

SenseGlove Unity Plugin v1.2

Generated by Doxygen 1.8.18

1 Namespace Index	1
1.1 Packages	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Namespace Documentation	9
4.1 SG Namespace Reference	9
4.1.1 Enumeration Type Documentation	11
4.1.1.1 AttachType	11
4.1.1.2 GloveSide	11
4.1.1.3 GrabType	13
4.1.1.4 MovementAxis	13
4.1.1.5 ReleaseMethod	13
4.1.1.6 SG_HandSection	14
4.2 SG.Calibration Namespace Reference	14
4.3 SG.Examples Namespace Reference	14
4.4 SG.Materials Namespace Reference	14
4.4.1 Enumeration Type Documentation	15
4.4.1.1 DisplaceType	15
4.4.1.2 VirtualMaterial	15
4.5 SG.Util Namespace Reference	15
5 Class Documentation	17
5.1 SG.SG_SenseGloveHardware.BuzzCmd Class Reference	17
5.1.1 Member Data Documentation	17
5.1.1.1 patterns	18
5.2 SG.Calibration.CalibrationPose Class Reference	18
5.2.1 Detailed Description	19
5.2.2 Constructor & Destructor Documentation	19
5.2.2.1 CalibrationPose() [1/2]	19
5.2.2.2 CalibrationPose() [2/2]	20
5.2.3 Member Function Documentation	20
5.2.3.1 CalibrateParameters()	20
5.2.3.2 GetFist()	20
5.2.3.3 GetFullFist()	21
5.2.3.4 GetFullOpen()	21
5.2.3.5 GetOpenHand()	22
5.2.3.6 GetThumbAbd()	22
5.2.3.7 GetThumbFlexed()	22
5.2.3.8 GetThumbNoAbd()	23

5.2.3.9 GetThumbsUp()	23
5.2.3.10 SetupArray()	23
5.2.3.11 SetupFloatArray()	24
5.2.4 Member Data Documentation	24
5.2.4.1 abd	24
5.2.4.2 calbrUsing	24
5.2.4.3 x0	24
5.2.4.4 xAffect	25
5.2.4.5 yAffect	25
5.2.4.6 yValue	25
5.3 SG.Materials.Deformation Struct Reference	25
5.3.1 Detailed Description	26
5.3.2 Constructor & Destructor Documentation	26
5.3.2.1 Deformation()	26
5.3.3 Member Data Documentation	26
5.3.3.1 absDeformPosition	26
5.3.3.2 absEntryVector	26
5.3.3.3 distance	27
5.4 SG.SG_DeviceManager.DeviceDetectedArgs Class Reference	27
5.4.1 Detailed Description	27
5.4.2 Constructor & Destructor Documentation	27
5.4.2.1 DeviceDetectedArgs()	27
5.4.3 Property Documentation	28
5.4.3.1 DeviceID	28
5.4.3.2 DeviceIndex	28
5.4.3.3 Type	28
5.5 SG.SG_DropZone.DropProps Class Reference	28
5.5.1 Detailed Description	29
5.6 SG.SG_DropZone.DropZoneArgs Class Reference	29
5.6.1 Constructor & Destructor Documentation	29
5.6.1.1 DropZoneArgs()	29
5.6.2 Member Data Documentation	30
5.6.2.1 grabable	30
5.7 SG.Util.FileIO Class Reference	30
5.7.1 Detailed Description	30
5.7.2 Member Function Documentation	30
5.7.2.1 ReadTxtFile()	30
5.7.2.2 SaveTxtFile()	31
5.8 SG.SG_SenseGloveHardware.GloveCalibrationArgs Class Reference	31
5.8.1 Detailed Description	32
5.8.2 Constructor & Destructor Documentation	32
5.8.2.1 GloveCalibrationArgs() [1/2]	32

5.8.2.2 GloveCalibrationArgs() [2/2]	32
5.8.3 Member Data Documentation	33
5.8.3.1 newData	33
5.8.3.2 oldData	33
5.9 SG.SG_HandDetector.GloveDetectionArgs Class Reference	33
5.9.1 Detailed Description	34
5.9.2 Constructor & Destructor Documentation	34
5.9.2.1 GloveDetectionArgs()	34
5.9.3 Member Data Documentation	34
5.9.3.1 handModel	34
5.10 SG.Materials.MaterialProps Struct Reference	34
5.10.1 Detailed Description	35
5.10.2 Constructor & Destructor Documentation	35
5.10.2.1 MaterialProps()	35
5.10.3 Member Function Documentation	36
5.10.3.1 Default()	36
5.10.3.2 Parse()	36
5.10.3.3 TryGetFloat()	36
5.10.3.4 TryGetRawValue()	37
5.10.4 Member Data Documentation	37
5.10.4.1 hapticDur	37
5.10.4.2 hapticForce	37
5.10.4.3 maxForce	37
5.10.4.4 maxForceDist	38
5.10.4.5 yieldDist	38
5.11 SG.SG_AutoHandAnimation Class Reference	38
5.11.1 Detailed Description	39
5.11.2 Member Function Documentation	39
5.11.2.1 CheckForScripts()	39
5.11.2.2 CollectFingerJoints()	39
5.11.2.3 Start()	39
5.11.3 Member Data Documentation	39
5.11.3.1 handModelInfo	40
5.12 SG.SG_BasicFeedback Class Reference	40
5.12.1 Detailed Description	41
5.12.2 Member Function Documentation	42
5.12.2.1 SendImpactFeedback()	42
5.12.2.2 SetupSelf()	42
5.12.2.3 UpdatePosition()	42
5.12.3 Member Data Documentation	42
5.12.3.1 cooldownTimer	42
5.12.3.2 handLocation	43

5.12.3.3 impactCooldown	43
5.12.3.4 impactFeedbackEnabled	43
5.12.3.5 impactProfile	43
5.12.3.6 lastPosition	43
5.12.3.7 linkedGlove	43
5.12.3.8 maxBuzzLevel	44
5.12.3.9 maxImpactSpeed	44
5.12.3.10 maxVelocityPoints	44
5.12.3.11 minBuzzLevel	44
5.12.3.12 minImpactSpeed	44
5.12.3.13 velocities	44
5.12.3.14 vibrationTime	45
5.12.4 Property Documentation	45
5.12.4.1 CanImpact	45
5.12.4.2 DebugEnabled	45
5.12.4.3 SmoothedVelocity	45
5.13 SG.SG_Breakable Class Reference	46
5.13.1 Detailed Description	47
5.13.2 Member Enumeration Documentation	47
5.13.2.1 UnbreakType	47
5.13.3 Member Function Documentation	48
5.13.3.1 Break()	48
5.13.3.2 CheckUnbreak()	48
5.13.3.3 IsBroken()	48
5.13.3.4 ObjectBrokenEventHandler()	48
5.13.3.5 ObjectUnBrokenEventHandler()	49
5.13.3.6 OnObjectBreaks()	49
5.13.3.7 OnObjectUnBreaks()	49
5.13.3.8 ResetObject()	49
5.13.3.9 UnBreak()	50
5.13.3.10 WholeMaterial_MaterialBreaks()	50
5.13.4 Member Data Documentation	50
5.13.4.1 breakParticles	50
5.13.4.2 breakSound	50
5.13.4.3 brokenDeform	50
5.13.4.4 brokenMaterial	51
5.13.4.5 brokenObject	51
5.13.4.6 checkTime	51
5.13.4.7 resetTime	51
5.13.4.8 unbreakMethod	51
5.13.4.9 wholeDeform	51
5.13.4.10 wholeMaterial	52

5.13.4.11 wholeObject	52
5.13.5 Event Documentation	52
5.13.5.1 ObjectBreaks	52
5.13.5.2 ObjectUnBreaks	52
5.14 SG.SG_BreakableContainer Class Reference	52
5.14.1 Detailed Description	54
5.14.2 Member Function Documentation	54
5.14.2.1 Break()	54
5.14.2.2 ResetContents()	54
5.14.2.3 ResetObject()	54
5.14.2.4 ResetShards()	54
5.14.2.5 SetColliders()	54
5.14.2.6 SetRB()	55
5.14.2.7 SpawnContents()	55
5.14.2.8 SpawnShards()	55
5.14.2.9 UnBreak()	55
5.14.3 Member Data Documentation	56
5.14.3.1 brokenShards	56
5.14.3.2 contentPositions	56
5.14.3.3 contentRotations	56
5.14.3.4 contents	56
5.14.3.5 contentsContainer	56
5.14.3.6 shardContainer	57
5.14.3.7 shardPositions	57
5.14.3.8 shardRotations	57
5.14.3.9 unbreakWithContents	57
5.15 SG.Calibration.SG_CalibrationSequence Class Reference	57
5.15.1 Detailed Description	59
5.15.2 Member Enumeration Documentation	59
5.15.2.1 CalPose	60
5.15.2.2 CalStage	60
5.15.3 Member Function Documentation	60
5.15.3.1 CalibrateCurrentStep()	60
5.15.3.2 CalibratePose()	60
5.15.3.3 EndGlobal()	60
5.15.3.4 GetCalibrationValues()	61
5.15.3.5 GoToMainStage()	61
5.15.3.6 LoadProfiles()	61
5.15.3.7 RegularExit()	61
5.15.3.8 ResetCalibration()	62
5.15.3.9 SaveAndExit()	62
5.15.3.10 SaveCalibration()	62

5.15.3.11 SkipStep()	62
5.15.3.12 StartGlobal()	62
5.15.4 Member Data Documentation	62
5.15.4.1 baseTxt	62
5.15.4.2 changes	63
5.15.4.3 endPopup	63
5.15.4.4 generalButtonTxt	63
5.15.4.5 hiddenBeforeStart	63
5.15.4.6 instrText	63
5.15.4.7 interpolator	63
5.15.4.8 leftAnimation	64
5.15.4.9 nextStepKey	64
5.15.4.10 openHandEx	64
5.15.4.11 poses	64
5.15.4.12 refinementMenu	64
5.15.4.13 rightHands	64
5.15.4.14 saveButton	65
5.15.4.15 skipButton	65
5.15.4.16 stage	65
5.15.4.17 subStage	65
5.15.4.18 wireFrame	65
5.16 SG.Calibration.SG_CalibrationStorage Class Reference	65
5.16.1 Detailed Description	66
5.16.2 Member Function Documentation	66
5.16.2.1 GetFilename()	66
5.16.2.2 LoadInterpolation()	67
5.16.2.3 StoreInterpolation() [1/2]	67
5.16.2.4 StoreInterpolation() [2/2]	67
5.16.3 Member Data Documentation	68
5.16.3.1 calibrDir	68
5.17 SG.SG_Debugger Class Reference	68
5.17.1 Detailed Description	69
5.17.2 Member Function Documentation	69
5.17.2.1 Instance_DebugMessageRecieved()	69
5.17.2.2 Log()	69
5.17.2.3 LogError()	70
5.17.2.4 LogWarning()	70
5.17.3 Member Data Documentation	70
5.17.3.1 DLL_debugLevel	70
5.17.3.2 unityEnabled	70
5.17.3.3 unityEnabled_S	71
5.18 SG.SG_DetectGrab Class Reference	71

5.18.1 Detailed Description	71
5.18.2 Member Function Documentation	71
5.18.2.1 InteractionBegin()	71
5.18.2.2 InteractionEnd()	72
5.19 SG.SG_DeviceLink Class Reference	72
5.19.1 Detailed Description	73
5.19.2 Member Function Documentation	73
5.19.2.1 DisposeDevice()	73
5.19.2.2 SetupDevice()	74
5.20 SG.SG_DeviceManager Class Reference	74
5.20.1 Detailed Description	75
5.20.2 Member Function Documentation	75
5.20.2.1 CheckConnections()	75
5.20.2.2 ClearConnections()	75
5.20.2.3 ListIndex()	75
5.20.2.4 ReportConnections()	76
5.20.2.5 UnlinkAll()	76
5.20.3 Member Data Documentation	76
5.20.3.1 devicesToLink	76
5.21 SG.SG_Dial Class Reference	77
5.21.1 Detailed Description	78
5.21.2 Member Function Documentation	78
5.21.2.1 AngleIndex()	78
5.21.2.2 GetAngle()	79
5.21.2.3 GetAxis()	79
5.21.2.4 InteractionBegin()	79
5.21.2.5 InteractionEnd()	79
5.21.2.6 SetAngle()	80
5.21.2.7 UpdateAngle()	80
5.21.2.8 UpdateInteraction()	80
5.21.2.9 ValidateAngle()	80
5.21.3 Member Data Documentation	81
5.21.3.1 _grabReference	81
5.21.3.2 angleIndex	81
5.21.3.3 anglOffset	81
5.21.3.4 currAngle	81
5.21.3.5 hinge	82
5.21.3.6 hingeAxis	82
5.21.3.7 hingePoint	82
5.21.3.8 maxAngle	82
5.21.3.9 minAngle	82
5.21.3.10 qBase	82

5.21.3.11 rotOffset	83
5.21.3.12 useLimits	83
5.22 SG.SG_Door Class Reference	83
5.22.1 Detailed Description	84
5.22.2 Member Function Documentation	84
5.22.2.1 OnDoorClosed()	84
5.22.2.2 OnDoorOpened()	84
5.22.3 Event Documentation	84
5.22.3.1 DoorClosed	84
5.22.3.2 DoorOpened	85
5.23 SG.SG_Drawer Class Reference	85
5.23.1 Detailed Description	87
5.23.2 Member Function Documentation	87
5.23.2.1 ForceClosed()	87
5.23.2.2 ForceOpen()	87
5.23.2.3 InteractionBegin()	87
5.23.2.4 InteractionEnd()	88
5.23.2.5 IsClosed()	88
5.23.2.6 IsOpen()	88
5.23.2.7 MoveAxis()	88
5.23.2.8 ResetObject()	89
5.23.2.9 SaveTransform()	89
5.23.2.10 SetInteractable()	89
5.23.2.11 SetMoveAxis()	89
5.23.2.12 UpdateInteraction()	90
5.23.3 Member Data Documentation	90
5.23.3.1 actualMoveDirection	90
5.23.3.2 grabOffset	90
5.23.3.3 grabReference	90
5.23.3.4 handles	90
5.23.3.5 maxDrawerDist	91
5.23.3.6 minDrawerDist	91
5.23.3.7 moveAxis	91
5.23.3.8 moveDirection	91
5.23.3.9 openEventFired	91
5.23.4 Event Documentation	91
5.23.4.1 DrawerClosed	91
5.23.4.2 DrawerOpened	92
5.24 SG.SG_DropZone Class Reference	92
5.24.1 Detailed Description	94
5.24.2 Member Function Documentation	94
5.24.2.1 AddObject()	94

5.24.2.2 AddTarget()	95
5.24.2.3 CallObjectDetect()	95
5.24.2.4 CallObjectRemoved()	95
5.24.2.5 CheckDetectionTimes()	95
5.24.2.6 CheckObjectEnter()	96
5.24.2.7 ClearObjects()	96
5.24.2.8 DropZoneEventHandler()	96
5.24.2.9 IsDetected()	96
5.24.2.10 IsTarget()	97
5.24.2.11 ListIndex()	97
5.24.2.12 RemoveObject() [1/2]	97
5.24.2.13 RemoveObject() [2/2]	98
5.24.2.14 ResetZoneAndObjects()	98
5.24.2.15 SetHighLight()	98
5.24.2.16 ValidateRB()	98
5.24.2.17 ValidateSettings()	99
5.24.3 Member Data Documentation	99
5.24.3.1 checkStayTime	99
5.24.3.2 checkStayTimer	99
5.24.3.3 detectHeldObjects	99
5.24.3.4 detectionTime	99
5.24.3.5 dropProperties	100
5.24.3.6 highLighters	100
5.24.3.7 objectsInside	100
5.24.3.8 objectsToGet	100
5.24.3.9 OnObjectDetected	100
5.24.3.10 OnObjectRemoved	100
5.24.3.11 physicsBody	101
5.24.3.12 setup	101
5.24.4 Property Documentation	101
5.24.4.1 AllObjectsDetected	101
5.24.4.2 NumberOfObjects	101
5.24.4.3 ObjectsInside	101
5.24.5 Event Documentation	102
5.24.5.1 ObjectDetected	102
5.24.5.2 ObjectRemoved	102
5.25 SG.SG_FingerFeedback Class Reference	102
5.25.1 Detailed Description	104
5.25.2 Member Function Documentation	104
5.25.2.1 AttachScript()	104
5.25.2.2 DetachScript()	104
5.25.2.3 FindForceDirection()	104

5.25.2.4 GetMaterialScript()	105
5.25.2.5 IsTouching() [1/3]	105
5.25.2.6 IsTouching() [2/3]	105
5.25.2.7 IsTouching() [3/3]	106
5.25.2.8 ObjectDisabled()	106
5.25.2.9 ResetForces()	106
5.25.2.10 SameScript()	106
5.25.2.11 SetupSelf()	107
5.25.2.12 UpdateFeedback()	107
5.25.3 Member Data Documentation	107
5.25.3.1 debugDirections	107
5.25.3.2 entryOrigin	107
5.25.3.3 entryPoint	108
5.25.4 Property Documentation	108
5.25.4.1 DistanceInCollider	108
5.25.4.2 ForceLevel	108
5.25.4.3 TouchedCollider	108
5.25.4.4 TouchedDeformScript	108
5.25.4.5 TouchedMaterialScript	109
5.25.4.6 TouchedObject	109
5.26 SG.SG_GestureGrabScript Class Reference	109
5.26.1 Detailed Description	110
5.26.2 Member Function Documentation	110
5.26.2.1 CanInteract()	110
5.26.2.2 IsTouching()	110
5.26.2.3 Setup()	111
5.26.2.4 UpdateGrabScript()	111
5.26.3 Member Data Documentation	111
5.26.3.1 baseGrabAngles	111
5.26.3.2 baseReleaseAngles	111
5.26.3.3 grabbing	111
5.26.3.4 hoverCollider	112
5.26.3.5 lastAngles	112
5.26.3.6 wantedGrab	112
5.27 SG.SG_Grabable Class Reference	112
5.27.1 Detailed Description	114
5.27.2 Member Function Documentation	114
5.27.2.1 BreakJoint()	114
5.27.2.2 CheckPickupRef()	115
5.27.2.3 ConnectJoint()	115
5.27.2.4 InteractionBegin()	115
5.27.2.5 InteractionEnd()	115

5.27.2.6 IsGrabbed()	116
5.27.2.7 ResetObject()	116
5.27.2.8 SaveRBParameters()	116
5.27.2.9 SaveTransform()	116
5.27.2.10 SetCollision()	116
5.27.2.11 SnapMeTo()	117
5.27.2.12 UpdateInteraction()	117
5.27.2.13 ZeroVelocity()	117
5.27.3 Member Data Documentation	117
5.27.3.1 attachMethod	117
5.27.3.2 canTransfer	118
5.27.3.3 grabOffset	118
5.27.3.4 grabReference	118
5.27.3.5 grabRotation	118
5.27.3.6 physicsBody	118
5.27.3.7 pickupMethod	118
5.27.3.8 pickupReference	119
5.27.3.9 snapReference	119
5.27.3.10 usedGravity	119
5.27.3.11 wasKinematic	119
5.27.4 Property Documentation	119
5.27.4.1 OriginalParent	119
5.27.4.2 UsedGravity	119
5.27.4.3 WasKinematic	120
5.28 SG.SG_GrabScript.SG_GrabEventArgs Class Reference	120
5.28.1 Detailed Description	120
5.28.2 Property Documentation	120
5.28.2.1 Interactable	120
5.29 SG.SG_GrabScript Class Reference	121
5.29.1 Detailed Description	123
5.29.2 Member Function Documentation	123
5.29.2.1 CanInteract()	123
5.29.2.2 CanRelease()	123
5.29.2.3 CheckForScripts()	124
5.29.2.4 ClearHeldObjects()	124
5.29.2.5 EndInteraction()	124
5.29.2.6 GetAngularVelocity()	124
5.29.2.7 GetHardware()	125
5.29.2.8 GetVelocity()	125
5.29.2.9 GrabEventHandler()	125
5.29.2.10 HeldObjects()	126
5.29.2.11 IsGrabbing() [1/2]	126

5.29.2.12 IsGrabbing() [2/2]	126
5.29.2.13 IsTouching()	126
5.29.2.14 ManualRelease()	127
5.29.2.15 OnGrabbedObject()	127
5.29.2.16 OnReleasedObject()	127
5.29.2.17 ReleaseObjectAt()	127
5.29.2.18 Setup()	128
5.29.2.19 TryGrabObject()	128
5.29.2.20 UpdateDynamics()	128
5.29.2.21 UpdateGrabScript()	129
5.29.3 Member Data Documentation	129
5.29.3.1 angularVelocities	129
5.29.3.2 elapsedTime	129
5.29.3.3 grabAnchor	129
5.29.3.4 grabReference	129
5.29.3.5 hardware	130
5.29.3.6 heldObjects	130
5.29.3.7 lastPosition	130
5.29.3.8 lastRotation	130
5.29.3.9 maxDataPoints	130
5.29.3.10 paused	130
5.29.3.11 pauseTime	131
5.29.3.12 setupFinished	131
5.29.3.13 velocities	131
5.29.4 Property Documentation	131
5.29.4.1 DebugEnabled	131
5.29.4.2 Hand	131
5.29.4.3 HardwareReady	131
5.29.5 Event Documentation	132
5.29.5.1 ObjectGrabbed	132
5.29.5.2 ObjectReleased	132
5.30 SG.SG_GrabZone Class Reference	132
5.30.1 Detailed Description	133
5.30.2 Member Function Documentation	133
5.30.2.1 ConnectionIndex()	133
5.30.2.2 ConnectTo()	134
5.30.2.3 InteractionBegin()	134
5.30.2.4 InteractionEnd()	134
5.30.2.5 ResetObject()	135
5.30.2.6 SaveTransform()	135
5.30.2.7 UpdateInteraction()	135
5.30.3 Member Data Documentation	135

5.30.3.1 connectedTo	135
5.31 SG.SG_HandAnimator Class Reference	136
5.31.1 Detailed Description	138
5.31.2 Member Function Documentation	138
5.31.2.1 CalibrateWrist()	138
5.31.2.2 CheckForScripts()	138
5.31.2.3 CollectCorrections()	138
5.31.2.4 CollectFingerJoints()	138
5.31.2.5 CollectHandParameters()	138
5.31.2.6 DifferenceFromWrist()	138
5.31.2.7 GetHardware()	139
5.31.2.8 ResizeHand()	139
5.31.2.9 SenseGlove_OnCalibrationFinished()	139
5.31.2.10 SenseGlove_OnGloveLoaded()	141
5.31.2.11 UpdateHand()	141
5.31.2.12 UpdateWrist()	141
5.31.3 Member Data Documentation	142
5.31.3.1 debugGroup	142
5.31.3.2 debugText	142
5.31.3.3 fingerCorrection	142
5.31.3.4 fingerJoints	142
5.31.3.5 foreArmTransfrom	142
5.31.3.6 resizeFingers	143
5.31.3.7 updateFingers	143
5.31.3.8 updateWrist	143
5.31.3.9 wristAngles	143
5.31.3.10 wristCalibration	143
5.31.3.11 wristCorrection	143
5.31.3.12 wristTransfrom	144
5.31.4 Property Documentation	144
5.31.4.1 Hand	144
5.31.4.2 HardwareReady	144
5.31.4.3 RelativeWrist	144
5.31.4.4 WristAngles	144
5.32 SG.SG_HandDetector Class Reference	145
5.32.1 Detailed Description	147
5.32.2 Member Enumeration Documentation	147
5.32.2.1 DetectionType	147
5.32.3 Member Function Documentation	147
5.32.3.1 AddEntry()	147
5.32.3.2 ContainsSenseGlove()	147
5.32.3.3 FireDetectEvent()	148

5.32.3.4 FireRemoveEvent()	148
5.32.3.5 GlovesInside()	148
5.32.3.6 HandModelIndex()	149
5.32.3.7 RemoveEntry()	149
5.32.3.8 SetHighLight()	149
5.32.3.9 ValidScript()	149
5.32.4 Member Data Documentation	150
5.32.4.1 activationThreshold	150
5.32.4.2 activationTime	150
5.32.4.3 detectedColliders	150
5.32.4.4 detectedGloves	150
5.32.4.5 detectIndex	151
5.32.4.6 detectionMethod	151
5.32.4.7 detectionTimes	151
5.32.4.8 detectMiddle	151
5.32.4.9 detectPinky	151
5.32.4.10 detectRing	151
5.32.4.11 detectThumb	152
5.32.4.12 eventFired	152
5.32.4.13 highLight	152
5.32.4.14 myCollider	152
5.32.4.15 myRigidbody	152
5.32.4.16 singleGlove	152
5.32.5 Event Documentation	153
5.32.5.1 GloveDetected	153
5.32.5.2 GloveRemoved	153
5.33 SG.SG_HandFeedback Class Reference	153
5.33.1 Detailed Description	154
5.33.2 Member Function Documentation	154
5.33.2.1 CheckForScripts()	154
5.33.2.2 GetHardware()	154
5.33.2.3 SetIgnoreCollision()	155
5.33.2.4 SetupScripts()	155
5.33.2.5 TouchingMaterial()	155
5.33.2.6 UpdateForces()	156
5.33.3 Member Data Documentation	156
5.33.3.1 connectedGlove	156
5.33.3.2 fingerFeedbackScripts	156
5.33.3.3 handModel	156
5.33.3.4 wristFeedbackScript	156
5.33.4 Property Documentation	156
5.33.4.1 ColliderDistances	157

5.33.4.2 DebugEnabled	157
5.33.4.3 Hand	157
5.33.4.4 HardwareReady	157
5.34 SG.SG_HandModelInfo Class Reference	157
5.34.1 Detailed Description	158
5.34.2 Member Function Documentation	158
5.34.2.1 GetFingerTip()	158
5.34.3 Member Data Documentation	159
5.34.3.1 fingerDebug	159
5.34.3.2 foreArmTransform	159
5.34.3.3 indexJoints	159
5.34.3.4 middleJoints	159
5.34.3.5 pinkyJoints	160
5.34.3.6 ringJoints	160
5.34.3.7 thumbJoints	160
5.34.3.8 wristDebug	160
5.34.3.9 wristTransform	160
5.34.4 Property Documentation	160
5.34.4.1 DebugEnabled	160
5.34.4.2 FingerJoints	161
5.35 SG.SG_HandRigidBodyes Class Reference	161
5.35.1 Detailed Description	162
5.35.2 Member Function Documentation	162
5.35.2.1 AddRigidBodyes()	162
5.35.2.2 CheckForScripts()	162
5.35.2.3 GetHardware()	163
5.35.2.4 RemoveRigidBodyes()	163
5.35.2.5 SetIgnoreCollision() [1/3]	163
5.35.2.6 SetIgnoreCollision() [2/3]	163
5.35.2.7 SetIgnoreCollision() [3/3]	164
5.35.2.8 SetupSelf()	164
5.35.3 Member Data Documentation	164
5.35.3.1 fingerObjs	164
5.35.3.2 handModel	164
5.35.3.3 wristObj	165
5.35.4 Property Documentation	165
5.35.4.1 CollisionsEnabled	165
5.35.4.2 DebugEnabled	165
5.35.4.3 Hand	165
5.35.4.4 HardwareReady	165
5.36 SG.SG_HandTrigger Class Reference	166
5.36.1 Detailed Description	167

5.36.2 Member Function Documentation	167
5.36.2.1 FireDetectEvent()	167
5.36.2.2 FireHapticFeedback()	167
5.36.2.3 FireRemoveEvent()	168
5.36.2.4 InUse()	168
5.36.2.5 SetEffectObject()	168
5.36.2.6 SetParticles()	169
5.36.3 Member Data Documentation	169
5.36.3.1 audioToPlay	169
5.36.3.2 buzz_CMD_Time	169
5.36.3.3 buzzTimer	169
5.36.3.4 effectToShow	169
5.36.3.5 hapticDuration	170
5.36.3.6 hapticFeedback	170
5.36.3.7 hapticForce	170
5.36.3.8 inUse	170
5.36.3.9 loop	170
5.36.3.10 particlesToPlay	170
5.36.3.11 whichFingers	171
5.37 SG.SG_Hinge Class Reference	171
5.37.1 Detailed Description	173
5.37.2 Member Function Documentation	173
5.37.2.1 CheckLimits()	173
5.37.2.2 GetAngle()	173
5.37.2.3 GetHingeAngle() [1/2]	173
5.37.2.4 GetHingeAngle() [2/2]	174
5.37.2.5 HingeRatio()	174
5.37.2.6 InteractionBegin()	174
5.37.2.7 InteractionEnd()	174
5.37.2.8 RotationAxis()	175
5.37.2.9 SetAngle()	175
5.37.2.10 SetInteractable()	175
5.37.2.11 SetupHinge()	176
5.37.2.12 StopPhysicsBody()	176
5.37.2.13 UpdateInteraction()	176
5.37.3 Member Data Documentation	176
5.37.3.1 autoSetup	176
5.37.3.2 grabReference	176
5.37.3.3 handles	177
5.37.3.4 hingeAxis	177
5.37.3.5 hingePoint	177
5.37.3.6 joint	177

5.37.3.7 maxAngle	177
5.37.3.8 minAngle	177
5.37.3.9 offsetAngle	178
5.37.3.10 physicsBody	178
5.37.3.11 usedGravity	178
5.37.3.12 wasKinematic	178
5.38 SG.SG_HoverCollider Class Reference	178
5.38.1 Detailed Description	179
5.38.2 Member Function Documentation	180
5.38.2.1 AddToList()	180
5.38.2.2 ClearTouchedObjects()	180
5.38.2.3 GetInteractableScript()	180
5.38.2.4 IsTouching() [1/3]	181
5.38.2.5 IsTouching() [2/3]	181
5.38.2.6 IsTouching() [3/3]	181
5.38.2.7 ListIndex()	182
5.38.2.8 MatchingObjects()	182
5.38.2.9 RemoveFromList()	182
5.38.2.10 SameScript()	183
5.38.3 Member Data Documentation	183
5.38.3.1 collidersInside	183
5.38.3.2 interactablesTouched	183
5.38.4 Property Documentation	183
5.38.4.1 TouchedObjects	183
5.39 SG.SG_Interactable Class Reference	184
5.39.1 Detailed Description	186
5.39.2 Member Function Documentation	186
5.39.2.1 BeginInteraction()	186
5.39.2.2 CanInteract()	186
5.39.2.3 EndInteractAllowed()	187
5.39.2.4 EndInteraction() [1/2]	187
5.39.2.5 EndInteraction() [2/2]	187
5.39.2.6 GetTouchIndex()	187
5.39.2.7 InteractingWith()	188
5.39.2.8 InteractionBegin()	188
5.39.2.9 InteractionEnd()	188
5.39.2.10 IsInteracting()	189
5.39.2.11 MustBeReleased()	189
5.39.2.12 ResetObject()	189
5.39.2.13 SaveTransform()	190
5.39.2.14 SetInteractable()	190
5.39.2.15 TouchedBy()	190

5.39.2.16 UnTouchedBy()	190
5.39.2.17 UpdateInteraction()	191
5.39.2.18 WithinBounds()	191
5.39.3 Member Data Documentation	191
5.39.3.1 _grabScript	191
5.39.3.2 fingerPalm	191
5.39.3.3 fingerThumb	191
5.39.3.4 isInteractable	192
5.39.3.5 originalDist	192
5.39.3.6 originalPos	192
5.39.3.7 originalRot	192
5.39.3.8 releaseDistance	192
5.39.3.9 releaseMethod	192
5.39.3.10 touchedColliders	193
5.39.3.11 touchedScripts	193
5.39.4 Property Documentation	193
5.39.4.1 GrabScript	193
5.39.5 Event Documentation	193
5.39.5.1 InteractionBegun	193
5.39.5.2 InteractionEnded	193
5.39.5.3 ObjectReset	194
5.39.5.4 Touched	194
5.39.5.5 UnTouched	194
5.40 SG.SG_InteractArgs Class Reference	194
5.40.1 Detailed Description	195
5.41 SG.SG_KeyBinds Class Reference	195
5.41.1 Detailed Description	195
5.42 SG.SG_Material Class Reference	196
5.42.1 Detailed Description	197
5.42.2 Member Function Documentation	198
5.42.2.1 CalculateForce()	198
5.42.2.2 CalculateHaptics()	198
5.42.2.3 CalculateResponseForce()	198
5.42.2.4 IsBroken()	199
5.42.2.5 LoadMaterialProps() [1/2]	199
5.42.2.6 LoadMaterialProps() [2/2]	199
5.42.2.7 OnDisable()	199
5.42.2.8 UnBreak()	200
5.42.3 Member Data Documentation	200
5.42.3.1 breakable	200
5.42.3.2 brokenBy	200
5.42.3.3 deformScript	200

5.42.3.4 hapticDuration	200
5.42.3.5 hapticFeedback	200
5.42.3.6 hapticMagnitude	201
5.42.3.7 isBroken	201
5.42.3.8 material	201
5.42.3.9 maxForce	201
5.42.3.10 maxForceDist	201
5.42.3.11 minimumFingers	201
5.42.3.12 mustBeGrabbed	202
5.42.3.13 myInteractable	202
5.42.3.14 raisedBreak	202
5.42.3.15 requiresThumb	202
5.42.3.16 yieldDistance	202
5.42.4 Event Documentation	202
5.42.4.1 MaterialBreaks	202
5.43 SG.SG_MeshDeform Class Reference	203
5.43.1 Detailed Description	204
5.43.2 Member Function Documentation	204
5.43.2.1 AddDeform()	204
5.43.2.2 AddDeformation()	204
5.43.2.3 ClearDeformations()	205
5.43.2.4 CollectMeshData()	205
5.43.2.5 DeformMesh()	205
5.43.2.6 RemoveDeform()	205
5.43.2.7 ResetMesh()	207
5.43.2.8 ResetPoints()	207
5.43.2.9 SameVertex()	207
5.43.2.10 SetDeform()	208
5.43.2.11 UpdateMesh()	208
5.43.2.12 UpdatePoint()	208
5.43.3 Member Data Documentation	208
5.43.3.1 atRest	208
5.43.3.2 deformationQueue	209
5.43.3.3 deforms	209
5.43.3.4 deformVerts	209
5.43.3.5 displaceType	209
5.43.3.6 maxDisplacement	209
5.43.3.7 meshFilter	209
5.43.3.8 myMesh	210
5.43.3.9 sameVertices	210
5.43.3.10 uniqueVertices	210
5.43.3.11 verts	210

5.44 SG.SG_PhysicsGrab Class Reference	210
5.44.1 Detailed Description	212
5.44.2 Member Function Documentation	212
5.44.2.1 CanInteract()	212
5.44.2.2 CanRelease()	212
5.44.2.3 CheckForScripts()	213
5.44.2.4 CheckGestures()	213
5.44.2.5 GetGrabables()	213
5.44.2.6 GetMatching() [1/2]	213
5.44.2.7 GetMatching() [2/2]	214
5.44.2.8 IsInside()	214
5.44.2.9 IsTouching()	215
5.44.2.10 Setup()	215
5.44.2.11 UpdateGrabScript()	215
5.44.3 Member Data Documentation	215
5.44.3.1 closedHandThresholds	215
5.44.3.2 handModel	216
5.44.3.3 indexTouch	216
5.44.3.4 middleTouch	216
5.44.3.5 openHandThresholds	216
5.44.3.6 palmTouch	216
5.44.3.7 thumbTouch	216
5.44.3.8 touchScripts	217
5.44.3.9 wantsGrab	217
5.44.4 Property Documentation	217
5.44.4.1 DebugEnabled	217
5.45 SG.Util.SG_QuitKey Class Reference	217
5.46 SG.Util.SG_ResetFloor Class Reference	218
5.47 SG.SG_SenseGloveData Class Reference	218
5.47.1 Detailed Description	220
5.47.2 Constructor & Destructor Documentation	221
5.47.2.1 SG_SenseGloveData() [1/2]	221
5.47.2.2 SG_SenseGloveData() [2/2]	221
5.47.3 Member Function Documentation	221
5.47.3.1 GetChainVariables()	221
5.47.3.2 GetFingerLengths()	222
5.47.3.3 GetJointPositions()	222
5.47.3.4 GetLinkVariables()	222
5.47.3.5 GetSide()	223
5.47.3.6 TotalGloveAngles()	223
5.47.3.7 UpdateVariables()	223
5.47.4 Member Data Documentation	223

