 PerformForceTests()

```
void GuidewireSim.ForceTestPerformer.PerformForceTests ( ) [private]
```

Performs each Force Test whose respective serialized boolean is set to true in the Unity inspector.

7.10.2.5 PerformForceTestThree()

```
IEnumerator GuidewireSim.ForceTestPerformer.PerformForceTestThree (
    Vector3 pullForce,
    float applyForceTime = 1f ) [private]
```

Performs force test three. This test applies an external force to one end of the guidewire for a fixed amount of time and then the opposite force at the same sphere for the same amount of time.

Parameters

<i>applyForceTime</i>	For how many seconds to apply the force to the particles.
-----------------------	---

7.10.2.6 PerformForceTestTwo()

```
void GuidewireSim.ForceTestPerformer.PerformForceTestTwo ( ) [private]
```

Performs force test two. This test applies an external force to one end of the guidewire.

7.10.2.7 PerformSingleLoopTest()

```
void GuidewireSim.ForceTestPerformer.PerformSingleLoopTest ( ) [private]
```

Performs the single loop test. This test shifts one end of the guidewire and runs the simulation for exactly one loop iteration to test constraint solving.

Note

Position of particle one stays at (0, 0, 0), while the section particle shifts to about (10, 2, 0). Expected result is that both particles move a bit towards each other and reestablish a distance of 10 between them.

7.10.2.8 Start()

```
void GuidewireSim.ForceTestPerformer.Start ( ) [private]
```

7.10.3 Member Data Documentation

7.10.3.1 doForceTestFour

```
bool GuidewireSim.ForceTestPerformer.doForceTestFour = false [private]
```

Whether to run Force Test Four. This test applies an external force to one end of the guidewire and the opposite force at the other end of the guidewire.

7.10.3.2 doForceTestOne

```
bool GuidewireSim.ForceTestPerformer.doForceTestOne = false [private]
```

Whether to run Force Test One. This test applies gravity to all spheres.

7.10.3.3 doForceTestThree

```
bool GuidewireSim.ForceTestPerformer.doForceTestThree = false [private]
```

Whether to run Force Test Three. This test applies an external force to one end of the guidewire for a fixed amount of time and then the opposite force at the same sphere for the same amount of time.

7.10.3.4 doForceTestTwo

```
bool GuidewireSim.ForceTestPerformer.doForceTestTwo = false [private]
```

Whether to run Force Test Two. This test applies an external force to one end of the guidewire.

7.10.3.5 doSingleLoopTest

```
bool GuidewireSim.ForceTestPerformer.doSingleLoopTest = false [private]
```

Whether to run the Single Loop Test. This test shifts one end of the guidewire and runs the simulation for exactly one loop iteration to test constraint solving.

7.10.3.6 pullForceTestThree

```
Vector3 GuidewireSim.ForceTestPerformer.pullForceTestThree = new Vector3(0f, 3f, 0f) [private]
```

External force that is applied in Force Test Three.

7.10.3.7 simulationLoop

```
SimulationLoop GuidewireSim.ForceTestPerformer.simulationLoop [private]
```

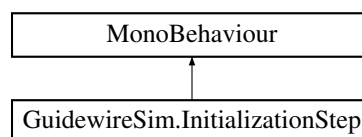
The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [ForceTestPerformer.cs](#)

7.11 GuidewireSim.InitializationStep Class Reference

Inheritance diagram for GuidewireSim.InitializationStep:



Public Member Functions

- void [InitSpherePositions](#) (GameObject[] spheres, int spheresCount, out Vector3[] spherePositions)
- void [InitSphereVelocities](#) (int spheresCount, out Vector3[] sphereVelocities)
- void [InitSphereInverseMasses](#) (int spheresCount, out float[] sphereInverseMasses)
- void [InitCylinderPositions](#) (int cylinderCount, Vector3[] spherePositions, out Vector3[] cylinderPositions)
- void [InitCylinderOrientations](#) (int cylinderCount, out BSM.Quaternion[] cylinderOrientations)
- void [InitDiscreteRestDarbouxVectors](#) (int cylinderCount, BSM.Quaternion[] cylinderOrientations, out Vector3[] discreteRestDarbouxVectors, float rodElementLength)
- void [InitCylinderAngularVelocities](#) (int cylinderCount, out Vector3[] cylinderAngularVelocities)
- void [InitCylinderScalarWeights](#) (int cylinderCount, out float[] cylinderScalarWeights)
- void [InitSphereExternalForces](#) (int spheresCount, out Vector3[] sphereExternalForces)
- void [InitSpherePositionPredictions](#) (GameObject[] spheres, int spheresCount, out Vector3[] spherePosition↔ Predictions)
- void [InitCylinderOrientationPredictions](#) (int cylinderCount, out BSM.Quaternion[] cylinderOrientation↔ Predictions)
- void [InitInertiaTensor](#) (out float[,] inertiaTensor)
- void [InitInverseInertiaTensor](#) (float[,] inertiaTensor, out float[,] inverseInertiaTensor)
- void [InitCylinderExternalTorques](#) (int cylinderCount, out Vector3[] cylinderExternalTorques)
- void [InitWorldSpaceBasis](#) (out BSM.Quaternion[] worldSpaceBasis)
- void [InitDirectors](#) (int cylinderCount, BSM.Quaternion[] worldSpaceBasis, out Vector3[][] directors)
- void [InitSphereColliders](#) (int spheresCount, GameObject[] spheres)

Private Member Functions

- void [Awake](#) ()

Private Attributes

- [CollisionHandler](#) `collisionHandler`
The component [CollisionHandler](#) that solves all collisions.
- [MathHelper](#) `mathHelper`
The component [MathHelper](#) that provides math related helper functions.
- float `materialDensity` = 7860
- float `materialRadius` = 0.001f

7.11.1 Detailed Description

This class is responsible for initializing all data with their initial values of the simulation at the start of the simulation.

7.11.2 Member Function Documentation

7.11.2.1 Awake()

```
void GuidewireSim.InitializationStep.Awake ( ) [private]
```

7.11.2.2 InitCylinderAngularVelocities()

```
void GuidewireSim.InitializationStep.InitCylinderAngularVelocities (
    int cylinderCount,
    out Vector3[] cylinderAngularVelocities )
```

Initializes `cylinderAngularVelocities` with the default value of zero at the start of the simulation.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/cylinder.

7.11.2.3 InitCylinderExternalTorques()

```
void GuidewireSim.InitializationStep.InitCylinderExternalTorques (
    int cylinderCount,
    out Vector3[] cylinderExternalTorques )
```

Initializes `cylinderExternalTorques` with the default value of zero at the start of the simulation.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderExternalTorques</i>	The sum of all current external torques that are applied per orientation element/ cylinder.

7.11.2.4 InitCylinderOrientationPredictions()

```
void GuidewireSim.InitializationStep.InitCylinderOrientationPredictions (
    int cylinderCount,
    out BSM.Quaternion[] cylinderOrientationPredictions )
```

Initializes `cylinderOrientationPredictions` with the default value of (0f, 0f, 0f, 1f) which equals the quaternion identity at the start of the simulation.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

7.11.2.5 InitCylinderOrientations()

```
void GuidewireSim.InitializationStep.InitCylinderOrientations (
    int cylinderCount,
    out BSM.Quaternion[] cylinderOrientations )
```

Initializes `cylinderOrientations` with the default value of (0f, 0f, 0f, 1f) which equals the quaternion identity at the start of the simulation.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientations</code> .
out	<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.

7.11.2.6 InitCylinderPositions()

```
void GuidewireSim.InitializationStep.InitCylinderPositions (
    int cylinderCount,
    Vector3[] spherePositions,
    out Vector3[] cylinderPositions )
```

Initializes `cylinderPositions` as middle points of the positions of `spheres` at the start of the simulation.

Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

7.11.2.7 InitCylinderScalarWeights()

```
void GuidewireSim.InitializationStep.InitCylinderScalarWeights (
    int cylinderCount,
    out float[] cylinderScalarWeights )
```

Initializes `cylinderScalarWeights` with the default value of one at the start of the simulation.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
out	<i>cylinderScalarWeights</i>	The constant scalar weights of each orientation/ quaternion similar to <code>sphereInverseMasses</code> .

7.11.2.8 InitDirectors()

```
void GuidewireSim.InitializationStep.InitDirectors (
    int cylinderCount,
    BSM.Quaternion[] worldSpaceBasis,
    out Vector3 directors[][] )
```

Initializes the `directors` array of arrays. The zero-th array defines all first directors of each director basis and so on.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
	<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.
out	<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

Note

Example: The (i, j)th element holds the (i-1)th director of orientation element j.

7.11.2.9 InitDiscreteRestDarbouxVectors()

```
void GuidewireSim.InitializationStep.InitDiscreteRestDarbouxVectors (
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    out Vector3[] discreteRestDarbouxVectors,
    float rodElementLength )
```

Calculates the discrete darboux vector for each orientation pair (two adjacent orientations) at its rest configuration, i.e. at frame 0.

Parameters

	<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
	<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
out	<i>discreteRestDarbouxVectors</i>	The array of all discrete Darboux Vectors at the rest configuration, i.e. at frame 0. Has (n-1) elements, if n is the number of orientations of the guidewire, because the darboux vector is taken of two adjacent orientations.
	<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Requirements `cylinderCount` should be at least one.
`rodElementLength` should be positive.

Note

All cylinder orientations must be computed for frame 0 first.

7.11.2.10 InitInertiaTensor()

```
void GuidewireSim.InitializationStep.InitInertiaTensor (
    out float inertiaTensor[,])
```

Initializes `inertiaTensor` so that all elements except the diagonal ones are zero. The first and second diagonal entry equal $\rho * \pi * \frac{r^2}{4}$, and the third diagonal entry equals $\rho * \pi * \frac{r^2}{2}$.

Parameters

out	<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
-----	----------------------	---

7.11.2.11 InitInverseInertiaTensor()

```
void GuidewireSim.InitializationStep.InitInverseInertiaTensor (
    float inertiaTensor[,],
    out float inverseInertiaTensor[,] )
```

Initializes `inverseInertiaTensor` as the inverse of `inertiaTensor`.

Parameters

	<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
out	<i>inverseInertiaTensor</i>	The inverse of <code>inertiaTensor</code> .

7.11.2.12 InitSphereColliders()

```
void GuidewireSim.InitializationStep.InitSphereColliders (
    int spheresCount,
    GameObject[] spheres )
```

7.11.2.13 InitSphereExternalForces()

```
void GuidewireSim.InitializationStep.InitSphereExternalForces (
    int spheresCount,
    out Vector3[] sphereExternalForces )
```

Initializes `sphereExternalForces` with the default value of zero at the start of the simulation.

Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>sphereExternalForces</i>	The sum of all current external forces that are applied per particle/ sphere.

7.11.2.14 InitSphereInverseMasses()

```
void GuidewireSim.InitializationStep.InitSphereInverseMasses (
```

```

    int spheresCount,
    out float[] sphereInverseMasses )

```

Initializes `sphereInverseMasses` with the default value of one at the start of the simulation.

Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>sphereInverseMasses</i>	The constant inverse masses of each sphere.

7.11.2.15 InitSpherePositionPredictions()

```

void GuidewireSim.InitializationStep.InitSpherePositionPredictions (
    GameObject[] spheres,
    int spheresCount,
    out Vector3[] spherePositionPredictions )

```

Initializes `spherePositionPredictions` with the default value of zero at the start of the simulation.

Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere.

7.11.2.16 InitSpherePositions()

```

void GuidewireSim.InitializationStep.InitSpherePositions (
    GameObject[] spheres,
    int spheresCount,
    out Vector3[] spherePositions )

```

Initializes `spherePositions` with the positions of `spheres` at the start of the simulation.

Parameters

	<i>spheres</i>	All spheres that are part of the guidewire.
	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
out	<i>spherePositions</i>	The position at the current frame of each sphere.

Requirements `spheresCount` should be at least one.

7.11.2.17 InitSphereVelocities()

```
void GuidewireSim.InitializationStep.InitSphereVelocities (
    int spheresCount,
    out Vector3[] sphereVelocities )
```

Initializes *sphereVelocities* with the default value of zero at the start of the simulation.

Parameters

	<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
out	<i>sphereVelocities</i>	The velocity of the current frame of each sphere.

Note

Velocities are set to zero at the start of the simulation.

7.11.2.18 InitWorldSpaceBasis()

```
void GuidewireSim.InitializationStep.InitWorldSpaceBasis (
    out BSM.Quaternion[] worldSpaceBasis )
```

Initializes the world space basis vectors (1, 0, 0), (0, 1, 0), (0, 0, 1) as embedded quaternions with scalar part zero.

Parameters

out	<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.
-----	------------------------	---

7.11.3 Member Data Documentation

7.11.3.1 collisionHandler

[CollisionHandler](#) GuidewireSim.InitializationStep.collisionHandler [private]

The component [CollisionHandler](#) that solves all collisions.

7.11.3.2 materialDensity

float GuidewireSim.InitializationStep.materialDensity = 7860 [private]

The density of the rod material. The value 7960 is taken from Table 2 of the CoRdE paper.

7.11.3.3 materialRadius

```
float GuidewireSim.InitializationStep.materialRadius = 0.001f [private]
```

The radius of the cross-section of the rod. Tha value 0.001 or 1mm is taken from Table 2 of the CoRdE paper.