5.47.4.1 absoluteCalibratedWrist	223
5.47.4.2 absoluteWrist	224
5.47.4.3 calibrationStep	224
5.47.4.4 commonOriginPos	224
5.47.4.5 commonOriginRot	224
5.47.4.6 dataLoaded	224
5.47.4.7 deviceId	224
5.47.4.8 firmwareVersion	225
5.47.4.9 gloveAngles	225
5.47.4.10 gloveLengths	225
5.47.4.11 glovePositions	225
5.47.4.12 gloveRotations	225
5.47.4.13 gloveSide	225
5.47.4.14 gloveValues	226
5.47.4.15 gloveVersion	226
5.47.4.16 handAngles	226
5.47.4.17 handLengths	226
5.47.4.18 handPositions	226
5.47.4.19 handRotations	226
5.47.4.20 imuCalibration	227
5.47.4.21 imuValues	227
5.47.4.22 numberOfSensors	227
5.47.4.23 packetsPerSecond	227
5.47.4.24 relativeWrist	227
5.47.4.25 totalCalibrationSteps	227
5.47.5 Property Documentation	228
5.47.5.1 Empty	228
5.48 SG.SG_SenseGloveHardware Class Reference	228
5.48.1 Detailed Description	233
5.48.2 Member Enumeration Documentation	233
5.48.2.1 ConnectionMethod	233
5.48.3 Member Function Documentation	233
5.48.3.1 CalibrationFinishedEventHandler()	233
5.48.3.2 CheckCalibration()	234
5.48.3.3 CheckConnection()	234
5.48.3.4 CheckForDeviceManager()	234
5.48.3.5 DisposeDevice()	234
5.48.3.6 FinishCalibration()	234
5.48.3.7 FlushCmds()	235
5.48.3.8 GloveEventHandler()	235
5.48.3.9 HasFunction()	235
5.48.3.10 MatchesConnection()	235

5.48.3.11 OnGloveConnect()	236
5.48.3.12 OnGloveDisconnect()	236
5.48.3.13 OnGloveLink()	236
5.48.3.14 OnGloveUnLink()	236
5.48.3.15 ReadyCalibration()	236
5.48.3.16 ResetKinematics()	237
5.48.3.17 SendBrakeCmd()	237
5.48.3.18 SendBuzzCmd() [1/4]	237
5.48.3.19 SendBuzzCmd() [2/4]	238
5.48.3.20 SendBuzzCmd() [3/4]	238
5.48.3.21 SendBuzzCmd() [4/4]	239
5.48.3.22 SendThumperCmd()	239
5.48.3.23 SetHandParameters()	239
5.48.3.24 SetupDevice()	240
5.48.3.25 StopBrakes()	240
5.48.3.26 StopBuzzMotors()	240
5.48.3.27 StopFeedback()	240
5.48.3.28 UpdateGlove()	241
5.48.3.29 WriteBrakeCmd()	241
5.48.4 Member Data Documentation	241
5.48.4.1 autoConnect	241
5.48.4.2 brakeQueue	241
5.48.4.3 buzz_Enabled	242
5.48.4.4 calibrationArguments	242
5.48.4.5 connectionMethod	242
5.48.4.6 deviceScannerPresent	242
5.48.4.7 lastBrakeLvls	242
5.48.4.8 limitFingers	242
5.48.4.9 linkedGlove	243
5.48.4.10 linkedGloveData	243
5.48.4.11 OnGloveLoad	243
5.48.4.12 sendHaptics	243
5.48.4.13 solver	243
5.48.4.14 thumpFFBThreshold	243
5.48.4.15 updateWrist	244
5.48.4.16 wasConnected	244
5.48.5 Property Documentation	244
5.48.5.1 FingerLengths	244
5.48.5.2 GloveData	244
5.48.5.3 GloveReady	244
5.48.5.4 IsCalibrating	244
5.48.5.5 IsConnected	245

5.48.5.6 IsRight	245
5.48.5.7 StartJointPositions	245
5.48.6 Event Documentation	245
5.48.6.1 CalibrationFinished	245
5.48.6.2 GloveLoaded	245
5.48.6.3 GloveUnLoaded	246
5.49 SG.SG_SimpleTracking Class Reference	246
5.49.1 Detailed Description	247
5.49.2 Member Enumeration Documentation	247
5.49.2.1 UpdateDuring	247
5.49.3 Member Function Documentation	247
5.49.3.1 SetIgnoreCollision()	247
5.49.3.2 SetTrackingTarget()	248
5.49.3.3 UpdatePosition()	248
5.49.4 Member Data Documentation	248
5.49.4.1 positionOffset	248
5.49.4.2 rotationOffset	248
5.49.4.3 trackingTarget	249
5.49.4.4 updateTime	249
5.49.5 Property Documentation	249
5.49.5.1 DebugEnabled	249
5.49.5.2 HasTarget	249
5.49.5.3 TargetPosition	249
5.49.5.4 TargetRotation	250
5.50 SG.SG_SnapDropZone Class Reference	250
5.50.1 Detailed Description	251
5.50.2 Member Enumeration Documentation	251
5.50.2.1 SnapMethod	251
5.50.3 Member Function Documentation	252
5.50.3.1 AddObject()	252
5.50.3.2 AttachObject()	252
5.50.3.3 CallObjectDetect()	252
5.50.3.4 Grabable_InteractionBegun()	253
5.50.3.5 Grabable_InteractionEnded()	253
5.50.3.6 Grabable_ObjectReset()	253
5.50.3.7 IsSnapped()	254
5.50.3.8 ReleaseObject() [1/2]	254
5.50.3.9 ReleaseObject() [2/2]	254
5.50.3.10 RemoveObject()	255
5.50.3.11 ValidateSettings()	255
5.50.4 Member Data Documentation	255
5.50.4.1 disablesInteraction	255

5.50.4.2 snapMethod	255
5.50.4.3 snapPoint	256
5.50.4.4 snapProperties	256
5.50.4.5 takesFromHand	256
5.51 SG.SG_TrackedBody Class Reference	256
5.51.1 Detailed Description	257
5.51.2 Member Function Documentation	257
5.51.2.1 TryAddRB()	257
5.51.2.2 TryRemoveRB()	258
5.51.2.3 UpdatePosition()	258
5.51.3 Member Data Documentation	258
5.51.3.1 physicsBody	258
5.51.3.2 resetDistance	258
5.51.3.3 resetTime	259
5.51.3.4 resetTimer	259
5.51.3.5 rotationSpeed	259
5.51.4 Property Documentation	259
5.51.4.1 CollisionEnabled	259
5.52 SG.SG_TrackedHand Class Reference	259
5.52.1 Detailed Description	261
5.52.2 Member Enumeration Documentation	261
5.52.2.1 TrackingHardware	261
5.52.2.2 TrackingMethod	262
5.52.3 Member Function Documentation	262
5.52.3.1 CheckForScripts()	262
5.52.3.2 SetupTracking()	262
5.52.3.3 SwapTracking()	263
5.52.3.4 UpdateTransformDefault()	263
5.52.3.5 UpdateTransformPhysics()	263
5.52.4 Member Data Documentation	263
5.52.4.1 feedbackScript	263
5.52.4.2 grabScript	263
5.52.4.3 handAnimation	264
5.52.4.4 handModel	264
5.52.4.5 handRB	264
5.52.4.6 hardware	264
5.52.4.7 ignoreGrabables	264
5.52.4.8 physicsTrackingLayer	264
5.52.4.9 physRotationSpeed	265
5.52.4.10 positionOffset	265
5.52.4.11 rigidBodyLayer	265
5.52.4.12 rotationOffset	265

5.52.4.13 trackedObject	265
5.52.4.14 trackingHardware	265
5.52.4.15 trackingMethod	266
5.52.5 Property Documentation	266
5.52.5.1 TargetPosition	266
5.52.5.2 TargetRotation	266
5.52.5.3 TracksRightHand	266
5.53 SG.SG_User Class Reference	267
5.53.1 Detailed Description	267
5.53.2 Member Function Documentation	267
5.53.2.1 SetupHands()	267
5.54 SG.SG_Util Class Reference	268
5.54.1 Detailed Description	269
5.54.2 Member Function Documentation	269
5.54.2.1 AppendButtonText()	269
5.54.2.2 Average() [1/2]	270
5.54.2.3 Average() [2/2]	270
5.54.2.4 AxisIndex()	270
5.54.2.5 CalculateAngularVelocity()	271
5.54.2.6 CalculateOffsets()	271
5.54.2.7 CheckForHandInfo()	272
5.54.2.8 CheckForTrackedHand()	272
5.54.2.9 GetAxis()	272
5.54.2.10 GetVector()	273
5.54.2.11 IsNegative()	273
5.54.2.12 Map()	273
5.54.2.13 NormalizeAngle() [1/2]	274
5.54.2.14 NormalizeAngle() [2/2]	274
5.54.2.15 NormalizeAngles()	274
5.54.2.16 SetChildren()	275
5.54.2.17 SpawnSphere()	275
5.54.2.18 ToEuler()	276
5.54.2.19 ToPosition() [1/2]	276
5.54.2.20 ToPosition() [2/2]	276
5.54.2.21 ToQuaternion()	277
5.54.2.22 ToString() [1/4]	277
5.54.2.23 ToString() [2/4]	277
5.54.2.24 ToString() [3/4]	278
5.54.2.25 ToString() [4/4]	278
5.54.2.26 ToUnityEuler()	278
5.54.2.27 ToUnityPosition() [1/2]	279
5.54.2.28 ToUnityPosition() [2/2]	279

5.54.2.29 ToUnityQuaternion()	279
5.54.2.30 TransformRigidBody()	281
5.54.2.31 TryAddRB()	281
5.54.2.32 TryRemoveRB()	282
5.55 SG.SG_WireFrame Class Reference	282
5.55.1 Detailed Description	284
5.55.2 Member Function Documentation	284
5.55.2.1 CollectFingerJoints()	284
5.55.2.2 ResizeHand()	284
5.55.2.3 SenseGlove_OnCalibrationFinished()	284
5.55.2.4 SenseGlove_OnGloveLoaded()	285
5.55.2.5 SetGlove()	285
5.55.2.6 SetHand()	285
5.55.2.7 SetupFingers()	285
5.55.2.8 SetupGlove()	286
5.55.2.9 SetupHandPalm()	286
5.55.2.10 UpdateHand()	286
5.55.3 Member Data Documentation	287
5.55.3.1 gloveBase	287
5.55.3.2 gloveJoints	287
5.55.3.3 gloveSectionModel	287
5.55.3.4 handBase	287
5.55.3.5 handPalmModel	287
5.55.3.6 phalangeModel	288
5.55.3.7 previewGroup	288
5.55.3.8 setupComplete	288
5.55.3.9 toggleGloveKey	288
5.55.3.10 toggleHandKey	288
5.56 SG.SGEvent Class Reference	289
5.57 SG.Examples.SGEx_Diagnostics Class Reference	289
5.57.1 Detailed Description	290
5.58 SG.Examples.SGEx_ForceFeedbackObjects Class Reference	290
5.59 SG.Examples.SGEx_HandLayerUI Class Reference	292
5.59.1 Member Data Documentation	293
5.59.1.1 instructionTexts	294
5.59.1.2 linkedLayers	294
5.60 SG.Examples.SGEx_RotateSimple Class Reference	295
5.60.1 Detailed Description	295
5.61 SG.Examples.SGEx_SelectHandModel Class Reference	295
5.62 SG.Examples.SGEx_ShowGloveAngles Class Reference	296
5.63 SG.SG_SnapDropZone.SnapProps Class Reference	297
5.63.1 Detailed Description	298

5.63.2 Constructor & Destructor Documentation	298
5.63.2.1 SnapProps()	298
5.63.3 Member Function Documentation	298
5.63.3.1 BreakJoint()	298
5.63.3.2 CreateJoint()	298
5.63.3.3 RestoreProperties()	299
5.63.4 Member Data Documentation	299
5.63.4.1 isSnapped	299
5.63.4.2 myJoint	299
5.63.4.3 oldParent	299
5.63.4.4 usedGravity	299
5.63.4.5 wasInteractable	300
5.63.4.6 wasKinematic	300
Index	301

Chapter 1

Namespace Index

1.1 Packages

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

SG	9
SG.Calibration	14
SG.Examples	14
SG.Materials	14
SG.Util	15

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

SG.SG_SenseGloveHardware.BuzzCmd	17
SG.Calibration.CalibrationPose	18
SG.Materials.Deformation	25
SG.SG_DropZone.DropProps	28
EventArgs	
SG.SG_DeviceManager.DeviceDetectedArgs	27
SG.SG_DropZone.DropZoneArgs	29
SG.SG_GrabScript.SG_GrabEventArgs	120
SG.SG_HandDetector.GloveDetectionArgs	33
SG.SG_InteractArgs	194
SG.SG_SenseGloveHardware.GloveCalibrationArgs	31
SG.Util.FileIO	30
SG.Materials.MaterialProps	34
MonoBehaviour	
SG.Calibration.SG_CalibrationSequence	57
SG.Examples.SGEx_Diagnostics	289
SG.Examples.SGEx_ForceFeedbackObjects	290
SG.Examples.SGEx_HandLayerUI	292
SG.Examples.SGEx_RotateSimple	295
SG.Examples.SGEx_SelectHandModel	295
SG.Examples.SGEx_ShowGloveAngles	296
SG.SG_Breakable	46
SG.SG_BreakableContainer	52
SG.SG_Debugger	68
SG.SG_DeviceLink	72
SG.SG_SenseGloveHardware	228
SG.SG_DeviceManager	74
SG.SG_DropZone	92
SG.SG_SnapDropZone	250
SG.SG_GrabScript	121
SG.SG_GestureGrabScript	109
SG.SG_PhysicsGrab	210
SG.SG_HandAnimator	136
SG.SG_AutoHandAnimation	38
SG.SG_WireFrame	282

SG.SG_HandDetector	145
SG.SG_HandTrigger	166
SG.SG_HandFeedback	153
SG.SG_HandModelInfo	157
SG.SG_HandRigidBodyes	161
SG.SG_Interactive	184
SG.SG_DetectGrab	71
SG.SG_Dial	77
SG.SG_Drawer	85
SG.SG_Grabable	112
SG.SG_GrabZone	132
SG.SG_Hinge	171
SG.SG_Door	83
SG.SG_KeyBinds	195
SG.SG_Material	196
SG.SG_MeshDeform	203
SG.SG_SimpleTracking	246
SG.SG_BasicFeedback	40
SG.SG_FingerFeedback	102
SG.SG_HoverCollider	178
SG.SG_TrackedBody	256
SG.SG_TrackedHand	259
SG.SG_User	267
SG.Util.SG_QuitKey	217
SG.Util.SG_ResetFloor	218
SG.Calibration.SG_CalibrationStorage	65
SG.SG_SenseGloveData	218
SG.SG_Util	268
SG.SG_SnapDropZone.SnapProps	297
UnityEvent	
SG.SGEvent	289

Chapter 3

Class Index

3.1 Class List

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

SG.SG_SenseGloveHardware.BuzzCmd	17
SG.Calibration.CalibrationPose Configurable Calibration poses for SenseGlove solvers. Tweak at thyne own risk	18
SG.Materials.Deformation Contains all variables needed to perform Deformations, and to evaluate two deformations	25
SG.SG_DeviceManager.DeviceDetectedArgs Arguments for the GloveDetected Event	27
SG.SG_DropZone.DropProps Properties that assist in object detection	28
SG.SG_DropZone.DropZoneArgs	29
SG.Util.FileIO Ensures that .txt files are properly handled by Unity	30
SG.SG_SenseGloveHardware.GloveCalibrationArgs CalibrationArguments, containing both old an new finger lengths and joint positions	31
SG.SG_HandDetector.GloveDetectionArgs EventArgs fired when a glove is detected in or removed from a SenseGlove_Detector	33
SG.Materials.MaterialProps Contains the editable Material Properties of a single SenseGlove_Material	34
SG.SG_AutoHandAnimation A HandAnimator that grabs its animation info from a SG_HandModelInfo script	38
SG.SG_BasicFeedback Attach to a collider and it will send haptic feedback to a SenseGlove on impact. Optionally tracks a GameObject. Extended by SG_Finger to apply more forces	40
SG.SG_Breakable A Gameobject that despawns an objects once its material breaks, and optionally replaces it with a 'broken' version	46
SG.SG_BreakableContainer A SenseGlove_Breakable that contains objects and optionally spawns shards of itself upon breaking	52
SG.Calibration.SG_CalibrationSequence Manobehaviour meant to run the user though two general calibration steps, and then allows them to refine their calibration	57
SG.Calibration.SG_CalibrationStorage Class responsible for storing and retrieving Sense Glove calibration on disk	65

SG.SG_Debugger	Utility Script that allows access to the internal debugger of the SenseGloveCs Library, and controls debug messages from the SenseGlove SDK specifically	68
SG.SG_DetectGrab	Attach this to any GameObject with a collider to have SG_Grabscripts detect it. Does not add any manipulation	71
SG.SG_DeviceLink	Link to a Sense Glove Device	72
SG.SG_DeviceManager	Class that links SenseGlove hardware to object in the Unity Engine	74
SG.SG_Dial	A knob that can be twisted along its axis. Used in intricate button panels	77
SG.SG_Door	A SenseGlove_Hinge that represents a door. Can raise opened / closed events and have hidden content	83
SG.SG_Drawer	A SG_Interactable that moves along one (local) axis	85
SG.SG_DropZone	Detects SenseGlove_Grabables within its volume	92
SG.SG_FingerFeedback	Extends impact feedback to also take into account force feedback from SG_Material 's. These scripts calculate their distance into a collider	102
SG.SG_GestureGrabScript	A simplified SenseGlove_GrabScript that grabs all objects within it's 'hover collider' when a grab gesture is made	109
SG.SG_Grabable	An object that can be picked up and dropped by the SenseGlove	112
SG.SG_GrabScript.SG_GrabEventArgs	Event Arguments for grabbing/releasing of objects	120
SG.SG_GrabScript	A Grabscrip that uses a number of the Sense Glove's data to start and end interactions	121
SG.SG_GrabZone	Creates a zone that extends its SG_Interactable methods to other objects, essentially creating a handle for (multiple) other Interactables	132
SG.SG_HandAnimator	A Generic Script that can be extended to work with most hand models. It requires the developer to assign the correct transforms for each joint. All of its methods can be overridden to create custom solutions	136
SG.SG_HandDetector	A class to detect a SG_HandAnimator based on its SG_Feedback colliders	145
SG.SG_HandFeedback	This script collects the Force Feedback from the hand and sends these to its connected Hardware	153
SG.SG_HandModelInfo	A script to assign information of hand joints, used by other scripts that use hand tracking . . .	157
SG.SG_HandRigidBody	A script to manage a set of Rigidbodies that represent the hand geometry	161
SG.SG_HandTrigger	A Detector that, when activated, triggers a series of in-game effects	166
SG.SG_Hinge	Represents an Interactable that can rotate around a specified point and axis. Used to extend doors and levers	171
SG.SG_HoverCollider	A script that keeps track of multiple SG_Interactable objects it collides with	178
SG.SG_Interactable	Represents an object that a SenseGlove Grabscrip can interact with. Extended by most of the Interaction scripts	184
SG.SG_InteractArgs	Contains event arguments	194

SG.SG_KeyBinds	A Keybinds component that can be attached to a TrackedHand so we may access certain functions through buttons or hotkeys	195
SG.SG_Material	A class that contains material properties for a virtual objects, which can be customized, hard-coded or loaded during runtime	196
SG.SG_MeshDeform	A class that can hook itself up to a SG_Interactable or material, and deform its mesh	203
SG.SG_PhysicsGrab	A simplified version of the original SenseGlove_PhysGrab script; If an object is touched by finger-thumb or by palm-finger	210
SG.Util.SG_QuitKey		217
SG.Util.SG_ResetFloor		218
SG.SG_SenseGloveData	Unity wrapper for the GloveData, which contains all a developer will need	218
SG.SG_SenseGloveHardware	After being linked to a proper Sense Glove via the SenseGlove_DeviceManager, this script is responsible for updating SG_SenseGloveData every frame, and for exposing feedback - and calibration methods	228
SG.SG_SimpleTracking	Attached to a GameObject to make it follow a 'target'	246
SG.SG_SnapDropZone	A DropZone that snaps a Grabable to a specific SnapPoint	250
SG.SG_TrackedBody	A Rigidbody that tracks a transform by adding velocity to the body, rather than directly applying positions. It reverts back to simpleTrackign if no Rigidbody is present	256
SG.SG_TrackedHand	A hand model with different layers, that follows a GameObject with a configurable offset	259
SG.SG_User	Utility Class to manage up to two SG_TrackedHands, and to swap their hands around	267
SG.SG_Util	Contains methods that make the SenseGloveCs library work with Unity	268
SG.SG_WireFrame	Type of SG_HandAnimator to debug hardware- and software models	282
SG.SGEvent		289
SG.Examples.SGEx_Diagnostics	Allows one to access certain Sense Glove fucntions using the keys on the keyboard	289
SG.Examples.SGEx_ForceFeedbackObjects		290
SG.Examples.SGEx_HandLayerUI		292
SG.Examples.SGEx_RotateSimple	A script to rotate an object around a specified axis	295
SG.Examples.SGEx_SelectHandModel		295
SG.Examples.SGEx_ShowGloveAngles		296
SG.SG_SnapDropZone.SnapProps	Contains parameters that assist in snapping/unsnapping to a SnapZone	297

Chapter 4

Namespace Documentation

4.1 SG Namespace Reference

Classes

- class [SG_AutoHandAnimation](#)
A HandAnimator that grabs its animation info from a [SG_HandModelInfo](#) script.
- class [SG_BasicFeedback](#)
Attach to a collider and it will send haptic feedback to a SenseGlove on impact. Optionally tracks a GameObject. Extended by [SG_Finger](#) to apply more forces.
- class [SG_Breakable](#)
A Gameobject that despawns an objects once its material breaks, and optionally replaces it with a 'broken' version.
- class [SG_BreakableContainer](#)
A SenseGlove_Breakable that contains objects and optionally spawns shards of itself upon breaking.
- class [SG_Debugger](#)
Utility Script that allows access to the internal debugger of the SenseGloveCs Library, and controls debug messages from the SenseGlove SDK specifically.
- class [SG_DetectGrab](#)
Attach this to any GameObject with a collider to have [SG_Grabscripts](#) detect it. Does not add any manipulation.
- class [SG_DeviceLink](#)
Link to a Sense Glove Device.
- class [SG_DeviceManager](#)
Class that links SenseGlove hardware to object in the Unity Engine.
- class [SG_Dial](#)
A knob that can be twisted along its axis. Used in intricate button panels.
- class [SG_Door](#)
A SenseGlove_Hinge that represents a door. Can raise opened / closed events and have hidden content.
- class [SG_Drawer](#)
A [SG_Interactive](#) that moves along one (local) axis.
- class [SG_DropZone](#)
Detects SenseGlove_Grabables within its volume.
- class [SG_FingerFeedback](#)
Extends impact feedback to also take into account force feedback from [SG_Material](#)'s. These scripts calculate their distance into a collider.
- class [SG_GestureGrabScript](#)
A simplified SenseGlove_GrabScript that grabs all objects within it's 'hover collider' when a grab gesture is made.

- class [SG_Grabable](#)
An object that can be picked up and dropped by the SenseGlove.
- class [SG_GrabScript](#)
A Grabscrip that uses a number of the Sense Glove's data to start and end interactions.
- class [SG_GrabZone](#)
Creates a zone that extends its [SG_Interactable](#) methods to other objects, essentially creating a handle for (multiple) other Interactables.
- class [SG_HandAnimator](#)
A Generic Script that can be extended to work with most hand models. It requires the developer to assign the correct transforms for each joint. All of its methods can be overridden to create custom solutions.
- class [SG_HandDetector](#)
A class to detect a [SG_HandAnimator](#) based on its [SG_Feedback](#) colliders
- class [SG_HandFeedback](#)
This script collects the Force Feedback from the hand and sends these to its connected Hardware.
- class [SG_HandModelInfo](#)
A script to assign information of hand joints, used by other scripts that use hand tracking.
- class [SG_HandRigidBodies](#)
A script to manage a set of Rigidbodies that represent the hand geometry.
- class [SG_HandTrigger](#)
A Detector that, when activated, triggers a series of in-game effects.
- class [SG_Hinge](#)
Represents an Interactable that can rotate around a specified point and axis. Used to extend doors and levers.
- class [SG_HoverCollider](#)
A script that keeps track of multiple [SG_Interactable](#) objects it collides with.
- class [SG_Interactable](#)
Represents an object that a SenseGlove Grabscrip can interact with. Extended by most of the Interaction scripts.
- class [SG_InteractArgs](#)
Contains event arguments
- class [SG_KeyBinds](#)
A Keybinds component that can be attached to a TrackedHand so we may access certain functions through buttons or hotkeys.
- class [SG_Material](#)
A class that contains material properties for a virtual objects, which can be customized, hard-coded or loaded during runtime.
- class [SG_MeshDeform](#)
A class that can hook itself up to a [SG_Interactable](#) or material, and deform its mesh.
- class [SG_PhysicsGrab](#)
A simplified version of the original SenseGlove_PhysGrab script; If an object is touched by finger-thumb or by palm-finger
- class [SG_SenseGloveData](#)
Unity wrapper for the GloveData, which contains all a developer will need.
- class [SG_SenseGloveHardware](#)
After being linked to a proper Sense Glove via the [SenseGlove_DeviceManager](#), this script is responsible for updating [SG_SenseGloveData](#) every frame, and for exposing feedback - and calibration methods.
- class [SG_SimpleTracking](#)
Attached to a GameObject to make it follow a 'target'
- class [SG_SnapDropZone](#)
A DropZone that snaps a Grabable to a specific SnapPoint.
- class [SG_TrackedBody](#)
A Rigidbody that tracks a transform by adding velocity to the body, rather than directly applying positions. It reverts back to simpleTrackign if no Rigidbody is present.
- class [SG_TrackedHand](#)

- class [SG_User](#)
A hand model with different layers, that follows a GameObject with a configurable offset
- class [SG_Util](#)
Utility Class to manage up to two SG_TrackedHands, and to swap their hands around.
- class [SG_WireFrame](#)
Contains methods that make the SenseGloveCs library work with Unity.
- class [SGEvent](#)
Type of SG_HandAnimator to debug hardware- and software models.

Enumerations

- enum [GloveSide](#) { [GloveSide.Unknown](#) = 0, [GloveSide.RightHand](#), [GloveSide.LeftHand](#) }
Whether this glove is left- or right handed.
- enum [SG_HandSection](#) { **Thumb** = 0, **Index**, **Middle**, **Ring**, **Pinky**, **Wrist**, **Unknown** }
Represents different sections of the hand, used to determine feedback location.
- enum [MovementAxis](#) { **X** = 0, **Y** = 1, **Z** = 2 }
The axis along which the drawer is moved.
- enum [GrabType](#) { [GrabType.Follow](#) = 0, [GrabType.FixedJoint](#), [GrabType.Parent](#) }
The way in which this Grabscrip picks up SG_Interactable objects.
- enum [AttachType](#) { [AttachType.Default](#) = 0, [AttachType.SnapToAnchor](#) }
The way that this SG_Grabable attaches to a GrabScript that tries to pick it up.
- enum [ReleaseMethod](#) { [ReleaseMethod.Default](#) = 0, [ReleaseMethod.MustOpenHand](#), [ReleaseMethod.FunctionCall](#) }
Parameter that determines how this object ends its interaction.

4.1.1 Enumeration Type Documentation

4.1.1.1 AttachType

```
enum SG.AttachType [strong]
```

The way that this [SG_Grabable](#) attaches to a GrabScript that tries to pick it up.

Enumerator

Default	Default. The object keeps its current position.
SnapToAnchor	The object snaps to the Grabscrip in a predefined position and orientation; useful for tools etc.

4.1.1.2 GloveSide

```
enum SG.GloveSide [strong]
```

Whether this glove is left- or right handed.

Enumerator

Unknown	No data about this glove is available yet.
RightHand	This is a right hand.
LeftHand	This is a left hand.

4.1.1.3 GrabType

```
enum SG.GrabType [strong]
```

The way in which this Grabscrip picks up [SG_Interactive](#) objects.

Enumerator

Follow	The grabbed object's transform follows that of the GrabReference through world coordinates. Does not interfere with VRTK scripts.
FixedJoint	A FixedJoint is created between the grabbed object and the GrabReference, which stops it from passing through rigidbodies.
Parent	The object becomes a child of the Grabreference. Its original parent is restored upon release.

4.1.1.4 MovementAxis

```
enum SG.MovementAxis [strong]
```

The axis along which the drawer is moved.

4.1.1.5 ReleaseMethod

```
enum SG.ReleaseMethod [strong]
```

Parameter that determines how this object ends its interaction.

Enumerator

Default	The Interactive behaves as determined by the GrabScript that interacts with it.
MustOpenHand	The Interactive may only be released if the Hand is sufficiently "open". Used to improve interaction of objects that move along specified paths.
FunctionCall	The interactable is only released when the EndInteraction or ResetObject functions are called.

4.1.1.6 SG_HandSection

enum [SG.SG_HandSection](#) [strong]

Represents different sections of the hand, used to determine feedback location.

4.2 SG.Calibration Namespace Reference

Classes

- class [CalibrationPose](#)
Configurable [Calibration](#) poses for SenseGlove solvers. Tweak at thyne own risk.
- class [SG_CalibrationSequence](#)
Manobehaviour meant to run the user though two general calibration steps, and then allows them to refine their calibration
- class [SG_CalibrationStorage](#)
Class responsible for storing and retrieving Sense Glove calibration on disk.

4.3 SG.Examples Namespace Reference

Classes

- class [SGEx_Diagnostics](#)
Allows one to access certain Sense Glove fuctions using the keys on the keyboard.
- class [SGEx_ForceFeedbackObjects](#)
- class [SGEx_HandLayerUI](#)
- class [SGEx_RotateSimple](#)
A script to rotate an object around a specified axis
- class [SGEx_SelectHandModel](#)
- class [SGEx_ShowGloveAngles](#)

4.4 SG.Materials Namespace Reference

Classes

- struct [Deformation](#)
Contains all variables needed to perform Deformations, and to evaluate two deformations.
- struct [MaterialProps](#)
Contains the editable Material Properties of a single SenseGlove_Material

Enumerations

- enum [VirtualMaterial](#) { [VirtualMaterial.Custom](#) = 0, [VirtualMaterial.Steel](#), [VirtualMaterial.Rubber](#), [VirtualMaterial.Egg](#) }
Determines how the material properties are loaded.
- enum [DisplaceType](#) { [DisplaceType.Plane](#) = 0 }
The method by which the mesh will be displaced using the SenseGlove_Feedback entry vector.

4.4.1 Enumeration Type Documentation

4.4.1.1 DisplaceType

enum [SG.Materials.DisplaceType](#) [strong]

The method by which the mesh will be displaced using the SenseGlove_Feedback entry vector.

Enumerator

Plane	Squashed the vertices as though they are pressed against a glass window.
-------	--

4.4.1.2 VirtualMaterial

enum [SG.Materials.VirtualMaterial](#) [strong]

Determines how the material properties are loaded.

Enumerator

Custom	Material Properties can be assigned via the inspector.
Steel	Assigns properties of the hardest material.
Rubber	Assigns properties of a medium-soft material.
Egg	Assigns properties of a soft material that is breakable.

4.5 SG.Util Namespace Reference

Classes

- class [FileIO](#)
Ensures that .txt files are properly handled by Unity.
- class [SG_QuitKey](#)
- class [SG_ResetFloor](#)

Chapter 5

Class Documentation

5.1 SG.SG_SenseGloveHardware.BuzzCmd Class Reference

Public Member Functions

- **BuzzCmd** (bool[] fin, int[] magn, int[] dur)
- void **Update** (float deltaTime)
- void **Merge** (ref int[] buffer)

Public Attributes

- bool[] **fingers**
- float[] **durations**
- float[] **times**
- int[] **magnitudes**

Static Public Attributes

- static readonly BuzzMotorPattern[] **patterns**

Protected Attributes

- int **elapsed** = 0

Properties

- bool **FullyElapsed** [get]

5.1.1 Member Data Documentation

5.1.1.1 patterns

```
readonly BuzzMotorPattern [ ] SG.SG_SenseGloveHardware.BuzzCmd.patterns [static]
```

Initial value:

```
= new BuzzMotorPattern[5]
    { BuzzMotorPattern.Constant, BuzzMotorPattern.Constant,
      BuzzMotorPattern.Constant, BuzzMotorPattern.Constant, BuzzMotorPattern.Constant }
```

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_SenseGloveHardware.cs

5.2 SG.Calibration.CalibrationPose Class Reference

Configurable [Calibration](#) poses for SenseGlove solvers. Tweak at thyne own risk.

Public Member Functions

- [CalibrationPose](#) (int[][] affects, int[][] valueIndices)
Create a new pose that does not affect output (y) components
- [CalibrationPose](#) (int[][] affects, int[][] valueIndices, int[][] yAffects, float[][] yValues)
Create a new pose that affects output (y) components
- void [CalibrateParameters](#) (Vector3[] calibrationValues, ref InterpolationSet_IMU interpolator)
Calibrate all parameters of an InterpolationSet, based on this pose's parameters and a set of input values.

Static Public Member Functions

- static [CalibrationPose GetFist](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to all fingers flexed (finger flexion calibration).
- static [CalibrationPose GetOpenHand](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to all fingers extended (finger extension calibration).
- static [CalibrationPose GetThumbsUp](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a thumb up (thumb extended calibration)
- static [CalibrationPose GetThumbFlexed](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a flexed thumb (thumb flexed calibration)
- static [CalibrationPose GetThumbAbd](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a thumb moved outwards (thumb abduction calibration)
- static [CalibrationPose GetThumbNoAbd](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a thumb flat against the hand palm (thumb adduction calibration)
- static [CalibrationPose GetFullOpen](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a fully opened hand (finger extension, thumb adduction calibration)
- static [CalibrationPose GetFullFist](#) (ref InterpolationSet_IMU interpolator)
Generates a calibration pose that corresponds to a fully gclosed hand (finger flexion, thumb abduction calibration)

Static Protected Member Functions

- static `int[][] SetupArray` (ref InterpolationSet_IMU interpolator)
Utility function that creates an array of integers of the appropriate size, with default values -1 (none)
- static `float[][] SetupFloatArray` (ref InterpolationSet_IMU interpolator)
Utility function that creates an array of floats of the appropriate size, with default values 0

Static Protected Attributes

- static readonly `int x0` = 0
Useful indices for interpolation
- static readonly `int abd` = 2
Useful indices for movements

Private Attributes

- `int[][] xAffect`
indicates this pose is meant to calibrate the x0 (0) or x1 (1) value for this finger, or no value at all (-1)
- `int[][] calbrUsing`
Which value to use for calibration (flexion, abduction, twist, none)
- `int[][] yAffect`
indicates this pose is meant to calibrate the y0 (0) or y1 (1) value for this finger, or no value at all (-1)
- `float[][] yValue`
//the y value to set, in case this motion sets the output (y) component.

Static Private Attributes

- static readonly `int x1` = 1
- static readonly `int none` = -1
- static readonly `int y0` = 0
- static readonly `int y1` = 1
- static readonly `int flex` = 1
- static readonly `int tw` = 0

5.2.1 Detailed Description

Configurable [Calibration](#) poses for SenseGlove solvers. Tweak at thyne own risk.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CalibrationPose() [1/2]

```
SG.Calibration.CalibrationPose.CalibrationPose (
    int affects[][],
    int valueIndices[][] )
```

Create a new pose that does not affect output (y) components

Parameters

<i>affects</i>	
<i>valueIndices</i>	

5.2.2.2 CalibrationPose() [2/2]

```
SG.Calibration.CalibrationPose.CalibrationPose (
    int affects[[]],
    int valueIndices[[]],
    int yAffects[[]],
    float yValues[[]] )
```

Create a new pose that affects output (y) components

Parameters

<i>affects</i>	
<i>valueIndices</i>	
<i>yAffects</i>	
<i>yValues</i>	

5.2.3 Member Function Documentation**5.2.3.1 CalibrateParameters()**

```
void SG.Calibration.CalibrationPose.CalibrateParameters (
    Vector3[] calibrationValues,
    ref InterpolationSet_IMU interpolator )
```

Calibrate all parameters of an InterpolationSet, based on this pose's parameters and a set of input values.

Parameters

<i>calibrationValues</i>	
<i>interpolator</i>	

5.2.3.2 GetFist()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetFist (
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to all fingers flexed (finger flexion calibration).

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.3 GetFullFist()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetFullFist (  
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a fully gclosed hand (finger flexion, thumb abduction calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.4 GetFullOpen()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetFullOpen (  
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a fully opened hand (finger extension, thumb adduction calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.5 GetOpenHand()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetOpenHand (
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to all fingers extended (finger extension calibration).

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.6 GetThumbAbd()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetThumbAbd (
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a thumb moved outwards (thumb abduction calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.7 GetThumbFlexed()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetThumbFlexed (
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a flexed thumb (thumb flexed calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.8 GetThumbNoAbd()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetThumbNoAbd (  
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a thumb flat against the hand palm (thumb adduction calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.9 GetThumbsUp()

```
static CalibrationPose SG.Calibration.CalibrationPose.GetThumbsUp (  
    ref InterpolationSet_IMU interpolator ) [static]
```

Generates a calibration pose that corresponds to a thumb up (thumb extended calibration)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.10 SetupArray()

```
static int [][] SG.Calibration.CalibrationPose.SetupArray (  
    ref InterpolationSet_IMU interpolator ) [static], [protected]
```

Utility function that creates an array of integers of the appropriate size, with default values -1 (none)

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.3.11 SetupFloatArray()

```
static float [][] SG.Calibration.CalibrationPose.SetupFloatArray (  
    ref InterpolationSet_IMU interpolator ) [static], [protected]
```

Utility function that creates an array of floats of the appropriate size, with default values 0

Parameters

<i>interpolator</i>	
---------------------	--

Returns

5.2.4 Member Data Documentation

5.2.4.1 abd

```
readonly int SG.Calibration.CalibrationPose.abd = 2 [static], [protected]
```

Useful indices for movements

5.2.4.2 calbrUsing

```
int [][] SG.Calibration.CalibrationPose.calbrUsing [private]
```

Which value to use for calibration (flexion, abduction, twist, none)

5.2.4.3 x0

```
readonly int SG.Calibration.CalibrationPose.x0 = 0 [static], [protected]
```

Useful indices for interpolation

5.2.4.4 xAffect

```
int [][] SG.Calibration.CalibrationPose.xAffect [private]
```

indicates this pose is meant to calibrate the x0 (0) or x1 (1) value for this finger, or no value at all (-1)

5.2.4.5 yAffect

```
int [][] SG.Calibration.CalibrationPose.yAffect [private]
```

indicates this pose is meant to calibrate the y0 (0) or y1 (1) value for this finger, or no value at all (-1)

5.2.4.6 yValue

```
float [][] SG.Calibration.CalibrationPose.yValue [private]
```

//the y value to set, in case this motion sets the output (y) component.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Calibration/Resources/SG_↔ CalibrationPoses.cs

5.3 SG.Materials.Deformation Struct Reference

Contains all variables needed to perform Deformations, and to evaluate two deformations.

Public Member Functions

- [Deformation](#) (Vector3 absEntryVect, Vector3 absDefPosition, float dist)
Create a new [Deformation](#) data struct.

Public Attributes

- Vector3 [absEntryVector](#)
The absolute entry vector of the [Deformation](#)
- Vector3 [absDeformPosition](#)
The (current) absolute position of the deformation.
- float [distance](#)
How far the abdDeformPosition is from the entry point

5.3.1 Detailed Description

Contains all variables needed to perform Deformations, and to evaluate two deformations.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 Deformation()

```
SG.Materials.Deformation.Deformation (
    Vector3 absEntryVect,
    Vector3 absDefPosition,
    float dist )
```

Create a new [Deformation](#) data struct.

Parameters

<i>absEntryVect</i>	
<i>absPosition</i>	
<i>dist</i>	

5.3.3 Member Data Documentation

5.3.3.1 absDeformPosition

```
Vector3 SG.Materials.Deformation.absDeformPosition
```

The (current) absolute position of the deformation.

5.3.3.2 absEntryVector

```
Vector3 SG.Materials.Deformation.absEntryVector
```

The absolute entry vector of the [Deformation](#)

5.3.3.3 distance

```
float SG.Materials.Deformation.distance
```

How far the abdDeformPosition is from the entry point

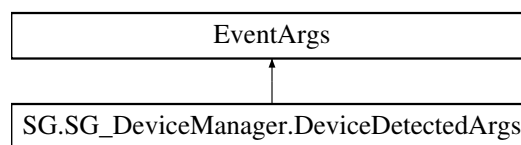
The documentation for this struct was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_Mesh↔ Deform.cs

5.4 SG.SG_DeviceManager.DeviceDetectedArgs Class Reference

Arguments for the GloveDetected Event.

Inheritance diagram for SG.SG_DeviceManager.DeviceDetectedArgs:



Public Member Functions

- [DeviceDetectedArgs](#) (string id, int gloveIndex, SenseGloveCs.DeviceType deviceType)
Create a new instance of the GloveDetectedArgs.

Properties

- string [DeviceID](#) [get, private set]
The unique hardware ID of the detected glove.
- int [DeviceIndex](#) [get, private set]
The index of the detected glove within the SenseGlove_DeviceManager memory.
- SenseGloveCs.DeviceType [Type](#) [get, private set]
The DeviceType that has been found

5.4.1 Detailed Description

Arguments for the GloveDetected Event.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 DeviceDetectedArgs()

```
SG.SG_DeviceManager.DeviceDetectedArgs.DeviceDetectedArgs (
    string id,
    int gloveIndex,
    SenseGloveCs.DeviceType deviceType )
```

Create a new instance of the GloveDetectedArgs.

Parameters

<i>glove</i>	
--------------	--

5.4.3 Property Documentation**5.4.3.1 DeviceID**

```
string SG.SG_DeviceManager.DeviceDetectedArgs.DeviceID [get], [private set]
```

The unique hardware ID of the detected glove.

5.4.3.2 DeviceIndex

```
int SG.SG_DeviceManager.DeviceDetectedArgs.DeviceIndex [get], [private set]
```

The index of the detected glove within the SenseGlove_DeviceManager memory.

5.4.3.3 Type

```
SenseGloveCs.DeviceType SG.SG_DeviceManager.DeviceDetectedArgs.Type [get], [private set]
```

The DeviceType that has been found

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_DeviceManager.cs

5.5 SG.SG_DropZone.DropProps Class Reference

Properties that assist in object detection.

Public Attributes

- float **insideTime**
- bool **detected**

5.5.1 Detailed Description

Properties that assist in object detection.

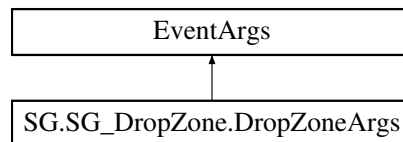
Placed inside a class to reduce the amount of List<> parameters.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_DropZone.cs

5.6 SG.SG_DropZone.DropZoneArgs Class Reference

Inheritance diagram for SG.SG_DropZone.DropZoneArgs:



Public Member Functions

- [DropZoneArgs](#) (SG_Grabable obj)
Create a new instance of the [DropZoneArgs](#).

Public Attributes

- [SG_Grabable](#) grabable
The object that was detected or removed.

5.6.1 Constructor & Destructor Documentation

5.6.1.1 DropZoneArgs()

```
SG.SG_DropZone.DropZoneArgs.DropZoneArgs (
    SG_Grabable obj )
```

Create a new instance of the [DropZoneArgs](#).

Parameters

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

5.6.2 Member Data Documentation

5.6.2.1 grabable

`SG_Grabable SG.SG_DropZone.DropZoneArgs.grabable`

The object that was detected or removed.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_DropZone.cs

5.7 SG.Util.FileIO Class Reference

Ensures that .txt files are properly handled by Unity.

Static Public Member Functions

- static bool `SaveTxtFile` (string dir, string fileName, string[] lines, bool append=false)
Attempt to save a string[] to a filename within a desired directory. Returns true if succesful.
- static bool `ReadTxtFile` (string path, out string[] lines)
Attempt to read all lines from a file and place them in the string[]. Returns true if succesful. If unable to open the file, the string[] will be empty.

5.7.1 Detailed Description

Ensures that .txt files are properly handled by Unity.

5.7.2 Member Function Documentation

5.7.2.1 ReadTxtFile()

```
static bool SG.Util.FileIO.ReadTxtFile (  
    string path,  
    out string[] lines ) [static]
```

Attempt to read all lines from a file and place them in the string[]. Returns true if succesful. If unable to open the file, the string[] will be empty.

Parameters

<i>path</i>	
<i>lines</i>	

Returns

5.7.2.2 SaveTxtFile()

```
static bool SG.Util.FileIO.SaveTxtFile (
    string dir,
    string fileName,
    string[] lines,
    bool append = false ) [static]
```

Attempt to save a string[] to a filename within a desired directory. Returns true if succesful.

Directory is added as a separate variable so we can more easily check for its existence.

Parameters

<i>dir</i>	
<i>fileName</i>	
<i>lines</i>	
<i>append</i>	

Returns

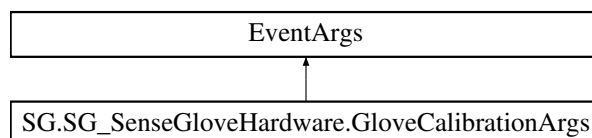
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_FileIO.cs

5.8 SG.SG_SenseGloveHardware.GloveCalibrationArgs Class Reference

CalibrationArguments, containing both old an new finger lengths and joint positions.

Inheritance diagram for SG.SG_SenseGloveHardware.GloveCalibrationArgs:



Public Member Functions

- [GloveCalibrationArgs](#) (SenseGloveCs.CalibrationArgs args)
Creates a new instance of the unity-friendly calibration args.
- [GloveCalibrationArgs](#) ([SG_SenseGloveData](#) oldD, [SG_SenseGloveData](#) newD)
Creates a new instance of the unity-friendly calibration args.

Public Attributes

- [SG_SenseGloveData](#) oldData
'Snapshot' of the old data, with old parameters
- [SG_SenseGloveData](#) newData
'Snapshot' of the new data, with updated parameters

5.8.1 Detailed Description

CalibrationArguments, containing both old an new finger lengths and joint positions.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 [GloveCalibrationArgs\(\)](#) [1/2]

```
SG.SG_SenseGloveHardware.GloveCalibrationArgs.GloveCalibrationArgs (
    SenseGloveCs.CalibrationArgs args )
```

Creates a new instance of the unity-friendly calibration args.

Parameters

<i>args</i>	
-------------	--

5.8.2.2 [GloveCalibrationArgs\(\)](#) [2/2]

```
SG.SG_SenseGloveHardware.GloveCalibrationArgs.GloveCalibrationArgs (
    SG\_SenseGloveData oldD,
    SG\_SenseGloveData newD )
```

Creates a new instance of the unity-friendly calibration args.

Parameters

<i>args</i>	
-------------	--

5.8.3 Member Data Documentation

5.8.3.1 newData

`SG_SenseGloveData` `SG.SG_SenseGloveHardware.GloveCalibrationArgs.newData`

'Snapshot' of the new data, with updated parameters

5.8.3.2 oldData

`SG_SenseGloveData` `SG.SG_SenseGloveHardware.GloveCalibrationArgs.oldData`

'Snapshot' of the old data, with old parameters

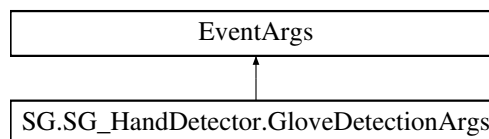
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_SenseGloveHardware.cs`

5.9 SG.SG_HandDetector.GloveDetectionArgs Class Reference

EventArgs fired when a glove is detected in or removed from a SenseGlove_Detector.

Inheritance diagram for SG.SG_HandDetector.GloveDetectionArgs:



Public Member Functions

- `GloveDetectionArgs` (`SG_SenseGloveHardware` model)
Create a new instance of the SenseGlove Detection Arguments

Public Attributes

- `SG_SenseGloveHardware` `handModel`
The Grabscrip that caused the event to fire.

5.9.1 Detailed Description

EventArgs fired when a glove is detected in or removed from a SenseGlove_Detector.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 GloveDetectionArgs()

```
SG.SG_HandDetector.GloveDetectionArgs.GloveDetectionArgs (
    SG_SenseGloveHardware model )
```

Create a new instance of the SenseGlove Detection Arguments

Parameters

<i>grab</i>	
-------------	--

5.9.3 Member Data Documentation

5.9.3.1 handModel

```
SG_SenseGloveHardware SG.SG_HandDetector.GloveDetectionArgs.handModel
```

The Grabscrip that caused the event to fire.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_Hand↵
Detector.cs

5.10 SG.Materials.MaterialProps Struct Reference

Contains the editable Material Properties of a single SenseGlove_Material

Public Member Functions

- [MaterialProps](#) (SG_Material material)

Convert a SenseGlove_Material into a [MaterialProps](#), which can be passed between scripts or stored later on.

Static Public Member Functions

- static [MaterialProps Default](#) ()
Retrieve a 'default' material.
- static [MaterialProps Parse](#) (List< string > dataBlock)
Parse a DataBlock into a [MaterialProps](#). Any missing variables will be set to their default value.

Public Attributes

- int [maxForce](#)
The maximum force that this material can put on the Sense Glove.
- float [maxForceDist](#)
The distance [m] where the maximum force has been reached. Setting it to 0 will instantly send maxForce on touch
- float [yieldDist](#)
The distance [m] at which the material breaks.
- int [hapticForce](#)
The magnitude [0..100%] of the buzz motor pulse
- int [hapticDur](#)
The duration of the Haptic Feedback, in milliseconds

Static Private Member Functions

- static bool [TryGetRawValue](#) (string line, out string raw)
Attempt to retrieve the (raw) value of this material property.
- static bool [TryGetFloat](#) (string line, out float res)
Attempt to convert a specific property to a floating point.

5.10.1 Detailed Description

Contains the editable Material Properties of a single SenseGlove_Material

5.10.2 Constructor & Destructor Documentation

5.10.2.1 MaterialProps()

```
SG.Materials.MaterialProps.MaterialProps (
    SG\_Material material )
```

Convert a SenseGlove_Material into a [MaterialProps](#), which can be passed between scripts or stored later on.

Parameters

<i>material</i>	
-----------------	--

5.10.3 Member Function Documentation

5.10.3.1 Default()

```
static MaterialProps SG.Materials.MaterialProps.Default ( ) [static]
```

Retrieve a 'default' material.

Returns

5.10.3.2 Parse()

```
static MaterialProps SG.Materials.MaterialProps.Parse (
    List< string > dataBlock ) [static]
```

Parse a DataBlock into a [MaterialProps](#). Any missing variables will be set to their default value.

Parameters

<i>dataBlock</i>	
------------------	--

Returns

5.10.3.3 TryGetFloat()

```
static bool SG.Materials.MaterialProps.TryGetFloat (
    string line,
    out float res ) [static], [private]
```

Attempt to convert a specific property to a floating point.

Parameters

<i>line</i>	
<i>res</i>	

Returns

5.10.3.4 TryGetRawValue()

```
static bool SG.Materials.MaterialProps.TryGetRawValue (
    string line,
    out string raw ) [static], [private]
```

Attempt to retrieve the (raw) value of this material property.

Parameters

<i>line</i>	
<i>raw</i>	

Returns

5.10.4 Member Data Documentation

5.10.4.1 hapticDur

```
int SG.Materials.MaterialProps.hapticDur
```

The duration of the Haptic Feedback, in milliseconds

5.10.4.2 hapticForce

```
int SG.Materials.MaterialProps.hapticForce
```

The magnitude [0..100%] of the buzz motor pulse

5.10.4.3 maxForce

```
int SG.Materials.MaterialProps.maxForce
```

The maximum force that this material can put on the Sense Glove.

5.10.4.4 maxForceDist

```
float SG.Materials.MaterialProps.maxForceDist
```

The distance [m] where the maximum force has been reached. Setting it to 0 will instantly send maxForce on touch

5.10.4.5 yieldDist

```
float SG.Materials.MaterialProps.yieldDist
```

The distance [m] at which the material breaks.

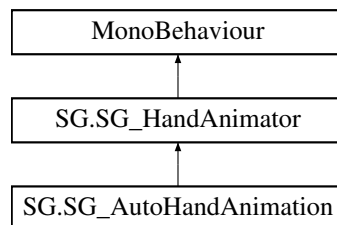
The documentation for this struct was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_Material.cs

5.11 SG.SG_AutoHandAnimation Class Reference

A HandAnimator that grabs its animation info from a [SG_HandModelInfo](#) script.

Inheritance diagram for SG.SG_AutoHandAnimation:



Public Attributes

- [SG_HandModelInfo](#) **handModelInfo**
teh HandModelInfo that this scripts animates.

Protected Member Functions

- override void [CollectFingerJoints](#) ()
Assign the joints of this script so that the [SG_HandAnimator](#) script takes over animation.
- override void [CheckForScripts](#) ()
Check for relevant linked scripts for this HandAnimator, specifically to the [SG_HandModelInfo](#).
- override void [Start](#) ()
If we have HandModelInfo, we can already collect joints

Additional Inherited Members

5.11.1 Detailed Description

A HandAnimator that grabs its animation info from a [SG_HandModelInfo](#) script.

5.11.2 Member Function Documentation

5.11.2.1 CheckForScripts()

```
override void SG.SG_AutoHandAnimation.CheckForScripts ( ) [protected], [virtual]
```

Check for relevant linked scripts for this HandAnimator, specifically to the [SG_HandModelInfo](#).

Reimplemented from [SG.SG_HandAnimator](#).

5.11.2.2 CollectFingerJoints()

```
override void SG.SG_AutoHandAnimation.CollectFingerJoints ( ) [protected], [virtual]
```

Assign the joints of this script so that the [SG_HandAnimator](#) script takes over animation.

Implements [SG.SG_HandAnimator](#).

5.11.2.3 Start()

```
override void SG.SG_AutoHandAnimation.Start ( ) [protected], [virtual]
```

If we have HandModelInfo, we can already collect joints

Reimplemented from [SG.SG_HandAnimator](#).

5.11.3 Member Data Documentation

5.11.3.1 handModelInfo

`SG_HandModelInfo` `SG.SG_AutoHandAnimation.handModelInfo`

teh HandModelInfo that this scripts animates.

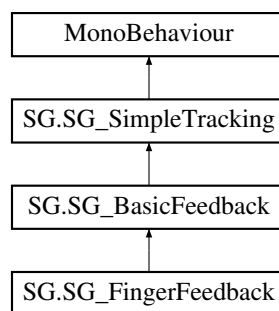
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_AutoHandAnimation.cs`

5.12 SG.SG_BasicFeedback Class Reference

Attach to a collider and it will send haptic feedback to a SenseGlove on impact. Optionally tracks a GameObject. Extended by `SG_Finger` to apply more forces.

Inheritance diagram for `SG.SG_BasicFeedback`:



Public Member Functions

- virtual void `SetupSelf` ()
Setup the `SG_BasicFeedback` script components
- void `SendImpactFeedback` (float impactVelocity)
Send an impact vibration to this script's connected glove, based on a speed in m/s.

Public Attributes

- `SG_SenseGloveHardware` `linkedGlove`
Sense Glove that will receive the feedback effect
- `SG_HandSection` `handLocation` = `SG_HandSection.Unknown`
The part of the hand that this script belongs to.
- bool `impactFeedbackEnabled` = true
If true, this script will send vibrotactile feedback on impact.
- float `impactCooldown` = 0.5f
The minimum time, in seconds, between impact vibration.
- float `minImpactSpeed` = 0.01f
The minimum speed, in m/s, that this object must make before an impact is played.
- float `maxImpactSpeed` = 0.1f
The speed, in m/s, where the maximum vibration level is sent.
- AnimationCurve `impactProfile` = `AnimationCurve.Linear(0, 0, 1, 1)`
A curve that determines how the impact vibration varies between the minimum and maximum impact speed. Set to constant (1) to have the same vibration no matter the speed.

Static Public Attributes

- static int `maxVelocityPoints` = 10
The maximum frames for which to keep track of velocity.

Protected Member Functions

- override void `UpdatePosition` ()
Update this collider's position, and register its velocity.
- override void `Awake` ()
- override void `FixedUpdate` ()
- virtual void `OnTriggerEnter` (Collider other)

Protected Attributes

- List< Vector3 > `velocities` = new List<Vector3>()
The xyz velocities during the last few frames, used to determine the average impact velocity.
- Vector3 `lastPosition` = Vector3.zero
This object's position during the last frame, used to determine velocity.
- float `cooldownTimer` = 0
Keeps track of time since last vibration

Static Protected Attributes

- static int `minBuzzLevel` = 50
The minimum vibration level at which an impact can be felt.
- static int `maxBuzzLevel` = 80
The maximum vibration level to represent an impact.
- static int `vibrationTime` = 100
The time to vibrate the buzz motors for.

Properties

- override bool `DebugEnabled` [get, set]
Used to show or hide this object's collider.
- bool `CanImpact` [get]
Returns true if this script can send an impact vibration
- Vector3 `SmoothedVelocity` [get]
Returns the average velocity over the last few frames

Additional Inherited Members

5.12.1 Detailed Description

Attach to a collider and it will send haptic feedback to a SenseGlove on impact. Optionally tracks a GameObject. Extended by SG_Finger to apply more forces.

5.12.2 Member Function Documentation

5.12.2.1 SendImpactFeedback()

```
void SG.SG_BasicFeedback.SendImpactFeedback (
    float impactVelocity )
```

Send an impact vibration to this script's connected glove, based on a speed in m/s.

Parameters

<i>impactVelocity</i>	
-----------------------	--

5.12.2.2 SetupSelf()

```
virtual void SG.SG_BasicFeedback.SetupSelf ( ) [virtual]
```

Setup the [SG_BasicFeedback](#) script components

Reimplemented in [SG.SG_FingerFeedback](#).

5.12.2.3 UpdatePosition()

```
override void SG.SG_BasicFeedback.UpdatePosition ( ) [protected], [virtual]
```

Update this collider's position, and register its velocity.

Reimplemented from [SG.SG_SimpleTracking](#).

5.12.3 Member Data Documentation

5.12.3.1 cooldownTimer

```
float SG.SG_BasicFeedback.cooldownTimer = 0 [protected]
```

Keeps track of time since last vibration

5.12.3.2 handLocation

```
SG_HandSection SG.SG_BasicFeedback.handLocation = SG_HandSection.Unknown
```

The part of the hand that this script belongs to.

5.12.3.3 impactCooldown

```
float SG.SG_BasicFeedback.impactCooldown = 0.5f
```

The minimum time, in seconds, between impact vibration.

5.12.3.4 impactFeedbackEnabled

```
bool SG.SG_BasicFeedback.impactFeedbackEnabled = true
```

If true, this script will send vibrotactile feedback on impact.

5.12.3.5 impactProfile

```
AnimationCurve SG.SG_BasicFeedback.impactProfile = AnimationCurve.Linear(0, 0, 1, 1)
```

A curve that determines how the impact vibration varies between the minimum and maximum impact speed. Set to constant (1) to have the same vibration no matter the speed.

5.12.3.6 lastPosition

```
Vector3 SG.SG_BasicFeedback.lastPosition = Vector3.zero [protected]
```

This object's position during the last frame, used to determine velocity.

5.12.3.7 linkedGlove

```
SG_SenseGloveHardware SG.SG_BasicFeedback.linkedGlove
```

Sense Glove that will receive the feedback effect

5.12.3.8 maxBuzzLevel

```
int SG.SG_BasicFeedback.maxBuzzLevel = 80 [static], [protected]
```

The maximum vibration level to represent an impact.

5.12.3.9 maxImpactSpeed

```
float SG.SG_BasicFeedback.maxImpactSpeed = 0.1f
```

The speed, in m/s, where the maximum vibration level is sent.

5.12.3.10 maxVelocityPoints

```
int SG.SG_BasicFeedback.maxVelocityPoints = 10 [static]
```

The maximum frames for which to keep track of velocity.

5.12.3.11 minBuzzLevel

```
int SG.SG_BasicFeedback.minBuzzLevel = 50 [static], [protected]
```

The minimum vibration level at which an impact can be felt.

5.12.3.12 minImpactSpeed

```
float SG.SG_BasicFeedback.minImpactSpeed = 0.01f
```

The minimum speed, in m/s, that this object must make before an impact is played.

5.12.3.13 velocities

```
List<Vector3> SG.SG_BasicFeedback.velocities = new List<Vector3>() [protected]
```

The xyz velocities during the last few frames, used to determine the average impact velocity.

5.12.3.14 vibrationTime

```
int SG.SG_BasicFeedback.vibrationTime = 100 [static], [protected]
```

The time to vibrate the buzz motors for.

5.12.4 Property Documentation

5.12.4.1 CanImpact

```
bool SG.SG_BasicFeedback.CanImpact [get]
```

Returns true if this script can send an impact vibration

5.12.4.2 DebugEnabled

```
override bool SG.SG_BasicFeedback.DebugEnabled [get], [set]
```

Used to show or hide this object's collider.

5.12.4.3 SmoothedVelocity

```
Vector3 SG.SG_BasicFeedback.SmoothedVelocity [get]
```

Returns the average velocity over the last few frames

Returns

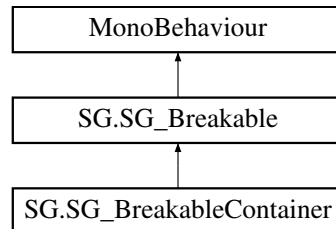
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_BasicFeedback.cs

5.13 SG.SG_Breakable Class Reference

A Gameobject that despawns an objects once its material breaks, and optionally replaces it with a 'broken' version.

Inheritance diagram for SG.SG_Breakable:



Public Types

- enum [UnbreakType](#) { [UnbreakType.None](#) = 0, [UnbreakType.Unbreak](#), [UnbreakType.Reset](#) }
- How the object will respond after it breaks.*

Public Member Functions

- bool [IsBroken](#) ()
Returns true if the wholeObject is currently in its broken state.
- virtual void [Break](#) ()
Break the object: Hide the whole object, optionally show the broken one and play the particle effect(s)
- virtual void [UnBreak](#) ()
Reset the object to before its unbroken state, at the same location of the current broken object.
- virtual void [ResetObject](#) ()
Reset this objects position and materials.
- virtual void [CheckUnbreak](#) ()
Check if this objects needs to be reset, depending on the state and unbreakMethod
- delegate void [ObjectBrokenEventHandler](#) (object source, System.EventArgs args)
Event delegate for the ObjectBreaks EventHandler
- delegate void [ObjectUnBrokenEventHandler](#) (object source, System.EventArgs args)
Event delegate for the ObjectUnBreaks EventHandler

Public Attributes

- [SG_Interactive](#) [wholeObject](#)
The Interactive with a material which can break. Represents the 'whole' object
- [SG_Interactive](#) [brokenObject](#)
The interactable in its broken state.
- ParticleSystem [breakParticles](#)
Optional Particle System that plays when the object breaks.
- AudioSource [breakSound](#)
Optional sound to play when the material breaks.
- [UnbreakType](#) [unbreakMethod](#) = [UnbreakType.None](#)
Determines if the Breakable resets back to the whole object after the desired timeframe.
- float [checkTime](#) = 1.0f
The time after which the breakable checks if it needs to reset.

Protected Member Functions

- virtual void **Start** ()
- virtual void **Update** ()
- void [OnObjectBreaks](#) ()
Calls the ObjectBreaks event handler.
- void [OnObjectUnBreaks](#) ()
Calls the ObjectUnBreaks event handler.

Events

- [ObjectBrokenEventHandler ObjectBreaks](#)
Fires when this objects [Break\(\)](#) function has been called.
- [ObjectUnBrokenEventHandler ObjectUnBreaks](#)
Fires when this objects [UnBreak\(\)](#) function has been called.

Private Member Functions

- void [WholeMaterial_MaterialBreaks](#) (object source, System.EventArgs args)
Fired when the associated material breaks.

Private Attributes

- float [resetTime](#) = 0
Timer to keep track of when this object resets.
- [SG_Material wholeMaterial](#)
SenseGlove_Material of the whole object. Used to catch the OnMaterialBreak event.
- [SG_Material brokenMaterial](#)
SenseGlove_Material of the broken object.
- [SG_MeshDeform wholeDeform](#)
(Optional) deform script of the whole object, to reset if the material breaks
- [SG_MeshDeform brokenDeform](#)
(Optional) deform script of the broken object, to reset if the material unbreaks

5.13.1 Detailed Description

A Gameobject that despawns an objects once its material breaks, and optionally replaces it with a 'broken' version.

5.13.2 Member Enumeration Documentation

5.13.2.1 UnbreakType

```
enum SG.SG\_Breakable.UnbreakType [strong]
```

How the object will respond after it breaks.

Enumerator

None	The object stays broken, and does nothing. Default value.
Unbreak	The object unbreaks after the timer elapses.
Reset	The object fully resets after the timer elapsed.

5.13.3 Member Function Documentation

5.13.3.1 Break()

```
virtual void SG.SG_Breakable.Break ( ) [virtual]
```

Break the object: Hide the whole object, optionally show the broken one and play the particle effect(s)

Reimplemented in [SG.SG_BreakableContainer](#).

5.13.3.2 CheckUnbreak()

```
virtual void SG.SG_Breakable.CheckUnbreak ( ) [virtual]
```

Check if this objects needs to be reset, depending on the state and unbreakMethod

5.13.3.3 IsBroken()

```
bool SG.SG_Breakable.IsBroken ( )
```

Returns true if the wholeObject is currently in its broken state.

Returns

5.13.3.4 ObjectBrokenEventHandler()

```
delegate void SG.SG_Breakable.ObjectBrokenEventHandler (
    object source,
    System.EventArgs args )
```

Event delegate for the ObjectBreaks EventHandler

Parameters

<i>source</i>	
<i>args</i>	

5.13.3.5 ObjectUnBrokenEventHandler()

```
delegate void SG.SG_Breakable.ObjectUnBrokenEventHandler (
    object source,
    System.EventArgs args )
```

Event delegate for the ObjectUnBreaks EventHandler

Parameters

<i>source</i>	
<i>args</i>	

5.13.3.6 OnObjectBreaks()

```
void SG.SG_Breakable.OnObjectBreaks ( ) [protected]
```

Calls the ObjectBreaks event handler.

5.13.3.7 OnObjectUnBreaks()

```
void SG.SG_Breakable.OnObjectUnBreaks ( ) [protected]
```

Calls the ObjectUnBreaks event handler.

5.13.3.8 ResetObject()

```
virtual void SG.SG_Breakable.ResetObject ( ) [virtual]
```

Reset this objects position and materials.

Reimplemented in [SG.SG_BreakableContainer](#).

5.13.3.9 UnBreak()

```
virtual void SG.SG_Breakable.UnBreak ( ) [virtual]
```

Reset the object to before its unbroken state, at the same location of the current broken object.

Reimplemented in [SG.SG_BreakableContainer](#).

5.13.3.10 WholeMaterial_MaterialBreaks()

```
void SG.SG_Breakable.WholeMaterial_MaterialBreaks (
    object source,
    System.EventArgs args ) [private]
```

Fired when the associated material breaks.

Parameters

<i>source</i>	
<i>args</i>	

5.13.4 Member Data Documentation

5.13.4.1 breakParticles

```
ParticleSystem SG.SG_Breakable.breakParticles
```

Optional Particle System that plays when the object breaks.

5.13.4.2 breakSound

```
AudioSource SG.SG_Breakable.breakSound
```

Optional sound to play when the material breaks.

5.13.4.3 brokenDeform

```
SG\_MeshDeform SG.SG_Breakable.brokenDeform [private]
```

(Optional) deform script of the broken object, to reset if the material unbreaks

5.13.4.4 brokenMaterial

`SG_Material` SG.SG_Breakable.brokenMaterial [private]

SenseGlove_Material of the broken object.

5.13.4.5 brokenObject

`SG_Interactive` SG.SG_Breakable.brokenObject

The interactable in its broken state.

5.13.4.6 checkTime

`float` SG.SG_Breakable.checkTime = 1.0f

The time after which the breakable checks if it needs to reset.

5.13.4.7 resetTime

`float` SG.SG_Breakable.resetTime = 0 [private]

Timer to keep track of when this object resets.

5.13.4.8 unbreakMethod

`UnbreakType` SG.SG_Breakable.unbreakMethod = `UnbreakType.None`

Determines if the Breakable resets back to the whole object after the desired timeframe.

5.13.4.9 wholeDeform

`SG_MeshDeform` SG.SG_Breakable.wholeDeform [private]

(Optional) deform script of the whole object, to reset if the material breaks

5.13.4.10 wholeMaterial

`SG_Material` `SG.SG_Breakable.wholeMaterial` [private]

SenseGlove_Material of the whole object. Used to catch the OnMaterialBreak event.

5.13.4.11 wholeObject

`SG_Interactive` `SG.SG_Breakable.wholeObject`

The Interactive with a material which can break. Represents the 'whole' object

5.13.5 Event Documentation

5.13.5.1 ObjectBreaks

`ObjectBrokenEventHandler` `SG.SG_Breakable.ObjectBreaks`

Fires when this objects `Break()` function has been called.

5.13.5.2 ObjectUnBreaks

`ObjectUnBrokenEventHandler` `SG.SG_Breakable.ObjectUnBreaks`

Fires when this objects `UnBreak()` function has been called.

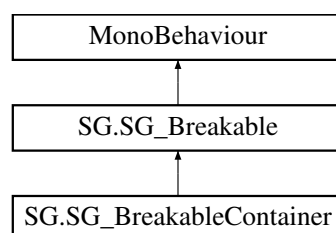
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Breakable.cs`↔

5.14 SG.SG_BreakableContainer Class Reference

A SenseGlove_Breakable that contains objects and optionally spawns shards of itself upon breaking.

Inheritance diagram for SG.SG_BreakableContainer:



Public Member Functions

- override void [Break](#) ()
Called when the breakable material of the wholeObject is broken
- override void [UnBreak](#) ()
Called when the breakable material is reset.
- override void [ResetObject](#) ()
This always resets contents, while Unbreak (called within ResetObjects) does not reset the contents.

Public Attributes

- GameObject [shardContainer](#)
Contains the GameObjects that represent the shards of the broken object.
- GameObject [contentsContainer](#)
Contains [SG_Interactable](#) objects that will be released upon the container breaking.
- bool [unbreakWithContents](#) = false
Determines if the contents are placed back into the container when the object is unbroken.

Protected Member Functions

- virtual void [Awake](#) ()
- void [SpawnShards](#) ()
Spawns the Shards when the container breaks.
- void [ResetShards](#) ()
Put all the shards back to their original (local) transform.
- void [SpawnContents](#) ()
Spawns Contents when the container breaks
- void [ResetContents](#) ()
Resets the contents back to their original (local) transforms.