7.11.3.4 mathHelper

```
MathHelper GuidewireSim.InitializationStep.mathHelper [private]
```

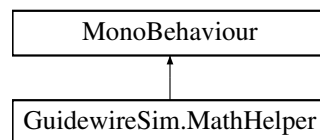
The component [MathHelper](#) that provides math related helper functions.

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

- [InitializationStep.cs](#)

7.12 GuidewireSim.MathHelper Class Reference

Inheritance diagram for GuidewireSim.MathHelper:



Public Member Functions

- void [CalculateCylinderPositions](#) (int cylinderCount, Vector3[] spherePositions, Vector3[] cylinderPositions)
- Vector3 [MatrixVectorMultiplication](#) (float[,] matrix, Vector3 vector)
- Vector3 [ColumnVectorMatrixMultiplication](#) (Vector3 columnVector, float[,] matrix)
- float[,] [VectorColumnVectorMultiplication](#) (Vector3 vectorOne, Vector3 columnVectorTwo)
- float[,] [ScalarMatrixMultiplication](#) (float scalar, float[,] matrix)
- float[,] [MatrixInverse](#) (float[,] matrix)
- BSM.Quaternion [EmbeddedVector](#) (Vector3 vector)
- Vector3 [ImaginaryPart](#) (BSM.Quaternion quaternion)
- Quaternion [QuaternionConversionFromBSM](#) (BSM.Quaternion bsmQuaternion)
- BSM.Quaternion [QuaternionConversionToBSM](#) (Quaternion quaternion)
- Vector3 [DiscreteDarbouxVector](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength)
- float [DarbouxSignFactor](#) (Vector3 currentDarbouxVector, Vector3 restDarbouxVector)
- float [VectorLength](#) (Vector3 vector)
- float [QuaternionLength](#) (BSM.Quaternion quaternion)
- float [RodElementLengthDeviation](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, float default← RodElementLength)
- float [StretchConstraintDeviation](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, BSM.Quaternion e_3, float rodElementLength, bool logIntermediateResults=false)
- float [BendTwistConstraintDeviation](#) (BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, bool logIntermediateResults=false)
- float [RodElementLength](#) (Vector3 particlePositionOne, Vector3 particlePositionTwo)
- Vector3[][] [UpdateDirectors](#) (int cylinderCount, BSM.Quaternion[] cylinderOrientations, Vector3[][] directors, BSM.Quaternion[] worldSpaceBasis)
- BSM.Quaternion [RandomUnitQuaternion](#) ()
- float [GetGaussianRandomNumber](#) ()

Private Member Functions

- float [SquaredNorm](#) (Vector3 vector)

7.12.1 Detailed Description

This class provides various helper methods for calculation.

7.12.2 Member Function Documentation

7.12.2.1 BendTwistConstraintDeviation()

```
float GuidewireSim.MathHelper.BendTwistConstraintDeviation (
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength,
    Vector3 discreteRestDarbouxVector,
    bool logIntermediateResults = false )
```

Returns the deviation of the bend twist constraint from zero.

Parameters

<i>orientationOne</i>	q of the equation (32).
<i>orientationTwo</i>	u of the equation (32).
<i>rodElementLength</i>	The Rod Element Length between <i>orientationOne</i> and <i>orientationTwo</i> . Used to calculate ℓ of the equation (32).
<i>discreteRestDarbouxVector</i>	\mathcal{D}^0 of the equation (32).
<i>logIntermediateResults</i>	Whether to output several logs that contain intermediate results of the calculation. Default is false.

Returns

The Deviation of the calculated bend twist constraint and zero.

Note

Check the Position and Orientation Based Cosserat Rods Paper (2016), equation (32), for more information on the bend twist constraint.

7.12.2.2 CalculateCylinderPositions()

```
void GuidewireSim.MathHelper.CalculateCylinderPositions (
    int cylinderCount,
```

```
Vector3[] spherePositions,  
Vector3[] cylinderPositions )
```

Calculates `cylinderPositions` as the middle points of two adjacent spheres.

Parameters

<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>spherePositions</i>	The position at the current frame of each sphere.
<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

Note

`cylinderPositions` is not marked as an out parameter, since `cylinderPositions` is not initialized in this method, but its values are changed.

7.12.2.3 ColumnVectorMatrixMultiplication()

```
Vector3 GuidewireSim.MathHelper.ColumnVectorMatrixMultiplication (
    Vector3 columnVector,
    float matrix[, ] )
```

Calculates the multiplication of $x^T M$, where M is the `matrix`, and x^T is the `columnVector` input.

Parameters

<i>matrix</i>	The matrix to be multiplied with the vector.
<i>vector</i>	The column vector x^T to be multiplied with the matrix.

Returns

The multiplication $x^T M$.

Requirements `matrix` must be a 3×3 matrix.

Attention

The input vector and the output vector are both column vectors.

7.12.2.4 DarbouxSignFactor()

```
float GuidewireSim.MathHelper.DarbouxSignFactor (
    Vector3 currentDarbouxVector,
    Vector3 restDarbouxVector )
```

Calculates the sign factor of the current discrete Darboux Vector and the rest Darboux Vector of the same orientations.

Note

Check the Position and Orientation Based Cosserat Rods Paper (2016) for more information on the sign factor.

Parameters

<i>currentDarbouxVector</i>	The discrete Darboux Vector of two fixed orientations at the current frame.
<i>restDarbouxVector</i>	The rest Darboux Vector of the same two orientations at frame 0.

Returns

The Sign Factor between these two entities.

7.12.2.5 DiscreteDarbouxVector()

```
Vector3 GuidewireSim.MathHelper.DiscreteDarbouxVector (
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength )
```

Calculates the discrete Darboux Vector of two adjacent orientations *orientationOne*, *orientationTwo*.

Parameters

<i>orientationOne</i>	The orientation with the lower index, e.g. <i>i</i> .
<i>orientationTwo</i>	The orientation with the higher index, e.g. <i>i</i> + 1.
<i>rodElementLength</i>	The distance between two spheres, also the distance between two orientations.

Returns

The discrete Darboux Vector between *orientationOne* and *orientationTwo*.

Note

There is only cylinderCount - 1 many darboux vectors. The *i*-th Darboux Vector is between orientation *i* and orientation *i*+1.

Attention

The order in which the orientations are entered matters. The Darboux Vector of q_1, q_2 is not the same as the Darboux Vector of q_2, q_1 .

7.12.2.6 EmbeddedVector()

```
BSM.Quaternion GuidewireSim.MathHelper.EmbeddedVector (
    Vector3 vector )
```

Returns a quaternion that is the embedded vector with scalar part zero.

Example

$$(x, y, z) \mapsto (x, y, z, 0).$$

Parameters

<i>vector</i>	The vector to be embedded.
---------------	----------------------------

Returns

The quaternion that is the embedded vector with scalar part zero.

7.12.2.7 GetGaussianRandomNumber()

```
float GuidewireSim.MathHelper.GetGaussianRandomNumber ( )
```

Provides a sample from $\mathcal{N}(0, 1)$ by using the Marsaglia polar method to transform a uniform distribution to a normal distribution.

Returns

A sample from $\mathcal{N}(0, 1)$.

Note

To understand this method, google the Marsaglia polar method. Note that unity does not provide a function to generate a random number following a gaussian distribution.

7.12.2.8 ImaginaryPart()

```
Vector3 GuidewireSim.MathHelper.ImaginaryPart (
    BSM.Quaternion quaternion )
```

Returns the imaginary part of a `quaternion`.

Example

$$(x, y, z, w) \mapsto (x, y, z).$$

Parameters

<i>quaternion</i>	The quaternion whose imaginary part to return.
-------------------	--

Returns

The imaginary part of a `quaternion`.

7.12.2.9 MatrixInverse()

```
float[,] GuidewireSim.MathHelper.MatrixInverse (
    float matrix[,] )
```

Calculates the inverse of a 3×3 matrix M .

Parameters

<i>matrix</i>	The matrix to
<i>columnVectorTwo</i>	The column vector x^T to be multiplied.

Returns

The resulting 3×3 matrix of the multiplication xv^T .

7.12.2.10 MatrixVectorMultiplication()

```
Vector3 GuidewireSim.MathHelper.MatrixVectorMultiplication (
    float matrix[,],
    Vector3 vector )
```

Calculates the multiplication of Mx , where M is the matrix, and x is the vector input.

Parameters

<i>matrix</i>	The matrix to be multiplied with the vector.
<i>vector</i>	The vector to be multiplied with the matrix.

Returns

The multiplication Mx .

Requirements `matrix` must be a 3×3 matrix.

7.12.2.11 QuaternionConversionFromBSM()

```
Quaternion GuidewireSim.MathHelper.QuaternionConversionFromBSM (
    BSM.Quaternion bsmQuaternion )
```

Takes as input a BSM.Quaternion and returns a UnityEngine.Quaternion.

Parameters

<i>bsmQuaternion</i>	The BSM.Quaternion to be converted.
----------------------	-------------------------------------

Returns

The converted UnityEngine.Quaternion.

7.12.2.12 QuaternionConversionToBSM()

```
BSM.Quaternion GuidewireSim.MathHelper.QuaternionConversionToBSM (
    Quaternion quaternion )
```

Takes as input a UnityEngine.Quaternion and returns a BSM.Quaternion.

Parameters

<i>bsmQuaternion</i>	The UnityEngine.Quaternion to be converted.
----------------------	---

Returns

The converted BSM.Quaternion.

7.12.2.13 QuaternionLength()

```
float GuidewireSim.MathHelper.QuaternionLength (
    BSM.Quaternion quaternion )
```

Returns the quaternion length of quaternion, i.e. $\sqrt{x^2 + y^2 + z^2 + w^2}$.

Parameters

<i>quaternion</i>	The quaternion whose length to return.
-------------------	--

Returns

The quaternion length of quaternion.

7.12.2.14 RandomUnitQuaternion()

```
BSM.Quaternion GuidewireSim.MathHelper.RandomUnitQuaternion ( )
```

Provides a random unit quaternion drawn from a gaussian distribution.

Returns

A random unit quaternion drawn from a gaussian distribution.

Note

This works by drawing four random, gaussian distributed, numbers, and filling the components of the quaternion with these numbers. Mathematically, this is equal to drawing a quaternion from a gaussian distribution in \mathcal{R}^4 , since the joint distribution of gaussian samples is again gaussian.

Requirements The length of the drawn quaternion is approximately equal to one.

7.12.2.15 RodElementLength()

```
float GuidewireSim.MathHelper.RodElementLength (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo )
```

Calculates the rod element length between *particlePositionOne* and *particlePositionTwo*.

Parameters

<i>particlePositionOne</i>	The first particle of the rod element under consideration.
<i>particlePositionTwo</i>	The first particle of the rod element under consideration.

Returns

The rod element length.

7.12.2.16 RodElementLengthDeviation()

```
float GuidewireSim.MathHelper.RodElementLengthDeviation (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    float defaultRodElementLength )
```

Returns the deviation between the actual distance of *particlePositionOne* and *particlePositionTwo* (current Rod Element Length) and the *defaultRodElementLength*.

Parameters

<i>particlePositionOne</i>	The first particle under consideration for the rod element length.
<i>particlePositionTwo</i>	The first particle under consideration for the rod element length.
<i>defaultRodElementLength</i>	The rod element length at rest state (i.e. frame 0) between these two particles.

Returns

The deviation between the actual rod element length and the default rod element length.

7.12.2.17 ScalarMatrixMultiplication()

```
float[,] GuidewireSim.MathHelper.ScalarMatrixMultiplication (
    float scalar,
    float matrix[,] )
```

Calculates the multiplication of aM , where a is a scalar , and M is a 3×3 matrix.

Parameters

<i>scalar</i>	The scalar to be multiplied.
<i>matrix</i>	The matrix M to be multiplied.

Returns

The resulting 3×3 matrix of the multiplication aM .

7.12.2.18 SquaredNorm()

```
float GuidewireSim.MathHelper.SquaredNorm (
    Vector3 vector ) [private]
```

Returns the squared norm of a vector.

Parameters

<i>vector</i>	The vector whose squared norm to return.
---------------	--

Returns

The Squared norm of *vector*.

7.12.2.19 StretchConstraintDeviation()

```
float GuidewireSim.MathHelper.StretchConstraintDeviation (
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
```

```
BSM.Quaternion e_3,
float rodElementLength,
bool logIntermediateResults = false )
```

Returns the deviation of the stretch constraint from zero.

Parameters

<i>particlePositionOne</i>	p_1 of the equation (31).
<i>particlePositionTwo</i>	p_2 of the equation (31).
<i>orientation</i>	q of the equation (31).
<i>e_3</i>	e_3 of the equation (31).
<i>rodElementLength</i>	l of the equation (31).
<i>logIntermediateResults</i>	Whether to output several logs that contain intermediate results of the calculation. Default is false.

Returns

The Deviation of the calculated stretch constraint and zero.

Note

Check the Position and Orientation Based Cosserat Rods Paper (2016), equation (31), for more information on the stretch constraint.

7.12.2.20 UpdateDirectors()

```
Vector3[][] GuidewireSim.MathHelper.UpdateDirectors (
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    Vector3 directors[][],
    BSM.Quaternion[] worldSpaceBasis )
```

Updates all directors of each orientation at the update step of the simulation loop.