Static Protected Member Functions

- static void [SetRB](#) (GameObject obj, bool gravity, bool kinematic)
Set the Rigidbody options of a particular gameObject, if the object has any.
- static void [SetColliders](#) (GameObject obj, bool trigger)
Set the Collider options of a particular GameObject, if the object has any.

Protected Attributes

- GameObject[] [brokenShards](#) = new GameObject[0]
All GameObjects within the shardsContainer. Will be spawned at the time of breaking.
- [SG_Interactable](#)[] [contents](#) = new [SG_Interactable](#)[0]
All [SenseGlove_Interactables](#) within the container. Will be set to interactable at the time of breaking.
- Quaternion[] [contentRotations](#)
The localRotations of the shards, applied on a reset.
- Vector3[] [contentPositions](#)
The localPositions of the shards, applied on reset.
- Quaternion[] [shardRotations](#)
The localRotations of the shards, applied on a reset.
- Vector3[] [shardPositions](#)
The localPositions of the shards, applied on reset.

Additional Inherited Members

5.14.1 Detailed Description

A SenseGlove_Breakable that contains objects and optionally spawns shards of itself upon breaking.

5.14.2 Member Function Documentation

5.14.2.1 Break()

```
override void SG.SG_BreakableContainer.Break ( ) [virtual]
```

Called when the breakable material of the wholeObject is broken

Reimplemented from [SG.SG_Breakable](#).

5.14.2.2 ResetContents()

```
void SG.SG_BreakableContainer.ResetContents ( ) [protected]
```

Resets the contents back to their original (local) transforms.

5.14.2.3 ResetObject()

```
override void SG.SG_BreakableContainer.ResetObject ( ) [virtual]
```

This always resets contents, while Unbreak (called within ResetObjects) does not reset the contents.

Reimplemented from [SG.SG_Breakable](#).

5.14.2.4 ResetShards()

```
void SG.SG_BreakableContainer.ResetShards ( ) [protected]
```

Put all the shards back to their original (local) transform.

5.14.2.5 SetColliders()

```
static void SG.SG_BreakableContainer.SetColliders (
    GameObject obj,
    bool trigger ) [static], [protected]
```

Set the Collider options of a particular GameObject, if the object has any.

Parameters

<i>obj</i>	
<i>trigger</i>	

5.14.2.6 SetRB()

```
static void SG.SG_BreakableContainer.SetRB (  
    GameObject obj,  
    bool gravity,  
    bool kinematic ) [static], [protected]
```

Set the Rigidbody options of a particular gameObject, if the object has any.

Parameters

<i>Obj</i>	
<i>gravity</i>	
<i>kinematic</i>	

5.14.2.7 SpawnContents()

```
void SG.SG_BreakableContainer.SpawnContents ( ) [protected]
```

Spawns Contents when the container breaks

The contents have been visible all along, they just havent been active.

5.14.2.8 SpawnShards()

```
void SG.SG_BreakableContainer.SpawnShards ( ) [protected]
```

Spawns the Shards when the container breaks.

5.14.2.9 UnBreak()

```
override void SG.SG_BreakableContainer.UnBreak ( ) [virtual]
```

Called when the breakable material is reset.

Reimplemented from [SG.SG_Breakable](#).

5.14.3 Member Data Documentation

5.14.3.1 brokenShards

```
GameObject [] SG.SG_BreakableContainer.brokenShards = new GameObject[0] [protected]
```

All GameObjects within the shardsContainer. Will be spawned at the time of breaking.

5.14.3.2 contentPositions

```
Vector3 [] SG.SG_BreakableContainer.contentPositions [protected]
```

The localPositions of the shards, applied on reset.

5.14.3.3 contentRotations

```
Quaternion [] SG.SG_BreakableContainer.contentRotations [protected]
```

The localRotations of the shards, applied on a reset.

5.14.3.4 contents

```
SG_Interactive [] SG.SG_BreakableContainer.contents = new SG_Interactive[0] [protected]
```

All SenseGlove_Interactables within the container. Will be set to interactable at the time of breaking.

5.14.3.5 contentsContainer

```
GameObject SG.SG_BreakableContainer.contentsContainer
```

Contains [SG_Interactive](#) objects that will be released upon the container breaking.

5.14.3.6 shardContainer

`GameObject SG.SG_BreakableContainer.shardContainer`

Contains the GameObjects that represent the shards of the broken object.

5.14.3.7 shardPositions

`Vector3 [] SG.SG_BreakableContainer.shardPositions [protected]`

The localPositions of the shards, applied on reset.

5.14.3.8 shardRotations

`Quaternion [] SG.SG_BreakableContainer.shardRotations [protected]`

The localRotations of the shards, applied on a reset.

5.14.3.9 unbreakWithContents

`bool SG.SG_BreakableContainer.unbreakWithContents = false`

Determines if the contents are placed back into the container when the object is unbroken.

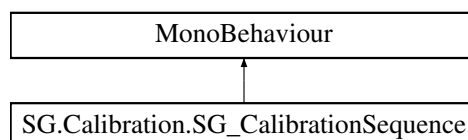
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Breakable↔ Container.cs

5.15 SG.Calibration.SG_CalibrationSequence Class Reference

Manobehaviour meant to run the user though two general calibration steps, and then allows them to refine their calibration

Inheritance diagram for SG.Calibration.SG_CalibrationSequence:



Public Types

- enum [CalStage](#) { **AwaitConnection**, **GlobalCalibration**, **LastRefinement**, **Saved** }
Stage of the [Calibration](#) sequence
- enum [CalPose](#) {
FingersExt = 0, **FingersFlexed**, **ThumbUp**, **ThumbFlex**,
AbdOut, **HandOpen**, **HandClosed**, **NoThumbAbd**,
All }
[Calibration](#) poses, used to access `SG_CalPoses` in an array.

Public Member Functions

- `Vector3[] GetCalibrationValues ()`
Get [Calibration](#) Values from the hardware, as the interpolation solver would.
- `void CalibratePose (int poseIndex)`
Calibrate the interpolator with a specified pose
- `void SaveCalibration ()`
Store calibration on disk so it may be used by other applications.
- `void ResetCalibration ()`
Reset Sense Glove [Calibration](#) back to its default values.
- `void StartGlobal ()`
Start Global [Calibration](#)
- `void EndGlobal ()`
End global calibration
- `void CalibrateCurrentStep ()`
Calibrate the current global calibration step
- `void SkipStep ()`
Skip the current calibration step, without calibrating.
- `void RegularExit ()`
Allow exiting of the application.
- `void SaveAndExit ()`
Save [Calibration](#), then exit the application

Public Attributes

- `SG_WireFrame wireFrame`
Wireframe model used to show the glove and access hardware.
- `CalStage stage = CalStage.AwaitConnection`
Stage of calibration we are currently in
- `int subStage = 0`
Sub-stage of calibration. Used for general calibration.
- `Text instrText`
UI element for general instructions.
- `GameObject refinementMenu`
Menu containing refinement steps for calibration, which is disabled at start.
- `Text generalButtonTxt`
Text of a "Calibrate Current Step" button, that can be altered
- `Button skipButton`
Button to skip the current calibration step, only available during general calibration
- `Button saveButton`

- Button to save calibration. Is disabled until changes are detected.
- GameObject **endPopup**
Popup to show if there are unsavedChanges
- GameObject **openHandEx**
Groups of GameObjects that show example poses of the hand
- KeyCode **nextStepKey** = KeyCode.Space
HotKey for calibrating the current global step.
- GameObject[] **hiddenBeforeStart** = new GameObject[0]
GameObject hidden until the SenseGlove is connected
- GameObject **leftAnimation**
HandModels for either a left or right hand, to move along the wireframe
- Transform[] **rightHands** = new Transform[0]
All GameObjects that represent a right hand, to be mirrored when a left hand is detected.

Protected Member Functions

- void **GoToMainStage** (int newStage)
Go to a substage within the Global [Calibration](#)

Private Member Functions

- void **LoadProfiles** (InterpolationSet_IMU interpolator)
Generates SG_CalibrationPoses for this interpolator.
- void **Start** ()
- void **Update** ()
- void **OnApplicationQuit** ()

Private Attributes

- GameObject **closedHandEx**
- GameObject **rightAnimation**
- bool **changes** = false
True if changes are detected, used to check when exiting.
- InterpolationSet_IMU **interpolator** = null
Interpolator clone of the Glove, which is updated and applied when calibrating.
- string **baseTxt** = ""
Base instruction text to add on top of other instructions
- [CalibrationPose](#)[] **poses** = new [CalibrationPose](#)[0]
[Calibration](#) poses used to calibrate the interpolator

5.15.1 Detailed Description

Manobehaviour meant to run the user through two general calibration steps, and then allows them to refine their calibration

5.15.2 Member Enumeration Documentation

5.15.2.1 CalPose

enum `SG.Calibration.SG_CalibrationSequence.CalPose` [strong]

[Calibration](#) poses, used to access SG_CalPoses in an array.

5.15.2.2 CalStage

enum `SG.Calibration.SG_CalibrationSequence.CalStage` [strong]

Stage of the [Calibration](#) sequence

5.15.3 Member Function Documentation

5.15.3.1 CalibrateCurrentStep()

void `SG.Calibration.SG_CalibrationSequence.CalibrateCurrentStep ()`

Calibrate the current global calibration step

5.15.3.2 CalibratePose()

void `SG.Calibration.SG_CalibrationSequence.CalibratePose (`
 int *poseIndex* `)`

Calibrate the interpolator with a specified pose

Parameters

<i>poseIndex</i>	
------------------	--

5.15.3.3 EndGlobal()

void `SG.Calibration.SG_CalibrationSequence.EndGlobal ()`

End global calibration

5.15.3.4 GetCalibrationValues()

```
Vector3 [ ] SG.Calibration.SG_CalibrationSequence.GetCalibrationValues ( )
```

Get [Calibration](#) Values from the hardware, as the interpolation solver would.

Returns

5.15.3.5 GoToMainStage()

```
void SG.Calibration.SG_CalibrationSequence.GoToMainStage (
    int newStage ) [protected]
```

Go to a substage within the Global [Calibration](#)

Parameters

<i>newStage</i>	
-----------------	--

5.15.3.6 LoadProfiles()

```
void SG.Calibration.SG_CalibrationSequence.LoadProfiles (
    InterpolationSet_IMU intepolator ) [private]
```

Generates SG_CalibrationPoses for this interpolator.

Parameters

<i>intepolator</i>	
--------------------	--

5.15.3.7 RegularExit()

```
void SG.Calibration.SG_CalibrationSequence.RegularExit ( )
```

Allow exiting of the application.

5.15.3.8 ResetCalibration()

```
void SG.Calibration.SG_CalibrationSequence.ResetCalibration ( )
```

Reset Sense Glove [Calibration](#) back to its default values.

5.15.3.9 SaveAndExit()

```
void SG.Calibration.SG_CalibrationSequence.SaveAndExit ( )
```

Save [Calibration](#), then exit the application

5.15.3.10 SaveCalibration()

```
void SG.Calibration.SG_CalibrationSequence.SaveCalibration ( )
```

Store calibration on disk so it may be used by other applications.

5.15.3.11 SkipStep()

```
void SG.Calibration.SG_CalibrationSequence.SkipStep ( )
```

Skip the current calibration step, without calibrating.

5.15.3.12 StartGlobal()

```
void SG.Calibration.SG_CalibrationSequence.StartGlobal ( )
```

Start Global [Calibration](#)

5.15.4 Member Data Documentation

5.15.4.1 baseTxt

```
string SG.Calibration.SG_CalibrationSequence.baseTxt = "" [private]
```

Base instruction text to add on top of other instructions

5.15.4.2 changes

```
bool SG.Calibration.SG_CalibrationSequence.changes = false [private]
```

True if changes are detected, used to check when exiting.

5.15.4.3 endPopup

```
GameObject SG.Calibration.SG_CalibrationSequence.endPopup
```

Popup to show if there are unsavedChanges

5.15.4.4 generalButtonText

```
Text SG.Calibration.SG_CalibrationSequence.generalButtonText
```

Text of a "Calibrate Current Step" button, that can be altered

5.15.4.5 hiddenBeforeStart

```
GameObject [] SG.Calibration.SG_CalibrationSequence.hiddenBeforeStart = new GameObject[0]
```

GameObject hidden until the SenseGlove is connected

5.15.4.6 instrText

```
Text SG.Calibration.SG_CalibrationSequence.instrText
```

UI element for general instructions.

5.15.4.7 interpolator

```
InterpolationSet_IMU SG.Calibration.SG_CalibrationSequence.interpolator = null [private]
```

Interpolator clone of the Glove, which is updated and applied when calibrating.

5.15.4.8 leftAnimation

```
GameObject SG.Calibration.SG_CalibrationSequence.leftAnimation
```

HandModels for either a left or right hand, to move along the wireframe

5.15.4.9 nextStepKey

```
KeyCode SG.Calibration.SG_CalibrationSequence.nextStepKey = KeyCode.Space
```

HotKey for calibrating the current global step.

5.15.4.10 openHandEx

```
GameObject SG.Calibration.SG_CalibrationSequence.openHandEx
```

Groups of GameObjects that show example poses of the hand

5.15.4.11 poses

```
CalibrationPose [ ] SG.Calibration.SG_CalibrationSequence.poses = new CalibrationPose[0] [private]
```

Calibration poses used to calibrate the interpolator

5.15.4.12 refinementMenu

```
GameObject SG.Calibration.SG_CalibrationSequence.refinementMenu
```

Menu containing refinement steps for calibration, which is disabled at start.

5.15.4.13 rightHands

```
Transform [ ] SG.Calibration.SG_CalibrationSequence.rightHands = new Transform[0]
```

All GameObjects that represent a right hand, to be mirrored when a left hand is detected.

5.15.4.14 saveButton

Button SG.Calibration.SG_CalibrationSequence.saveButton

Button to save calibration. Is disabled until changes are detected.

5.15.4.15 skipButton

Button SG.Calibration.SG_CalibrationSequence.skipButton

Button to skip the current calibration step, only available during general calibration

5.15.4.16 stage

CalStage SG.Calibration.SG_CalibrationSequence.stage = CalStage.AwaitConnection

Stage of calibration we are currently in

5.15.4.17 subStage

int SG.Calibration.SG_CalibrationSequence.subStage = 0

Sub-stage of calibration. Used for general calibration.

5.15.4.18 wireFrame

SG_WireFrame SG.Calibration.SG_CalibrationSequence.wireFrame

Wireframe model used to show the glove and access hardware.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Calibration/Resources/SG_CalibrationSequence.cs

5.16 SG.Calibration.SG_CalibrationStorage Class Reference

Class responsible for storing and retrieving Sense Glove calibration on disk.

Static Public Member Functions

- static void [StoreInterpolation](#) (SenseGloveCs.Kinematics.InterpolationSet_IMU interpolator, SenseGloveCs.DeviceType type, [GloveSide](#) side)
Stores a deserialized value of an interpolator onto a disk.
- static void [StoreInterpolation](#) (string interpolator, SenseGloveCs.DeviceType type, [GloveSide](#) side)
Stores a serialized value of an interpolator onto a disk.
- static bool [LoadInterpolation](#) (SenseGloveCs.DeviceType type, [GloveSide](#) side, out string output)
Retrieves an interpolator from the disk. Returns true if one is actually available.

Static Private Member Functions

- static string [GetFilename](#) (SenseGloveCs.DeviceType type, [GloveSide](#) side)
Generate a new filename for this calibration profile.

Static Private Attributes

- static readonly string [calibrDir](#)
Default location for storing calibration data.

5.16.1 Detailed Description

Class responsible for storing and retrieving Sense Glove calibration on disk.

5.16.2 Member Function Documentation

5.16.2.1 GetFilename()

```
static string SG.Calibration.SG_CalibrationStorage.GetFilename (
    SenseGloveCs.DeviceType type,
    GloveSide side ) [static], [private]
```

Generate a new filename for this calibration profile.

Parameters

<i>side</i>	
-------------	--

Returns

5.16.2.2 LoadInterpolation()

```
static bool SG.Calibration.SG_CalibrationStorage.LoadInterpolation (
    SenseGloveCs.DeviceType type,
    GloveSide side,
    out string output ) [static]
```

Retrieves an interpolator from the disk. Returns true if one is actually available.

Parameters

<i>side</i>	
-------------	--

Returns

5.16.2.3 StoreInterpolation() [1/2]

```
static void SG.Calibration.SG_CalibrationStorage.StoreInterpolation (
    SenseGloveCs.Kinematics.InterpolationSet_IMU interpolator,
    SenseGloveCs.DeviceType type,
    GloveSide side ) [static]
```

Stores a deserialized value of an interpolator onto a disk.

Parameters

<i>interpolator</i>	
<i>side</i>	

5.16.2.4 StoreInterpolation() [2/2]

```
static void SG.Calibration.SG_CalibrationStorage.StoreInterpolation (
    string interpolator,
    SenseGloveCs.DeviceType type,
    GloveSide side ) [static]
```

Stores a serialized value of an interpolator onto a disk.

Parameters

<i>interpolator</i>	
<i>side</i>	

5.16.3 Member Data Documentation

5.16.3.1 calibrDir

```
readonly string SG.Calibration.SG_CalibrationStorage.calibrDir [static], [private]
```

Initial value:

```
= System.Environment.GetFolderPath(System.Environment.SpecialFolder.MyDocuments)
+ "/SenseGlove/Calibration/"
```

Default location for storing calibration data.

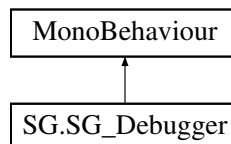
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Calibration/Resources/SG_CalibrationStorage.cs

5.17 SG.SG_Debugger Class Reference

Utility Script that allows access to the internal debugger of the SenseGloveCs Library, and controls debug messages from the SenseGlove SDK specifically.

Inheritance diagram for SG.SG_Debugger:



Static Public Member Functions

- static void [Log](#) (string message)
Write a message to the [SG_Debugger](#).
- static void [LogWarning](#) (string message)
Write a message to the [SG_Debugger](#) to appear as a warning.
- static void [LogError](#) (string message)
Write a message to the [SG_Debugger](#) to appear as an error.

Public Attributes

- DebugLevel [DLL_debugLevel](#) = SenseGloveCs.Diagnostics.Debugger.defaultDebugLvl
The level of debug messages that one will receive from the DLL.
- bool [unityEnabled](#) = true
Enables or disables debug messages from the Unity SDK scripts.

Private Member Functions

- void **Awake** ()
- void **LateUpdate** ()
- void **OnDestroy** ()
- void **OnApplicationQuit** ()
- void [Instance_DebugMessageRecieved](#) (object source, DebugArgs args)

Fires when our debugger reports that a new message has been recieved.

Static Private Attributes

- static bool [unityEnabled_S](#) = true

Copies the unityEnabled boolean so it works in a static method.

5.17.1 Detailed Description

Utility Script that allows access to the internal debugger of the SenseGloveCs Library, and controls debug messages from the SenseGlove SDK specifically.

5.17.2 Member Function Documentation

5.17.2.1 Instance_DebugMessageRecieved()

```
void SG.SG_Debugger.Instance_DebugMessageRecieved (
    object source,
    DebugArgs args ) [private]
```

Fires when our debugger reports that a new message has been recieved.

Parameters

<i>source</i>	
<i>args</i>	

5.17.2.2 Log()

```
static void SG.SG_Debugger.Log (
    string message ) [static]
```

Write a message to the [SG_Debugger](#).

Parameters

<i>message</i>	
----------------	--

5.17.2.3 LogError()

```
static void SG.SG_Debugger.LogError (
    string message ) [static]
```

Write a message to the [SG_Debugger](#) to appear as an error.

Parameters

<i>message</i>	
----------------	--

5.17.2.4 LogWarning()

```
static void SG.SG_Debugger.LogWarning (
    string message ) [static]
```

Write a message to the [SG_Debugger](#) to appear as a warning.

Parameters

<i>message</i>	
----------------	--

5.17.3 Member Data Documentation

5.17.3.1 DLL_debugLevel

```
DebugLevel SG.SG_Debugger.DLL_debugLevel = SenseGloveCs.Diagnostics.Debugger.defaultDebugLvl
```

The level of debug messages that one will receive from the DLL.

5.17.3.2 unityEnabled

```
bool SG.SG_Debugger.unityEnabled = true
```

Enables or disables debug messages from the Unity SDK scripts.

5.17.3.3 unityEnabled_S

```
bool SG.SG_Debugger.unityEnabled_S = true [static], [private]
```

Copies the unityEnabled boolean so it works in a static method.

Becomes troublesome if you're using multiple [SG_Debugger](#) scripts. Still, I would like to be able to control my debug messages via the inspector.

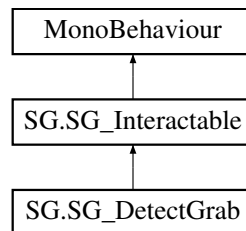
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_Debugger.cs

5.18 SG.SG_DetectGrab Class Reference

Attach this to any GameObject with a collider to have SG_Grabscripts detect it. Does not add any manipulation.

Inheritance diagram for SG.SG_DetectGrab:



Protected Member Functions

- override bool [InteractionBegin](#) ([SG_GrabScript](#) grabScript, bool fromExternal)
Called when the Interaction begins on this Interactable.
- override bool [InteractionEnd](#) ([SG_GrabScript](#) grabScript, bool fromExternal)
Called when the Interaction ends on this Interactable.

Additional Inherited Members

5.18.1 Detailed Description

Attach this to any GameObject with a collider to have SG_Grabscripts detect it. Does not add any manipulation.

5.18.2 Member Function Documentation

5.18.2.1 InteractionBegin()

```
override bool SG.SG_DetectGrab.InteractionBegin (
    SG\_GrabScript grabScript,
    bool fromExternal ) [protected], [virtual]
```

Called when the Interaction begins on this Interactable.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Returns

True if a succesfull connection has been established.

Implements [SG.SG_Interactive](#).

5.18.2.2 InteractionEnd()

```
override bool SG.SG_DetectGrab.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal ) [protected], [virtual]
```

Called when the Interaction ends on this Interactive.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Returns

True if the interaction has been ended.

Implements [SG.SG_Interactive](#).

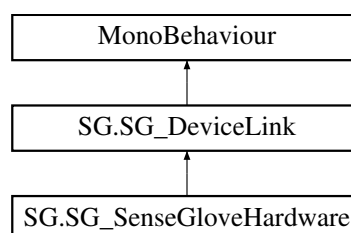
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_DetectGrab.cs

5.19 SG.SG_DeviceLink Class Reference

Link to a Sense Glove Device.

Inheritance diagram for SG.SG_DeviceLink:



Public Member Functions

- virtual SenseGloveCs.IODevice **GetInternalObject** ()
- bool **LinkDevice** (SenseGloveCs.IODevice device, int index)
- void **UnlinkDevice** ()

Protected Member Functions

- virtual bool **CanLinkTo** (SenseGloveCs.IODevice device)
- virtual void **SetupDevice** ()
When linked, this function is run for first time setup.
- virtual void **DisposeDevice** ()
Run when the device is unliked, a.k.a. when the DeviceList shuts down / during OnDestroy
- virtual void **OnDestroy** ()

Protected Attributes

- SenseGloveCs.IODevice **linkedDevice** = null

Properties

- int **DeviceIndex** [get, protected set]
- virtual bool **IsConnected** [get]
- virtual bool **IsLinked** [get]

5.19.1 Detailed Description

Link to a Sense Glove Device.

5.19.2 Member Function Documentation

5.19.2.1 DisposeDevice()

```
virtual void SG.SG_DeviceLink.DisposeDevice ( ) [protected], [virtual]
```

Run when the device is unliked, a.k.a. when the DeviceList shuts down / during OnDestroy

Reimplemented in [SG.SG_SenseGloveHardware](#).

5.19.2.2 SetupDevice()

```
virtual void SG.SG_DeviceLink.SetupDevice ( ) [protected], [virtual]
```

When linked, this function is run for first time setup.

Reimplemented in [SG.SG_SenseGloveHardware](#).

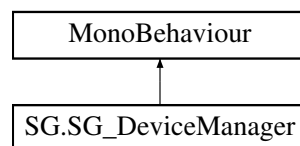
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_DeviceLink.cs

5.20 SG.SG_DeviceManager Class Reference

Class that links SenseGlove hardware to object in the Unity Engine.

Inheritance diagram for SG.SG_DeviceManager:



Classes

- class [DeviceDetectedArgs](#)
Arguments for the GloveDetected Event.

Public Member Functions

- void [CheckConnections](#) ()
Check if any new connections have come in, and should be linked.
- int [ListIndex](#) ([SG_DeviceLink](#) link)
- void [AddToWatchList](#) ([SG_DeviceLink](#) link)
- void [ClearConnections](#) ()
Clear the current connections to devices so we can try again...

Static Public Member Functions

- static string [ReportConnections](#) ()
Reports all internal connections for debugging purposes.

Public Attributes

- bool **debug** = false
- KeyCode **clearDevicesKey** = KeyCode.None

Protected Member Functions

- void **SetupScanner** ()
- void **DisposeScanner** ()
- void **Log** (string msg)
- void **UnlinkAll** ()
Unlink devices from the list so they can be connected again in other scenes.
- void **Start** ()
- void **Update** ()
- void **OnDestroy** ()
- void **OnApplicationQuit** ()

Protected Attributes

- int **lastAvailable** = 0
- List< bool > **linked** = new List<bool>()
- int **objectsLinked** = 0

Private Attributes

- List< [SG_DeviceLink](#) > **devicesToLink** = new List<[SG_DeviceLink](#)>()
Sense Glove related object to link.

5.20.1 Detailed Description

Class that links SenseGlove hardware to object in the Unity Engine.

5.20.2 Member Function Documentation

5.20.2.1 CheckConnections()

```
void SG.SG_DeviceManager.CheckConnections ( )
```

Check if any new connections have come in, and should be linked.

5.20.2.2 ClearConnections()

```
void SG.SG_DeviceManager.ClearConnections ( )
```

Clear the current connections to devices so we can try again...

5.20.2.3 ListIndex()

```
int SG.SG_DeviceManager.ListIndex (
    SG\_DeviceLink link )
```

Parameters

<i>link</i>	
-------------	--

Using Index as opposed to bool because it might be useful later on

Returns**5.20.2.4 ReportConnections()**

```
static string SG.SG_DeviceManager.ReportConnections ( ) [static]
```

Reports all internal connections for debugging purposes.

Returns**5.20.2.5 UnlinkAll()**

```
void SG.SG_DeviceManager.UnlinkAll ( ) [protected]
```

Unlink devices from the list so they can be connected again in other scenes.

5.20.3 Member Data Documentation**5.20.3.1 devicesToLink**

```
List<SG_DeviceLink> SG.SG_DeviceManager.devicesToLink = new List<SG_DeviceLink>() [private]
```

Sense Glove related object to link.

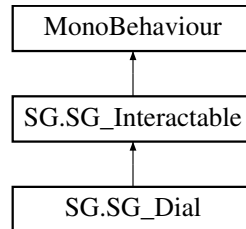
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_Device↵
Manager.cs

5.21 SG.SG_Dial Class Reference

A knob that can be twisted along its axis. Used in intricate button panels.

Inheritance diagram for SG.SG_Dial:



Public Member Functions

- override void [UpdateInteraction](#) ()
Update the dial while it is held by the glove.
- float [GetAngle](#) ()
Retrieve the latest angle of the dial
- float [SetAngle](#) (float angle)
Set the angle of the dial manually. Returns the angle that was set.
- float [ValidateAngle](#) (float angle)
Validate the dial angle before applying it.

Static Public Member Functions

- static Vector3 [GetAxis](#) ([MovementAxis](#) axis)
Retrieve a Vector3 representation of this dial's local rotation axis.
- static int [AngleIndex](#) ([MovementAxis](#) axis)
Retrieve the index (x, y or z) of the movementAxis.

Public Attributes

- bool [useLimits](#) = false
Whether the dial is limited in any direction or not.
- float [minAngle](#) = -180
The minimum angle of the dial, when using limits
- float [maxAngle](#) = 180
The maximum angle of the dial, when using limits

Protected Member Functions

- override bool [InteractionBegin](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Start an interaction between this dial and a sense glove.
- override bool [InteractionEnd](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
End an interaction between this dial and a Sense Glove.
- virtual void [UpdateAngle](#) ()
Contained in a separate method for child classes.
- virtual void **Start** ()
- virtual void **Update** ()

Protected Attributes

- Transform [_grabReference](#)
Grab reference of the grabscript that is currently interacting with this Dial.
- Transform [hingePoint](#)
The point / object around which the object pivots.
- [MovementAxis hingeAxis](#) = MovementAxis.X
The (local) axis of the hingePoint around which this dial pivots.
- Quaternion [qBase](#) = Quaternion.identity
Base rotation at startup, which is considered 0
- Vector3 [hinge](#) = new Vector3(1, 0, 0)
local hinge vector, updated when changing the hingeAxis.
- float [currAngle](#) = 0
The last assigned angle; used for quick access.
- Quaternion [rotOffset](#) = Quaternion.identity
Offset between the grabreference and the hinge point when the object was touched.
- float [anglOffset](#) = 0
The position of the dial when it was first touched.
- int [angleIndex](#) = 0
Index [x=0. y=1.z=2] by which to access the proper (local) euler angle.

Additional Inherited Members

5.21.1 Detailed Description

A knob that can be twisted along its axis. Used in intricate button panels.

5.21.2 Member Function Documentation

5.21.2.1 AngleIndex()

```
static int SG.SG_Dial.AngleIndex (
    MovementAxis axis ) [static]
```

Retrieve the index (x, y or z) of the movementAxis.

Parameters

<i>axis</i>	
-------------	--

Returns

5.21.2.2 GetAngle()

```
float SG.SG_Dial.GetAngle ( )
```

Retrieve the latest angle of the dial

Returns

5.21.2.3 GetAxis()

```
static Vector3 SG.SG_Dial.GetAxis (
    MovementAxis axis ) [static]
```

Retrieve a Vector3 representation of this dial's local rotation axis.

Parameters

<i>axis</i>	
-------------	--

Returns

5.21.2.4 InteractionBegin()

```
override bool SG.SG_Dial.InteractionBegin (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Start an interaction between this dial and a sense glove.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactable](#).

5.21.2.5 InteractionEnd()

```
override bool SG.SG_Dial.InteractionEnd (
```

```
SG_GrabScript grabScript,
bool fromExternal = false ) [protected], [virtual]
```

End an interaction between this dial and a Sense Glove.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactive](#).

5.21.2.6 SetAngle()

```
float SG.SG_Dial.SetAngle (
    float angle )
```

Set the angle of the dial manually. Returns the angle that was set.

Parameters

<i>angle</i>	
--------------	--

Returns

5.21.2.7 UpdateAngle()

```
virtual void SG.SG_Dial.UpdateAngle ( ) [protected], [virtual]
```

Contained in a separate method for child classes.

5.21.2.8 UpdateInteraction()

```
override void SG.SG_Dial.UpdateInteraction ( ) [virtual]
```

Update the dial while it is held by the glove.

Reimplemented from [SG.SG_Interactive](#).

5.21.2.9 ValidateAngle()

```
float SG.SG_Dial.ValidateAngle (
    float angle )
```

Validate the dial angle before applying it.

Parameters

<i>angle</i>	
--------------	--

Returns

5.21.3 Member Data Documentation

5.21.3.1 `_grabReference`

Transform SG.SG_Dial._grabReference [protected]

Grab reference of the grabscript that is currently interacting with this Dial.

5.21.3.2 `angleIndex`

int SG.SG_Dial.angleIndex = 0 [protected]

Index [x=0. y=1.z=2] by which to access the proper (local) euler angle.

5.21.3.3 `anglOffset`

float SG.SG_Dial.anglOffset = 0 [protected]

The position of the dial when it was first touched.

5.21.3.4 `currAngle`

float SG.SG_Dial.currAngle = 0 [protected]

The last assigned angle; used for quick access.

5.21.3.5 hinge

```
Vector3 SG.SG_Dial.hinge = new Vector3(1, 0, 0) [protected]
```

local hinge vector, updated when changing the hingeAxis.

5.21.3.6 hingeAxis

```
MovementAxis SG.SG_Dial.hingeAxis = MovementAxis.X [protected]
```

The (local) axis of the hingePoint around which this dial pivots.

5.21.3.7 hingePoint

```
Transform SG.SG_Dial.hingePoint [protected]
```

The point / object around which the object pivots.

5.21.3.8 maxAngle

```
float SG.SG_Dial.maxAngle = 180
```

The maximum angle of the dial, when using limits

5.21.3.9 minAngle

```
float SG.SG_Dial.minAngle = -180
```

The minimum angle of the dial, when using limits

5.21.3.10 qBase

```
Quaternion SG.SG_Dial.qBase = Quaternion.identity [protected]
```

Base rotation at startup, which is considered 0

5.21.3.11 rotOffset

```
Quaternion SG.SG_Dial.rotOffset = Quaternion.identity [protected]
```

Offset between the grabreference and the hinge point when the object was touched.

5.21.3.12 useLimits

```
bool SG.SG_Dial.useLimits = false
```

Whether the dial is limited in any direction or not.

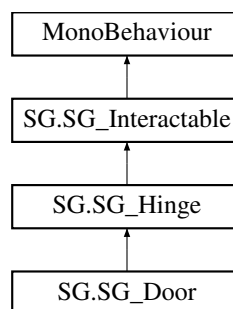
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Dial.cs

5.22 SG.SG_Door Class Reference

A SenseGlove_Hinge that represents a door. Can raise opened / closed events and have hidden content.

Inheritance diagram for SG.SG_Door:



Public Member Functions

- delegate void **DoorClosedEventHandler** (object source, EventArgs args)
- delegate void **DoorOpenedEventHandler** (object source, EventArgs args)

Protected Member Functions

- void **OnDoorClosed** ()
Raise the DoorClosed event
- void **OnDoorOpened** ()
Raise the DoorOpened Event

Events

- DoorClosedEventHandler [DoorClosed](#)
Fires the door returns to its initial position.
- DoorOpenedEventHandler [DoorOpened](#)
Fires the Door returns to its maxLimit position?

Additional Inherited Members

5.22.1 Detailed Description

A SenseGlove_Hinge that represents a door. Can raise opened / closed events and have hidden content.

5.22.2 Member Function Documentation

5.22.2.1 OnDoorClosed()

```
void SG.SG_Door.OnDoorClosed ( ) [protected]
```

Raise the DoorClosed event

5.22.2.2 OnDoorOpened()

```
void SG.SG_Door.OnDoorOpened ( ) [protected]
```

Raise the DoorOpened Event

5.22.3 Event Documentation

5.22.3.1 DoorClosed

```
DoorClosedEventHandler SG.SG_Door.DoorClosed
```

Fires the door returns to its initial position.

5.22.3.2 DoorOpened

DoorOpenedEventHandler SG.SG_Door.DoorOpened

Fires the Door returns to its maxLimit position?

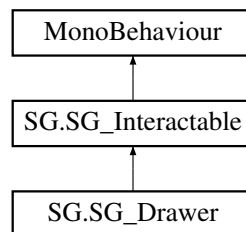
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Door.cs

5.23 SG.SG_Drawer Class Reference

A [SG_Interactive](#) that moves along one (local) axis.

Inheritance diagram for SG.SG_Drawer:



Public Member Functions

- override void [UpdateInteraction](#) ()
Called when the grabreference of the [SG_GrabScript](#) has been updated during the LateUpdate function.
- Vector3 [MoveAxis](#) ()
Retrieve the current movement axis [0, 0, 1].
- void [ForceOpen](#) (bool raiseEvent=false)
Force this drawer to its open (maxDist) position.
- void [ForceClosed](#) (bool raiseEvent=false)
Force this drawer to its original closed (minDist) position
- void [SetMoveAxis](#) ([MovementAxis](#) newAxis)
Set the moveDirection of this drawer. This method is cleaner than doing it via the public property
- bool [IsOpen](#) ()
Wheck if this drawer is currently open
- bool [IsClosed](#) ()
Check if this drawer is currently closed.
- override void [SaveTransform](#) ()
Save this drawer's current position when the ResetObject is called.
- override void [ResetObject](#) ()
Reset the drawer (and its contents?) To their original position.
- delegate void **DrawerClosedEventHandler** (object source, EventArgs args)
- delegate void **DrawerOpenedEventHandler** (object source, EventArgs args)
- override void [SetInteractive](#) (bool canInteract)
Sets the object to be interactable (or not).

Public Attributes

- **MovementAxis** `moveDirection` = MovementAxis.X
The movement axis along which the SenseGlove_Drawer slides.
- List< **SG_GrabZone** > `handles` = new List<SG_GrabZone>()
The handles connected to this drawer.
- float `minDrawerDist` = 0
The minimum distance that this drawer can move from its starting position.
- float `maxDrawerDist` = 1
The maximum distance that this drawer can move from its starting position.

Protected Member Functions

- virtual void **Awake** ()
- virtual void **Start** ()
- virtual void **Update** ()
- override bool **InteractionBegin** (SG_GrabScript grabScript, bool fromExternal=false)
Called when a new SG_GrabScript engages in an interaction with this Drawer
- override bool **InteractionEnd** (SG_GrabScript grabScript, bool fromExternal=false)
Called when a SG_GrabScript ends the interaction with this drawer.
- void **OnDrawerClosed** ()
- void **OnDrawerOpened** ()

Properties

- Vector3 **InitPos** [get]
- float **DrawerRatio** [get]

Events

- DrawerClosedEventHandler **DrawerClosed**
Fires the Drawer returns to its initial position.
- DrawerOpenedEventHandler **DrawerOpened**
Fires when the drawer reached its maximum extension.

Private Attributes

- bool `openEventFired` = false
Used to ensure the open and closed events are not fired every time.
- bool `closeEventFired` = true
- GameObject `grabReference`
The Grabreference of the SG_GrabScript that is attached to this drawer.
- Vector3 `grabOffset` = Vector3.zero
The offset between the grabReference at the time this drawer's interaction began.
- Vector3 `moveAxis`
The movement axis of this drawer. Will always be normalized (size is 1)
- **MovementAxis** `actualMoveDirection` = MovementAxis.X
Automatically recalculates the MoveAxis when one changes the moveDirection via the public property.

Additional Inherited Members

5.23.1 Detailed Description

A [SG_Interactive](#) that moves along one (local) axis.

5.23.2 Member Function Documentation

5.23.2.1 ForceClosed()

```
void SG.SG_Drawer.ForceClosed (
    bool raiseEvent = false )
```

Force this drawer to its original closed (minDist) position

5.23.2.2 ForceOpen()

```
void SG.SG_Drawer.ForceOpen (
    bool raiseEvent = false )
```

Force this drawer to its open (maxDist) position.

5.23.2.3 InteractionBegin()

```
override bool SG.SG_Drawer.InteractionBegin (
    SG\_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Called when a new [SG_GrabScript](#) engages in an interaction with this Drawer

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactive](#).

5.23.2.4 InteractionEnd()

```
override bool SG.SG_Drawer.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Called when a [SG_GrabScript](#) ends the interaction with this drawer.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactable](#).

5.23.2.5 IsClosed()

```
bool SG.SG_Drawer.IsClosed ( )
```

Check if this drawer is currently closed.

Returns

5.23.2.6 IsOpen()

```
bool SG.SG_Drawer.IsOpen ( )
```

Wheck if this drawer is currently open

Returns

5.23.2.7 MoveAxis()

```
Vector3 SG.SG_Drawer.MoveAxis ( )
```

Retrieve the current movement axis [0, 0, 1].

Returns

5.23.2.8 ResetObject()

```
override void SG.SG_Drawer.ResetObject ( ) [virtual]
```

Reset the drawer (and its contents?) To their original position.

Reimplemented from [SG.SG_Interactive](#).

5.23.2.9 SaveTransform()

```
override void SG.SG_Drawer.SaveTransform ( ) [virtual]
```

Save this drawer's current position when the ResetObject is called.

Reimplemented from [SG.SG_Interactive](#).

5.23.2.10 SetInteractive()

```
override void SG.SG_Drawer.SetInteractive (
    bool canInteract ) [virtual]
```

Sets the object to be interactable (or not).

May be overridden by sub-classes.

Parameters

<i>canInteract</i>	
--------------------	--

Reimplemented from [SG.SG_Interactive](#).

5.23.2.11 SetMoveAxis()

```
void SG.SG_Drawer.SetMoveAxis (
    MovementAxis newAxis )
```

Set the moveDirection of this drawer. This method is cleaner than doing it via the public property

Parameters

<i>newAxis</i>	
----------------	--

5.23.2.12 UpdateInteraction()

```
override void SG.SG_Drawer.UpdateInteraction ( ) [virtual]
```

Called when the grabreference of the [SG_GrabScript](#) has been updated during the LateUpdate function.

Reimplemented from [SG.SG_Interactable](#).

5.23.3 Member Data Documentation

5.23.3.1 actualMoveDirection

```
MovementAxis SG.SG_Drawer.actualMoveDirection = MovementAxis.X [private]
```

Automatically recalculates the MoveAxis when one changes the moveDirection via the public property.

5.23.3.2 grabOffset

```
Vector3 SG.SG_Drawer.grabOffset = Vector3.zero [private]
```

The offset between the grabReference at the time this drawer's interaction began.

5.23.3.3 grabReference

```
GameObject SG.SG_Drawer.grabReference [private]
```

The Grabreference of the [SG_GrabScript](#) that is attached to this drawer.

5.23.3.4 handles

```
List<SG_GrabZone> SG.SG_Drawer.handles = new List<SG_GrabZone>()
```

The handles connected to this drawer.

5.23.3.5 maxDrawerDist

```
float SG.SG_Drawer.maxDrawerDist = 1
```

The maximum distance that this drawer can move from its starting position.

5.23.3.6 minDrawerDist

```
float SG.SG_Drawer.minDrawerDist = 0
```

The minimum distance that this drawer can move from its starting position.

5.23.3.7 moveAxis

```
Vector3 SG.SG_Drawer.moveAxis [private]
```

The movement axis of this drawer. Will always be normalized (size is 1)

5.23.3.8 moveDirection

```
MovementAxis SG.SG_Drawer.moveDirection = MovementAxis.X
```

The movement axis along which the SenseGlove_Drawer slides.

5.23.3.9 openEventFired

```
bool SG.SG_Drawer.openEventFired = false [private]
```

Used to ensure the open and closed events are not fired every time.

5.23.4 Event Documentation

5.23.4.1 DrawerClosed

```
DrawerClosedEventHandler SG.SG_Drawer.DrawerClosed
```

Fires the Drawer returns to its initial position.

5.23.4.2 DrawerOpened

DrawerOpenedEventHandler SG.SG_Drawer.DrawerOpened

Fires when the drawer reached its maximum extension.

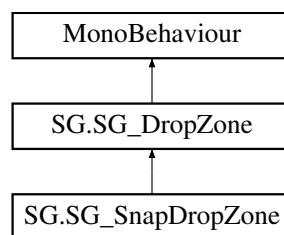
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Drawer.cs

5.24 SG.SG_DropZone Class Reference

Detects SenseGlove_Grabables within its volume.

Inheritance diagram for SG.SG_DropZone:



Classes

- class [DropProps](#)
Properties that assist in object detection.
- class [DropZoneArgs](#)

Public Member Functions

- virtual void [ValidateSettings](#) ()
Validates the settings of this DropZone.
- bool [IsDetected](#) (SG_Grabable obj)
Check if this Object has already been detected.
- bool [IsTarget](#) (SG_Grabable obj)
Check if this SG_SenseGloveHardware is one of the "goal" objects;
- void [AddTarget](#) (SG_Grabable obj)
Add a target object.
- virtual void [AddObject](#) (SG_Grabable grabable)
Adds an object to this SenseGlove_DropZone. Does not fire the eventTime.
- virtual void [RemoveObject](#) (SG_Grabable grabable)
Removes a specific object from this SenseGlove_DropZone
- virtual void [ClearObjects](#) ()
Clear all objects currently detected within this space.
- void [SetHighLight](#) (bool active)
Turn the Highlighter(s) of this DropZone on or off.
- virtual void [ResetZoneAndObjects](#) ()
Resets both the zone and its objects to their original state.
- delegate void [DropZoneEventHandler](#) (object source, [DropZoneArgs](#) args)
Event Delegate for DropZones.

Static Public Member Functions

- static int [ListIndex](#) (SG_Grabable obj, List< [SG_Grabable](#) > grabables)
Retrieve the index of a Grabable within a list of Grabables.

Public Attributes

- List< [SG_Grabable](#) > [objectsToGet](#) = new List<[SG_Grabable](#)>()
The objects that should be inside this DropZone. Leave it empty to snap to all SenseGlove_Grabables.
- float [detectionTime](#) = 0.2f
The time (in s) that a Grabable must be inside this zone before it is considered 'inside'.
- bool [detectHeldObjects](#) = true
Determines if objects that are still being held are detected.
- MeshRenderer[] [highLighters](#)
An optional highlight for this snapzone that can be turned on or off.

Protected Member Functions

- void [ValidateRB](#) ()
Check if all Rigidbody settings allow us to pick up objects.
- virtual void [RemoveObject](#) (int index)
Raises a removed event and then remove an object from all associated lists
- virtual void [CheckObjectEnter](#) (GameObject obj)
Check if a newly incoming object belongs to our targets.
- virtual void [CheckObjectExit](#) (GameObject obj)
- virtual void [CheckDetectionTimes](#) ()
Checks Detection times of the Grabables within this zone.
- virtual void [CallObjectDetect](#) ([SG_Grabable](#) detectedObject)
Calls the ObjectDetected event
- virtual void [CallObjectRemoved](#) ([SG_Grabable](#) removedObject)
Calls the ObjectRemoved event
- virtual void [Update](#) ()
- virtual void [OnDestroy](#) ()
- virtual void [OnEnable](#) ()
- virtual void [OnTriggerEnter](#) (Collider other)
- virtual void [OnTriggerStay](#) (Collider other)
- virtual void [OnTriggerExit](#) (Collider other)

Protected Attributes

- [SGEvent OnObjectDetected](#)
Fires when an Object is Detected.
- [SGEvent OnObjectRemoved](#)
Fires when an Object is Removed.
- bool [setup](#) = false
Whether or not this script has run setup before.
- float [checkStayTimer](#) = 0
Timer variable to check OnCollisionStay.
- Rigidbody [physicsBody](#)
The Rigidbody connected to this DropZone.
- List< [SG_Grabable](#) > [objectsInside](#) = new List<[SG_Grabable](#)>()
The list of objects currently inside this dropZone
- List< [DropProps](#) > [dropProperties](#) = new List<[DropProps](#)>()
Contains all properties for dropZone logic.

Static Protected Attributes

- static float [checkStayTime](#) = 0.2f
The time, in seconds, for which to check OnCollisionStay

Properties

- [SG_Grabable\[\] ObjectsInside](#) [get]
Get a list of all objects inside this DropZone.
- [SG_Grabable\[\] TargetObjects](#) [get]
- int [NumberOfObjects](#) [get]
Check the amount of objects within this DropZone.
- bool [AllObjectsDetected](#) [get]
Check if all desired objects have been detected.

Events

- [DropZoneEventHandler ObjectDetected](#)
Fires when an object has been detected inside this dropZone.
- [DropZoneEventHandler ObjectRemoved](#)
Fires when an object has been removed from this dropZone.

5.24.1 Detailed Description

Detects SenseGlove_Grabables within its volume.

5.24.2 Member Function Documentation

5.24.2.1 AddObject()

```
virtual void SG.SG_DropZone.AddObject (
    SG\_Grabable grabable ) [virtual]
```

Adds an object to this SenseGlove_DropZone. Does not fire the eventTime.

Parameters

grabable	
--------------------------	--

Reimplemented in [SG.SG_SnapDropZone](#).

5.24.2.2 AddTarget()

```
void SG.SG_DropZone.AddTarget (
    SG_Grabable obj )
```

Add a target object.

Parameters

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

5.24.2.3 CallObjectDetect()

```
virtual void SG.SG_DropZone.CallObjectDetect (
    SG_Grabable detectedObject ) [protected], [virtual]
```

Calls the ObjectDetected event

Parameters

<i>detectedObject</i>	
-----------------------	--

Reimplemented in [SG.SG_SnapDropZone](#).

5.24.2.4 CallObjectRemoved()

```
virtual void SG.SG_DropZone.CallObjectRemoved (
    SG_Grabable removedObject ) [protected], [virtual]
```

Calls the ObjectRemoved event

Parameters

<i>removedObject</i>	
----------------------	--

5.24.2.5 CheckDetectionTimes()

```
virtual void SG.SG_DropZone.CheckDetectionTimes ( ) [protected], [virtual]
```

Checks Detection times of the Grabables within this zone.

5.24.2.6 CheckObjectEnter()

```
virtual void SG.SG_DropZone.CheckObjectEnter (
    GameObject obj ) [protected], [virtual]
```

Check if a newly incoming object belongs to our targets.

Parameters

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

5.24.2.7 ClearObjects()

```
virtual void SG.SG_DropZone.ClearObjects ( ) [virtual]
```

Clear all objects currently detected within this space.

5.24.2.8 DropZoneEventHandler()

```
delegate void SG.SG_DropZone.DropZoneEventHandler (
    object source,
    DropZoneArgs args )
```

Event Delegate for DropZones.

Parameters

<i>source</i>	
<i>args</i>	

5.24.2.9 IsDetected()

```
bool SG.SG_DropZone.IsDetected (
    SG_Grabable obj )
```

Check if this Object has already been detected.

Parameters

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

Returns

5.24.2.10 IsTarget()

```
bool SG.SG_DropZone.IsTarget (
    SG_Grabable obj )
```

Check if this [SG_SenseGloveHardware](#) is one of the "goal" objects;

Parameters

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

Returns

5.24.2.11 ListIndex()

```
static int SG.SG_DropZone.ListIndex (
    SG_Grabable obj,
    List< SG_Grabable > grabables ) [static]
```

Retrieve the index of a Grabable within a list of Grabables.

Parameters

<i>obj</i>	
<i>grabables</i>	

Returns

Returns -1 if obj does not exist in grabables.

5.24.2.12 RemoveObject() [1/2]

```
virtual void SG.SG_DropZone.RemoveObject (
    int index ) [protected], [virtual]
```

Raises a removed event and then remove an object from all associated lists

Parameters

<i>index</i>	
--------------	--

Reimplemented in [SG.SG_SnapDropZone](#).

5.24.2.13 RemoveObject() [2/2]

```
virtual void SG.SG_DropZone.RemoveObject (
    SG_Grabable grabable ) [virtual]
```

Removes a specific object from this SenseGlove_DropZone

Parameters

<i>grabable</i>	
-----------------	--

5.24.2.14 ResetZoneAndObjects()

```
virtual void SG.SG_DropZone.ResetZoneAndObjects ( ) [virtual]
```

Resets both the zone and its objects to their original state.

5.24.2.15 SetHighLight()

```
void SG.SG_DropZone.SetHighLight (
    bool active )
```

Turn the Highlighter(s) of this DropZone on or off.

Parameters

<i>active</i>	
---------------	--

5.24.2.16 ValidateRB()

```
void SG.SG_DropZone.ValidateRB ( ) [protected]
```

Check if all RigidBody settings allow us to pick up objects.

5.24.2.17 ValidateSettings()

```
virtual void SG.SG_DropZone.ValidateSettings ( ) [virtual]
```

Validates the settings of this DropZone.

Reimplemented in [SG.SG_SnapDropZone](#).

5.24.3 Member Data Documentation

5.24.3.1 checkStayTime

```
float SG.SG_DropZone.checkStayTime = 0.2f [static], [protected]
```

The time, in seconds, for which to check OnCollisionStay

In case the collider is enabled with an object already inside

5.24.3.2 checkStayTimer

```
float SG.SG_DropZone.checkStayTimer = 0 [protected]
```

Timer variable to check OnCollisionStay.

5.24.3.3 detectHeldObjects

```
bool SG.SG_DropZone.detectHeldObjects = true
```

Determines if objects that are still being held are detected.

5.24.3.4 detectionTime

```
float SG.SG_DropZone.detectionTime = 0.2f
```

The time (in s) that a Grabable must be inside this zone before it is considered 'inside'.

5.24.3.5 dropProperties

```
List<DropProps> SG.SG_DropZone.dropProperties = new List<DropProps>() [protected]
```

Contains all properties for dropZone logic.

5.24.3.6 highLighters

```
MeshRenderer [] SG.SG_DropZone.highLighters
```

An optional highlight for this snapzone that can be turned on or off.

5.24.3.7 objectsInside

```
List<SG_Grabable> SG.SG_DropZone.objectsInside = new List<SG_Grabable>() [protected]
```

The list of objects currently inside this dropZone

5.24.3.8 objectsToGet

```
List<SG_Grabable> SG.SG_DropZone.objectsToGet = new List<SG_Grabable>()
```

The objects that should be inside this DropZone. Leave it empty to snap to all SenseGlove_Grabables.

5.24.3.9 OnObjectDetected

```
SGEvent SG.SG_DropZone.OnObjectDetected [protected]
```

Fires when an Object is Detected.

5.24.3.10 OnObjectRemoved

```
SGEvent SG.SG_DropZone.OnObjectRemoved [protected]
```

Fires when an Object is Removed.

5.24.3.11 physicsBody

Rigidbody SG.DropZone.physicsBody [protected]

The Rigidbody connected to this DropZone.

5.24.3.12 setup

bool SG.DropZone.setup = false [protected]

Whether or not this script has run setup before.

5.24.4 Property Documentation

5.24.4.1 AllObjectsDetected

bool SG.DropZone.AllObjectsDetected [get]

Check if all desired objects have been detected.

Returns

5.24.4.2 NumberOfObjects

int SG.DropZone.NumberOfObjects [get]

Check the amount of objects within this DropZone.

Returns

5.24.4.3 ObjectsInside

SG_Grabable [] SG.DropZone.ObjectsInside [get]

Get a list of all objects inside this DropZone.

Returns

5.24.5 Event Documentation

5.24.5.1 ObjectDetected

`DropZoneEventHandler SG.SG_DropZone.ObjectDetected`

Fires when an object has been detected inside this dropZone.

5.24.5.2 ObjectRemoved

`DropZoneEventHandler SG.SG_DropZone.ObjectRemoved`

Fires when an object has been removed from this dropZone.

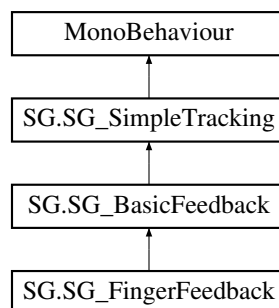
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_DropZone.cs

5.25 SG.SG_FingerFeedback Class Reference

Extends impact feedback to also take into account force feedback from [SG_Material](#)'s. These scripts calculate their distance into a collider.

Inheritance diagram for SG.SG_FingerFeedback:



Public Member Functions

- void `ResetForces` ()
Reset the forces and distances
- override void `SetupSelf` ()
Setup this collider's properties
- bool `IsTouching` ()
Check if this collider is touching a valid GameObject
- bool `IsTouching` (GameObject obj)
Check if this script is touching a specific gameobject
- bool `IsTouching` (Collider collider)
Check if this collider is touching a specific collider
- void `DetachScript` ()
Remove this script's reference to its [SG_Material](#) so that it is free to find another

Static Public Member Functions

- static bool [GetMaterialScript](#) (Collider col, out [SG_Material](#) materialScript, bool favourSpecific=true)
Utility function to find a [SG_Material](#) script attached to a collider. Returns true if such a script exists.
- static bool [SameScript](#) (Collider col, [SG_Material](#) touchedMat)
Utility function to check if a collider has a specific [SG_Material](#) collider attached.

Public Attributes

- bool [debugDirections](#) = false
If true, the force vectors of this script are rendered into the Scene view.

Protected Member Functions

- bool [ObjectDisabled](#) ()
Returns true if this script's touchedObject has been disabled or destroyed.
- void [FindForceDirection](#) (Collider col)
Calculated an 'entry vector' between this object and a collider.
- void [AttachScript](#) (Collider collider, [SG_Material](#) material)
Connect this script to a [SG_Material](#), and link any other possible components
- override void [FixedUpdate](#) ()
- override void [OnTriggerEnter](#) (Collider other)
- virtual void [OnTriggerExit](#) (Collider other)

Protected Attributes

- Vector3 [entryOrigin](#) = Vector3.zero
The position of the collider the moment it entered a new object. Used to determine collider normal.
- Vector3 [entryPoint](#) = Vector3.zero
A point of the collider of the touchedObject on the moment that collision was detected. Used to determine collider normal.

Properties

- GameObject [TouchedObject](#) [get, protected set]
The object that is currently touched by this SenseGlove_Touch script.
- [SG_Material](#) [TouchedMaterialScript](#) [get, protected set]
The Material of the last touched object. If set to null, it may have been deleted.
- [SG_MeshDeform](#) [TouchedDeformScript](#) [get, protected set]
The Mesh Deform of the last touched object, if available. Used to deform an object based on its SenseGlove-Material Properties.
- Collider [TouchedCollider](#) [get, protected set]
The collider that activated the feedback
- float [DistanceInCollider](#) [get, protected set]
The distance [in m] that the finger collider has penetrated into the object.
- int [ForceLevel](#) [get, protected set]
The current force-feedback level as determined by the material properties of the object we are touching.

Private Member Functions

- void [UpdateFeedback](#) ()

Calculate the force feedback levels based on material properties.

Additional Inherited Members

5.25.1 Detailed Description

Extends impact feedback to also take into account force feedback from [SG_Material](#)'s. These scripts calculate their distance into a collider.

5.25.2 Member Function Documentation

5.25.2.1 AttachScript()

```
void SG.SG_FingerFeedback.AttachScript (
    Collider collider,
    SG\_Material material ) [protected]
```

Connect this script to a [SG_Material](#), and link any other possible components

Parameters

<i>collider</i>	
<i>material</i>	

5.25.2.2 DetachScript()

```
void SG.SG_FingerFeedback.DetachScript ( )
```

Remove this script's reference to its [SG_Material](#) so that it is free to find another

5.25.2.3 FindForceDirection()

```
void SG.SG_FingerFeedback.FindForceDirection (
    Collider col ) [protected]
```

Calculated an 'entry vector' between this object and a collider.

Parameters

<i>col</i>	
------------	--

5.25.2.4 GetMaterialScript()

```
static bool SG.SG_FingerFeedback.GetMaterialScript (
    Collider col,
    out SG\_Material materialScript,
    bool favourSpecific = true ) [static]
```

Utility function to find a [SG_Material](#) script attached to a collider. Returns true if such a script exists.

Parameters

<i>col</i>	
<i>materialScript</i>	
<i>favourSpecific</i>	

Returns

5.25.2.5 IsTouching() [1/3]

```
bool SG.SG_FingerFeedback.IsTouching ( )
```

Check if this collider is touching a valid GameObject

5.25.2.6 IsTouching() [2/3]

```
bool SG.SG_FingerFeedback.IsTouching (
    Collider collider )
```

Check if this collider is touching a specific collider

Parameters

<i>collider</i>	
-----------------	--

Returns

5.25.2.7 IsTouching() [3/3]

```
bool SG.SG_FingerFeedback.IsTouching (
    GameObject obj )
```

Check if this script is touching a specific gameobject

Parameters

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

Returns

5.25.2.8 ObjectDisabled()

```
bool SG.SG_FingerFeedback.ObjectDisabled ( ) [protected]
```

Returns true if this script's touchedObject has been disabled or destroyed.

Returns

5.25.2.9 ResetForces()

```
void SG.SG_FingerFeedback.ResetForces ( )
```

Reset the forces and distances

5.25.2.10 SameScript()

```
static bool SG.SG_FingerFeedback.SameScript (
    Collider col,
    SG_Material touchedMat ) [static]
```

Utility function to check if a collider has a specific [SG_Material](#) collider attached.

Parameters

<i>col</i>	
<i>touchedMat</i>	

Returns

5.25.2.11 SetupSelf()

```
override void SG.SG_FingerFeedback.SetupSelf ( ) [virtual]
```

Setup this collider's properties

Reimplemented from [SG.SG_BasicFeedback](#).

5.25.2.12 UpdateFeedback()

```
void SG.SG_FingerFeedback.UpdateFeedback ( ) [private]
```

Calculate the force feedback levels based on material properties.

5.25.3 Member Data Documentation

5.25.3.1 debugDirections

```
bool SG.SG_FingerFeedback.debugDirections = false
```

If true, the force vectors of this script are rendered into the Scene view.

5.25.3.2 entryOrigin

```
Vector3 SG.SG_FingerFeedback.entryOrigin = Vector3.zero [protected]
```

The position of the collider the moment it entered a new object. Used to determine collider normal.

5.25.3.3 `entryPoint`

```
Vector3 SG.SG_FingerFeedback.entryPoint = Vector3.zero [protected]
```

A point of the collider of the touchedObject on the moment that collision was detected. Used to determine collider normal.

5.25.4 Property Documentation

5.25.4.1 `DistanceInCollider`

```
float SG.SG_FingerFeedback.DistanceInCollider [get], [protected set]
```

The distance [in m] that the finger collider has penetrated into the object.

5.25.4.2 `ForceLevel`

```
int SG.SG_FingerFeedback.ForceLevel [get], [protected set]
```

The current force-feedback level as determined by the material properties of the object we are touching.

5.25.4.3 `TouchedCollider`

```
Collider SG.SG_FingerFeedback.TouchedCollider [get], [protected set]
```

The collider that activated the feedback

5.25.4.4 `TouchedDeformScript`

```
SG\_MeshDeform SG.SG_FingerFeedback.TouchedDeformScript [get], [protected set]
```

The Mesh Deform of the last touched object, if available. Used to deform an object based on its SenseGlove-↔ Material Properties.

5.25.4.5 TouchedMaterialScript

`SG_Material` `SG.SG_FingerFeedback.TouchedMaterialScript` [get], [protected set]

The Material of the last touched object. If set to null, it may have been deleted.

5.25.4.6 TouchedObject

`GameObject` `SG.SG_FingerFeedback.TouchedObject` [get], [protected set]

The object that is currently touched by this SenseGlove_Touch script.

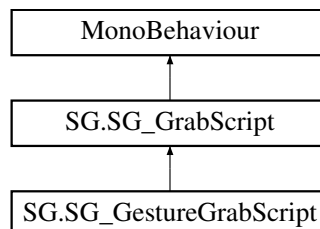
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_FingerFeedback.cs

5.26 SG.SG_GestureGrabScript Class Reference

A simplified SenseGlove_GrabScript that grabs all objects within it's 'hover collider' when a grab gesture is made.

Inheritance diagram for SG.SG_GestureGrabScript:



Public Member Functions

- override bool `CanInteract` ()
Returns true if this GrabScript is all set up to go.
- override bool `IsTouching` ()
Returns true if our HoverCollider is hovering above a valid object
- override bool `Setup` ()
Setup this GrabScript's components.
- override void `UpdateGrabScript` ()
Update this GrabScript's behaviour.

Public Attributes

- `SG_HoverCollider` `hoverCollider`
A collider in the hand palm that checks for SenseGlove_Interactable objects near the hand.

Protected Attributes

- float[] [lastAngles](#) = new float[5]
Angles during the last update, used to check for grab/release events.
- bool[] [grabbing](#) = new bool[5]
Whether each finger can be considered to be 'grasping' or 'grabbing'
- bool [wantedGrab](#) = false
Whether a grab action was desired during the last frame.

Static Protected Attributes

- static float[] [baseGrabAngles](#) = new float[5] { -60, -45, -45, -45, -90 }
Total flexion must fall below these values to consider 'grabbing'. Sorted thumb to pinky
- static float[] [baseReleaseAngles](#) = new float[5] { -20, -20, -20, -20, -45 }
Total flexion must fall below these values to consider 'releasing'. Sorted thumb to pinky

Additional Inherited Members

5.26.1 Detailed Description

A simplified SenseGlove_GrabScript that grabs all objects within it's 'hover collider' when a grab gesture is made.

5.26.2 Member Function Documentation

5.26.2.1 CanInteract()

```
override bool SG.SG_GestureGrabScript.CanInteract ( ) [virtual]
```

Returns true if this GrabScript is all set up to go.

Returns

Implements [SG.SG_GrabScript](#).

5.26.2.2 IsTouching()

```
override bool SG.SG_GestureGrabScript.IsTouching ( ) [virtual]
```

Returns true if our HoverCollider is hovering above a valid object

Returns

Implements [SG.SG_GrabScript](#).

5.26.2.3 Setup()

```
override bool SG.SG_GestureGrabScript.Setup ( ) [virtual]
```

Setup this GrabScript's components.

Returns

Implements [SG.SG_GrabScript](#).

5.26.2.4 UpdateGrabScript()

```
override void SG.SG_GestureGrabScript.UpdateGrabScript ( ) [virtual]
```

Update this GrabScript's behaviour.

Implements [SG.SG_GrabScript](#).

5.26.3 Member Data Documentation

5.26.3.1 baseGrabAngles

```
float [ ] SG.SG_GestureGrabScript.baseGrabAngles = new float[5] { -60, -45, -45, -45, -90 }  
[static], [protected]
```

Total flexion must fall below these values to consider 'grabbing'. Sorted thumb to pinky

5.26.3.2 baseReleaseAngles

```
float [ ] SG.SG_GestureGrabScript.baseReleaseAngles = new float[5] { -20, -20, -20, -20, -45 }  
[static], [protected]
```

Total flexion must fall below these values to consider 'releasing'. Sorted thumb to pinky

5.26.3.3 grabbing

```
bool [ ] SG.SG_GestureGrabScript.grabbing = new bool[5] [protected]
```

Whether each finger can be considered to be 'grasping' or 'grabbing'

5.26.3.4 hoverCollider

`SG_HoverCollider` `SG.SG_GestureGrabScript.hoverCollider`

A collider in the hand palm that checks for SenseGlove_Interactive objects near the hand.

5.26.3.5 lastAngles

`float []` `SG.SG_GestureGrabScript.lastAngles = new float[5] [protected]`

Angles during the last update, used to check for grab/release events.

5.26.3.6 wantedGrab

`bool` `SG.SG_GestureGrabScript.wantedGrab = false [protected]`

Whether a grab action was desired during the last frame.

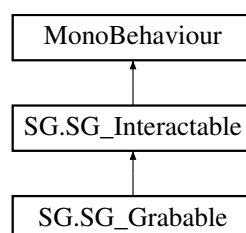
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Grabbing/SG_GestureGrabScript.cs`

5.27 SG.SG_Grabable Class Reference

An object that can be picked up and dropped by the SenseGlove.

Inheritance diagram for `SG.SG_Grabable`:



Public Member Functions

- override void [UpdateInteraction](#) ()
Called when this object is being held and the GrabReference is updated.
- void [SnapMeTo](#) (Transform originToMatch)
Moves this Grbabale such that its snapReference matches the rotation and position of the originToMatch.
- override void [SaveTransform](#) ()
Save this object's position and orientation, in case the ResetObject function is called.
- override void [ResetObject](#) ()
Reset this object back to its original position. Removes all connections between this and grabscripts.
- bool [IsGrabbed](#) ()
Check if this Interactable is currently being held by a SenseGlove GrabScript.
- void [ZeroVelocity](#) ()
Set the Velocities of this script to 0. Stops the grabable from rotating / flying away.
- bool [ConnectJoint](#) (Rigidbody other, float breakForce=SG_Grabable.defaultBreakForce)
Connect this Grabable's rigidBody to another using a FixedJoint
- void [BreakJoint](#) ()
Remove a fixedJoint connection between this object and another.
- void [SetCollision](#) (bool active)
Enable/Disable rigidbody collision of this Grabable.
- virtual void [CheckPickupRef](#) ()
Check the PickupReference of this Grabable
- virtual void [SaveRBParameters](#) ()
Store the Rigidbody parameters of this Grabable
- virtual void [Awake](#) ()

Public Attributes

- [GrabType](#) pickupMethod = [GrabType.Parent](#)
The way that this object is be picked up by a GrabScript.
- [AttachType](#) attachMethod = [AttachType.Default](#)
The way this object connects itself to the grabscript.
- Transform [snapReference](#)
If this object has an attachType of SnapToAnchor, this transform is used as a refrence.
- bool [canTransfer](#) = true
Whether or not this object can be picked up by another Grabscrip while it is being held.
- Transform [pickupReference](#)
The transform that is grabbed instead of this object. Useful when dealing with a grabable that is a child of another grabable.
- Rigidbody [physicsBody](#)
The rigidBody to which velocity, gravity and kinematic options are applied.

Static Public Attributes

- const float [defaultBreakForce](#) = 4000

Protected Member Functions

- override bool [InteractionBegin](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Called when a [SG_GrabScript](#) initiates an interaction with this grabable.
- override bool [InteractionEnd](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Called when a [SG_GrabScript](#) no longer wishes to interact with this grabable.
- virtual void **Update** ()

Protected Attributes

- GameObject [grabReference](#)
The gameObject used as a reference for the Grabable's transform updates.
- Vector3 [grabOffset](#) = Vector3.zero
The xyz offset of this Grabable's transform to the grabReference, on the moment it was picked up.
- Quaternion [grabRotation](#) = Quaternion.identity
The quaternion offset of this Grabable's transform to the grabReference, on the moment it was picked up.
- Transform **originalParent**
- Joint **connection**
- bool [wasKinematic](#)
Whether this grabable's physicsBody was kinematic before it was picked up.
- bool [usedGravity](#)
Whether this grabable's physicsBody was used gravity before it was picked up.

Properties

- Transform [OriginalParent](#) [get, set]
The original parent of this Grabable, before any GrabScripts picked it up.
- bool [UsedGravity](#) [get, set]
Whether this Grabable used gravity before it was picked up
- bool [WasKinematic](#) [get, set]
Whether this Grabable was marked as Kinematic before it was picked up

Additional Inherited Members

5.27.1 Detailed Description

An object that can be picked up and dropped by the SenseGlove.

5.27.2 Member Function Documentation

5.27.2.1 BreakJoint()

```
void SG.SG_Grabable.BreakJoint ( )
```

Remove a fixedJoint connection between this object and another.

5.27.2.2 CheckPickupRef()

```
virtual void SG.SG_Grabable.CheckPickupRef ( ) [virtual]
```

Check the PickupReference of this Grabable

5.27.2.3 ConnectJoint()

```
bool SG.SG_Grabable.ConnectJoint (
    Rigidbody other,
    float breakForce = SG_Grabable.defaultBreakForce )
```

Connect this Grabable's rigidBody to another using a FixedJoint

Parameters

<i>other</i>	
--------------	--

Returns

True, if the connection was sucesfully made.

5.27.2.4 InteractionBegin()

```
override bool SG.SG_Grabable.InteractionBegin (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Called when a [SG_GrabScript](#) initiates an interaction with this grabable.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactable](#).