Example

The directors d_1, d_2, d_3 are calculated as $d_i = q \cdot e_i \cdot \bar{q}$ for each orientation q , where e_i is the i -th world space basis vector. In quaternion calculus, this means rotating e_i by q .

Parameters

<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <code>cylinderOrientationPredictions</code> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.
<i>worldSpaceBasis</i>	The three basis vectors of the world coordinate system.

Returns

All directors of each orientation.

7.12.2.21 VectorColumnVectorMultiplication()

```
float[,] GuidewireSim.MathHelper.VectorColumnVectorMultiplication (
    Vector3 vectorOne,
    Vector3 columnVectorTwo )
```

Calculates the multiplication of xv^T , where x is a (row) vector , and v^T is a column vector.

Parameters

<i>vectorOne</i>	The (row) vector to be multiplied.
<i>columnVectorTwo</i>	The column vector x^T to be multiplied.

Returns

The resulting 3×3 matrix of the multiplication xv^T .

7.12.2.22 VectorLength()

```
float GuidewireSim.MathHelper.VectorLength (
    Vector3 vector )
```

Returns the vector length of `vector`, i.e. $\sqrt{x_1^2 + x_2^2 + x_3^2}$ for a three-dimensional vector.

Parameters

<i>vector</i>	The vector whose length to return.
---------------	------------------------------------

Returns

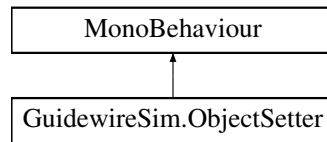
The vector length of `vector`.

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

- [MathHelper.cs](#)

7.13 GuidewireSim.ObjectSetter Class Reference

Inheritance diagram for GuidewireSim.ObjectSetter:



Public Member Functions

- void [SetSpherePositions](#) (GameObject[] spheres, int spheresCount, Vector3[] spherePositions)
- void [SetCylinderPositions](#) (GameObject[] cylinders, int cylinderCount, Vector3[] cylinderPositions)
- void [SetCylinderOrientations](#) (GameObject[] cylinders, int cylinderCount, BSM.Quaternion[] cylinderOrientations, Vector3[][] directors)

Private Member Functions

- void [Awake](#) ()

Private Attributes

- [MathHelper](#) mathHelper

The component [MathHelper](#) that provides math related helper functions.

7.13.1 Detailed Description

This class is responsible for setting the transformation positions of the GameObjects in the scene to their respective simulation data like `spherePositions`.

7.13.2 Member Function Documentation

7.13.2.1 Awake()

```
void GuidewireSim.ObjectSetter.Awake ( ) [private]
```

7.13.2.2 SetCylinderOrientations()

```
void GuidewireSim.ObjectSetter.SetCylinderOrientations (
    GameObject[] cylinders,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    Vector3 directors[][] )
```

Rotates each cylinder GameObject such that its centerline is parallel with the line segment that is spanned by the two adjacent sphere's center of masses.

Parameters

<i>cylinders</i>	All cylinders that are part of the guidewire.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>directors</i>	The orthonormal basis of each orientation element / cylinder, also called directors.

Note

appliedTransformation is the rotation that aligns the y-axis of the cylinder with the z-axis of the orientations (the third director). This is needed, because the y-axis of the cylinder is parallel with its centerline, while the z-axis of the orientations (the third director) is also defined as being parallel with the cylinder's centerline. Thus *appliedTransformation* is necessary.

7.13.2.3 SetCylinderPositions()

```
void GuidewireSim.ObjectSetter.SetCylinderPositions (
    GameObject[] cylinders,
    int cylinderCount,
    Vector3[] cylinderPositions )
```

Sets the positions of the GameObjects *cylinders* to their respective *cylinderPositions*.

Parameters

<i>cylinders</i>	All cylinders that are part of the guidewire.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderPositions</i>	The position/ center of mass of each cylinder.

7.13.2.4 SetSpherePositions()

```
void GuidewireSim.ObjectSetter.SetSpherePositions (
    GameObject[] spheres,
    int spheresCount,
    Vector3[] spherePositions )
```

Sets the positions of the GameObjects *spheres* to their respective *spherePositions*.

Parameters

<i>spheres</i>	All spheres that are part of the guidewire.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositions</i>	The position at the current frame of each sphere.

7.13.3 Member Data Documentation

7.13.3.1 mathHelper

`MathHelper` GuidewireSim.ObjectSetter.mathHelper [private]

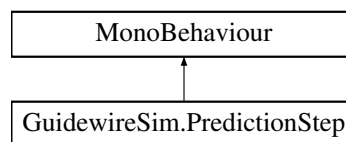
The component `MathHelper` that provides math related helper functions.

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

- [ObjectSetter.cs](#)

7.14 GuidewireSim.PredictionStep Class Reference

Inheritance diagram for GuidewireSim.PredictionStep:



Public Member Functions

- `Vector3[]` [PredictSphereVelocities](#) (`Vector3[]` sphereVelocities, `float[]` sphereInverseMasses, `Vector3[]` sphereExternalForces)
- `Vector3[]` [PredictSpherePositions](#) (`Vector3[]` spherePositionPredictions, `int` spheresCount, `Vector3[]` spherePositions, `Vector3[]` sphereVelocities)
- `Vector3[]` [PredictAngularVelocities](#) (`Vector3[]` cylinderAngularVelocities, `int` cylinderCount, `float[,]` inertiaTensor, `Vector3[]` cylinderExternalTorques, `float[,]` inverseInertiaTensor)
- `BSM.Quaternion[]` [PredictCylinderOrientations](#) (`BSM.Quaternion[]` cylinderOrientationPredictions, `int` cylinderCount, `Vector3[]` cylinderAngularVelocities, `BSM.Quaternion[]` cylinderOrientations)

Private Member Functions

- `void` [Awake](#) ()

Private Attributes

- `MathHelper` `mathHelper`

The component `MathHelper` that provides math related helper functions.

7.14.1 Detailed Description

This class implements the prediction step of the algorithm.

7.14.2 Member Function Documentation

7.14.2.1 Awake()

```
void GuidewireSim.PredictionStep.Awake ( ) [private]
```

7.14.2.2 PredictAngularVelocities()

```
Vector3[] GuidewireSim.PredictionStep.PredictAngularVelocities (
    Vector3[] cylinderAngularVelocities,
    int cylinderCount,
    float inertiaTensor[,],
    Vector3[] cylinderExternalTorques,
    float inverseInertiaTensor[,] )
```

Calculates the predictions for the angular velocities for the prediction step of the algorithm.

Parameters

<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>inertiaTensor</i>	The inertia tensor. Entries are approximates as in the CoRdE paper.
<i>cylinderExternalTorques</i>	The sum of all current external torques that are applied per orientation element/ cylinder.
<i>inverseInertiaTensor</i>	The inverse of <i>inertiaTensor</i> .

Returns

The angular velocity of the current frame of each orientation element/ cylinder, i.e. *cylinderAngularVelocities*.

Note

The predictions are again stored in *cylinderAngularVelocities*.

7.14.2.3 PredictCylinderOrientations()

```
BSM.Quaternion[] GuidewireSim.PredictionStep.PredictCylinderOrientations (
    BSM.Quaternion[] cylinderOrientationPredictions,
    int cylinderCount,
    Vector3[] cylinderAngularVelocities,
    BSM.Quaternion[] cylinderOrientations )
```

Calculates the predictions for the cylinder orientations for the prediction step of the algorithm.

Parameters

<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.

Returns

The prediction of the orientation of each cylinder at its center of mass, i.e. *cylinderOrientationPredictions*.

7.14.2.4 PredictSpherePositions()

```
Vector3[] GuidewireSim.PredictionStep.PredictSpherePositions (
    Vector3[] spherePositionPredictions,
    int spheresCount,
    Vector3[] spherePositions,
    Vector3[] sphereVelocities )
```

Calculates the predictions for the sphere positions for the prediction step of the algorithm.

Parameters

<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositions</i>	The position at the current frame of each sphere.
<i>sphereVelocities</i>	The velocity of the current frame of each sphere.

Returns

The prediction of the position at the current frame of each sphere, i.e. *spherePositionPredictions*.

7.14.2.5 PredictSphereVelocities()

```
Vector3[] GuidewireSim.PredictionStep.PredictSphereVelocities (
    Vector3[] sphereVelocities,
    float[] sphereInverseMasses,
    Vector3[] sphereExternalForces )
```

Calculates the predictions for the sphere velocities for the prediction step of the algorithm.

Parameters

<i>sphereVelocities</i>	The velocity of the current frame of each sphere.
<i>sphereInverseMasses</i>	The constant inverse masses of each sphere.
<i>sphereExternalForces</i>	The sum of all current external forces that are applied per particle/ sphere.

Returns

The predictions of the positions of the spheres, i.e. `spherePositionPredictions`.

Note

The predictions are again stored in `sphereVelocities`.

7.14.3 Member Data Documentation

7.14.3.1 mathHelper

`MathHelper` `GuidewireSim.PredictionStep.mathHelper` [private]

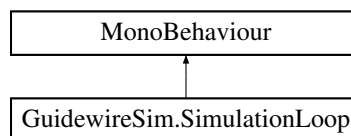
The component `MathHelper` that provides math related helper functions.

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

- `PredictionStep.cs`

7.15 GuidewireSim.SimulationLoop Class Reference

Inheritance diagram for `GuidewireSim.SimulationLoop`:



Public Member Functions

- void `PerformSimulationLoop` ()

Public Attributes

- `GameObject[]` [spheres](#)
- `Vector3[]` [spherePositions](#)
The position at the current frame of each sphere.
- `Vector3[]` [sphereVelocities](#)
The velocity of the current frame of each sphere. Initalized with zero entries.
- `float[]` [sphereInverseMasses](#)
- `Vector3[]` [sphereExternalForces](#)
The sum of all current external forces that are applied per particle/ sphere.
- `Vector3[]` [cylinderPositions](#)
The center of mass of each cylinder.
- `float[]` [cylinderScalarWeights](#)
- `Vector3[]` [cylinderExternalTorques](#)
The sum of all current external torques that are applied per orientation element/ cylinder.
- `Vector3[][]` [directors](#)
- `bool` [solveStretchConstraints](#) = true
Whether or not to perform the constraint solving of the stretch constraint.
- `bool` [solveBendTwistConstraints](#) = true
Whether or not to perform the constraint solving of the bend twist constraint.
- `bool` [solveCollisionConstraints](#) = true
Whether or not to perform the constraint solving of collision constraints.

Properties

- `bool` [ExecuteSingleLoopTest](#) = false [get, set]
- `int` [ConstraintSolverSteps](#) [get, set]
- `float` [TimeStep](#) [get]
- `int` [SpheresCount](#) [get, private set]
The count of all [spheres](#) of the guidewire.
- `int` [CylinderCount](#) [get, private set]
The count of all [cylinders](#) of the guidewire.

Private Member Functions

- `void` [Awake](#) ()
- `void` [Start](#) ()
- `void` [FixedUpdate](#) ()
- `void` [PerformInitializationStep](#) ()
- `void` [PerformConstraintSolvingStep](#) ()
- `void` [PerformUpdateStep](#) ()
- `void` [PerformPredictionStep](#) ()
- `void` [AdaptCalculations](#) ()
- `void` [SetCollidersStep](#) ()

Private Attributes

- [InitializationStep initializationStep](#)
The component *InitializationStep* that is responsible for initializing the simulation.
- [PredictionStep predictionStep](#)
The component *PredictionStep* that is responsible for executing the Prediction Step of the algorithm.
- [ConstraintSolvingStep constraintSolvingStep](#)
- [UpdateStep updateStep](#)
The component *UpdateStep* that is responsible for executing the Update Step of the algorithm.
- [ObjectSetter objectSetter](#)
The component *ObjectSetter* that is responsible for setting all positions and rotations the the GameObjects.
- [MathHelper mathHelper](#)
The component *MathHelper* that provides math related helper functions.
- [CollisionSolvingStep collisionSolvingStep](#)
The component *CollisionSolvingStep* that solves all collisions.
- [CollisionHandler collisionHandler](#)
The component *CollisionHandler* that tracks all collisions.
- `GameObject[] cylinders`
- `Vector3[] spherePositionPredictions`
The prediction of the position at the current frame of each sphere.
- `BSM.Quaternion[] cylinderOrientations`
The orientation of each cylinder at its center of mass.
- `BSM.Quaternion[] cylinderOrientationPredictions`
The prediction of the orientation of each cylinder at its center of mass.
- `Vector3[] discreteRestDarbouxVectors`
- `Vector3[] cylinderAngularVelocities`
- `float[,] inertiaTensor`
The inertia tensor. Entries are approximates as in the CoRdE paper.
- `float[,] inverseInertiaTensor`
The inverse of *inertiaTensor*.
- `BSM.Quaternion[] worldSpaceBasis`
- `float rodElementLength = 10f`
- `int constraintSolverSteps = 1000`
- `float timeStep = 0.01f`

7.15.1 Detailed Description

This class executes the outer simulation loop of the algorithm and calls the implementations of each algorithm step and manages all coherent data.

7.15.2 Member Function Documentation

7.15.2.1 AdaptCalculations()

```
void GuidewireSim.SimulationLoop.AdaptCalculations ( ) [private]
```

Adapts the data to the Unity GameObjects. For example, sets the positions of the GameObjects [spheres](#) to [spherePositions](#).

Requirements Sets the positions of the GameObjects [spheres](#) to [spherePositions](#).
Calculates [cylinderPositions](#) based on [spherePositions](#).
Sets the positions of the GameObjects [cylinders](#) to [cylinderPositions](#).
Sets the rotations of the GameObjects [cylinders](#) to [cylinderOrientations](#).

7.15.2.2 Awake()

```
void GuidewireSim.SimulationLoop.Awake ( ) [private]
```

7.15.2.3 FixedUpdate()

```
void GuidewireSim.SimulationLoop.FixedUpdate ( ) [private]
```

Requirements Execute the simulation loop if and only if [ExecuteSingleLoopTest](#) is false.

7.15.2.4 PerformConstraintSolvingStep()

```
void GuidewireSim.SimulationLoop.PerformConstraintSolvingStep ( ) [private]
```

Performs the constraint solving step of the algorithm.

Requirements Performs the constraint solving of every constraint `#solverStep` many times.
Solve stretch constraints, if and only if [solveStretchConstraints](#) is true.
Solve bend twist constraints, if and only if [solveBendTwistConstraints](#) is true.
If [solveStretchConstraints](#), then [SpheresCount](#) is at least two.
If [solveStretchConstraints](#), then [CylinderCount](#) is at least one.
If [solveBendTwistConstraints](#), then [SpheresCount](#) is at least three.
If [solveBendTwistConstraints](#), then [CylinderCount](#) is at least two.
If [solveStretchConstraints](#), after the step is complete the deviation between the actual rod element length and the default (rest state) [rodElementLength](#) should be close to zero.
If [solveStretchConstraints](#), after the step is complete the deviation of the stretch constraint to zero should be close to zero.
If [solveCollisionConstraints](#), after this step is complete clear the list `registeredCollisions`, since these collisions are now resolved.

7.15.2.5 PerformInitializationStep()

```
void GuidewireSim.SimulationLoop.PerformInitializationStep ( ) [private]
```

Calls every step that is mandatory to declare and initialize all data.

Requirements Set [SpheresCount](#) to the length of [spheres](#).
Set [CylinderCount](#) to the length of [cylinders](#).
Call every init method of [initializationStep](#).

7.15.2.6 PerformPredictionStep()

```
void GuidewireSim.SimulationLoop.PerformPredictionStep ( ) [private]
```

Performs the prediction step of the algorithm.

Requirements Predict the [sphereVelocities](#).
Predict the [spherePositionPredictions](#).
Predict the [cylinderAngularVelocities](#).
Predict the [cylinderOrientationPredictions](#).

7.15.2.7 PerformSimulationLoop()

```
void GuidewireSim.SimulationLoop.PerformSimulationLoop ( )
```

Performs the outer simulation loop of the algorithm.

Note

In a late version, CollisionDetection and GenerateCollisionConstraints will be added to the algorithm.

7.15.2.8 PerformUpdateStep()

```
void GuidewireSim.SimulationLoop.PerformUpdateStep ( ) [private]
```

Performs the update step of the algorithm.

Requirements Update [sphereVelocities](#).
Update [spherePositions](#).
Update [cylinderAngularVelocities](#).
Update [cylinderOrientations](#).
Update [directors](#).

7.15.2.9 SetCollidersStep()

```
void GuidewireSim.SimulationLoop.SetCollidersStep ( ) [private]
```

Sets the position of the collider of each sphere of the guidewire to the sphere's position prediction.

7.15.2.10 Start()

```
void GuidewireSim.SimulationLoop.Start ( ) [private]
```

7.15.3 Member Data Documentation

7.15.3.1 collisionHandler

```
CollisionHandler GuidewireSim.SimulationLoop.collisionHandler [private]
```

The component [CollisionHandler](#) that tracks all collisions.

7.15.3.2 collisionSolvingStep

```
CollisionSolvingStep GuidewireSim.SimulationLoop.collisionSolvingStep [private]
```

The component [CollisionSolvingStep](#) that solves all collisions.

7.15.3.3 constraintSolverSteps

```
int GuidewireSim.SimulationLoop.constraintSolverSteps = 1000 [private]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

Attention

This value must be positive.

7.15.3.4 constraintSolvingStep

```
ConstraintSolvingStep GuidewireSim.SimulationLoop.constraintSolvingStep [private]
```

The component [ConstraintSolvingStep](#) that is responsible for correcting the predictions with the collision and model constraints.

7.15.3.5 cylinderAngularVelocities

```
Vector3 [] GuidewireSim.SimulationLoop.cylinderAngularVelocities [private]
```

The angular velocity of the current frame of each orientation element/ cylinder. Initalized with zero entries.

7.15.3.6 cylinderExternalTorques

```
Vector3 [] GuidewireSim.SimulationLoop.cylinderExternalTorques
```

The sum of all current external torques that are applied per orientation element/ cylinder.

7.15.3.7 cylinderOrientationPredictions

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.cylinderOrientationPredictions [private]
```

The prediction of the orientation of each cylinder at its center of mass.

7.15.3.8 cylinderOrientations

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.cylinderOrientations [private]
```

The orientation of each cylinder at its center of mass.

7.15.3.9 cylinderPositions

```
Vector3 [] GuidewireSim.SimulationLoop.cylinderPositions
```

The center of mass of each cylinder.

7.15.3.10 cylinders

```
GameObject [] GuidewireSim.SimulationLoop.cylinders [private]
```

All cylinders that are part of the guidewire.

Attention

The order in which the cylinders are assigned matters. Assign them such that two adjacent cylinders are adjacent in the array as well.

7.15.3.11 cylinderScalarWeights

```
float [ ] GuidewireSim.SimulationLoop.cylinderScalarWeights
```

The constant scalar weights of each orientation/ quaternion similar to [sphereInverseMasses](#).

Note

Set to 1 for moving orientations (so that angular motion can be applied) and to 0 for fixed orientations.

7.15.3.12 directors

```
Vector3 [ ] [ ] GuidewireSim.SimulationLoop.directors
```

The orthonormal basis of each orientation element / cylinder, also called directors.

Note

In the 0th row are the first directors of each orientation element, not in the 1th row. Example: The (i, j)th element holds the (i-1)th director of orientation element j.

7.15.3.13 discreteRestDarbouxVectors

```
Vector3 [ ] GuidewireSim.SimulationLoop.discreteRestDarbouxVectors [private]
```

The discrete Darboux Vector at the rest configuration, i.e. at frame 0.

Note

It is important to only take the imaginary part in the calculation for the discrete Darboux Vector, thus we only save it as a Vector3. To use it in a quaternion setting, embedd the Vector3 with scalar part 0, i.e. with EmbeddedVector().

Attention

There is only CylinderCount - 1 many darboux vectors. The i-th Darboux Vector is between orientation i and orientation i+1.

7.15.3.14 inertiaTensor

```
float [,] GuidewireSim.SimulationLoop.inertiaTensor [private]
```

The inertia tensor. Entries are approximates as in the CoRdE paper.

7.15.3.15 initializationStep

`InitializationStep` GuidewireSim.SimulationLoop.initializationStep [private]

The component `InitializationStep` that is responsible for initializing the simulation.

7.15.3.16 inverseInertiaTensor

`float [,]` GuidewireSim.SimulationLoop.inverseInertiaTensor [private]

The inverse of `inertiaTensor`.

7.15.3.17 mathHelper

`MathHelper` GuidewireSim.SimulationLoop.mathHelper [private]

The component `MathHelper` that provides math related helper functions.

7.15.3.18 objectSetter

`ObjectSetter` GuidewireSim.SimulationLoop.objectSetter [private]

The component `ObjectSetter` that is responsible for setting all positions and rotations the the GameObjects.

7.15.3.19 predictionStep

`PredictionStep` GuidewireSim.SimulationLoop.predictionStep [private]

The component `PredictionStep` that is responsible for executing the Prediction Step of the algorithm.

7.15.3.20 rodElementLength

`float` GuidewireSim.SimulationLoop.rodElementLength = 10f [private]

The distance between two spheres, also the distance between two orientations. Also the length of one cylinder.

Note

This should be two times the radius of a sphere.

Attention

Make sure that the guidewire setup fulfills that the distance between two adjacent spheres is `rodElementLength`.

7.15.3.21 solveBendTwistConstraints

```
bool GuidewireSim.SimulationLoop.solveBendTwistConstraints = true
```

Whether or not to perform the constraint solving of the bend twist constraint.

7.15.3.22 solveCollisionConstraints

```
bool GuidewireSim.SimulationLoop.solveCollisionConstraints = true
```

Whether or not to perform the constraint solving of collision constraints.

7.15.3.23 solveStretchConstraints

```
bool GuidewireSim.SimulationLoop.solveStretchConstraints = true
```

Whether or not to perform the constraint solving of the stretch constraint.

7.15.3.24 sphereExternalForces

```
Vector3 [] GuidewireSim.SimulationLoop.sphereExternalForces
```

The sum of all current external forces that are applied per particle/ sphere.

7.15.3.25 sphereInverseMasses

```
float [] GuidewireSim.SimulationLoop.sphereInverseMasses
```

The constant inverse masses of each sphere.

Note

Set to 1 for moving spheres and to 0 for fixed spheres.

7.15.3.26 spherePositionPredictions

```
Vector3 [] GuidewireSim.SimulationLoop.spherePositionPredictions [private]
```

The prediction of the position at the current frame of each sphere.

7.15.3.27 spherePositions

```
Vector3 [] GuidewireSim.SimulationLoop.spherePositions
```

The position at the current frame of each sphere.

7.15.3.28 spheres

```
GameObject [] GuidewireSim.SimulationLoop.spheres
```

All spheres that are part of the guidewire.

Attention

The order in which the spheres are assigned matters. Assign them such that two adjacent spheres are adjacent in the array as well.

7.15.3.29 sphereVelocities

```
Vector3 [] GuidewireSim.SimulationLoop.sphereVelocities
```

The velocity of the current frame of each sphere. Initalized with zero entries.

7.15.3.30 timeStep

```
float GuidewireSim.SimulationLoop.timeStep = 0.01f [private]
```

The fixed time step in seconds at which the simulation runs.

Note

A lower timestep than 0.002 can not be guaranteed by the test hardware to be executed in time. Only choose a lower timestep if you are certain your hardware can handle it.

7.15.3.31 updateStep

```
UpdateStep GuidewireSim.SimulationLoop.updateStep [private]
```

The component [UpdateStep](#) that is responsible for executing the Update Step of the algorithm.

7.15.3.32 worldSpaceBasis

```
BSM.Quaternion [] GuidewireSim.SimulationLoop.worldSpaceBasis [private]
```

The three basis vectors of the world coordinate system as embedded quaternions with scalar part 0. E.g. the first basis vector is (1, 0, 0), the second (0, 1, 0) and the third (0, 0, 1).

7.15.4 Property Documentation

7.15.4.1 ConstraintSolverSteps

```
int GuidewireSim.SimulationLoop.ConstraintSolverSteps [get], [set]
```

7.15.4.2 CylinderCount

```
int GuidewireSim.SimulationLoop.CylinderCount [get], [private set]
```

The count of all [cylinders](#) of the guidewire.

7.15.4.3 ExecuteSingleLoopTest

```
bool GuidewireSim.SimulationLoop.ExecuteSingleLoopTest = false [get], [set]
```

Whether or not to execute the Single Loop Test, in which the outer simulation loop is exactly executed once.

7.15.4.4 SpheresCount

```
int GuidewireSim.SimulationLoop.SpheresCount [get], [private set]
```

The count of all [spheres](#) of the guidewire.

7.15.4.5 TimeStep

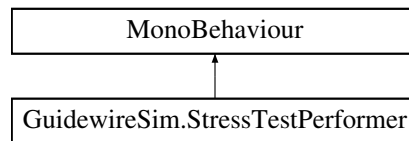
```
float GuidewireSim.SimulationLoop.TimeStep [get]
```

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

- [SimulationLoop.cs](#)

7.16 GuidewireSim.StressTestPerformer Class Reference

Inheritance diagram for GuidewireSim.StressTestPerformer:



Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformStressTests](#) ()
- IEnumerator [PerformStressTestOne](#) (float applyForceTime=1f)

Private Attributes

- [SimulationLoop](#) [simulationLoop](#)
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- bool [doStressTestOne](#) = false

7.16.1 Detailed Description

This class enables the user to test the impact of multiple external forces and external torques with one button within the Unity inspector.

Attention

In the current version, the user is not able to fix positions or orientations of the guidewire, which is necessary e.g. for stress test one.

7.16.2 Member Function Documentation

7.16.2.1 Awake()

```
void GuidewireSim.StressTestPerformer.Awake ( ) [private]
```

7.16.2.2 PerformStressTestOne()

```
IEnumerator GuidewireSim.StressTestPerformer.PerformStressTestOne (
    float applyForceTime = 1f ) [private]
```

Performs stress test one. This test fixes the position of one end of the guidewire, and applies `pullForce` at the other end for `applyForceTime` seconds, and then applies `- pullForce` for another `applyForceTime` seconds.

Parameters

<i>applyForceTime</i>	For how many seconds to apply the force to the particles.
-----------------------	---

Attention

In the current version, the user is not able to fix positions or orientations of the guidewire, which is necessary e.g. for stress test one.

Requirements Output a log message when no further forces are applied to the guidewire.

7.16.2.3 PerformStressTests()