5.27.2.5 InteractionEnd()

```
override bool SG.SG_Grabable.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Called when a [SG_GrabScript](#) no longer wishes to interact with this grabable.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Implements [SG.SG_Interactable](#).

5.27.2.6 IsGrabbed()

```
bool SG.SG_Grabable.IsGrabbed ( )
```

Check if this Interactable is currently being held by a SenseGlove GrabScript.

Returns**5.27.2.7 ResetObject()**

```
override void SG.SG_Grabable.ResetObject ( ) [virtual]
```

Reset this object back to its original position. Removes all connections between this and grabscripts.

Reimplemented from [SG.SG_Interactable](#).

5.27.2.8 SaveRBParameters()

```
virtual void SG.SG_Grabable.SaveRBParameters ( ) [virtual]
```

Store the RigidBody parameters of this Grabable

5.27.2.9 SaveTransform()

```
override void SG.SG_Grabable.SaveTransform ( ) [virtual]
```

Save this object's position and orientation, in case the ResetObject function is called.

Reimplemented from [SG.SG_Interactable](#).

5.27.2.10 SetCollision()

```
void SG.SG_Grabable.SetCollision (
    bool active )
```

Enable/Disable rigidbody collision of this Grabable.

Parameters

<i>active</i>	
---------------	--

5.27.2.11 SnapMeTo()

```
void SG.SG_Grabable.SnapMeTo (
    Transform originToMatch )
```

Moves this Grbabale such that its snapReference matches the rotation and position of the originToMatch.

Parameters

<i>originToMatch</i>	
----------------------	--

5.27.2.12 UpdateInteraction()

```
override void SG.SG_Grabable.UpdateInteraction ( ) [virtual]
```

Called when this object is being held and the GrabReference is updated.

Reimplemented from [SG.SG_Interactive](#).

5.27.2.13 ZeroVelocity()

```
void SG.SG_Grabable.ZeroVelocity ( )
```

Set the Velocities of this script to 0. Stops the grabable from rotating / flying away.

5.27.3 Member Data Documentation**5.27.3.1 attachMethod**

```
AttachType SG.SG_Grabable.attachMethod = AttachType.Default
```

The way this object connects itself to the grabscript.

5.27.3.2 canTransfer

```
bool SG.SG_Grabable.canTransfer = true
```

Whether or not this object can be picked up by another Grabscript while it is being held.

5.27.3.3 grabOffset

```
Vector3 SG.SG_Grabable.grabOffset = Vector3.zero [protected]
```

The xyz offset of this Grabable's transform to the grabReference, on the moment it was picked up.

5.27.3.4 grabReference

```
GameObject SG.SG_Grabable.grabReference [protected]
```

The gameObject used as a reference for the Grabable's transform updates.

5.27.3.5 grabRotation

```
Quaternion SG.SG_Grabable.grabRotation = Quaternion.identity [protected]
```

The quaternion offset of this Grabable's transform to the grabReference, on the moment it was picked up.

5.27.3.6 physicsBody

```
Rigidbody SG.SG_Grabable.physicsBody
```

The rigidBody to which velocity, gravity and kinematic options are applied.

5.27.3.7 pickupMethod

```
GrabType SG.SG_Grabable.pickupMethod = GrabType.Parent
```

The way that this object is be picked up by a GrabScript.

5.27.3.8 pickupReference

`Transform SG.SG_Grabable.pickupReference`

The transform that is grabbed instead of this object. Useful when dealing with a grabable that is a child of another grabable.

5.27.3.9 snapReference

`Transform SG.SG_Grabable.snapReference`

If this object has an attachType of SnapToAnchor, this transform is used as a reference.

5.27.3.10 usedGravity

`bool SG.SG_Grabable.usedGravity [protected]`

Whether this grabable's physicsBody was used gravity before it was picked up.

5.27.3.11 wasKinematic

`bool SG.SG_Grabable.wasKinematic [protected]`

Whether this grabable's physicsBody was kinematic before it was picked up.

5.27.4 Property Documentation

5.27.4.1 OriginalParent

`Transform SG.SG_Grabable.OriginalParent [get], [set]`

The original parent of this Grabable, before any GrabScripts picked it up.

5.27.4.2 UsedGravity

`bool SG.SG_Grabable.UsedGravity [get], [set]`

Whether this Grabable used gravity before it was picked up

5.27.4.3 WasKinematic

```
bool SG.SG_Grabable.WasKinematic [get], [set]
```

Whether this Grabable was marked as Kinematic before it was picked up

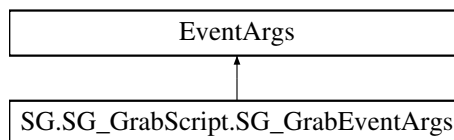
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Grabable.cs

5.28 SG.SG_GrabScript.SG_GrabEventArgs Class Reference

Event Arguments for grabbing/releasing of objects.

Inheritance diagram for SG.SG_GrabScript.SG_GrabEventArgs:



Public Member Functions

- **SG_GrabEventArgs** ([SG_Interactive](#) obj)

Properties

- [SG_Interactive Interactive](#) [get, protected set]
The object that is being grabbed or released

5.28.1 Detailed Description

Event Arguments for grabbing/releasing of objects.

5.28.2 Property Documentation

5.28.2.1 Interactive

```
SG\_Interactive SG.SG_GrabScript.SG_GrabEventArgs.Interactable [get], [protected set]
```

The object that is being grabbed or released

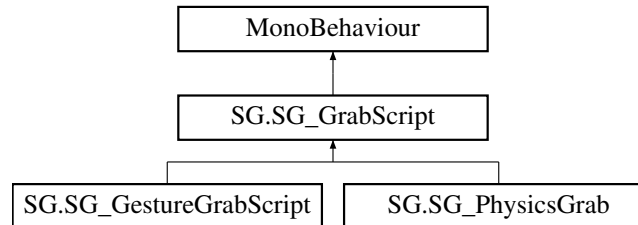
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Grabbing/SG_GrabScript.cs

5.29 SG.SG_GrabScript Class Reference

A Grabscrip that uses a number of the Sense Glove's data to start and end interactions.

Inheritance diagram for SG.SG_GrabScript:



Classes

- class [SG_GrabEventArgs](#)
Event Arguments for grabbing/releasing of objects.

Public Member Functions

- virtual bool [GetHardware](#) (out [SG_SenseGloveHardware](#) hardware)
Returns true if this GrabScript is connected to Sense Glove Hardware and returns a reference to it. Used in an if statement for safety
- virtual void [CheckForScripts](#) ()
Check for relevant scripts connected to this one, that may not yet have been assigned.
- Vector3 [GetVelocity](#) ()
Retrieve the Velocity of this Grabscrip in m/s
- Vector3 [GetAngularVelocity](#) ()
Retrieve the angular velocity of this Grabscrip in rad/s
- abstract bool [Setup](#) ()
Run setup on this grabscrip; creating and/or resizing the proper colliders etc.
- virtual void [ManualRelease](#) (float timeToReactivate=1.0f)
Manually force the SenseGlove_PhysGrab to drop whatever it is holding.
- abstract bool [CanInteract](#) ()
Returns true if this grabscrip can currently pickup an object
- virtual [SG_Interactable](#)[] [HeldObjects](#) ()
Return a list of GameObjects that this script is Currently Interacting with.
- virtual bool [IsGrabbing](#) ()
Returns true if this grabscrip is currently holding an object
- virtual bool [IsGrabbing](#) ([SG_Interactable](#) obj)
Returns true if this GrabScript is grabbing a specific [SG_Interactable](#).
- abstract bool [IsTouching](#) ()
Returns true if the grabscrip is touching an object
- virtual void [ClearHeldObjects](#) ()
Remove any references to held objects, restoring the GrabScript as though it has not touched anything yet.
- abstract void [UpdateGrabScript](#) ()
Update the Grabscrip logic; called automatically every Update() frame
- virtual void [EndInteraction](#) ([SG_Interactable](#) obj)
If this grabscrip is holding obj, end its interaction with it.
- delegate void [GrabEventHandler](#) (object source, [SG_GrabEventArgs](#) args)
Event Handler for grabbing/releasing objects

Public Attributes

- [SG_SenseGloveHardware](#) `hardware`
A [SG_SenseGloveHardware](#) for gloveData related shenanigans.
- `GameObject` [grabReference](#)
When an object is picked up, this `GameObject` (Typically the wrist) is used as a reference for its movement / parent / fixedJoint.
- `Rigidbody` [grabAnchor](#)
A `Rigidbody` that is used as an anchor when interacting with an object via a `FixedJoint`.

Protected Member Functions

- virtual void [UpdateDynamics](#) ()
Update the dynamics (velocity, angular velocity) of the grabreference.
- virtual bool [CanRelease](#) ([SG_Interactable](#) obj)
Check if this `GrabScript` is allowed to release an object, based on its release parameters.
- void [OnGrabbedObject](#) ([SG_Interactable](#) obj)
Calls the `ObjectGrabbed` event
- void [OnReleasedObject](#) ([SG_Interactable](#) obj)
Calls an `ObjectReleased` event.
- virtual void [TryGrabObject](#) ([SG_Interactable](#) obj)
Attempt to grab an `Interactable`. If succesful, fire the `ObjectGrabbed` event.
- virtual int [ReleaseObjectAt](#) (int index)
Attempt to release an `Interactable` in `heldObjects`. If succesful, fire the `ObjectReleased` event.
- virtual void **Awake** ()
- virtual void **Start** ()
- virtual void **Update** ()
- virtual void **LateUpdate** ()
- virtual void **OnDisable** ()

Protected Attributes

- bool [setupFinished](#) = false
Becomes true after the colliders have been succesfully assigned.
- `List< SG_Interactable >` [heldObjects](#) = new `List<SG_Interactable>(2)`
The object(s) that are being held by this script.
- `List< Vector3 >` [velocities](#) = new `List<Vector3>()`
The velocity during the previous frames.
- `List< Vector3 >` [angularVelocities](#) = new `List<Vector3>()`
The angular velocity during the previous frames.
- `Vector3` [lastPosition](#) = `Vector3.zero`
The `grabReference`'s position during the last frame.
- `Quaternion` [lastRotation](#) = `Quaternion.identity`
The `grabReference`'s rotation during the last frame.
- bool [paused](#) = false
If paused, the `GrabScript` will no longer raise events or grab objects untill the `pauseTime` has elapsed.
- float [pauseTime](#) = 1.0f
The time [s] that needs to elapse before the `GrabScript` can pick up another object.
- float [elapsedTime](#) = 0
The amount of time that has elapsed since the `Manual Release` function was called.

Static Protected Attributes

- static int [maxDataPoints](#) = 5
The maximum frames for which to keep track of velocities.

Properties

- virtual bool [DebugEnabled](#) [set]
Show/Hide the debug elements (colliders, DrawLines) of this GrabScript.
- [SG_TrackedHand Hand](#) [get, protected set]
The TrackedHand this GrabScript is connected to, used to access animation, hardware, etc.
- virtual bool [HardwareReady](#) [get]
Returns true if this GrabScript is connected to Hardware that is ready to go

Events

- [GrabEventHandler ObjectGrabbed](#)
Fires when a [SG_GrabScript](#)'s grabs an object.
- [GrabEventHandler ObjectReleased](#)
Fires when a [SG_GrabScript](#)'s releases an object.

5.29.1 Detailed Description

A Grabscrip that uses a number of the Sense Glove's data to start and end interactions.

5.29.2 Member Function Documentation

5.29.2.1 CanInteract()

```
abstract bool SG.SG_GrabScript.CanInteract ( ) [pure virtual]
```

Returns true if this grabscrip can currently pickup an object

Returns

Implemented in [SG.SG_PhysicsGrab](#), and [SG.SG_GestureGrabScript](#).

5.29.2.2 CanRelease()

```
virtual bool SG.SG_GrabScript.CanRelease (
    SG\_Interactable obj ) [protected], [virtual]
```

Check if this GrabScript is allowed to release an object, based on its release parameters.

Parameters

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

Returns

Reimplemented in [SG.SG_PhysicsGrab](#).

5.29.2.3 CheckForScripts()

```
virtual void SG.SG_GrabScript.CheckForScripts ( ) [virtual]
```

Check for relevant scripts connected to this one, that may not yet have been assigned.

Reimplemented in [SG.SG_PhysicsGrab](#).

5.29.2.4 ClearHeldObjects()

```
virtual void SG.SG_GrabScript.ClearHeldObjects ( ) [virtual]
```

Remove any references to held objects, restoring the GrabScript as though it has not touched anything yet.

5.29.2.5 EndInteraction()

```
virtual void SG.SG_GrabScript.EndInteraction (
    SG\_Interactive obj ) [virtual]
```

If this grabscript is holding obj, end its interaction with it.

Parameters

<i>obj</i>	
<i>callEvent</i>	Call the EndInteraction on this object.

5.29.2.6 GetAngularVelocity()

```
Vector3 SG.SG_GrabScript.GetAngularVelocity ( )
```

Retrieve the angular velocity of this Grabscrip in rad/s

Returns

5.29.2.7 GetHardware()

```
virtual bool SG.SG_GrabScript.GetHardware (
    out SG_SenseGloveHardware hardware ) [virtual]
```

Returns true if this GrabScript is connected to Sense Glove Hardware and returns a reference to it. Used in an if statement for safety

Parameters

<i>hardware</i>	
-----------------	--

Returns

5.29.2.8 GetVelocity()

```
Vector3 SG.SG_GrabScript.GetVelocity ( )
```

Retrieve the Velocity of this Grabscrip in m/s

Returns

5.29.2.9 GrabEventHandler()

```
delegate void SG.SG_GrabScript.GrabEventHandler (
    object source,
    SG_GrabEventArgs args )
```

Event Handler for grabbing/releasing objects

Parameters

<i>source</i>	
<i>args</i>	

5.29.2.10 HeldObjects()

```
virtual SG\_Interactive [ ] SG.SG_GrabScript.HeldObjects ( ) [virtual]
```

Return a list of GameObjects that this script is Currently Interacting with.

Returns

5.29.2.11 IsGrabbing() [1/2]

```
virtual bool SG.SG_GrabScript.IsGrabbing ( ) [virtual]
```

Returns true if this grabscript is currently holding an object

5.29.2.12 IsGrabbing() [2/2]

```
virtual bool SG.SG_GrabScript.IsGrabbing (
    SG\_Interactive obj ) [virtual]
```

Returns true if this GrabScript is grabbing a specific [SG_Interactive](#).

Parameters

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

Returns

5.29.2.13 IsTouching()

```
abstract bool SG.SG_GrabScript.IsTouching ( ) [pure virtual]
```

Returns true if the grabscript is touching an object

Returns

Implemented in [SG.SG_PhysicsGrab](#), and [SG.SG_GestureGrabScript](#).

5.29.2.14 ManualRelease()

```
virtual void SG.SG_GrabScript.ManualRelease (
    float timeToReactivate = 1.0f ) [virtual]
```

Manually force the SenseGlove_PhysGrab to drop whatever it is holding.

Parameters

<i>time</i>	The amount of time before the Grabscrip can pick up objects again
-------------	---

5.29.2.15 OnGrabbedObject()

```
void SG.SG_GrabScript.OnGrabbedObject (
    SG_Interactive obj ) [protected]
```

Calls the ObjectGrabbed event

Parameters

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

5.29.2.16 OnReleasedObject()

```
void SG.SG_GrabScript.OnReleasedObject (
    SG_Interactive obj ) [protected]
```

Calls an ObjectReleased event.

Parameters

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

5.29.2.17 ReleaseObjectAt()

```
virtual int SG.SG_GrabScript.ReleaseObjectAt (
    int index ) [protected], [virtual]
```

Attempt to release an Interactive in heldObjects. If succesful, fire the ObjectReleased event.

Parameters

<i>index</i>	
--------------	--

Returns**5.29.2.18 Setup()**

```
abstract bool SG.SG_GrabScript.Setup ( ) [pure virtual]
```

Run setup on this grabscript; creating and/or resizing the proper colliders etc.

Returns

Implemented in [SG.SG_PhysicsGrab](#), and [SG.SG_GestureGrabScript](#).

5.29.2.19 TryGrabObject()

```
virtual void SG.SG_GrabScript.TryGrabObject (
    SG\_Interactive obj ) [protected], [virtual]
```

Attempt to grab an Interactive. If succesful, fire the ObjectGrabbed event.

Parameters

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

5.29.2.20 UpdateDynamics()

```
virtual void SG.SG_GrabScript.UpdateDynamics ( ) [protected], [virtual]
```

Update the dynamics (velocity, angular velocity) of the grabreference.

5.29.2.21 UpdateGrabScript()

```
abstract void SG.SG_GrabScript.UpdateGrabScript ( ) [pure virtual]
```

Update the Grabscrip logic; called automatically every Update() frame

Implemented in [SG.SG_PhysicsGrab](#), and [SG.SG_GestureGrabScript](#).

5.29.3 Member Data Documentation

5.29.3.1 angularVelocities

```
List<Vector3> SG.SG_GrabScript.angularVelocities = new List<Vector3>() [protected]
```

The angular velocity during the previous frames.

5.29.3.2 elapsedTime

```
float SG.SG_GrabScript.elapsedTime = 0 [protected]
```

The amount of time that has elapsed since the Manual Release function was called.

5.29.3.3 grabAnchor

```
Rigidbody SG.SG_GrabScript.grabAnchor
```

A Rigidbody that is used as an anchor when interacting with an object via a FixedJoint.

5.29.3.4 grabReference

```
GameObject SG.SG_GrabScript.grabReference
```

When an object is picked up, this GameObject (Typically the wrist) is used as a reference for its movement / parent / fixedJoint.

5.29.3.5 hardware

`SG_SenseGloveHardware SG.SG_GrabScript.hardware`

A [SG_SenseGloveHardware](#) for gloveData related shenanigans.

5.29.3.6 heldObjects

`List<SG_Interactable> SG.SG_GrabScript.heldObjects = new List<SG_Interactable>(2) [protected]`

The object(s) that are being held by this script.

5.29.3.7 lastPosition

`Vector3 SG.SG_GrabScript.lastPosition = Vector3.zero [protected]`

The grabReference's position during the last frame.

5.29.3.8 lastRotation

`Quaternion SG.SG_GrabScript.lastRotation = Quaternion.identity [protected]`

The grabReference's rotation during the last frame.

5.29.3.9 maxDataPoints

`int SG.SG_GrabScript.maxDataPoints = 5 [static], [protected]`

The maximum frames for which to keep track of velocities.

5.29.3.10 paused

`bool SG.SG_GrabScript.paused = false [protected]`

If paused, the GrabScript will no longer raise events or grab objects until the pauseTime has elapsed.

5.29.3.11 pauseTime

```
float SG.SG_GrabScript.pauseTime = 1.0f [protected]
```

The time [s] that needs to elapse before the GrabScript can pick up another object.

5.29.3.12 setupFinished

```
bool SG.SG_GrabScript.setupFinished = false [protected]
```

Becomes true after the colliders have been successfully assigned.

5.29.3.13 velocities

```
List<Vector3> SG.SG_GrabScript.velocities = new List<Vector3>() [protected]
```

The velocity during the previous frames.

5.29.4 Property Documentation

5.29.4.1 DebugEnabled

```
virtual bool SG.SG_GrabScript.DebugEnabled [set]
```

Show/Hide the debug elements (colliders, DrawLines) of this GrabScript.

5.29.4.2 Hand

```
SG_TrackedHand SG.SG_GrabScript.Hand [get], [protected set]
```

The TrackedHand this GrabScript is connected to, used to access animation, hardware, etc.

5.29.4.3 HardwareReady

```
virtual bool SG.SG_GrabScript.HardwareReady [get]
```

Returns true if this GrabScript is connected to Hardware that is ready to go

5.29.5 Event Documentation

5.29.5.1 ObjectGrabbed

`GrabEventHandler SG.SG_GrabScript.ObjectGrabbed`

Fires when a [SG_GrabScript](#)'s grabs an object.

5.29.5.2 ObjectReleased

`GrabEventHandler SG.SG_GrabScript.ObjectReleased`

Fires when a [SG_GrabScript](#)'s releases an object.

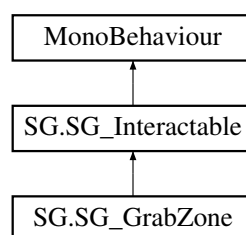
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Grabbing/SG_GrabScript.cs

5.30 SG.SG_GrabZone Class Reference

Creates a zone that extends its [SG_Interactive](#) methods to other objects, essentially creating a handle for (multiple) other Interactables.

Inheritance diagram for SG.SG_GrabZone:



Public Member Functions

- bool [ConnectTo](#) ([SG_Interactive](#) obj)
Connect a new Interactive to this GrabZone. Returns true if succesful.
- override void [UpdateInteraction](#) ()
Pass the updateInteraction on to all connected SenseGlove_Interactables.
- override void [ResetObject](#) ()
Pass the ResetObject on to all connected SenseGlove_Interactables.
- override void [SaveTransform](#) ()
Pass the SaveTransform function to all connected Interactables.

Public Attributes

- List< [SG_Interactive](#) > [connectedTo](#) = new List<[SG_Interactive](#)>()
The Interactables that this Grabzone is connected to.

Protected Member Functions

- override bool [InteractionBegin](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Pass the BeginInteraction on to all connected SenseGlove_Interactables.
- override bool [InteractionEnd](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Pass the EndInteraction on to all connected SenseGlove_Interactables.

Private Member Functions

- void **Awake** ()
- int [ConnectionIndex](#) ([SG_Interactive](#) obj)
Check if a [SG_Interactive](#) is already connected to this GrabZone.

Additional Inherited Members

5.30.1 Detailed Description

Creates a zone that extends its [SG_Interactive](#) methods to other objects, essentially creating a handle for (multiple) other Interactables.

5.30.2 Member Function Documentation

5.30.2.1 ConnectionIndex()

```
int SG.SG_GrabZone.ConnectionIndex (
    SG\_Interactive obj ) [private]
```

Check if a [SG_Interactive](#) is already connected to this GrabZone.

Parameters

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

Returns

5.30.2.2 ConnectTo()

```
bool SG.SG_GrabZone.ConnectTo (
    SG_Interactive obj )
```

Connect a new Interactive to this GrabZone. Returns true if succesful.

Parameters

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

Returns

5.30.2.3 InteractionBegin()

```
override bool SG.SG_GrabZone.InteractionBegin (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Pass the BeginInteraction on to all connected SenseGlove_Interactables.

Parameters

<i>grabScript</i>	
-------------------	--

Implements [SG.SG_Interactive](#).

5.30.2.4 InteractionEnd()

```
override bool SG.SG_GrabZone.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Pass the EndInteraction on to all connected SenseGlove_Interactables.

Parameters

<i>grabScript</i>	
-------------------	--

Implements [SG.SG_Interactive](#).

5.30.2.5 ResetObject()

```
override void SG.SG_GrabZone.ResetObject ( ) [virtual]
```

Pass the ResetObject on to all connected SenseGlove_Interactables.

Reimplemented from [SG.SG_Interactive](#).

5.30.2.6 SaveTransform()

```
override void SG.SG_GrabZone.SaveTransform ( ) [virtual]
```

Pass the SaveTransform function to all connected Interactables.

Reimplemented from [SG.SG_Interactive](#).

5.30.2.7 UpdateInteraction()

```
override void SG.SG_GrabZone.UpdateInteraction ( ) [virtual]
```

Pass the updateInteraction on to all connected SenseGlove_Interactables.

Reimplemented from [SG.SG_Interactive](#).

5.30.3 Member Data Documentation

5.30.3.1 connectedTo

```
List<SG\_Interactive> SG.SG_GrabZone.connectedTo = new List<SG\_Interactive>()
```

The Interactables that this Grabzone is connected to.

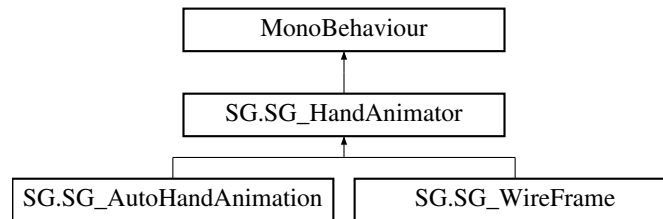
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_GrabZone.cs

5.31 SG.SG_HandAnimator Class Reference

A Generic Script that can be extended to work with most hand models. It requires the developer to assign the correct transforms for each joint. All of its methods can be overridden to create custom solutions.

Inheritance diagram for SG.SG_HandAnimator:



Public Member Functions

- virtual bool [GetHardware](#) (out [SG_SenseGloveHardware](#) hardware)
Returns true if this Animator is connected to Sense Glove Hardware. Used in an if statement for safety
- virtual void [CollectHandParameters](#) ()
collects the starting positions and rotations of the VHM, which can later be applied to Sense Glove models
- void [CalibrateWrist](#) ()
Calibrate the wrist model of this handModel.
- virtual void [UpdateHand](#) ([SG_SenseGloveData](#) data)
Update the (absolute) finger orientations, which move realtive to the (absolute) wrist transform. Note: This method is called after [UpdateWrist\(\)](#) is called.
- virtual void [UpdateWrist](#) ([SG_SenseGloveData](#) data)
Update the (absolute) wrist orientation, which moves realtive to the (absolute) lower arm transform. Note: This method is called before [UpdateFingers\(\)](#) is called.
- virtual void [ResizeHand](#) (float[][] newLengths)
Resize the finger lengths of this hand model to reflect that of the current user.

Static Public Member Functions

- static Vector3 [DifferenceFromWrist](#) (Transform [wristTransfrom](#), Vector3 absPos)
Calculates the difference between an absolute position and the wrist transform, without scaling.

Public Attributes

- [SG_SenseGloveHardware](#) [senseGlove](#)
The Sense Glove that controls this hand model. /summary>
- bool [updateWrist](#) = false
Whether or not to update the wrist of this Hand Model.
- Transform [foreArmTransfrom](#)
The GameObject representing the Forearm.
- Transform [wristTransfrom](#)
The GameObject representing the Wrist, moves relative to the foreArm.

Protected Member Functions

- virtual void [CheckForScripts](#) ()
Check for Scripts relevant for this Animator
- virtual void [SenseGlove_OnGloveLoaded](#) (object source, System.EventArgs args)
Utility method when the Sense Glove finishes loading. Determine left / right, for example.
- virtual void [SenseGlove_OnCalibrationFinished](#) (object source, [SG_SenseGloveHardware.GloveCalibrationArgs](#) args)
Call the ResizeFingers function.
- abstract void [CollectFingerJoints](#) ()
Collect a proper (finger x joint) array, and assign it to this.fingerJoints(). Use the handRoot variable to help you iterate.
- virtual void [CollectCorrections](#) ()
Collect the absolute angles of the fingers in their 'calibration' pose, correct these with the current wrist orientation.
- virtual void **Start** ()
- virtual void **Update** ()

Protected Attributes

- bool [updateFingers](#) = true
Whether or not to update the fingers of this Hand Model.
- bool [resizeFingers](#) = false
Whether or not to resize the fingers after calibration completes.
- Transform[][] [fingerJoints](#) = new Transform[0][]
The list of finger joint transforms, used to manipulate the angles. Assigned in the [CollectFingerJoints\(\)](#) function.
- List< List< Quaternion > > [fingerCorrection](#) = new List<List<Quaternion>>()
The initial angles of the hand model, corresponding to (0, 0, 0) rotation of the fingers.
- Quaternion [wristCorrection](#) = Quaternion.identity
Offset between the wrist and lower arm, used when updating the wrist transform.
- Quaternion [wristCalibration](#) = Quaternion.identity
Quaternion that aligns the lower arm with the wrist at the moment of calibration.
- Quaternion [wristAngles](#) = Quaternion.identity
The relative angles between wrist and lower arm transforms.
- GameObject [debugGroup](#)
A container for the motor level debug texts to easily toggle it on/off.
- TextMesh[] [debugText](#)
Show the motor levels as determine by the feedback colliders on the fingers.
- Vector3[] [_jointPositions](#) = new Vector3[0]
- Vector3[][] [_handLengths](#) = new Vector3[0][]

Properties

- [SG_TrackedHand Hand](#) [get, protected set]
The TrackedHand this Animator takes its data from, used to access grabscript, hardware, etc.
- virtual bool [HardwareReady](#) [get]
Returns true if this Animator is connected to Hardware that is ready to go
- Quaternion [RelativeWrist](#) [get]
Retrieve the Quaterion rotation between this model's foreArm and Wrist.
- Vector3 [WristAngles](#) [get]
Retrive the euler angles between this model's foreArm and Wrist.

5.31.1 Detailed Description

A Generic Script that can be extended to work with most hand models. It requires the developer to assign the correct transforms for each joint. All of its methods can be overridden to create custom solutions.

5.31.2 Member Function Documentation

5.31.2.1 CalibrateWrist()

```
void SG.SG_HandAnimator.CalibrateWrist ( )
```

Calibrate the wrist model of this handModel.

5.31.2.2 CheckForScripts()

```
virtual void SG.SG_HandAnimator.CheckForScripts ( ) [protected], [virtual]
```

Check for Scripts relevant for this Animator

Reimplemented in [SG.SG_AutoHandAnimation](#).

5.31.2.3 CollectCorrections()

```
virtual void SG.SG_HandAnimator.CollectCorrections ( ) [protected], [virtual]
```

Collect the absolute angles of the fingers in their 'calibration' pose, correct these with the current wrist orientation.

5.31.2.4 CollectFingerJoints()

```
abstract void SG.SG_HandAnimator.CollectFingerJoints ( ) [protected], [pure virtual]
```

Collect a proper (finger x joint) array, and assign it to this.fingerJoints(). Use the handRoot variable to help you iterate.

Implemented in [SG.SG_WireFrame](#), and [SG.SG_AutoHandAnimation](#).

5.31.2.5 CollectHandParameters()

```
virtual void SG.SG_HandAnimator.CollectHandParameters ( ) [virtual]
```

collects the starting positions and rotations of the VHM, which can later be applied to Sense Glove models

5.31.2.6 DifferenceFromWrist()

```
static Vector3 SG.SG_HandAnimator.DifferenceFromWrist (
    Transform wristTransform,
    Vector3 absPos ) [static]
```

Calculates the difference between an absolute position and the wrist transform, without scaling.

Parameters

<i>wristTransform</i>	
<i>absPos</i>	

Returns

5.31.2.7 GetHardware()

```
virtual bool SG.SG_HandAnimator.GetHardware (
    out SG\_SenseGloveHardware hardware ) [virtual]
```

Returns true if this Animator is connected to Sense Glove Hardware. Used in an if statement for safety

Parameters

<i>hardware</i>	
-----------------	--

Returns

5.31.2.8 ResizeHand()

```
virtual void SG.SG_HandAnimator.ResizeHand (
    float newLengths[ ][ ] ) [virtual]
```

Resize the finger lengths of this hand model to reflect that of the current user.

Parameters

<i>newLengths</i>	
-------------------	--

Reimplemented in [SG.SG_WireFrame](#).

5.31.2.9 SenseGlove_OnCalibrationFinished()

```
virtual void SG.SG_HandAnimator.SenseGlove_OnCalibrationFinished (
    object source,
    SG\_SenseGloveHardware.GloveCalibrationArgs args ) [protected], [virtual]
```

Call the `ResizeFingers` function.

Parameters

<i>source</i>	
<i>args</i>	

Reimplemented in [SG.SG_WireFrame](#).

5.31.2.10 SenseGlove_OnGloveLoaded()

```
virtual void SG.SG_HandAnimator.SenseGlove_OnGloveLoaded (
    object source,
    System.EventArgs args ) [protected], [virtual]
```

Utility method when the Sense Glove finishes loading. Determine left / right, for example.

Parameters

<i>source</i>	
<i>args</i>	

5.31.2.11 UpdateHand()

```
virtual void SG.SG_HandAnimator.UpdateHand (
    SG_SenseGloveData data ) [virtual]
```

Update the (absolute) finger orientations, which move relative to the (absolute) wrist transform. Note: This method is called after [UpdateWrist\(\)](#) is called.

Parameters

<i>data</i>	
-------------	--

Reimplemented in [SG.SG_WireFrame](#).

5.31.2.12 UpdateWrist()

```
virtual void SG.SG_HandAnimator.UpdateWrist (
    SG_SenseGloveData data ) [virtual]
```

Update the (absolute) wrist orientation, which moves relative to the (absolute) lower arm transform. Note: This method is called before [UpdateFingers\(\)](#) is called.

Parameters

<i>data</i>	
-------------	--

5.31.3 Member Data Documentation

5.31.3.1 debugGroup

`GameObject SG.SG_HandAnimator.debugGroup [protected]`

A container for the motor level debug texts to easily toggle it on/off.

5.31.3.2 debugText

`TextMesh [] SG.SG_HandAnimator.debugText [protected]`

Show the motor levels as determine by the feedback colliders on the fingers.

5.31.3.3 fingerCorrection

`List<List<Quaternion> > SG.SG_HandAnimator.fingerCorrection = new List<List<Quaternion>>() [protected]`

The initial angles of the hand model, corresponding to (0, 0, 0) rotation of the fingers.

5.31.3.4 fingerJoints

`Transform [][] SG.SG_HandAnimator.fingerJoints = new Transform[0][] [protected]`

The list of finger joint transforms, used to manipulate the angles. Assigned in the [CollectFingerJoints\(\)](#) function.

5.31.3.5 foreArmTransfrom

`Transform SG.SG_HandAnimator.foreArmTransfrom`

The GameObject representing the Forearm.

5.31.3.6 `resizeFingers`

```
bool SG.SG_HandAnimator.resizeFingers = false [protected]
```

Whether or not to resize the fingers after calibration completes.

5.31.3.7 `updateFingers`

```
bool SG.SG_HandAnimator.updateFingers = true [protected]
```

Whether or not to update the fingers of this Hand Model.

5.31.3.8 `updateWrist`

```
bool SG.SG_HandAnimator.updateWrist = false
```

Whether or not to update the wrist of this Hand Model.

5.31.3.9 `wristAngles`

```
Quaternion SG.SG_HandAnimator.wristAngles = Quaternion.identity [protected]
```

The relative angles between wrist and lower arm transforms.

5.31.3.10 `wristCalibration`

```
Quaternion SG.SG_HandAnimator.wristCalibration = Quaternion.identity [protected]
```

Quaternion that aligns the lower arm with the wrist at the moment of calibration.

5.31.3.11 `wristCorrection`