```
void GuidewireSim.StressTestPerformer.PerformStressTests ( ) [private]
```

Performs each Stress Test whose respective serialized boolean is set to true in the Unity inspector.

7.16.2.4 Start()

```
void GuidewireSim.StressTestPerformer.Start ( ) [private]
```

7.16.3 Member Data Documentation

7.16.3.1 doStressTestOne

```
bool GuidewireSim.StressTestPerformer.doStressTestOne = false [private]
```

Whether to run Stress Test One. This test fixes the position of one end of the guidewire, and applies `pullForce` at the other end for `applyForceTime` seconds, and then applies `- pullForce` for another `applyForceTime` seconds.

7.16.3.2 simulationLoop

```
SimulationLoop GuidewireSim.StressTestPerformer.simulationLoop [private]
```

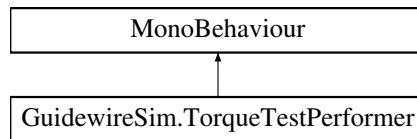
The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [StressTestPerformer.cs](#)

7.17 GuidewireSim.TorqueTestPerformer Class Reference

Inheritance diagram for GuidewireSim.TorqueTestPerformer:



Private Member Functions

- void [Awake](#) ()
- void [Start](#) ()
- void [PerformTorqueTests](#) ()
- void [PerformTorqueTestOne](#) ()
- IEnumerator [PerformTorqueTestTwo](#) (Vector3 [pullTorque](#), float applyTorqueTime=1f)
- IEnumerator [PerformTorqueTestThree](#) (Vector3 [pullTorque](#), float applyTorqueTime=10f)

Private Attributes

- [SimulationLoop](#) [simulationLoop](#)
The [SimulationLoop](#) component that executes all steps of the simulation loop.
- Vector3 [pullTorque](#) = new Vector3(0f, 0.3f, 0f)
- bool [doTorqueTestOne](#) = false
Whether to run Torque Test One. This test applies an external torque to one end of the guidewire.
- bool [doTorqueTestTwo](#) = false
- bool [doTorqueTestThree](#) = false

7.17.1 Detailed Description

This class enables the user to test the impact of external torques with one button within the Unity inspector.

7.17.2 Member Function Documentation

7.17.2.1 Awake()

```
void GuidewireSim.TorqueTestPerformer.Awake ( ) [private]
```

7.17.2.2 PerformTorqueTestOne()

```
void GuidewireSim.TorqueTestPerformer.PerformTorqueTestOne ( ) [private]
```

Performs torque test one. This test applies an external torque to one end of the guidewire.

7.17.2.3 PerformTorqueTests()

```
void GuidewireSim.TorqueTestPerformer.PerformTorqueTests ( ) [private]
```

Performs each Torque Test whose respective serialized boolean is set to true in the Unity inspector.

7.17.2.4 PerformTorqueTestThree()

```
IEnumerator GuidewireSim.TorqueTestPerformer.PerformTorqueTestThree (
    Vector3 pullTorque,
    float applyTorqueTime = 10f ) [private]
```

Performs torque test three. This test applies an external torque to one end of the guidewire and at the same time the opposite torque at the other end of the guidewire. The applied torque starts at 0 and linearly interpolates until it reaches pullTorque at applyTorqueTime seconds.

Parameters

<i>pullTorque</i>	The external torque that is applied to one end of the guidewire.
<i>applyTorqueTime</i>	For how many seconds to apply the torque to the orientations.

Requirements Output a log message when no further torques are applied to the guidewire.

7.17.2.5 PerformTorqueTestTwo()

```
IEnumerator GuidewireSim.TorqueTestPerformer.PerformTorqueTestTwo (
    Vector3 pullTorque,
    float applyTorqueTime = 1f ) [private]
```

Performs torque test two. This test applies an external torque to one end of the guidewire for a fixed amount of time and then the opposite torque at the same orientation for the same amount of time.

Parameters

<i>pullTorque</i>	The external torque that is applied to one end of the guidewire.
<i>applyTorqueTime</i>	For how many seconds to apply the torque to the orientations.

Requirements Output a log message when no further torques are applied to the guidewire.

7.17.2.6 Start()

```
void GuidewireSim.TorqueTestPerformer.Start ( ) [private]
```

7.17.3 Member Data Documentation

7.17.3.1 doTorqueTestOne

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestOne = false [private]
```

Whether to run Torque Test One. This test applies an external torque to one end of the guidewire.

7.17.3.2 doTorqueTestThree

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestThree = false [private]
```

Whether to run Torque Test Three. This test applies an external torque to one end of the guidewire and at the same time the opposite torque at the other end of the guidewire. The applied torque starts at 0 and linearly interpolates until it reaches `pullTorque` at `applyTorqueTime` seconds.

7.17.3.3 doTorqueTestTwo

```
bool GuidewireSim.TorqueTestPerformer.doTorqueTestTwo = false [private]
```

Whether to run Torque Test Two. This test applies an external torque to one end of the guidewire for a fixed amount of time and then the opposite torque at the same orientation for the same amount of time.

7.17.3.4 pullTorque

```
Vector3 GuidewireSim.TorqueTestPerformer.pullTorque = new Vector3(0f, 0.3f, 0f) [private]
```

The external torque that is applied to the respective parts of the guidewire, depending on the test.

7.17.3.5 simulationLoop

```
SimulationLoop GuidewireSim.TorqueTestPerformer.simulationLoop [private]
```

The [SimulationLoop](#) component that executes all steps of the simulation loop.

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

- [TorqueTestPerformer.cs](#)

7.18 UnitTest_SolveBendTwistConstraint Class Reference

Public Member Functions

- IEnumerator [PerformUnitTests](#) ()

Private Member Functions

- void [Test_SolveBendTwistConstraint](#) (int iterations, BSM.Quaternion orientationOne, BSM.Quaternion orientationTwo, float rodElementLength, Vector3 discreteRestDarbouxVector, [GuidewireSim.MathHelper](#) mathHelper, [GuidewireSim.ConstraintSolvingStep](#) constraintSolvingStep)

Private Attributes

- int [sampleSize](#) = 10
- int [constraintSolverSteps](#) = 50

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

7.18.1 Detailed Description

This class provides unit tests that test the method `SolveBendTwistConstraint()` of `ConstraintSolvingStep`. Executing this test once generates `sampleSize` many random value pairs and executes the unit test with each of these pairs.

7.18.2 Member Function Documentation

7.18.2.1 PerformUnitTests()

```
IEnumerator UnitTest_SolveBendTwistConstraint.PerformUnitTests ( )
```

Arranges all necessary data, generates [sampleSize](#) many random value pairs, and then passes all data to [Test_SolveBendTwistConstraint\(\)](#), where the unit tests are executed.

Note

Only tests the case that all rod elements are aligned at rest state. If you want to test deformed rods at rest state, change `discreteRestDarbouxVector` accordingly.

7.18.2.2 Test_SolveBendTwistConstraint()

```
void UnitTest_SolveBendTwistConstraint.Test_SolveBendTwistConstraint (
    int iterations,
    BSM.Quaternion orientationOne,
    BSM.Quaternion orientationTwo,
    float rodElementLength,
    Vector3 discreteRestDarbouxVector,
    GuidewireSim.MathHelper mathHelper,
    GuidewireSim.ConstraintSolvingStep constraintSolvingStep ) [private]
```

Executes `SolveBendTwistConstraint()` of `ConstraintSolvingStep` `iterations` many times for one values pair, and then asserts whether the results of the algorithm of `SolveBendTwistConstraint()` converged towards the expected values.

Parameters

<i>iterations</i>	The number of iterations that SolveBendTwistConstraint() of ConstraintSolvingStep is executed.
<i>orientationOne</i>	The first orientation for SolveBendTwistConstraint().
<i>orientationOne</i>	The second orientation for SolveBendTwistConstraint().
<i>rodElementLength</i>	The rod element length for SolveBendTwistConstraint().
<i>discreteRestDarbouxVector</i>	The discrete Darboux Vector at rest state for SolveBendTwistConstraint().
<i>mathHelper</i>	The component MathHelper.
<i>constraintSolvingStep</i>	The component ConstraintSolvingStep.

Requirements `orientationOne` and `orientationTwo` are still unit quaternions at the end of the test.

The deviation between the bend twist constraint and zero is lower than a reasonable tolerance, i.e. close to zero., which means that the algorithm of `SolveBendTwistConstraint()` converges towards the fulfillment of the bend twist constraint.

7.18.3 Member Data Documentation

7.18.3.1 constraintSolverSteps

```
int UnitTest_SolveBendTwistConstraint.constraintSolverSteps = 50 [private]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

7.18.3.2 sampleSize

```
int UnitTest_SolveBendTwistConstraint.sampleSize = 10 [private]
```

The number of value-pairs the test is executed with. E.g. if `sampleSize` is 10, then the unit test is executed with 10 randomly drawn value-pairs. A higher number needs more time to execute.

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

- [UnitTest_SolveBendTwistConstraint.cs](#)

7.19 UnitTest_SolveStretchConstraint Class Reference

Public Member Functions

- IEnumerator [PerformUnitTests](#) ()

Private Member Functions

- void [PickRandomPositions](#) (out Vector3 particlePositionOne, out Vector3 particlePositionTwo)
- void [Test_SolveStretchConstraint](#) (int iterations, Vector3 particlePositionOne, Vector3 particlePositionTwo, BSM.Quaternion orientation, [GuidewireSim.MathHelper](#) mathHelper, [GuidewireSim.ConstraintSolvingStep](#) constraintSolvingStep)

Private Attributes

- float [maximalDistanceOffset](#) = 1f
- int [sampleSize](#) = 10
- int [constraintSolverSteps](#) = 1000
How often the constraint solver iterates over each constraint during the Constraint Solving Step.
- float [rodElementLength](#) = 10f
The distance between two spheres, also the distance between two orientations.

7.19.1 Detailed Description

This class provides unit tests that test the method `SolveStretchConstraint()` of `ConstraintSolvingStep`. Executing this test once generates `sampleSize` many random value pairs and executes the unit test with each of these pairs.

7.19.2 Member Function Documentation

7.19.2.1 PerformUnitTests()

```
IEnumerator UnitTest_SolveStretchConstraint.PerformUnitTests ( )
```

Arranges all necessary data, generates [sampleSize](#) many random value pairs, and then passes all data to [Test_SolveStretchConstraint\(\)](#), where the unit tests are executed.

7.19.2.2 PickRandomPositions()

```
void UnitTest_SolveStretchConstraint.PickRandomPositions (
    out Vector3 particlePositionOne,
    out Vector3 particlePositionTwo ) [private]
```

Picks the first particle position uniformly distributed with $x, y, z \in [-5, 5]$ and the second uniformly distributed around the first position with a uniformly distributed distance in $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$.

Note

The method for picking the second position is inspired by <https://math.stackexchange.com/q/50482>

Parameters

out	<i>particlePositionOne</i>	The first particle position that got picked.
out	<i>particlePositionTwo</i>	The second particle position that got picked.

Requirements Picks the first particle position uniformly distributed so that $x, y, z \in [-5, 5]$.

Picks a distance between the two particles that is uniformly distributed in the interval $[rodElementLength - maximalDistanceOffset, rodElementLength + maximalDistanceOffset]$.

Picks the second particle position uniformly distributed on the surface of the sphere with center *particlePositionOne* and radius *startDistance*.

7.19.2.3 Test_SolveStretchConstraint()

```
void UnitTest_SolveStretchConstraint.Test_SolveStretchConstraint (
    int iterations,
    Vector3 particlePositionOne,
    Vector3 particlePositionTwo,
    BSM.Quaternion orientation,
    GuidewireSim.MathHelper mathHelper,
    GuidewireSim.ConstraintSolvingStep constraintSolvingStep ) [private]
```

Executes *SolveStretchConstraint()* of *ConstraintSolvingStep* *iterations* many times for one values pair, and then asserts whether the results of the algorithm of *SolveStretchConstraint()* converged towards the expected values.

Parameters

<i>iterations</i>	The number of iterations that <i>SolveStretchConstraint()</i> of <i>ConstraintSolvingStep</i> is executed.
<i>particlePositionOne</i>	The first particle position for <i>SolveStretchConstraint()</i> .
<i>particlePositionTwo</i>	The second particle position for <i>SolveStretchConstraint()</i> .
<i>orientation</i>	The orientation for <i>SolveStretchConstraint()</i> .
<i>mathHelper</i>	The component <i>MathHelper</i> .
<i>constraintSolvingStep</i>	The component <i>ConstraintSolvingStep</i> .

Requirements *orientation* is still a unit quaternion at the end of the test.

The deviation between the stretch constraint and zero is lower than the tolerance 0.1, which means that the algorithm of *SolveStretchConstraint()* converges towards the fulfillment of the stretch constraint.

The deviation between the actual distance of *particlePositionOne* and *particlePositionTwo* and the rest rod element length is lower than a reasonable tolerance, i.e. close to zero.

Attention

The fulfillment of the requirement that the rod element length converges towards the rest rod element length depends on the initial deviation of both particle positions from each other and is just a byproduct of converging towards the constraint fulfillment. If this requirement is not fulfilled, the initial offset or the number of iterations was probably simply to high or low, respectively.

7.19.3 Member Data Documentation

7.19.3.1 constraintSolverSteps

```
int UnitTest_SolveStretchConstraint.constraintSolverSteps = 1000 [private]
```

How often the constraint solver iterates over each constraint during the Constraint Solving Step.

7.19.3.2 maximalDistanceOffset

```
float UnitTest_SolveStretchConstraint.maximalDistanceOffset = 1f [private]
```

The maximal deviation from the rest `rodElementLength`.

Example

Let `rodElementLength` be 10 and [maximalDistanceOffset](#) be 2. Then the two random particle positions drawn will have a distance between 8 and 12.

7.19.3.3 rodElementLength

```
float UnitTest_SolveStretchConstraint.rodElementLength = 10f [private]
```

The distance between two spheres, also the distance between two orientations.

7.19.3.4 sampleSize

```
int UnitTest_SolveStretchConstraint.sampleSize = 10 [private]
```

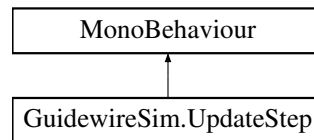
The number of value-pairs the test is executed with. E.g. if [sampleSize](#) is 10, then the unit test is executed with 10 randomly drawn value-pairs. A higher number needs more time to execute.

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

- [UnitTest_SolveStretchConstraint.cs](#)

7.20 GuidewireSim.UpdateStep Class Reference

Inheritance diagram for GuidewireSim.UpdateStep:



Public Member Functions

- `Vector3[] UpdateSphereVelocities (Vector3[] sphereVelocities, int spheresCount, Vector3[] spherePosition↔ Predictions, Vector3[] spherePositions)`
- `Vector3[] UpdateSpherePositions (Vector3[] spherePositions, int spheresCount, Vector3[] spherePosition↔ Predictions)`
- `Vector3[] UpdateCylinderAngularVelocities (Vector3[] cylinderAngularVelocities, int cylinderCount, BSM.↔ Quaternion[] cylinderOrientations, BSM.Quaternion[] cylinderOrientationPredictions)`
- `BSM.Quaternion[] UpdateCylinderOrientations (BSM.Quaternion[] cylinderOrientations, int cylinderCount, BSM.Quaternion[] cylinderOrientationPredictions)`

Private Member Functions

- `void Awake ()`

Private Attributes

- `MathHelper mathHelper`

The component `MathHelper` that provides math related helper functions.

7.20.1 Detailed Description

This class implements the update step of the algorithm.

7.20.2 Member Function Documentation

7.20.2.1 Awake()

```
void GuidewireSim.UpdateStep.Awake ( ) [private]
```

7.20.2.2 UpdateCylinderAngularVelocities()

```
Vector3[] GuidewireSim.UpdateStep.UpdateCylinderAngularVelocities (
    Vector3[] cylinderAngularVelocities,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientations,
    BSM.Quaternion[] cylinderOrientationPredictions )
```

Updates the cylinder angular velocities for the update step of the simulation.

Parameters

<i>cylinderAngularVelocities</i>	The angular velocity of the current frame of each orientation element/ cylinder.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

Returns

The angular velocity of the current frame of each orientation element/ cylinder, i.e. *cylinderAngularVelocities*.

7.20.2.3 UpdateCylinderOrientations()

```
BSM.Quaternion[] GuidewireSim.UpdateStep.UpdateCylinderOrientations (
    BSM.Quaternion[] cylinderOrientations,
    int cylinderCount,
    BSM.Quaternion[] cylinderOrientationPredictions )
```

Updates the cylinder orientations given the current orientation predictions for the update step of the simulation.

Parameters

<i>cylinderOrientations</i>	The orientation of each cylinder at its center of mass.
<i>cylinderCount</i>	The count of all cylinders of the guidewire. Equals the length of <i>cylinderOrientationPredictions</i> .
<i>cylinderOrientationPredictions</i>	The prediction of the orientation of each cylinder at its center of mass.

Returns

The orientation of each cylinder at its center of mass, i.e. *cylinderOrientations*.

7.20.2.4 UpdateSpherePositions()

```
Vector3[] GuidewireSim.UpdateStep.UpdateSpherePositions (
    Vector3[] spherePositions,
    int spheresCount,
    Vector3[] spherePositionPredictions )
```

Updates the sphere positions given the current position predictions.

Parameters

<i>spherePositions</i>	The position at the current frame of each sphere.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <i>spherePositionPredictions</i> .
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).

Returns

The position at the current frame of each sphere, i.e. `spherePositions`.

7.20.2.5 UpdateSphereVelocities()

```
Vector3[] GuidewireSim.UpdateStep.UpdateSphereVelocities (
    Vector3[] sphereVelocities,
    int spheresCount,
    Vector3[] spherePositionPredictions,
    Vector3[] spherePositions )
```

Updates the sphere velocities given the current prediction and the current position.

Parameters

<i>sphereVelocities</i>	The velocity of the current frame of each sphere.
<i>spheresCount</i>	The count of all spheres of the guidewire. Equals the length of <code>spherePositionPredictions</code> .
<i>spherePositionPredictions</i>	The prediction of the position at the current frame of each sphere (in this case of the last frame).
<i>spherePositions</i>	The position at the current frame of each sphere.

Returns

The velocity of the current frame of each sphere, i.e. `sphereVelocities`.

7.20.3 Member Data Documentation

7.20.3.1 mathHelper

`MathHelper` `GuidewireSim.UpdateStep.mathHelper` [private]

The component `MathHelper` that provides math related helper functions.

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

- [UpdateStep.cs](#)

Chapter 8

File Documentation

8.1 CollisionDetectionPrimitive.cs File Reference

Classes

- class [GuidewireSim.CollisionDetectionPrimitive](#)

Namespaces

- namespace [GuidewireSim](#)

8.2 CollisionDetectionVessel.cs File Reference

Classes

- class [GuidewireSim.CollisionDetectionVessel](#)

Namespaces

- namespace [GuidewireSim](#)

8.3 CollisionHandler.cs File Reference

Classes

- class [GuidewireSim.CollisionHandler](#)

Namespaces

- namespace [GuidewireSim](#)

8.4 CollisionPair.cs File Reference

Classes

- struct [GuidewireSim.CollisionPair](#)

Namespaces

- namespace [GuidewireSim](#)

8.5 CollisionSolvingStep.cs File Reference

Classes

- class [GuidewireSim.CollisionSolvingStep](#)

Namespaces

- namespace [GuidewireSim](#)

8.6 CollisionTestPerformer.cs File Reference

Classes

- class [GuidewireSim.CollisionTestPerformer](#)

Namespaces

- namespace [GuidewireSim](#)

8.7 ConstraintSolvingStep.cs File Reference

Classes

- class [GuidewireSim.ConstraintSolvingStep](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.7.1 Typedef Documentation

8.7.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.8 DebugExtension.cs File Reference

Classes

- class [DebugExtension](#)
Debug Extension

8.9 DirectorsDrawer.cs File Reference

Classes

- class [GuidewireSim.DirectorsDrawer](#)

Namespaces

- namespace [GuidewireSim](#)

8.10 ForceTestPerformer.cs File Reference

Classes

- class [GuidewireSim.ForceTestPerformer](#)

Namespaces

- namespace [GuidewireSim](#)

8.11 InitializationStep.cs File Reference

Classes

- class [GuidewireSim.InitializationStep](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.11.1 Typedef Documentation

8.11.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.12 MathHelper.cs File Reference

Classes

- class [GuidewireSim.MathHelper](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.12.1 Typedef Documentation

8.12.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.13 ObjectSetter.cs File Reference

Classes

- class [GuidewireSim.ObjectSetter](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.13.1 Typedef Documentation

8.13.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.14 PredictionStep.cs File Reference

Classes

- class [GuidewireSim.PredictionStep](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.14.1 Typedef Documentation

8.14.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.15 SimulationLoop.cs File Reference

Classes

- class [GuidewireSim.SimulationLoop](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.15.1 Typedef Documentation

8.15.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.16 StressTestPerformer.cs File Reference

Classes

- class [GuidewireSim.StressTestPerformer](#)

Namespaces

- namespace [GuidewireSim](#)

8.17 TorqueTestPerformer.cs File Reference

Classes

- class [GuidewireSim.TorqueTestPerformer](#)

Namespaces

- namespace [GuidewireSim](#)

8.18 UpdateStep.cs File Reference

Classes

- class [GuidewireSim.UpdateStep](#)

Namespaces

- namespace [GuidewireSim](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.18.1 Typedef Documentation

8.18.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.19 UnitTest_SolveBendTwistConstraint.cs File Reference

Classes

- class [UnitTest_SolveBendTwistConstraint](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.19.1 Typedef Documentation

8.19.1.1 BSM

```
using BSM = BulletSharp.Math
```

8.20 UnitTest_SolveStretchConstraint.cs File Reference

Classes

- class [UnitTest_SolveStretchConstraint](#)

Typedefs

- using [BSM](#) = BulletSharp.Math

8.20.1 Typedef Documentation

8.20.1.1 BSM

```
using BSM = BulletSharp.Math
```

Index

- AdaptCalculations
 - GuidewireSim.SimulationLoop, [107](#)
- arrowHeadAngle
 - GuidewireSim.DirectorsDrawer, [75](#)
- arrowHeadPercentage
 - GuidewireSim.DirectorsDrawer, [75](#)
- AssignSphereID
 - GuidewireSim.CollisionDetectionPrimitive, [16](#)
- Awake
 - GuidewireSim.CollisionDetectionPrimitive, [16](#)
 - GuidewireSim.CollisionDetectionVessel, [18](#)
 - GuidewireSim.CollisionSolvingStep, [24](#)
 - GuidewireSim.CollisionTestPerformer, [28](#)
 - GuidewireSim.ConstraintSolvingStep, [31](#)
 - GuidewireSim.DirectorsDrawer, [73](#)
 - GuidewireSim.ForceTestPerformer, [77](#)
 - GuidewireSim.InitializationStep, [81](#)
 - GuidewireSim.ObjectSetter, [100](#)
 - GuidewireSim.PredictionStep, [103](#)
 - GuidewireSim.SimulationLoop, [108](#)
 - GuidewireSim.StressTestPerformer, [117](#)
 - GuidewireSim.TorqueTestPerformer, [119](#)
 - GuidewireSim.UpdateStep, [127](#)
- bendStiffness
 - GuidewireSim.ConstraintSolvingStep, [39](#)
- BendTwistConstraintDeviation
 - GuidewireSim.MathHelper, [89](#)
- BSM
 - ConstraintSolvingStep.cs, [133](#)
 - InitializationStep.cs, [134](#)
 - MathHelper.cs, [134](#)
 - ObjectSetter.cs, [135](#)
 - PredictionStep.cs, [136](#)
 - SimulationLoop.cs, [136](#)
 - UnitTest_SolveBendTwistConstraint.cs, [138](#)
 - UnitTest_SolveStretchConstraint.cs, [138](#)
 - UpdateStep.cs, [137](#)
- CalculateArrowHeadPositions
 - GuidewireSim.DirectorsDrawer, [73](#)
- CalculateCylinderPositions
 - GuidewireSim.MathHelper, [89](#)
- CalculateDeltaPosition
 - GuidewireSim.CollisionSolvingStep, [24](#)
- CollisionDetectionPrimitive.cs, [131](#)
- CollisionDetectionVessel.cs, [131](#)
- collisionHandler
 - GuidewireSim.CollisionDetectionPrimitive, [16](#)
 - GuidewireSim.CollisionDetectionVessel, [18](#)
 - GuidewireSim.CollisionSolvingStep, [26](#)
 - GuidewireSim.InitializationStep, [87](#)
 - GuidewireSim.SimulationLoop, [110](#)
- CollisionHandler.cs, [131](#)
- collisionMargin
 - GuidewireSim.CollisionSolvingStep, [26](#)
- collisionNormal
 - GuidewireSim.CollisionPair, [22](#)
- CollisionPair
 - GuidewireSim.CollisionPair, [22](#)
- CollisionPair.cs, [132](#)
- collisionSolvingStep
 - GuidewireSim.SimulationLoop, [110](#)
- CollisionSolvingStep.cs, [132](#)
- collisionStiffness
 - GuidewireSim.CollisionSolvingStep, [26](#)
- CollisionTestPerformer.cs, [132](#)
- ColumnVectorMatrixMultiplication
 - GuidewireSim.MathHelper, [91](#)
- ConstraintSolverSteps
 - GuidewireSim.SimulationLoop, [116](#)
- constraintSolverSteps
 - GuidewireSim.SimulationLoop, [110](#)
 - UnitTest_SolveBendTwistConstraint, [123](#)
 - UnitTest_SolveStretchConstraint, [126](#)
- constraintSolvingStep
 - GuidewireSim.SimulationLoop, [110](#)
- ConstraintSolvingStep.cs, [132](#)
- BSM, [133](#)
- contactPoint
 - GuidewireSim.CollisionPair, [22](#)
- CorrectBendTwistPredictions
 - GuidewireSim.ConstraintSolvingStep, [31](#)
- CorrectCollisionPredictions
 - GuidewireSim.CollisionSolvingStep, [24](#)
- CorrectStretchPredictions
 - GuidewireSim.ConstraintSolvingStep, [32](#)
- cylinderAngularVelocities
 - GuidewireSim.SimulationLoop, [110](#)
- CylinderCount
 - GuidewireSim.SimulationLoop, [116](#)
- cylinderExternalTorques
 - GuidewireSim.SimulationLoop, [111](#)
- cylinderOrientationPredictions
 - GuidewireSim.SimulationLoop, [111](#)
- cylinderOrientations
 - GuidewireSim.SimulationLoop, [111](#)
- cylinderPositions
 - GuidewireSim.SimulationLoop, [111](#)