```
Quaternion SG.SG_HandAnimator.wristCorrection = Quaternion.identity [protected]
```

Offset between the wrist and lower arm, used when updating the wrist transform.

5.31.3.12 wristTransform

`Transform SG.SG_HandAnimator.wristTransform`

The GameObject representing the Wrist, moves relative to the foreArm.

5.31.4 Property Documentation

5.31.4.1 Hand

`SG_TrackedHand SG.SG_HandAnimator.Hand [get], [protected set]`

The TrackedHand this Animator takes its data from, used to access grabscript, hardware, etc.

5.31.4.2 HardwareReady

`virtual bool SG.SG_HandAnimator.HardwareReady [get]`

Returns true if this Animator is connected to Hardware that is ready to go

5.31.4.3 RelativeWrist

`Quaternion SG.SG_HandAnimator.RelativeWrist [get]`

Retrieve the Quaterion rotation between this model's foreArm and Wrist.

5.31.4.4 WristAngles

`Vector3 SG.SG_HandAnimator.WristAngles [get]`

Retrive the euler angles between this model's foreArm and Wrist.

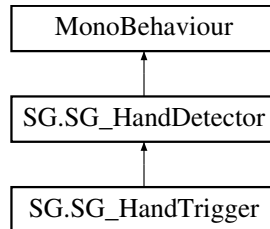
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_Hand↔Animator.cs`

5.32 SG.SG_HandDetector Class Reference

A class to detect a [SG_HandAnimator](#) based on its SG_Feedback colliders

Inheritance diagram for SG.SG_HandDetector:



Classes

- class [GloveDetectionArgs](#)
EventArgs fired when a glove is detected in or removed from a SenseGlove_Detector.

Public Types

- enum [DetectionType](#) { **AnyFinger** = 0, **SpecificFingers** }
How a Sense Glove is detected through its Feedback scripts.

Public Member Functions

- void [SetHighLight](#) (bool active)
Set the highlight of this Sense Glove on or off.
- bool [ContainsSenseGlove](#) ()
Returns true if there is a Sense Glove contained within this detector.
- [SG_SenseGloveHardware\[\]](#) [GlovesInside](#) ()
Get a list of all gloves within this detection area.
- delegate void **GloveDetectedEventHandler** (object source, [GloveDetectionArgs](#) args)
- delegate void **OnGloveRemovedEventHandler** (object source, [GloveDetectionArgs](#) args)
- void **ResetParameters** ()

Public Attributes

- [DetectionType](#) [detectionMethod](#) = DetectionType.AnyFinger
General Colliders or Specific fingers.
- int [activationThreshold](#) = 1
How many SG_Feedback colliders must enter the Detector before the GloveDetected event is raised.
- bool [detectThumb](#) = true
Whether or not this detector is activated by a thumb when detecting specific fingers only.
- bool [detectIndex](#) = true
Whether or not this detector is activated by an index finger when detecting specific fingers only.
- bool [detectMiddle](#) = true
Whether or not this detector is activated by a middle finger when detecting specific fingers only.

- bool [detectRing](#) = true
Whether or not this detector is activated by a ring finger when detecting specific fingers only.
- bool [detectPinky](#) = true
Whether or not this detector is activated by a pinky finger when detecting specific fingers only.
- float [activationTime](#) = 0
Optional: The time in seconds that the Sense Glove must be inside the detector for before the GloveDetected event is called.
- bool [singleGlove](#) = false
If set to true, the detector will not raise events if a second handModel joins in.
- Renderer [highLight](#)
An optional Highlight of this Detector that can be enabled / disabled.

Protected Member Functions

- virtual void **Start** ()
- virtual void **LateUpdate** ()
- virtual void **FireDetectEvent** ([SG_SenseGloveHardware](#) model)
A step in between events that can be overridden by sub-classes of the SenseGlove_Detector
- virtual void **FireRemoveEvent** ([SG_SenseGloveHardware](#) model)
A step in between events that can be overridden by sub-classes of the SenseGlove_Detector
- void **OnGloveDetected** ([SG_SenseGloveHardware](#) model)
- void **OnGloveRemoved** ([SG_SenseGloveHardware](#) model)

Protected Attributes

- List< [SG_SenseGloveHardware](#) > [detectedGloves](#) = new List<[SG_SenseGloveHardware](#)>()
All of the grabscripts currently interacting with this detector, in order of appearance.
- List< bool > [eventFired](#) = new List<bool>()
Used to determine if the activationtheshold had been reached before. Prevents the script from firing multiple times.

Events

- GloveDetectedEventHandler [GloveDetected](#)
Fires when a new [SG_GrabScript](#) enters this detection zone and fullfils the detector's conditions.
- OnGloveRemovedEventHandler [GloveRemoved](#)
Fires when a [SG_GrabScript](#) exits this detection zone and fullfils the detector's conditions.

Private Member Functions

- void **OnTriggerEnter** (Collider col)
- void **OnTriggerExit** (Collider col)
- int [HandModelIndex](#) ([SG_SenseGloveHardware](#) model)
Returns the index of the [SG_HandAnimator](#) in this detector's detectedGloves. Returns -1 if it is not in the list.
- void [AddEntry](#) ([SG_SenseGloveHardware](#) model)
Add a newly detected SenseGlove to the list of detected gloves.
- void [RemoveEntry](#) (int scriptIndex)
Remove a handmodel at the specified index from the list of detected gloves.
- bool [ValidScript](#) ([SG_HandSection](#) handSection)
Check if this scriptIndex is detectable by this Detector.
- bool **ValidScript** ([SG_BasicFeedback](#) touch)

Private Attributes

- List< int > [detectedColliders](#) = new List<int>()
The amount of SenseGlove_Touch colliders of each grabscript that are currently in the detection area
- List< float > [detectionTimes](#) = new List<float>()
Used to keep track of the time that each glove have been inside this detector.
- Collider [myCollider](#)
The collider of this detection area. Assigned on startup
- Rigidbody [myRigidbody](#)
The rigidbody of this detection area. Assigned on StartUp

5.32.1 Detailed Description

A class to detect a [SG_HandAnimator](#) based on its SG_Feedback colliders

5.32.2 Member Enumeration Documentation

5.32.2.1 DetectionType

```
enum SG.SG_HandDetector.DetectionType [strong]
```

How a Sense Glove is detected through its Feedback scripts.

5.32.3 Member Function Documentation

5.32.3.1 AddEntry()

```
void SG.SG_HandDetector.AddEntry (
    SG_SenseGloveHardware model ) [private]
```

Add a newly detected SenseGlove to the list of detected gloves.

Parameters

<i>model</i>	
--------------	--

5.32.3.2 ContainsSenseGlove()

```
bool SG.SG_HandDetector.ContainsSenseGlove ( )
```

Returns true if there is a Sense Glove contained within this detector.

Returns

5.32.3.3 FireDetectEvent()

```
virtual void SG.SG_HandDetector.FireDetectEvent (
    SG_SenseGloveHardware model ) [protected], [virtual]
```

A step in between events that can be overridden by sub-classes of the SenseGlove_Detector

Parameters

<i>model</i>	
--------------	--

Reimplemented in [SG.SG_HandTrigger](#).

5.32.3.4 FireRemoveEvent()

```
virtual void SG.SG_HandDetector.FireRemoveEvent (
    SG_SenseGloveHardware model ) [protected], [virtual]
```

A step in between events that can be overridden by sub-classes of the SenseGlove_Detector

Parameters

<i>model</i>	
--------------	--

Reimplemented in [SG.SG_HandTrigger](#).

5.32.3.5 GlovesInside()

```
SG_SenseGloveHardware [] SG.SG_HandDetector.GlovesInside ( )
```

Get a list of all gloves within this detection area.

Returns

5.32.3.6 HandModelIndex()

```
int SG.SG_HandDetector.HandModelIndex (
    SG_SenseGloveHardware model ) [private]
```

Returns the index of the [SG_HandAnimator](#) in this detector's detectedGloves. Returns -1 if it is not in the list.

Parameters

<i>grab</i>	
-------------	--

Returns

5.32.3.7 RemoveEntry()

```
void SG.SG_HandDetector.RemoveEntry (
    int scriptIndex ) [private]
```

Remove a handmodel at the specified index from the list of detected gloves.

Parameters

<i>scriptIndex</i>	
--------------------	--

5.32.3.8 SetHighLight()

```
void SG.SG_HandDetector.SetHighLight (
    bool active )
```

Set the highlight of this Sense Glove on or off.

Parameters

<i>active</i>	
---------------	--

5.32.3.9 ValidScript()

```
bool SG.SG_HandDetector.ValidScript (
    SG_HandSection handSection ) [private]
```

Check if this scriptIndex is detectable by this Detector.

Parameters

<i>scriptIndex</i>	
--------------------	--

Returns

5.32.4 Member Data Documentation

5.32.4.1 activationThreshold

```
int SG_SG_HandDetector.activationThreshold = 1
```

How many SG_Feedback colliders must enter the Detector before the GloveDetected event is raised.

5.32.4.2 activationTime

```
float SG_SG_HandDetector.activationTime = 0
```

Optional: The time in seconds that the Sense Glove must be inside the detector for before the GloveDetected event is called.

5.32.4.3 detectedColliders

```
List<int> SG_SG_HandDetector.detectedColliders = new List<int>() [private]
```

The amount of SenseGlove_Touch colliders of each grabscript that are currently in the detection area

5.32.4.4 detectedGloves

```
List<SG_SenseGloveHardware> SG_SG_HandDetector.detectedGloves = new List<SG_SenseGloveHardware>() [protected]
```

All of the grabscripts currently interacting with this detector, in order of appearance.

5.32.4.5 detectIndex

```
bool SG.SG_HandDetector.detectIndex = true
```

Whether or not this detector is activated by an index finger when detecting specific fingers only.

5.32.4.6 detectionMethod

```
DetectionType SG.SG_HandDetector.detectionMethod = DetectionType.AnyFinger
```

General Colliders or Specific fingers.

5.32.4.7 detectionTimes

```
List<float> SG.SG_HandDetector.detectionTimes = new List<float>() [private]
```

Used to keep track of the time that each glove have been inside this detector.

5.32.4.8 detectMiddle

```
bool SG.SG_HandDetector.detectMiddle = true
```

Whether or not this detector is activated by a middle finger when detecting specific fingers only.

5.32.4.9 detectPinky

```
bool SG.SG_HandDetector.detectPinky = true
```

Whether or not this detector is activated by a pinky finger when detecting specific fingers only.

5.32.4.10 detectRing

```
bool SG.SG_HandDetector.detectRing = true
```

Whether or not this detector is activated by a ring finger when detecting specific fingers only.

5.32.4.11 detectThumb

```
bool SG.SG_HandDetector.detectThumb = true
```

Whether or not this detector is activated by a thumb when detecting specific fingers only.

5.32.4.12 eventFired

```
List<bool> SG.SG_HandDetector.eventFired = new List<bool>() [protected]
```

Used to determine if the activationthreshold had been reached before. Prevents the script from firing multiple times.

5.32.4.13 highLight

```
Renderer SG.SG_HandDetector.highLight
```

An optional Highlight of this Detector that can be enabled / disabled.

5.32.4.14 myCollider

```
Collider SG.SG_HandDetector.myCollider [private]
```

The collider of this detection area. Assigned on startup

5.32.4.15 myRigidbody

```
Rigidbody SG.SG_HandDetector.myRigidbody [private]
```

The rigidbody of this detection area. Assigned on StartUp

5.32.4.16 singleGlove

```
bool SG.SG_HandDetector.singleGlove = false
```

If set to true, the detector will not raise events if a second handModel joins in.

5.32.5 Event Documentation

5.32.5.1 GloveDetected

GloveDetectedEventHandler SG.SG_HandDetector.GloveDetected

Fires when a new [SG_GrabScript](#) enters this detection zone and fulfils the detector's conditions.

5.32.5.2 GloveRemoved

OnGloveRemovedEventHandler SG.SG_HandDetector.GloveRemoved

Fires when a [SG_GrabScript](#) exits this detection zone and fulfils the detector's conditions.

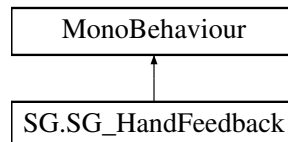
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_HandDetector.cs

5.33 SG.SG_HandFeedback Class Reference

This script collects the Force Feedback from the hand and sends these to its connected Hardware.

Inheritance diagram for SG.SG_HandFeedback:



Public Member Functions

- bool [GetHardware](#) (out [SG_SenseGloveHardware](#) hardware)
Returns true if this FeedbackScript is connected to Sense Glove Hardware and returns a link to it. Used in an if statement for safety
- bool [TouchingMaterial](#) ()
Returns true if at least one collider is touching a material.
- void [SetIgnoreCollision](#) ([SG_HandRigidBodies](#) otherLayer, bool ignoreCollision)
Set ignoreCollision between this layer and another set of rigidbodies.
- void [SetIgnoreCollision](#) (GameObject obj, bool ignoreCollision)
- void [SetIgnoreCollision](#) (Collider col, bool ignoreCollision)
- void [SetupScripts](#) ()
Sets up this script's components to link to the same glove and the appropriate hand section.
- void [UpdateForces](#) ()
Retrieve the forces for each finger and send these to the glove.
- virtual void [CheckForScripts](#) ()
Checks for scripts that might be connected to this GameObject. Used in editor and during startup.

Public Attributes

- [SG_SenseGloveHardware](#) `connectedGlove`
The hardware that this script will send its Force-Feedback commands to
- [SG_HandModelInfo](#) `handModel`
Information about the 3D model this script is connected to. Used to set up tracking for the fingers/wrist.
- [SG_BasicFeedback](#) `wristFeedbackScript`
Impact script for the wrist, should be linked to this connectedGlove.
- [SG_FingerFeedback\[\]](#) `fingerFeedbackScripts`
Feedback colliders on each of the fingers, sorted from thumb to pinky.

Properties

- [SG_TrackedHand](#) `Hand` [get, protected set]
The TrackedHand this FeedbackScript takes its data from, used to access other components like grabscript, hardware, etc.
- bool [HardwareReady](#) [get]
Returns true if this FeedbackScript is connected to Hardware that is ready to go
- bool [DebugEnabled](#) [set]
Used to show/hide the feedback colliders of this hand.
- float[] [ColliderDistances](#) [get]
returns the distance (in m) of the fingers inside a [SG_Material](#) collider, provided they are touching one.

Private Member Functions

- void **Awake** ()
- void **Update** ()

5.33.1 Detailed Description

This script collects the Force Feedback from the hand and sends these to its connected Hardware.

5.33.2 Member Function Documentation

5.33.2.1 CheckForScripts()

```
virtual void SG.SG_HandFeedback.CheckForScripts ( ) [virtual]
```

Checks for scripts that might be connected to this GameObject. Used in editor and during startup.

5.33.2.2 GetHardware()

```
bool SG.SG_HandFeedback.GetHardware (
    out SG\_SenseGloveHardware hardware )
```

Returns true if this FeedbackScript is connected to Sense Glove Hardware and returns a link to it. Used in an if statement for safety

Parameters

<i>hardware</i>	
-----------------	--

Returns**5.33.2.3 SetIgnoreCollision()**

```
void SG.SG_HandFeedback.SetIgnoreCollision (
    SG_HandRigidBodies otherLayer,
    bool ignoreCollision )
```

Set ignoreCollision between this layer and another set of rigidbodies.

Parameters

<i>otherLayer</i>	
<i>ignoreCollision</i>	

5.33.2.4 SetupScripts()

```
void SG.SG_HandFeedback.SetupScripts ( )
```

Sets up this script's components to link to the same glove and the appropriate hand section.

5.33.2.5 TouchingMaterial()

```
bool SG.SG_HandFeedback.TouchingMaterial ( )
```

Returns true if at least one collider is touching a material.

Returns

5.33.2.6 UpdateForces()

```
void SG_SG_HandFeedback.UpdateForces ( )
```

Retrieve the forces for each finger and send these to the glove.

5.33.3 Member Data Documentation

5.33.3.1 connectedGlove

```
SG_SenseGloveHardware SG_SG_HandFeedback.connectedGlove
```

The hardware that this script will send its Force-Feedback commands to

5.33.3.2 fingerFeedbackScripts

```
SG_FingerFeedback [ ] SG_SG_HandFeedback.fingerFeedbackScripts
```

Feedback colliders on each of the fingers, sorted from thumb to pinky.

5.33.3.3 handModel

```
SG_HandModelInfo SG_SG_HandFeedback.handModel
```

Information about the 3D model this script is connected to. Used to set up tracking for the fingers/wrist.

5.33.3.4 wristFeedbackScript

```
SG_BasicFeedback SG_SG_HandFeedback.wristFeedbackScript
```

Impact script for the wrist, should be linked to this connectedGlove.

5.33.4 Property Documentation

5.33.4.1 ColliderDistances

```
float [ ] SG.SG_HandFeedback.ColliderDistances [get]
```

returns the distance (in m) of the fingers inside a [SG_Material](#) collider, provided they are touching one.

5.33.4.2 DebugEnabled

```
bool SG.SG_HandFeedback.DebugEnabled [set]
```

Used to show/hide the feedback colliders of this hand.

5.33.4.3 Hand

```
SG\_TrackedHand SG.SG_HandFeedback.Hand [get], [protected set]
```

The TrackedHand this FeedbackScript takes its data from, used to access other components like grabscript, hardware, etc.

5.33.4.4 HardwareReady

```
bool SG.SG_HandFeedback.HardwareReady [get]
```

Returns true if this FeedbackScript is connected to Hardware that is ready to go

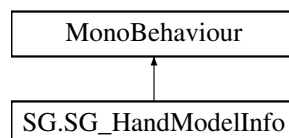
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_Hand↔ Feedback.cs

5.34 SG.SG_HandModelInfo Class Reference

A script to assign information of hand joints, used by other scripts that use hand tracking.

Inheritance diagram for SG.SG_HandModelInfo:



Public Member Functions

- bool [GetFingerTip](#) ([SG_HandSection](#) finger, out Transform fingerTip)
Retrieve the fingertip transform of this Hand Model.

Public Attributes

- Transform [foreArmTransform](#)
The forearm of the hand model, usually the parent of the wrist transform.
- Transform [wristTransform](#)
The transform of the wrist. Should be distinct from the foreArmTransform if wrist animation is not required.
- Transform[] [thumbJoints](#) = new Transform[0]
The thumb joint transforms, preferably including the fingertip.
- Transform[] [indexJoints](#) = new Transform[0]
The index joint transforms, preferably including the fingertip.
- Transform[] [middleJoints](#) = new Transform[0]
The middle joint transforms, preferably including the fingertip.
- Transform[] [ringJoints](#) = new Transform[0]
The ring joint transforms, preferably including the fingertip.
- Transform[] [pinkyJoints](#) = new Transform[0]
The pinky joint transforms, preferably including the fingertip.

Protected Attributes

- GameObject[][] [fingerDebug](#) = null
Debug objects to show the user where the finger joint transforms are.
- GameObject [wristDebug](#) = null
Debug objects to show the user where the wrist transform is

Properties

- Transform[][] [FingerJoints](#) [get]
Retreive all finger joints as an array of Transforms, sorted from thumb to pinky.
- bool [DebugEnabled](#) [get, set]
Create/Destroy a set of small spheres on each of the hand model transforms.

5.34.1 Detailed Description

A script to assign information of hand joints, used by other scripts that use hand tracking.

5.34.2 Member Function Documentation

5.34.2.1 GetFingerTip()

```
bool SG.SG_HandModelInfo.GetFingerTip (
    SG_HandSection finger,
    out Transform fingerTip )
```

Retrieve the fingertip transform of this Hand Model.

Parameters

<i>finger</i>	
<i>fingerTip</i>	

Returns

5.34.3 Member Data Documentation

5.34.3.1 fingerDebug

```
GameObject [][] SG.SG_HandModelInfo.fingerDebug = null [protected]
```

Debug objects to show the user where the finger joint transforms are.

5.34.3.2 foreArmTransform

```
Transform SG.SG_HandModelInfo.foreArmTransform
```

The forearm of the hand model, usually the parent of the wrist transform.

5.34.3.3 indexJoints

```
Transform [] SG.SG_HandModelInfo.indexJoints = new Transform[0]
```

The index joint transforms, preferably including the fingertip.

5.34.3.4 middleJoints

```
Transform [] SG.SG_HandModelInfo.middleJoints = new Transform[0]
```

The middle joint transforms, preferably including the fingertip.

5.34.3.5 pinkyJoints

```
Transform [ ] SG.SG_HandModelInfo.pinkyJoints = new Transform[0]
```

The pinky joint transforms, preferably including the fingertip.

5.34.3.6 ringJoints

```
Transform [ ] SG.SG_HandModelInfo.ringJoints = new Transform[0]
```

The ring joint transforms, preferably including the fingertip.

5.34.3.7 thumbJoints

```
Transform [ ] SG.SG_HandModelInfo.thumbJoints = new Transform[0]
```

The thumb joint transforms, preferably including the fingertip.

5.34.3.8 wristDebug

```
GameObject SG.SG_HandModelInfo.wristDebug = null [protected]
```

Debug objects to show the user where the wrist transform is

5.34.3.9 wristTransform

```
Transform SG.SG_HandModelInfo.wristTransform
```

The transform of the wrist. Should be distinct from the foreArmTransform if wrist animation is not required.

5.34.4 Property Documentation

5.34.4.1 DebugEnabled

```
bool SG.SG_HandModelInfo.DebugEnabled [get], [set]
```

Create/Destroy a set of small spheres on each of the hand model transforms.

5.34.4.2 FingerJoints

`Transform [][] SG.SG_HandModelInfo.FingerJoints [get]`

Retrieve all finger joints as an array of Transforms, sorted from thumb to pinky.

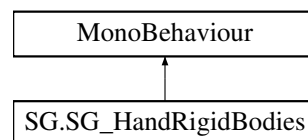
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_HandModelInfo.cs`

5.35 SG.SG_HandRigidBodyClass Reference

A script to manage a set of Rigidbodies that represent the hand geometry.

Inheritance diagram for SG.SG_HandRigidBodyClass:



Public Member Functions

- virtual bool [GetHardware](#) (out [SG_SenseGloveHardware](#) hardware)
Returns true if this Animator is connected to Sense Glove Hardware. Used in an if statement for safety
- void [SetIgnoreCollision](#) ([SG_HandRigidBodyClass](#) otherLayer, bool ignoreCollision)
Set ignoreCollision between this layer and another set of rigidbodies.
- void [SetIgnoreCollision](#) (GameObject obj, bool ignoreCollision)
Set the ignoreCollision between this layer and a specific gameobject
- void [SetIgnoreCollision](#) (Collider col, bool ignoreCollision)
Set the ignoreCollision between this layer and a specific collider
- void [AddRigidBody](#) (bool useGrav=false, bool kinematic=false)
Add Rigidbodies with proper parameters for this layer.
- void [RemoveRigidBody](#) ()
Removes rigidbodies from this layer, so their collision can become part of a different RigidBody.

Public Attributes

- [SG_HandModelInfo](#) handModel
The hand model information, used to assign tracking information. If left unassigned, you'll need to assign them manually.
- [SG_TrackedBody](#) wristObj
The managed rigidbody of the wrist
- [SG_TrackedBody\[\]](#) fingerObjs = new [SG_TrackedBody](#)[0]
The managed rigidbody of the fingers, from thumb to pinky.

Protected Member Functions

- virtual void [CheckForScripts](#) ()
Assign scripts relevant to this script's functioning.
- void [SetupSelf](#) ()
Setup the tracking / parameters of this script's components.
- void **Awake** ()

Properties

- [SG_TrackedHand Hand](#) [get, protected set]
The TrackedHand this Animator takes its data from, used to access grabscript, hardware, etc.
- virtual bool [HardwareReady](#) [get]
Returns true if this Animator is connected to Hardware that is ready to go
- bool [DebugEnabled](#) [set]
Show/Hide the the rigidbodies in this layer.
- bool [CollisionsEnabled](#) [set]
Enable/Disable the overall collision of the rigidbodies in this layer.

5.35.1 Detailed Description

A script to manage a set of Rigidbodies that represent the hand geometry.

5.35.2 Member Function Documentation

5.35.2.1 AddRigidBodies()

```
void SG.SG_HandRigidBodies.AddRigidBodies (
    bool useGrav = false,
    bool kinematic = false )
```

Add Rigidbodies with proper parameters for this layer.

Parameters

<i>useGrav</i>	
<i>kinematic</i>	

5.35.2.2 CheckForScripts()

```
virtual void SG.SG_HandRigidBodies.CheckForScripts ( ) [protected], [virtual]
```

Assign scripts relevant to this script's functioning.

5.35.2.3 GetHardware()

```
virtual bool SG.SG_HandRigidBodyClass.GetHardware (
    out SG_SenseGloveHardware hardware ) [virtual]
```

Returns true if this Animator is connected to Sense Glove Hardware. Used in an if statement for safety

Parameters

<i>hardware</i>	
-----------------	--

Returns

5.35.2.4 RemoveRigidBody()

```
void SG.SG_HandRigidBodyClass.RemoveRigidBody ( )
```

Removes rigidbodies from this layer, so their collision can become part of a different RigidBody.

5.35.2.5 SetIgnoreCollision() [1/3]

```
void SG.SG_HandRigidBodyClass.SetIgnoreCollision (
    Collider col,
    bool ignoreCollision )
```

Set the ignoreCollision between this layer and a specific collider

Parameters

<i>obj</i>	
<i>ignoreCollision</i>	

5.35.2.6 SetIgnoreCollision() [2/3]

```
void SG.SG_HandRigidBodyClass.SetIgnoreCollision (
    GameObject obj,
    bool ignoreCollision )
```

Set the ignoreCollision between this layer and a specific gameobject

Parameters

<i>obj</i>	
<i>ignoreCollision</i>	

5.35.2.7 SetIgnoreCollision() [3/3]

```
void SG.SG_HandRigidBodies.SetIgnoreCollision (
    SG_HandRigidBodies otherLayer,
    bool ignoreCollision )
```

Set ignoreCollision between this layer and another set of rigidbodies.

Parameters

<i>otherLayer</i>	
<i>ignoreCollision</i>	

5.35.2.8 SetupSelf()

```
void SG.SG_HandRigidBodies.SetupSelf ( ) [protected]
```

Setup the tracking / parameters of this script's components.

5.35.3 Member Data Documentation**5.35.3.1 fingerObjs**

```
SG_TrackedBody [ ] SG.SG_HandRigidBodies.fingerObjs = new SG_TrackedBody[0]
```

The managed rigidbody of the fingers, from thumb to pinky.

5.35.3.2 handModel

```
SG_HandModelInfo SG.SG_HandRigidBodies.handModel
```

The hand model information, used to assign tracking information. If left unassigned, you'll need to assign them manually.

5.35.3.3 wristObj

`SG_TrackedBody` `SG.SG_HandRigidBodies.wristObj`

The managed rigidbody of the wrist

5.35.4 Property Documentation

5.35.4.1 CollisionsEnabled

`bool` `SG.SG_HandRigidBodies.CollisionsEnabled` `[set]`

Enable/Disable the overall collision of the rigidbodies in this layer.

5.35.4.2 DebugEnabled

`bool` `SG.SG_HandRigidBodies.DebugEnabled` `[set]`

Show/Hide the the rigidbodies in this layer.

5.35.4.3 Hand

`SG_TrackedHand` `SG.SG_HandRigidBodies.Hand` `[get]`, `[protected set]`

The TrackedHand this Animator takes its data from, used to access grabscript, hardware, etc.

5.35.4.4 HardwareReady

`virtual bool` `SG.SG_HandRigidBodies.HardwareReady` `[get]`

Returns true if this Animator is connected to Hardware that is ready to go

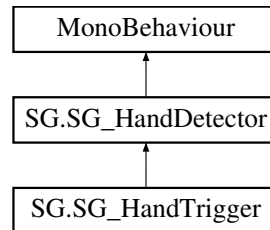
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_HandRigidBodies.cs`

5.36 SG.SG_HandTrigger Class Reference

A Detector that, when activated, triggers a series of in-game effects.

Inheritance diagram for SG.SG_HandTrigger:



Public Member Functions

- bool [InUse](#) ()
Check if the trigger is in use by one or more sense gloves.
- void **SetAudio** (bool play)
- void [SetParticles](#) (bool play)
Start / Stop the particleeffect
- void [SetEffectObject](#) (bool active)
Enable/disable the "effectToShow" Gameobject

Public Attributes

- ParticleSystem [particlesToPlay](#)
Particle effects that are shown when the glove is detected
- AudioSource [audioToPlay](#)
(Optional) Audio to play if the glove is detected
- GameObject [effectToShow](#)
(group of) game objects to show when the glove is detected.
- bool [hapticFeedback](#) = false
(Optional) tells the glove to give haptic feedback
- int [hapticForce](#) = 100
The magnitude of the haptic Feedback
- int [hapticDuration](#) = 200
The duration of a haptic feedback pulse.
- bool[] [whichFingers](#) = new bool[5] { true, false, false, false, false }
Which fingers to apply the Haptic feedback to
- bool [loop](#) = false
If set to true, the haptic feedback is continuous while the glove is inside the trigger

Protected Member Functions

- override void [FireDetectEvent](#) (SG_SenseGloveHardware model)
A step in between events that can be overridden by sub-classes of the SenseGlove_Detector
- override void [FireRemoveEvent](#) (SG_SenseGloveHardware model)
A step in between events that can be overridden by sub-classes of the SenseGlove_Detector
- override void **Start** ()
- virtual void **Update** ()

Private Member Functions

- void [FireHapticFeedback](#) (bool stopAll=false)
Fire the haptic feedback pulse or loop.

Private Attributes

- int [inUse](#) = 0
The amount of gloves that are using this trigger.
- float [buzz_CMD_Time](#) = 1
If loop is set to true, send a new command every X seconds.
- float [buzzTimer](#) = 0
Used to keep track of new buzz commands.

Additional Inherited Members

5.36.1 Detailed Description

A Detector that, when activated, triggers a series of in-game effects.

5.36.2 Member Function Documentation

5.36.2.1 FireDetectEvent()

```
override void SG.SG_HandTrigger.FireDetectEvent (
    SG_SenseGloveHardware model ) [protected], [virtual]
```

A step in between events that can be overridden by sub-classes of the SenseGlove_Detector

Parameters

<i>model</i>	
--------------	--

Reimplemented from [SG.SG_HandDetector](#).

5.36.2.2 FireHapticFeedback()

```
void SG.SG_HandTrigger.FireHapticFeedback (
    bool stopAll = false ) [private]
```

Fire the haptic feedback pulse or loop.

Parameters

<i>stopAll</i>	
----------------	--

5.36.2.3 FireRemoveEvent()

```
override void SG.SG_HandTrigger.FireRemoveEvent (
    SG_SenseGloveHardware model ) [protected], [virtual]
```

A step in between events that can be overridden by sub-classes of the SenseGlove_Detector

Parameters

<i>model</i>	
--------------	--

Reimplemented from [SG.SG_HandDetector](#).

5.36.2.4 InUse()

```
bool SG.SG_HandTrigger.InUse ( )
```

Check if the trigger is in use by one or more sense gloves.

Returns

5.36.2.5 SetEffectObject()

```
void SG.SG_HandTrigger.SetEffectObject (
    bool active )
```

Enable/disable the "effectToShow" Gameobject

Parameters

<i>active</i>	
---------------	--

5.36.2.6 SetParticles()

```
void SG.SG_HandTrigger.SetParticles (
    bool play )
```

Start / Stop the particleeffect

Parameters

<i>play</i>	
-------------	--

5.36.3 Member Data Documentation

5.36.3.1 audioToPlay

```
AudioSource SG.SG_HandTrigger.audioToPlay
```

(Optional) Audio to play if the glove is detected

5.36.3.2 buzz_CMD_Time

```
float SG.SG_HandTrigger.buzz_CMD_Time = 1 [private]
```

If loop is set to true, send a new command every X seconds.

5.36.3.3 buzzTimer

```
float SG.SG_HandTrigger.buzzTimer = 0 [private]
```

Used to keep track of new buzz commands.

5.36.3.4 effectToShow

```
GameObject SG.SG_HandTrigger.effectToShow
```

(group of) game objects to show when the glove is detected.

5.36.3.5 hapticDuration

```
int SG.SG_HandTrigger.hapticDuration = 200
```

The duration of a haptic feedback pulse.

5.36.3.6 hapticFeedback

```
bool SG.SG_HandTrigger.hapticFeedback = false
```

(Optional) tells the glove to give haptic feedback

5.36.3.7 hapticForce

```
int SG.SG_HandTrigger.hapticForce = 100
```

The magnitude of the haptic Feedback

5.36.3.8 inUse

```
int SG.SG_HandTrigger.inUse = 0 [private]
```

The amount of gloves that are using this trigger.

5.36.3.9 loop

```
bool SG.SG_HandTrigger.loop = false
```

If set to true, the haptic feedback is continuous while the glove is inside the trigger

5.36.3.10 particlesToPlay

```
ParticleSystem SG.SG_HandTrigger.particlesToPlay
```

Particle effects that are shown when the glove is detected

5.36.3.11 whichFingers

```
bool [] SG.SG_HandTrigger.whichFingers = new bool[5] { true, false, false, false, false }
```

Which fingers to apply the Haptic feedback to

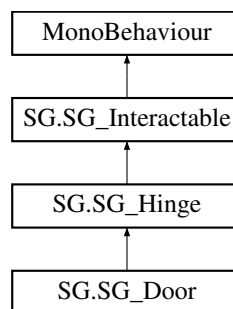
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_HandTrigger.cs

5.37 SG.SG_Hinge Class Reference

Represents an Interactable that can rotate around a specified point and axis. Used to extend doors and levers.

Inheritance diagram for SG.SG_Hinge:



Public Member Functions

- override void [UpdateInteraction](#) ()
Update the interaction with this Interactable.
- override void [SetInteractable](#) (bool interactable)
Set this drawer and its handles to active / inactive.
- void [SetupHinge](#) ()
Setup the hinge with the chosen options and verify them.
- void [StopPhysicsBody](#) ()
Stop the hinge body's movement before setting the angle(s)
- void [SetAngle](#) (float newAngle, bool freezeBody=false)
Set the hinge angle to the desired value (in degrees), using its localRotation.
- float [GetHingeAngle](#) ()
Retrieve the local rotation angle of the hingePoint
- float [GetHingeAngle](#) (Vector3 absPosition)
Retrieve the angle that the hinge should face to reach the chosen position.
- float [HingeRatio](#) ()
Retrieve the ratio (0 .. 1) of this Hinge Joint, from minAngle (0) to maxAngle (1). Used for events / animations.

Public Attributes

- Transform [hingePoint](#)
The point around which the hinge moves.
- [MovementAxis](#) [hingeAxis](#) = [MovementAxis.Y](#)
The axis of the hinge point around which the hinge moves.
- [HingeJoint](#) [joint](#)
The (optional) physics-based hinge joint that controls the hinge's movement when not interacting.
- bool [autoSetup](#) = true
Set to true if you want the Sense Glove to be automatically set up. False to stop the SenseGlove from messing with your script(s).
- int [minAngle](#) = -180
The minimum hinge angle, in degrees
- int [maxAngle](#) = 180
The maximum hinge angle, in degrees
- List< [SG_GrabZone](#) > [handles](#) = new List<[SG_GrabZone](#)>()
The handles connected to this Interactable.

Protected Member Functions

- virtual void **Awake** ()
- virtual void **Start** ()
- virtual void **Update** ()
- virtual void **FixedUpdate** ()
- override bool [InteractionBegin](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Begin the interaction with this Interactable
- override bool [InteractionEnd](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Ends the interaction between the grabscript and this hinge

Private Member Functions

- float [GetAngle](#) (Vector3 absPosition)
Calculate the angle of an absolute position relative to the hinge [Internal use]
- void [CheckLimits](#) ()
Check if the hinge is still within its working limits.
- Vector3 [RotationAxis](#) ()
Returns the (absolute) rotation axis of this hinge.

Private Attributes

- Rigidbody [physicsBody](#)
The (optional) rigidbody of the hinge that moves it around when not interacting.
- GameObject [grabReference](#)
The reference of the GrabScript that is holding this hinge
- float [offsetAngle](#) = 0
The offset
- bool [usedGravity](#) = false
Whether the hinge used gravity before interaction started.
- bool [wasKinematic](#) = true
Whether the hinge was kinematic before any interaction started.

Additional Inherited Members

5.37.1 Detailed Description

Represents an Interactable that can rotate around a specified point and axis. Used to extend doors and levers.

5.37.2 Member Function Documentation

5.37.2.1 CheckLimits()

```
void SG.SG_Hinge.CheckLimits ( ) [private]
```

Check if the hinge is still within its working limits.

5.37.2.2 GetAngle()

```
float SG.SG_Hinge.GetAngle (
    Vector3 absPosition ) [private]
```

Calculate the angle of an absolute position relative to the hinge [Internal use]

Parameters

<i>absPosition</i>	
--------------------	--

Returns

5.37.2.3 GetHingeAngle() [1/2]

```
float SG.SG_Hinge.GetHingeAngle ( )
```

Retrieve the local rotation angle of the hingePoint

Returns

5.37.2.4 GetHingeAngle() [2/2]

```
float SG.SG_Hinge.GetHingeAngle (
    Vector3 absPosition )
```

Retrieve the angle that the hinge should face to reach the chosen position.

Parameters

<i>absPosition</i>	
--------------------	--

Returns

5.37.2.5 HingeRatio()

```
float SG.SG_Hinge.HingeRatio ( )
```

Retrieve the ratio (0 .. 1) of this Hinge Joint, from minAngle (0) to maxAngle (1). Used for events / animations.

Returns

5.37.2.6 InteractionBegin()

```
override bool SG.SG_Hinge.InteractionBegin (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Begin the interaction with this Interactable

Parameters

<i>grabScript</i>	
-------------------	--

Implements [SG.SG_Interactable](#).

5.37.2.7 InteractionEnd()

```
override bool SG.SG_Hinge.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal = false ) [protected], [virtual]
```

Ends the interaction between the grabscript and this hinge

Parameters

<i>grabScript</i>	
-------------------	--

Implements [SG.SG_Interactive](#).

5.37.2.8 RotationAxis()

```
Vector3 SG.SG_Hinge.RotationAxis ( ) [private]
```

Returns the (absolute) rotation axis of this hinge.

Returns

5.37.2.9 SetAngle()

```
void SG.SG_Hinge.SetAngle (
    float newAngle,
    bool freezeBody = false )
```

Set the hinge angle to the desired value (in degrees), using its localRotation.