- cylinders
 - GuidewireSim.SimulationLoop, 111
- cylinderScalarWeights
 - GuidewireSim.SimulationLoop, 111
- DarbouxSignFactor
 - GuidewireSim.MathHelper, 91
- DebugArrow
 - DebugExtension, 42, 43
- DebugBounds
 - DebugExtension, 43, 44
- DebugCapsule
 - DebugExtension, 44, 45
- DebugCircle
 - DebugExtension, 46–48
- DebugCone
 - DebugExtension, 49–51
- DebugCylinder
 - DebugExtension, 51, 52
- DebugExtension, 40
 - DebugArrow, 42, 43
 - DebugBounds, 43, 44
 - DebugCapsule, 44, 45
 - DebugCircle, 46–48
 - DebugCone, 49–51
 - DebugCylinder, 51, 52
 - DebugLocalCube, 53–55
 - DebugPoint, 56
 - DebugWireSphere, 58, 59
 - DrawArrow, 59, 60
 - DrawBounds, 60
 - DrawCapsule, 61, 62
 - DrawCircle, 62–64
 - DrawCone, 64–66
 - DrawCylinder, 66, 67
 - DrawLocalCube, 68, 69
 - DrawPoint, 70
 - MethodsOfObject, 71
 - MethodsOfType, 71
- DebugExtension.cs, 133
- DebugLocalCube
 - DebugExtension, 53–55
- DebugPoint
 - DebugExtension, 56
- DebugWireSphere
 - DebugExtension, 58, 59
- deltaOrientation
 - GuidewireSim.ConstraintSolvingStep, 39
- deltaOrientationOne
 - GuidewireSim.ConstraintSolvingStep, 39
- deltaOrientationTwo
 - GuidewireSim.ConstraintSolvingStep, 39
- deltaPosition
 - GuidewireSim.CollisionSolvingStep, 27
- deltaPositionOne
 - GuidewireSim.ConstraintSolvingStep, 39
- deltaPositionTwo
 - GuidewireSim.ConstraintSolvingStep, 39
- directorColors
 - GuidewireSim.DirectorsDrawer, 75
- directorOneColor
 - GuidewireSim.DirectorsDrawer, 75
- directors
 - GuidewireSim.SimulationLoop, 112
- DirectorsDrawer.cs, 133
- directorThreeColor
 - GuidewireSim.DirectorsDrawer, 75
- directorTwoColor
 - GuidewireSim.DirectorsDrawer, 76
- DiscreteDarbouxVector
 - GuidewireSim.MathHelper, 92
- discreteRestDarbouxVectors
 - GuidewireSim.SimulationLoop, 112
- doCollisionTestFour
 - GuidewireSim.CollisionTestPerformer, 29
- doCollisionTestOne
 - GuidewireSim.CollisionTestPerformer, 29
- doCollisionTestThree
 - GuidewireSim.CollisionTestPerformer, 29
- doCollisionTestTwo
 - GuidewireSim.CollisionTestPerformer, 29
- doForceTestFour
 - GuidewireSim.ForceTestPerformer, 79
- doForceTestOne
 - GuidewireSim.ForceTestPerformer, 79
- doForceTestThree
 - GuidewireSim.ForceTestPerformer, 79
- doForceTestTwo
 - GuidewireSim.ForceTestPerformer, 79
- doSingleLoopTest
 - GuidewireSim.ForceTestPerformer, 79
- doStressTestOne
 - GuidewireSim.StressTestPerformer, 118
- doTorqueTestOne
 - GuidewireSim.TorqueTestPerformer, 121
- doTorqueTestThree
 - GuidewireSim.TorqueTestPerformer, 121
- doTorqueTestTwo
 - GuidewireSim.TorqueTestPerformer, 121
- DrawArrow
 - DebugExtension, 59, 60
- DrawArrowHeadConnectionLines
 - GuidewireSim.DirectorsDrawer, 73
- DrawArrowHeadLines
 - GuidewireSim.DirectorsDrawer, 74
- DrawBounds
 - DebugExtension, 60
- DrawCapsule
 - DebugExtension, 61, 62
- DrawCircle
 - DebugExtension, 62–64
- DrawCollisionInformation
 - GuidewireSim.CollisionSolvingStep, 25
- DrawCone
 - DebugExtension, 64–66
- DrawCylinder
 - DebugExtension, 66, 67

- DrawDirectors
 - GuidewireSim.DirectorsDrawer, 74
- DrawLocalCube
 - DebugExtension, 68, 69
- DrawPoint
 - DebugExtension, 70
- EmbeddedVector
 - GuidewireSim.MathHelper, 92
- executeInBilateralOrder
 - GuidewireSim.ConstraintSolvingStep, 40
- ExecuteSingleLoopTest
 - GuidewireSim.SimulationLoop, 116
- FixedUpdate
 - GuidewireSim.SimulationLoop, 108
- ForceTestPerformer.cs, 133
- GetGaussianRandomNumber
 - GuidewireSim.MathHelper, 93
- GuidewireSim, 13
- GuidewireSim.CollisionDetectionPrimitive, 15
 - AssignSphereID, 16
 - Awake, 16
 - collisionHandler, 16
 - OnCollisionEnter, 16
 - OnCollisionStay, 16
 - simulationLoop, 16
 - sphereID, 17
 - Start, 16
- GuidewireSim.CollisionDetectionVessel, 17
 - Awake, 18
 - collisionHandler, 18
 - simulationLoop, 18
 - YieldSphereID, 18
- GuidewireSim.CollisionHandler, 19
 - RegisterCollision, 19
 - registeredCollisions, 20
 - ResetRegisteredCollisions, 20
 - SetCollidersToPredictions, 20
 - sphereColliders, 21
 - sphereRadius, 21
 - Start, 20
- GuidewireSim.CollisionPair, 21
 - collisionNormal, 22
 - CollisionPair, 22
 - contactPoint, 22
 - sphere, 22
 - sphereID, 22
- GuidewireSim.CollisionSolvingStep, 23
 - Awake, 24
 - CalculateDeltaPosition, 24
 - collisionHandler, 26
 - collisionMargin, 26
 - collisionStiffness, 26
 - CorrectCollisionPredictions, 24
 - deltaPosition, 27
 - DrawCollisionInformation, 25
 - initialPositionPrediction, 27
 - mathHelper, 27
 - SolveCollisionConstraint, 25
 - SolveCollisionConstraints, 26
 - sphereRadius, 27
- GuidewireSim.CollisionTestPerformer, 27
 - Awake, 28
 - doCollisionTestFour, 29
 - doCollisionTestOne, 29
 - doCollisionTestThree, 29
 - doCollisionTestTwo, 29
 - PerformCollisionTestFour, 28
 - PerformCollisionTestOne, 28
 - PerformCollisionTests, 28
 - PerformCollisionTestThree, 29
 - PerformCollisionTestTwo, 29
 - pullForce, 30
 - simulationLoop, 30
 - Start, 29
 - startTime, 30
- GuidewireSim.ConstraintSolvingStep, 30
 - Awake, 31
 - bendStiffness, 39
 - CorrectBendTwistPredictions, 31
 - CorrectStretchPredictions, 32
 - deltaOrientation, 39
 - deltaOrientationOne, 39
 - deltaOrientationTwo, 39
 - deltaPositionOne, 39
 - deltaPositionTwo, 39
 - executeInBilateralOrder, 40
 - mathHelper, 40
 - SolveBendTwistConstraint, 33
 - SolveBendTwistConstraints, 33
 - SolveBendTwistConstraintsInBilateralOrder, 34
 - SolveBendTwistConstraintsInNaiveOrder, 35
 - SolveStretchConstraint, 35
 - SolveStretchConstraints, 36
 - SolveStretchConstraintsInBilateralOrder, 37
 - SolveStretchConstraintsInNaiveOrder, 38
 - stretchStiffness, 40
- GuidewireSim.DirectorsDrawer, 72
 - arrowHeadAngle, 75
 - arrowHeadPercentage, 75
 - Awake, 73
 - CalculateArrowHeadPositions, 73
 - directorColors, 75
 - directorOneColor, 75
 - directorThreeColor, 75
 - directorTwoColor, 76
 - DrawArrowHeadConnectionLines, 73
 - DrawArrowHeadLines, 74
 - DrawDirectors, 74
 - scaleFactor, 76
 - simulationLoop, 76
 - Update, 75
- GuidewireSim.ForceTestPerformer, 76
 - Awake, 77
 - doForceTestFour, 79

- doForceTestOne, 79
- doForceTestThree, 79
- doForceTestTwo, 79
- doSingleLoopTest, 79
- PerformForceTestFour, 77
- PerformForceTestOne, 77
- PerformForceTests, 78
- PerformForceTestThree, 78
- PerformForceTestTwo, 78
- PerformSingleLoopTest, 78
- pullForceTestThree, 79
- simulationLoop, 80
- Start, 78
- GuidewireSim.InitializationStep, 80
 - Awake, 81
 - collisionHandler, 87
 - InitCylinderAngularVelocities, 81
 - InitCylinderExternalTorques, 81
 - InitCylinderOrientationPredictions, 82
 - InitCylinderOrientations, 82
 - InitCylinderPositions, 83
 - InitCylinderScalarWeights, 83
 - InitDirectors, 83
 - InitDiscreteRestDarbouxVectors, 84
 - InitInertiaTensor, 84
 - InitInverseInertiaTensor, 85
 - InitSphereColliders, 85
 - InitSphereExternalForces, 85
 - InitSphereInverseMasses, 85
 - InitSpherePositionPredictions, 86
 - InitSpherePositions, 86
 - InitSphereVelocities, 86
 - InitWorldSpaceBasis, 87
 - materialDensity, 87
 - materialRadius, 87
 - mathHelper, 88
- GuidewireSim.MathHelper, 88
 - BendTwistConstraintDeviation, 89
 - CalculateCylinderPositions, 89
 - ColumnVectorMatrixMultiplication, 91
 - DarbouxSignFactor, 91
 - DiscreteDarbouxVector, 92
 - EmbeddedVector, 92
 - GetGaussianRandomNumber, 93
 - ImaginaryPart, 93
 - MatrixInverse, 93
 - MatrixVectorMultiplication, 94
 - QuaternionConversionFromBSM, 94
 - QuaternionConversionToBSM, 95
 - QuaternionLength, 95
 - RandomUnitQuaternion, 95
 - RodElementLength, 96
 - RodElementLengthDeviation, 96
 - ScalarMatrixMultiplication, 97
 - SquaredNorm, 97
 - StretchConstraintDeviation, 97
 - UpdateDirectors, 98
 - VectorColumnVectorMultiplication, 99
 - VectorLength, 99
- GuidewireSim.ObjectSetter, 99
 - Awake, 100
 - mathHelper, 102
 - SetCylinderOrientations, 100
 - SetCylinderPositions, 101
 - SetSpherePositions, 101
- GuidewireSim.PredictionStep, 102
 - Awake, 103
 - mathHelper, 105
 - PredictAngularVelocities, 103
 - PredictCylinderOrientations, 103
 - PredictSpherePositions, 104
 - PredictSphereVelocities, 104
- GuidewireSim.SimulationLoop, 105
 - AdaptCalculations, 107
 - Awake, 108
 - collisionHandler, 110
 - collisionSolvingStep, 110
 - ConstraintSolverSteps, 116
 - constraintSolverSteps, 110
 - constraintSolvingStep, 110
 - cylinderAngularVelocities, 110
 - CylinderCount, 116
 - cylinderExternalTorques, 111
 - cylinderOrientationPredictions, 111
 - cylinderOrientations, 111
 - cylinderPositions, 111
 - cylinders, 111
 - cylinderScalarWeights, 111
 - directors, 112
 - discreteRestDarbouxVectors, 112
 - ExecuteSingleLoopTest, 116
 - FixedUpdate, 108
 - inertiaTensor, 112
 - initializationStep, 112
 - inverseInertiaTensor, 113
 - mathHelper, 113
 - objectSetter, 113
 - PerformConstraintSolvingStep, 108
 - PerformInitializationStep, 108
 - PerformPredictionStep, 109
 - PerformSimulationLoop, 109
 - PerformUpdateStep, 109
 - predictionStep, 113
 - rodElementLength, 113
 - SetCollidersStep, 109
 - solveBendTwistConstraints, 113
 - solveCollisionConstraints, 114
 - solveStretchConstraints, 114
 - sphereExternalForces, 114
 - sphereInverseMasses, 114
 - spherePositionPredictions, 114
 - spherePositions, 114
 - spheres, 115
 - SpheresCount, 116
 - sphereVelocities, 115
 - Start, 110

- TimeStep, 116
- timeStep, 115
- updateStep, 115
- worldSpaceBasis, 115
- GuidewireSim.StressTestPerformer, 117
 - Awake, 117
 - doStressTestOne, 118
 - PerformStressTestOne, 117
 - PerformStressTests, 118
 - simulationLoop, 118
 - Start, 118
- GuidewireSim.TorqueTestPerformer, 119
 - Awake, 119
 - doTorqueTestOne, 121
 - doTorqueTestThree, 121
 - doTorqueTestTwo, 121
 - PerformTorqueTestOne, 119
 - PerformTorqueTests, 119
 - PerformTorqueTestThree, 120
 - PerformTorqueTestTwo, 120
 - pullTorque, 121
 - simulationLoop, 121
 - Start, 120
- GuidewireSim.UpdateStep, 127
 - Awake, 127
 - mathHelper, 129
 - UpdateCylinderAngularVelocities, 127
 - UpdateCylinderOrientations, 128
 - UpdateSpherePositions, 128
 - UpdateSphereVelocities, 129
- ImaginaryPart
 - GuidewireSim.MathHelper, 93
- inertiaTensor
 - GuidewireSim.SimulationLoop, 112
- InitCylinderAngularVelocities
 - GuidewireSim.InitializationStep, 81
- InitCylinderExternalTorques
 - GuidewireSim.InitializationStep, 81
- InitCylinderOrientationPredictions
 - GuidewireSim.InitializationStep, 82
- InitCylinderOrientations
 - GuidewireSim.InitializationStep, 82
- InitCylinderPositions
 - GuidewireSim.InitializationStep, 83
- InitCylinderScalarWeights
 - GuidewireSim.InitializationStep, 83
- InitDirectors
 - GuidewireSim.InitializationStep, 83
- InitDiscreteRestDarbouxVectors
 - GuidewireSim.InitializationStep, 84
- initializationStep
 - GuidewireSim.SimulationLoop, 112
- InitializationStep.cs, 134
 - BSM, 134
- initialPositionPrediction
 - GuidewireSim.CollisionSolvingStep, 27
- InitInertiaTensor
 - GuidewireSim.InitializationStep, 84
- InitInverseInertiaTensor
 - GuidewireSim.InitializationStep, 85
- InitSphereColliders
 - GuidewireSim.InitializationStep, 85
- InitSphereExternalForces
 - GuidewireSim.InitializationStep, 85
- InitSphereInverseMasses
 - GuidewireSim.InitializationStep, 85
- InitSpherePositionPredictions
 - GuidewireSim.InitializationStep, 86
- InitSpherePositions
 - GuidewireSim.InitializationStep, 86
- InitSphereVelocities
 - GuidewireSim.InitializationStep, 86
- InitWorldSpaceBasis
 - GuidewireSim.InitializationStep, 87
- inverseInertiaTensor
 - GuidewireSim.SimulationLoop, 113
- materialDensity
 - GuidewireSim.InitializationStep, 87
- materialRadius
 - GuidewireSim.InitializationStep, 87
- mathHelper
 - GuidewireSim.CollisionSolvingStep, 27
 - GuidewireSim.ConstraintSolvingStep, 40
 - GuidewireSim.InitializationStep, 88
 - GuidewireSim.ObjectSetter, 102
 - GuidewireSim.PredictionStep, 105
 - GuidewireSim.SimulationLoop, 113
 - GuidewireSim.UpdateStep, 129
- MathHelper.cs, 134
 - BSM, 134
- MatrixInverse
 - GuidewireSim.MathHelper, 93
- MatrixVectorMultiplication
 - GuidewireSim.MathHelper, 94
- maximalDistanceOffset
 - UnitTest_SolveStretchConstraint, 126
- MethodsOfObject
 - DebugExtension, 71
- MethodsOfType
 - DebugExtension, 71
- objectSetter
 - GuidewireSim.SimulationLoop, 113
- ObjectSetter.cs, 135
 - BSM, 135
- OnCollisionEnter
 - GuidewireSim.CollisionDetectionPrimitive, 16
- OnCollisionStay
 - GuidewireSim.CollisionDetectionPrimitive, 16
- PerformCollisionTestFour
 - GuidewireSim.CollisionTestPerformer, 28
- PerformCollisionTestOne
 - GuidewireSim.CollisionTestPerformer, 28
- PerformCollisionTests
 - GuidewireSim.CollisionTestPerformer, 28

- PerformCollisionTestThree
 - GuidewireSim.CollisionTestPerformer, 29
- PerformCollisionTestTwo
 - GuidewireSim.CollisionTestPerformer, 29
- PerformConstraintSolvingStep
 - GuidewireSim.SimulationLoop, 108
- PerformForceTestFour
 - GuidewireSim.ForceTestPerformer, 77
- PerformForceTestOne
 - GuidewireSim.ForceTestPerformer, 77
- PerformForceTests
 - GuidewireSim.ForceTestPerformer, 78
- PerformForceTestThree
 - GuidewireSim.ForceTestPerformer, 78
- PerformForceTestTwo
 - GuidewireSim.ForceTestPerformer, 78
- PerformInitializationStep
 - GuidewireSim.SimulationLoop, 108
- PerformPredictionStep
 - GuidewireSim.SimulationLoop, 109
- PerformSimulationLoop
 - GuidewireSim.SimulationLoop, 109
- PerformSingleLoopTest
 - GuidewireSim.ForceTestPerformer, 78
- PerformStressTestOne
 - GuidewireSim.StressTestPerformer, 117
- PerformStressTests
 - GuidewireSim.StressTestPerformer, 118
- PerformTorqueTestOne
 - GuidewireSim.TorqueTestPerformer, 119
- PerformTorqueTests
 - GuidewireSim.TorqueTestPerformer, 119
- PerformTorqueTestThree
 - GuidewireSim.TorqueTestPerformer, 120
- PerformTorqueTestTwo
 - GuidewireSim.TorqueTestPerformer, 120
- PerformUnitTests
 - UnitTest_SolveBendTwistConstraint, 122
 - UnitTest_SolveStretchConstraint, 124
- PerformUpdateStep
 - GuidewireSim.SimulationLoop, 109
- PickRandomPositions
 - UnitTest_SolveStretchConstraint, 124
- PredictAngularVelocities
 - GuidewireSim.PredictionStep, 103
- PredictCylinderOrientations
 - GuidewireSim.PredictionStep, 103
- predictionStep
 - GuidewireSim.SimulationLoop, 113
- PredictionStep.cs, 135
 - BSM, 136
- PredictSpherePositions
 - GuidewireSim.PredictionStep, 104
- PredictSphereVelocities
 - GuidewireSim.PredictionStep, 104
- pullForce
 - GuidewireSim.CollisionTestPerformer, 30
- pullForceTestThree
 - GuidewireSim.ForceTestPerformer, 79
- pullTorque
 - GuidewireSim.TorqueTestPerformer, 121
- QuaternionConversionFromBSM
 - GuidewireSim.MathHelper, 94
- QuaternionConversionToBSM
 - GuidewireSim.MathHelper, 95
- QuaternionLength
 - GuidewireSim.MathHelper, 95
- RandomUnitQuaternion
 - GuidewireSim.MathHelper, 95
- RegisterCollision
 - GuidewireSim.CollisionHandler, 19
- registeredCollisions
 - GuidewireSim.CollisionHandler, 20
- ResetRegisteredCollisions
 - GuidewireSim.CollisionHandler, 20
- RodElementLength
 - GuidewireSim.MathHelper, 96
- rodElementLength
 - GuidewireSim.SimulationLoop, 113
 - UnitTest_SolveStretchConstraint, 126
- RodElementLengthDeviation
 - GuidewireSim.MathHelper, 96
- sampleSize
 - UnitTest_SolveBendTwistConstraint, 123
 - UnitTest_SolveStretchConstraint, 126
- ScalarMatrixMultiplication
 - GuidewireSim.MathHelper, 97
- scaleFactor
 - GuidewireSim.DirectorsDrawer, 76
- SetCollidersStep
 - GuidewireSim.SimulationLoop, 109
- SetCollidersToPredictions
 - GuidewireSim.CollisionHandler, 20
- SetCylinderOrientations
 - GuidewireSim.ObjectSetter, 100
- SetCylinderPositions
 - GuidewireSim.ObjectSetter, 101
- SetSpherePositions
 - GuidewireSim.ObjectSetter, 101
- simulationLoop
 - GuidewireSim.CollisionDetectionPrimitive, 16
 - GuidewireSim.CollisionDetectionVessel, 18
 - GuidewireSim.CollisionTestPerformer, 30
 - GuidewireSim.DirectorsDrawer, 76
 - GuidewireSim.ForceTestPerformer, 80
 - GuidewireSim.StressTestPerformer, 118
 - GuidewireSim.TorqueTestPerformer, 121
- SimulationLoop.cs, 136
 - BSM, 136
- SolveBendTwistConstraint
 - GuidewireSim.ConstraintSolvingStep, 33
- SolveBendTwistConstraints
 - GuidewireSim.ConstraintSolvingStep, 33
- solveBendTwistConstraints

- GuidewireSim.SimulationLoop, 113
- SolveBendTwistConstraintsInBilateralOrder
 - GuidewireSim.ConstraintSolvingStep, 34
- SolveBendTwistConstraintsInNaiveOrder
 - GuidewireSim.ConstraintSolvingStep, 35
- SolveCollisionConstraint
 - GuidewireSim.CollisionSolvingStep, 25
- SolveCollisionConstraints
 - GuidewireSim.CollisionSolvingStep, 26
- solveCollisionConstraints
 - GuidewireSim.SimulationLoop, 114
- SolveStretchConstraint
 - GuidewireSim.ConstraintSolvingStep, 35
- SolveStretchConstraints
 - GuidewireSim.ConstraintSolvingStep, 36
- solveStretchConstraints
 - GuidewireSim.SimulationLoop, 114
- SolveStretchConstraintsInBilateralOrder
 - GuidewireSim.ConstraintSolvingStep, 37
- SolveStretchConstraintsInNaiveOrder
 - GuidewireSim.ConstraintSolvingStep, 38
- sphere
 - GuidewireSim.CollisionPair, 22
- sphereColliders
 - GuidewireSim.CollisionHandler, 21
- sphereExternalForces
 - GuidewireSim.SimulationLoop, 114
- sphereID
 - GuidewireSim.CollisionDetectionPrimitive, 17
 - GuidewireSim.CollisionPair, 22
- sphereInverseMasses
 - GuidewireSim.SimulationLoop, 114
- spherePositionPredictions
 - GuidewireSim.SimulationLoop, 114
- spherePositions
 - GuidewireSim.SimulationLoop, 114
- sphereRadius
 - GuidewireSim.CollisionHandler, 21
 - GuidewireSim.CollisionSolvingStep, 27
- spheres
 - GuidewireSim.SimulationLoop, 115
- SpheresCount
 - GuidewireSim.SimulationLoop, 116
- sphereVelocities
 - GuidewireSim.SimulationLoop, 115
- SquaredNorm
 - GuidewireSim.MathHelper, 97
- Start
 - GuidewireSim.CollisionDetectionPrimitive, 16
 - GuidewireSim.CollisionHandler, 20
 - GuidewireSim.CollisionTestPerformer, 29
 - GuidewireSim.ForceTestPerformer, 78
 - GuidewireSim.SimulationLoop, 110
 - GuidewireSim.StressTestPerformer, 118
 - GuidewireSim.TorqueTestPerformer, 120
- startTime
 - GuidewireSim.CollisionTestPerformer, 30
- StressTestPerformer.cs, 136
- StretchConstraintDeviation
 - GuidewireSim.MathHelper, 97
- stretchStiffness
 - GuidewireSim.ConstraintSolvingStep, 40
- Test_SolveBendTwistConstraint
 - UnitTest_SolveBendTwistConstraint, 122
- Test_SolveStretchConstraint
 - UnitTest_SolveStretchConstraint, 125
- TimeStep
 - GuidewireSim.SimulationLoop, 116
- timeStep
 - GuidewireSim.SimulationLoop, 115
- TorqueTestPerformer.cs, 137
- UnitTest_SolveBendTwistConstraint, 121
 - constraintSolverSteps, 123
 - PerformUnitTests, 122
 - sampleSize, 123
 - Test_SolveBendTwistConstraint, 122
- UnitTest_SolveBendTwistConstraint.cs, 137
 - BSM, 138
- UnitTest_SolveStretchConstraint, 123
 - constraintSolverSteps, 126
 - maximalDistanceOffset, 126
 - PerformUnitTests, 124
 - PickRandomPositions, 124
 - rodElementLength, 126
 - sampleSize, 126
 - Test_SolveStretchConstraint, 125
- UnitTest_SolveStretchConstraint.cs, 138
 - BSM, 138
- Update
 - GuidewireSim.DirectorsDrawer, 75
- UpdateCylinderAngularVelocities
 - GuidewireSim.UpdateStep, 127
- UpdateCylinderOrientations
 - GuidewireSim.UpdateStep, 128
- UpdateDirectors
 - GuidewireSim.MathHelper, 98
- UpdateSpherePositions
 - GuidewireSim.UpdateStep, 128
- UpdateSphereVelocities
 - GuidewireSim.UpdateStep, 129
- updateStep
 - GuidewireSim.SimulationLoop, 115
- UpdateStep.cs, 137
 - BSM, 137
- VectorColumnVectorMultiplication
 - GuidewireSim.MathHelper, 99
- VectorLength
 - GuidewireSim.MathHelper, 99
- worldSpaceBasis
 - GuidewireSim.SimulationLoop, 115
- YieldSphereID
 - GuidewireSim.CollisionDetectionVessel, 18