Parameters

<i>newAngle</i>	
-----------------	--

5.37.2.10 SetInteractive()

```
override void SG.SG_Hinge.SetInteractive (
    bool interactable ) [virtual]
```

Set this drawer and its handles to active / inactive.

Parameters

<i>interactable</i>	
---------------------	--

Reimplemented from [SG.SG_Interactive](#).

5.37.2.11 SetupHinge()

```
void SG.SG_Hinge.SetupHinge ( )
```

Setup the hinge with the chosen options and verify them.

5.37.2.12 StopPhysicsBody()

```
void SG.SG_Hinge.StopPhysicsBody ( )
```

Stop the hinge body's movement before setting the angle(s)

5.37.2.13 UpdateInteraction()

```
override void SG.SG_Hinge.UpdateInteraction ( ) [virtual]
```

Update the interaction with this Interactable.

Reimplemented from [SG.SG_Interactable](#).

5.37.3 Member Data Documentation

5.37.3.1 autoSetup

```
bool SG.SG_Hinge.autoSetup = true
```

Set to true if you want the Sense Glove to be automatically set up. False to stop the SenseGlove from messing with your script(s).

5.37.3.2 grabReference

```
GameObject SG.SG_Hinge.grabReference [private]
```

The reference of the GrabScript that is holding this hinge

5.37.3.3 handles

```
List<SG_GrabZone> SG.SG_Hinge.handles = new List<SG_GrabZone>()
```

The handles connected to this Interactable.

5.37.3.4 hingeAxis

```
MovementAxis SG.SG_Hinge.hingeAxis = MovementAxis.Y
```

The axis of the hinge point around which the hinge moves.

5.37.3.5 hingePoint

```
Transform SG.SG_Hinge.hingePoint
```

The point around which the hinge moves.

5.37.3.6 joint

```
HingeJoint SG.SG_Hinge.joint
```

The (optional) physics-based hinge joint that controls the hinge's movement when not interacting.

5.37.3.7 maxAngle

```
int SG.SG_Hinge.maxAngle = 180
```

The maximum hinge angle, in degrees

5.37.3.8 minAngle

```
int SG.SG_Hinge.minAngle = -180
```

The minimum hinge angle, in degrees

5.37.3.9 offsetAngle

```
float SG.SG_Hinge.offsetAngle = 0 [private]
```

The offset

The offset angle between the grabreference and the hinge (handle)

5.37.3.10 physicsBody

```
Rigidbody SG.SG_Hinge.physicsBody [private]
```

The (optional) rigidbody of the hinge that moves it around when not interacting.

5.37.3.11 usedGravity

```
bool SG.SG_Hinge.usedGravity = false [private]
```

Whether the hinge used gravity before interaction started.

5.37.3.12 wasKinematic

```
bool SG.SG_Hinge.wasKinematic = true [private]
```

Whether the hinge was kinematic before any interaction started.

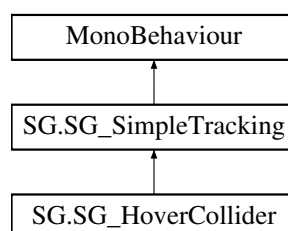
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Hinge.cs

5.38 SG.SG_HoverCollider Class Reference

A script that keeps track of multiple [SG_Interactive](#) objects it collides with.

Inheritance diagram for SG.SG_HoverCollider:



Public Member Functions

- bool [IsTouching](#) ()
Return true if this script is touching an object
- bool [IsTouching](#) (GameObject obj)
Returns true if this script is touching a specific GameObject.
- bool [IsTouching](#) (SG_Interactable interactable)
Returns true if this script is touching a specific SG_Interactable.
- [SG_Interactable\[\] MatchingObjects](#) (SG_HoverCollider other)
Returns a list of interactables that are touched by both this hoverCollider and another hoverCollider
- void [ClearTouchedObjects](#) ()
Clear this scripts references to other scripts.

Static Public Member Functions

- static bool [GetInteractableScript](#) (Collider col, out [SG_Interactable](#) interactable, bool favourSpecific=true)
Retrieve a SG_Interactable object from a collider. Returns true if one is found.
- static bool [SameScript](#) (Collider col, [SG_Interactable](#) touchedScript)
Checks if a collider is connected to a specific touchedScript

Protected Member Functions

- int [ListIndex](#) ([SG_Interactable](#) iScript)
Returns the index of an SG_Interactable in this script's touchedObjects.
- void [AddToList](#) ([SG_Interactable](#) script)
Add a new (collider of) an SG_Interactable script to this script's touchedObjects
- void [RemoveFromList](#) (Collider col)
Remove a collider from this script's touchedObjects
- override void **Awake** ()
- virtual void **OnTriggerEnter** (Collider other)
- virtual void **OnTriggerExit** (Collider other)

Protected Attributes

- List< [SG_Interactable](#) > [interactablesTouched](#) = new List<[SG_Interactable](#)>()
The list of interactables that are currently being touched.
- List< int > [collidersInside](#) = new List<int>()
The number of colliders for each interactable that this script is touching.

Properties

- [SG_Interactable\[\] TouchedObjects](#) [get]
The interactable objects that this script is currently touching

Additional Inherited Members

5.38.1 Detailed Description

A script that keeps track of multiple [SG_Interactable](#) objects it collides with.

5.38.2 Member Function Documentation

5.38.2.1 AddToList()

```
void SG.SG_HoverCollider.AddToList (
    SG_Interactive script ) [protected]
```

Add a new (collider of) an [SG_Interactive](#) script to this script's touchedObjects

Parameters

<i>script</i>	
---------------	--

5.38.2.2 ClearTouchedObjects()

```
void SG.SG_HoverCollider.ClearTouchedObjects ( )
```

Clear this scripts references to other scripts.

5.38.2.3 GetInteractiveScript()

```
static bool SG.SG_HoverCollider.GetInteractiveScript (
    Collider col,
    out SG_Interactive interactable,
    bool favourSpecific = true ) [static]
```

Retrieve a [SG_Interactive](#) object from a collider. Returns true if one is found.

Parameters

<i>col</i>	
<i>interactable</i>	
<i>favourSpecific</i>	

Returns

5.38.2.4 IsTouching() [1/3]

```
bool SG.SG_HoverCollider.IsTouching ( )
```

Return true if this script is touching an object

Returns

5.38.2.5 IsTouching() [2/3]

```
bool SG.SG_HoverCollider.IsTouching (
    GameObject obj )
```

Returns true if this script is touching a specific GameObject.

Parameters

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

Returns

5.38.2.6 IsTouching() [3/3]

```
bool SG.SG_HoverCollider.IsTouching (
    SG\_Interactive interactable )
```

Returns true if this script is touching a specific [SG_Interactive](#).

Parameters

<i>interactable</i>	
---------------------	--

Returns

5.38.2.7 ListIndex()

```
int SG.SG_HoverCollider.ListIndex (
    SG_Interactive iScript ) [protected]
```

Returns the index of an [SG_Interactive](#) in this script's touchedObjects.

Parameters

<i>iScript</i>	
----------------	--

Returns

5.38.2.8 MatchingObjects()

```
SG_Interactive [ ] SG.SG_HoverCollider.MatchingObjects (
    SG_HoverCollider other )
```

Returns a list of interactables that are touched by both this hoverCollider and another hoverCollider

Parameters

<i>other</i>	
--------------	--

Returns

5.38.2.9 RemoveFromList()

```
void SG.SG_HoverCollider.RemoveFromList (
    Collider col ) [protected]
```

Remove a collider from this script's touchedObjects

Parameters

<i>col</i>	
------------	--

5.38.2.10 SameScript()

```
static bool SG.SG_HoverCollider.SameScript (
    Collider col,
    SG_Interactive touchedScript ) [static]
```

Checks if a collider is connected to a specific touchedScript

Parameters

<i>col</i>	
<i>touchedScript</i>	

Returns

5.38.3 Member Data Documentation

5.38.3.1 collidersInside

```
List<int> SG.SG_HoverCollider.collidersInside = new List<int>() [protected]
```

The number of colliders for each interactable that this script is touching.

5.38.3.2 interactablesTouched

```
List<SG_Interactive> SG.SG_HoverCollider.interactablesTouched = new List<SG_Interactive>()
[protected]
```

The list of interactables that are currently being touched.

5.38.4 Property Documentation

5.38.4.1 TouchedObjects

```
SG_Interactive [ ] SG.SG_HoverCollider.TouchedObjects [get]
```

The interactable objects that this script is currently touching

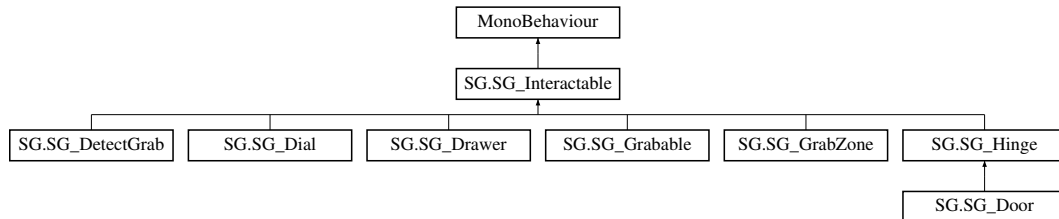
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Grabbing/SG_Hover↔ Collider.cs

5.39 SG.SG_Interactable Class Reference

Represents an object that a SenseGlove Grabsript can interact with. Extended by most of the Interaction scripts.

Inheritance diagram for SG.SG_Interactable:



Public Member Functions

- virtual void [SetInteractable](#) (bool canInteract)
Sets the object to be interactable (or not).
- virtual bool [CanInteract](#) ()
Check if this object can be interacted with at this moment.
- virtual bool [WithinBounds](#) ()
Check if this object is still within acceptable distance of the grabsript.
- virtual bool [MustBeReleased](#) ()
Returns true if this script is not longer active.
- virtual bool [EndInteractAllowed](#) ()
Check if this interactable allows a grabScript to end an interaction.
- bool [BeginInteraction](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
Begin the interaction between this object and a GrabScript.
- virtual void [EndInteraction](#) ()
(Manually) ends all interaction with this object's GrabScript(s)
- bool [EndInteraction](#) ([SG_GrabScript](#) grabScript, bool fromExternal=false)
(Manually) End the interaction with this GrabScript
- virtual void [UpdateInteraction](#) ()
Called by the grabsript after it has updated. Ensures that the FollowObject always updates last.
- void [TouchedBy](#) ([SG_BasicFeedback](#) touchScript)
Called by SG_Feedback when it touches an interactable. Informs this Interactable that it is being touched.
- void [UnTouchedBy](#) ([SG_BasicFeedback](#) touchScript)
Called by SG_Feedback when it touches an interactable. Informs this Interactable that it is no longer being touched
- virtual void [ResetObject](#) ()
Reset this object to its original state.
- virtual void [SaveTransform](#) ()
Save the current "state" of the interactable, to which it will return when ResetObject is called.
- virtual bool [InteractingWith](#) ([SG_GrabScript](#) grabScript)
Check if this Interactable is (already) interacting with a specified grabsript.
- virtual bool [IsInteracting](#) ()
Check if this object is being interacted with.
- delegate void [InteractBeginEventHandler](#) (object source, [SG_InteractArgs](#) args)
- delegate void [InteractEndEventHandler](#) (object source, [SG_InteractArgs](#) args)
- delegate void [ResetEventHandler](#) (object source, System.EventArgs args)
- delegate void [TouchedEventHandler](#) (object source, System.EventArgs args)

Public Attributes

- bool `fingerThumb` = true
This object can be picked up between a thumb and finger collider.
- bool `fingerPalm` = true
This object can be picked up between the palm collider and a finger (including the thumb).
- `ReleaseMethod releaseMethod` = `ReleaseMethod.Default`
Determines special conditions that must be fulfilled to release this object.

Protected Member Functions

- abstract bool `InteractionBegin` (`SG_GrabScript` grabScript, bool fromExternal)
Called when the Interaction begins on this Interactable.
- abstract bool `InteractionEnd` (`SG_GrabScript` grabScript, bool fromExternal)
Called when the Interaction ends on this Interactable.
- int `GetTouchIndex` (`SG_SenseGloveHardware` grabScript)
Get the index
- virtual void `OnInteractBegin` (`SG_GrabScript` grabScript, bool fromExternal)
- virtual void `OnInteractEnd` (`SG_GrabScript` grabScript, bool fromExternal)
- void `OnObjectReset` ()
- virtual void `OnTouched` ()
- virtual void `OnUntouched` ()

Protected Attributes

- bool `isInteractable` = true
Indicates if this object can be interacted with at this moment.
- float `releaseDistance` = 0.10f
Force then EndInteraction if the handModel ever passes more than this distance (in m) from the original grab location.
- `SG_GrabScript _grabScript`
A reference to the GrabScript that is currently interacting with this SenseGlove.
- Vector3 `originalPos`
The original (absolute) position of this GameObject, stored on Awake()
- Quaternion `originalRot`
The original (absolute) rotation of this GameObject, stored on Awake()
- float `originalDist`
The original distance between grabreference and my pickupReference.
- List< `SG_SenseGloveHardware` > `touchedScripts` = new List<`SG_SenseGloveHardware`>()
The list of touchScripts that are currently touching this object.
- List< int > `touchedColliders` = new List<int>()
The number of colliders of a given grabscript that are touching this Interactable.

Properties

- virtual `SG_GrabScript GrabScript` [get]
Access the grabscript that is currently interacting with this object.

Events

- InteractBeginEventHandler [InteractionBegun](#)
Fires after this interactable begins an interaction with a specific Grabscript.
- InteractEndEventHandler [InteractionEnded](#)
Fires after this interactable ends an interaction with a specific GrabScript.
- ResetEventHandler [ObjectReset](#)
Fires when this Object is reset to its original position.
- TouchedEventHandler [Touched](#)
Fires when this Interactable is first touched by a Sense Glove_Touch collider.
- TouchedEventHandler [UnTouched](#)
Fires when all colliders have stopped touching this Interactable.

5.39.1 Detailed Description

Represents an object that a SenseGlove Grabscript can interact with. Extended by most of the Interaction scripts.

5.39.2 Member Function Documentation

5.39.2.1 BeginInteraction()

```
bool SG.SG_Interactable.BeginInteraction (
    SG_GrabScript grabScript,
    bool fromExternal = false )
```

Begin the interaction between this object and a GrabScript.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

5.39.2.2 CanInteract()

```
virtual bool SG.SG_Interactable.CanInteract ( ) [virtual]
```

Check if this object can be interacted with at this moment.

May be overridden by sub-classes.

Returns

5.39.2.3 EndInteractAllowed()

```
virtual bool SG.SG_Interactable.EndInteractAllowed ( ) [virtual]
```

Check if this interactable allows a grabScript to end an interaction.

Returns

5.39.2.4 EndInteraction() [1/2]

```
virtual void SG.SG_Interactable.EndInteraction ( ) [virtual]
```

(Manually) ends all interaction with this object's GrabScript(s)

5.39.2.5 EndInteraction() [2/2]

```
bool SG.SG_Interactable.EndInteraction (
    SG_GrabScript grabScript,
    bool fromExternal = false )
```

(Manually) End the interaction with this GrabScript

Parameters

<i>fromExternal</i>	
<i>grabScript</i>	

5.39.2.6 GetTouchIndex()

```
int SG.SG_Interactable.GetTouchIndex (
    SG_SenseGloveHardware grabScript ) [protected]
```

Get the index

Parameters

<i>grabScript</i>	
-------------------	--

Returns

5.39.2.7 InteractingWith()

```
virtual bool SG.SG_Interactable.InteractingWith (
    SG_GrabScript grabScript ) [virtual]
```

Check if this Interactable is (already) interacting with a specified grabscript.

Parameters

<i>grabScript</i>	
-------------------	--

Returns

5.39.2.8 InteractionBegin()

```
abstract bool SG.SG_Interactable.InteractionBegin (
    SG_GrabScript grabScript,
    bool fromExternal ) [protected], [pure virtual]
```

Called when the Interaction begins on this Interactable.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Returns

True if a succesfull connection has been established.

Implemented in [SG.SG_Drawer](#), [SG.SG_Hinge](#), [SG.SG_Grabable](#), [SG.SG_GrabZone](#), [SG.SG_Dial](#), and [SG.SG_DetectGrab](#).

5.39.2.9 InteractionEnd()

```
abstract bool SG.SG_Interactable.InteractionEnd (
    SG_GrabScript grabScript,
    bool fromExternal ) [protected], [pure virtual]
```

Called when the Interaction ends on this Interactable.

Parameters

<i>grabScript</i>	
<i>fromExternal</i>	

Returns

True if the interaction has been ended.

Implemented in [SG.SG_Grabable](#), [SG.SG_Hinge](#), [SG.SG_Drawer](#), [SG.SG_GrabZone](#), [SG.SG_Dial](#), and [SG.SG_DetectGrab](#).

5.39.2.10 IsInteracting()

```
virtual bool SG.SG_Interactable.IsInteracting ( ) [virtual]
```

Check if this object is being interacted with.

Returns

5.39.2.11 MustBeReleased()

```
virtual bool SG.SG_Interactable.MustBeReleased ( ) [virtual]
```

Returns true if this script is not longer active.

Returns

5.39.2.12 ResetObject()

```
virtual void SG.SG_Interactable.ResetObject ( ) [virtual]
```

Reset this object to its original state.

Reimplemented in [SG.SG_Drawer](#), [SG.SG_Grabable](#), and [SG.SG_GrabZone](#).

5.39.2.13 SaveTransform()

```
virtual void SG.SG_Interactable.SaveTransform ( ) [virtual]
```

Save the current "state" of the interactable, to which it will return when ResetObject is called.

Reimplemented in [SG.SG_Drawer](#), [SG.SG_Grabable](#), and [SG.SG_GrabZone](#).

5.39.2.14 SetInteractable()

```
virtual void SG.SG_Interactable.SetInteractable (
    bool canInteract ) [virtual]
```

Sets the object to be interactable (or not).

May be overridden by sub-classes.

Parameters

<i>canInteract</i>	
--------------------	--

Reimplemented in [SG.SG_Hinge](#), and [SG.SG_Drawer](#).

5.39.2.15 TouchedBy()

```
void SG.SG_Interactable.TouchedBy (
    SG\_BasicFeedback touchScript )
```

Called by SG_Feedback when it touches an interactable. Informs this Interactable that it is being touched.

Parameters

<i>touchScript</i>	
--------------------	--

5.39.2.16 UnTouchedBy()

```
void SG.SG_Interactable.UnTouchedBy (
    SG\_BasicFeedback touchScript )
```

Called by SG_Feedback when it touches an interactable. Informs this Interactable that it is no longer being touched

Parameters

<i>touchScript</i>	
--------------------	--

5.39.2.17 UpdateInteraction()

```
virtual void SG.SG_Interactable.UpdateInteraction ( ) [virtual]
```

Called by the grabscript after it has updated. Ensures that the FollowObject always updates last.

Reimplemented in [SG.SG_Drawer](#), [SG.SG_Grabable](#), [SG.SG_Hinge](#), [SG.SG_GrabZone](#), and [SG.SG_Dial](#).

5.39.2.18 WithinBounds()

```
virtual bool SG.SG_Interactable.WithinBounds ( ) [virtual]
```

Check if this object is still within acceptable distance of the grabscript.

5.39.3 Member Data Documentation**5.39.3.1 _grabScript**

```
SG\_GrabScript SG.SG_Interactable._grabScript [protected]
```

A reference to the GrabScript that is currently interacting with this SenseGlove.

5.39.3.2 fingerPalm

```
bool SG.SG_Interactable.fingerPalm = true
```

This object can be picked up between the palm collider and a finger (including the thumb).

5.39.3.3 fingerThumb

```
bool SG.SG_Interactable.fingerThumb = true
```

This object can be picked up between a thumb and finger collider.

5.39.3.4 isInteractable

```
bool SG.SG_Interactable.isInteractable = true [protected]
```

Indicates if this object can be interacted with at this moment.

5.39.3.5 originalDist

```
float SG.SG_Interactable.originalDist [protected]
```

The original distance between grabrefrence and my pickupRefrence.

5.39.3.6 originalPos

```
Vector3 SG.SG_Interactable.originalPos [protected]
```

The original (absolute) position of this GameObject, stored on Awake()

5.39.3.7 originalRot

```
Quaternion SG.SG_Interactable.originalRot [protected]
```

The original (absolute) rotation of this GameObject, stored on Awake()

5.39.3.8 releaseDistance

```
float SG.SG_Interactable.releaseDistance = 0.10f [protected]
```

Force then EndInteraction if the handModel ever passes more than this distance (in m) from the original grab location.

Mostly relevant for drawers and levers, or other controls that move along a specific path.

5.39.3.9 releaseMethod

```
ReleaseMethod SG.SG_Interactable.releaseMethod = ReleaseMethod.Default
```

Determines special conditions that must be fulfilled to release this object.

5.39.3.10 touchedColliders

```
List<int> SG.SG_Interactable.touchedColliders = new List<int>() [protected]
```

The number of colliders of a given grabscript that are touching this Interactable.

5.39.3.11 touchedScripts

```
List<SG_SenseGloveHardware> SG.SG_Interactable.touchedScripts = new List<SG_SenseGloveHardware>() [protected]
```

The list of touchScripts that are currently touching this object.

5.39.4 Property Documentation

5.39.4.1 GrabScript

```
virtual SG_GrabScript SG.SG_Interactable.GrabScript [get]
```

Access the grabscript that is currently interacting with this object.

Returns

5.39.5 Event Documentation

5.39.5.1 InteractionBegun

```
InteractBeginEventHandler SG.SG_Interactable.InteractionBegun
```

Fires after this interactable begins an interaction with a specific Grabscript.

5.39.5.2 InteractionEnded

```
InteractEndEventHandler SG.SG_Interactable.InteractionEnded
```

Fires after this interactable ends an interaction with a specific GrabScript.

5.39.5.3 ObjectReset

`ResetEventHandler SG.SG_Interactable.ObjectReset`

Fires when this Object is reset to its original position.

5.39.5.4 Touched

`TouchedEventHandler SG.SG_Interactable.Touched`

Fires when this Interactable is first touched by a Sense Glove_Touch collider.

5.39.5.5 UnTouched

`TouchedEventHandler SG.SG_Interactable.UnTouched`

Fires when all colliders have stopped touching this Interactable.

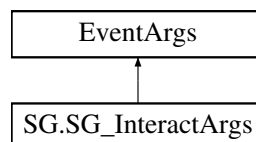
The documentation for this class was generated from the following file:

- [D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Interactable.cs](#)

5.40 SG.SG_InteractArgs Class Reference

Contains event arguments

Inheritance diagram for SG.SG_InteractArgs:



Public Member Functions

- **SG_InteractArgs** ([SG_GrabScript](#) script, bool fromExternal)

Properties

- [SG_GrabScript](#) **GrabScript** [get, private set]
- bool **Forced** [get, private set]

5.40.1 Detailed Description

Contains event arguments

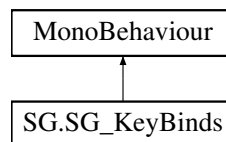
The documentation for this class was generated from the following file:

- [D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Interaction/SG_Interactable.cs](#)↔

5.41 SG.SG_KeyBinds Class Reference

A Keybinds component that can be attached to a TrackedHand so we may access certain functions through buttons or hotkeys.

Inheritance diagram for SG.SG_KeyBinds:



Public Member Functions

- void **LinkScripts** ()
- void **TryCalibrateWrist** ()
- void **TryManualRelease** ()

Public Attributes

- [SG_TrackedHand](#) **senseGloveHand**
- KeyCode **calibrateWristKey** = KeyCode.P
- KeyCode **releaseObjectKey** = KeyCode.E

Protected Member Functions

- void **Start** ()
- void **Update** ()

5.41.1 Detailed Description

A Keybinds component that can be attached to a TrackedHand so we may access certain functions through buttons or hotkeys.

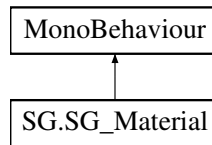
The documentation for this class was generated from the following file:

- [D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_KeyBinds.cs](#)

5.42 SG.SG_Material Class Reference

A class that contains material properties for a virtual objects, which can be customized, hard-coded or loaded during runtime.

Inheritance diagram for SG.SG_Material:



Public Member Functions

- bool [IsBroken](#) ()
Check if this material is broken
- void [UnBreak](#) ()
Unbreak the material, allowing it to give feedback and raise the break event again.
- int [CalculateForce](#) (float displacement, int fingerIndex)
Calculates the force on the finger based on material properties.
- int [CalculateHaptics](#) ()
Calculate the haptic pulse based on material properties.
- void [LoadMaterialProps](#) (SG.Materials.VirtualMaterial ofMaterial)
Load the hard-coded properties of the material
- delegate void **MaterialBreaksEventHandler** (object source, System.EventArgs args)

Static Public Member Functions

- static int [CalculateResponseForce](#) (float disp, int [maxForce](#), float [maxForceDist](#))
The actual method to calculate things, used by both default and custom materials.

Public Attributes

- [SG.Materials.VirtualMaterial](#) [material](#) = [SG.Materials.VirtualMaterial.Custom](#)
The material-type of the SenseGlove_Material.
- int [maxForce](#) = 100
The maximum brake force [0..100%] that the material provides at maxForceDist.
- float [maxForceDist](#) = 0.00f
The distance [in m] before the maximum force is reached.
- float [yieldDistance](#) = 0.03f
The distance [in m] before the material calls an OnBreak event.
- bool [hapticFeedback](#) = false
Whether or not the material should give any haptic feedback through the buzzMotors.
- int [hapticMagnitude](#) = 100
The magnitude of the haptic pulse [0..100%]
- int [hapticDuration](#) = 100
(maximum) duration in ms of the haptic pulse
- bool [breakable](#) = false

- Indicates that this material can raise an OnBreak event.*
 - bool `mustBeGrabbed` = false
this object must first be picked up before it can be broken.
 - bool `requiresThumb` = false
This object must be crushed by the thumb before it can be broken
 - int `minimumFingers` = 1
The minimum amount of fingers (not thumb) that 'break' this object before it actually breaks.

Protected Member Functions

- void `OnMaterialBreak` ()
- virtual void `Start` ()
- virtual void `OnDisable` ()
Unbreak this material if it is disabled.

Protected Attributes

- `SG_MeshDeform deformScript`
(Optional) Connected Material Deformation Script, used to pass deformation paraeters?

Events

- MaterialBreaksEventHandler `MaterialBreaks`
Fires when the material breaks under the conditions set through the Material Properties.

Private Member Functions

- void `LoadMaterialProps` (`SG.Materials.MaterialProps` props)
Actually apply materialProps to this Material.

Private Attributes

- bool `isBroken` = false
Check whether or not this object is broken.
- `SG_Interactable myInteractable`
My (optional) interactable script
- bool[] `raisedBreak` = new bool[5]
[thumb/palm, index, middle, pinky, ring]
- int `brokenBy` = 0
How many fingers [not thumb] have raised break events.

5.42.1 Detailed Description

A class that contains material properties for a virtual objects, which can be customized, hard-coded or loaded during runtime.

5.42.2 Member Function Documentation

5.42.2.1 CalculateForce()

```
int SG.SG_Material.CalculateForce (
    float displacement,
    int fingerIndex )
```

Calculates the force on the finger based on material properties.

Parameters

<i>displacement</i>	
<i>fingerIndex</i>	

Returns

5.42.2.2 CalculateHaptics()

```
int SG.SG_Material.CalculateHaptics ( )
```

Calculate the haptic pulse based on material properties.

Returns

5.42.2.3 CalculateResponseForce()

```
static int SG.SG_Material.CalculateResponseForce (
    float disp,
    int maxForce,
    float maxForceDist ) [static]
```

The actual method to calculate things, used by both default and custom materials.

Returns

5.42.2.4 IsBroken()

```
bool SG.SG_Material.IsBroken ( )
```

Check if this material is broken

Returns

5.42.2.5 LoadMaterialProps() [1/2]

```
void SG.SG_Material.LoadMaterialProps (
    SG.Materials.MaterialProps props ) [private]
```

Actually apply materialProps to this Material.

Parameters

<i>props</i>	
--------------	--

5.42.2.6 LoadMaterialProps() [2/2]

```
void SG.SG_Material.LoadMaterialProps (
    SG.Materials.VirtualMaterial ofMaterial )
```

Load the hard-coded properties of the material

Parameters

<i>ofMaterial</i>	
-------------------	--

5.42.2.7 OnDisable()

```
virtual void SG.SG_Material.OnDisable ( ) [protected], [virtual]
```

Unbreak this material if it is disabled.

5.42.2.8 UnBreak()

```
void SG.SG_Material.UnBreak ( )
```

Unbreak the material, allowing it to give feedback and raise the break event again.

5.42.3 Member Data Documentation

5.42.3.1 breakable

```
bool SG.SG_Material.breakable = false
```

Indicates that this material can raise an OnBreak event.

5.42.3.2 brokenBy

```
int SG.SG_Material.brokenBy = 0 [private]
```

How many fingers [not thumb] have raised break events.

5.42.3.3 deformScript

```
SG_MeshDeform SG.SG_Material.deformScript [protected]
```

(Optional) Connected Material Deformation Script, used to pass deformation paraeters?

5.42.3.4 hapticDuration

```
int SG.SG_Material.hapticDuration = 100
```

(maximum) duration in ms of the haptic pulse

5.42.3.5 hapticFeedback

```
bool SG.SG_Material.hapticFeedback = false
```

Whether or not the material should give any haptic feedback through the buzzMotors.

5.42.3.6 hapticMagnitude

```
int SG.SG_Material.hapticMagnitude = 100
```

The magnitude of the haptic pulse [0..100%]

5.42.3.7 isBroken

```
bool SG.SG_Material.isBroken = false [private]
```

Check whether or not this object is broken.

5.42.3.8 material

```
SG.Materials.VirtualMaterial SG.SG_Material.material = SG.Materials.VirtualMaterial.Custom
```

The material-type of the SenseGlove_Material.

5.42.3.9 maxForce

```
int SG.SG_Material.maxForce = 100
```

The maximum brake force [0..100%] that the material provides at maxForceDist.

5.42.3.10 maxForceDist

```
float SG.SG_Material.maxForceDist = 0.00f
```

The distance [in m] before the maximum force is reached.

5.42.3.11 minimumFingers

```
int SG.SG_Material.minimumFingers = 1
```

The minimum amount of fingers (not thumb) that 'break' this object before it actually breaks.

5.42.3.12 mustBeGrabbed

```
bool SG.SG_Material.mustBeGrabbed = false
```

this object must first be picked up before it can be broken.

5.42.3.13 myInteractable

```
SG_Interactable SG.SG_Material.myInteractable [private]
```

My (optional) interactable script

5.42.3.14 raisedBreak

```
bool [ ] SG.SG_Material.raisedBreak = new bool[5] [private]
```

[thumb/palm, index, middle, pinky, ring]

5.42.3.15 requiresThumb

```
bool SG.SG_Material.requiresThumb = false
```

This object must be crushed by the thumb before it can be broken

5.42.3.16 yieldDistance

```
float SG.SG_Material.yieldDistance = 0.03f
```

The distance [in m] before the material calls an OnBreak event.

5.42.4 Event Documentation

5.42.4.1 MaterialBreaks

```
MaterialBreaksEventHandler SG.SG_Material.MaterialBreaks
```

Fires when the material breaks under the conditions set through the Material Properties.

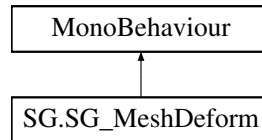
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_Material.cs

5.43 SG.SG_MeshDeform Class Reference

A class that can hook itself up to a [SG_Interactive](#) or material, and deform its mesh.

Inheritance diagram for SG.SG_MeshDeform:



Public Member Functions

- bool [SameVertex](#) (Vector3 v1, Vector3 v2)
Check if one Vertex equals another
- void [AddDeformation](#) (Vector3 absEntryVector, Vector3 absDeformPoint, float dist)
Add a deformation to calculate at the end of the fixedUpdate function.
- void [ResetMesh](#) ()
Reset the points in the mesh to their original vertices.

Public Attributes

- MeshFilter [meshFilter](#)
Will be used to extract the Mesh variable without exposing it to other classes.
- [SG.Materials.DisplaceType](#) [displaceType](#) = [SG.Materials.DisplaceType.Plane](#)
Determines how the Vertices respond to the collider(s)
- float [maxDisplacement](#) = 0.01f
The Maximum that a vertex can displace from its original position

Protected Member Functions

- void [SetDeform](#) (bool meshDeforms)
Enable / Disable mesh deformation of this script. Default set to true.
- void [CollectMeshData](#) ()
Collect the Mesh Data and find its unique vertices.
- void [AddDeform](#) (Vector3 absEntryVector, Vector3 absDeformPoint, float dist)
Add a single Deformation to the queue
- void [RemoveDeform](#) (int index)
Remove a deformation from the queue
- void [ClearDeformations](#) ()
Clear the list of deforms after everything;s been applied.
- void [ResetPoints](#) (bool resetAll)
Reset all (unique) vertices.
- void [DeformMesh](#) (Vector3 absEntryVector, Vector3 absDeformPoint)
Actually deform the mesh
- void [UpdatePoint](#) (int uniqueVertIndex, Vector3 newPos)
Update a vertex in the uniqueVertices array, and its associated sameVertices.
- void [UpdateMesh](#) ()
Apply all deformation in the Queue
- virtual void **Start** ()
- virtual void **FixedUpdate** ()
- virtual void **OnDisable** ()

Protected Attributes

- Mesh [myMesh](#)
The actual Mesh to manipulate.
- Vector3[] [verts](#)
The original vertices of the mesh, used for Deformation Logic
- Vector3[] [deformVerts](#)
The deformed mesh vertices, which are used to update the Mesh
- bool [atRest](#) = true
Indicated that the Mesh should be defroming. No need to recalculate unless they are being touched by a Feedback Collider.
- int[] [uniqueVertices](#)
The indices (in myMesh.vertices) that represent points that may be shared with others.
- int[][] [sameVertices](#)
The points shared by the Vertices at each indes of uniqueVertices.
- List< [SG.Materials.Deformation](#) > [deformationQueue](#) = new List<[SG.Materials.Deformation](#)>()
The queue of deformations that will be aplied during the next update frame.
- bool [deforms](#) = true
Used to enable/disable the mesh deformation.

5.43.1 Detailed Description

A class that can hook itself up to a [SG_Interactable](#) or material, and deform its mesh.

5.43.2 Member Function Documentation

5.43.2.1 AddDeform()

```
void SG.SG_MeshDeform.AddDeform (
    Vector3 absEntryVector,
    Vector3 absDeformPoint,
    float dist ) [protected]
```

Add a single Deformation to the queue

Parameters

<i>absEntryVector</i>	
<i>absDeformPoint</i>	
<i>dist</i>	

5.43.2.2 AddDeformation()

```
void SG.SG_MeshDeform.AddDeformation (
```

```
Vector3 absEntryVector,  
Vector3 absDeformPoint,  
float dist )
```

Add a deformation to calculate at the end of the fixedUpdate function.

Parameters

<i>absEntryVector</i>	
<i>absDeformPoint</i>	

5.43.2.3 ClearDeformations()

```
void SG.SG_MeshDeform.ClearDeformations ( ) [protected]
```

Clear the list of deforms after everything;s been applied.

5.43.2.4 CollectMeshData()

```
void SG.SG_MeshDeform.CollectMeshData ( ) [protected]
```

Collect the Mesh Data and find its unique vertices.

Placed in a separate function so one can re-analyze the mesh data on the fly.

5.43.2.5 DeformMesh()

```
void SG.SG_MeshDeform.DeformMesh (  
    Vector3 absEntryVector,  
    Vector3 absDeformPoint ) [protected]
```

Actually deform the mesh

Parameters

<i>absEntryVector</i>	
<i>absDeformPoint</i>	

5.43.2.6 RemoveDeform()

```
void SG.SG_MeshDeform.RemoveDeform (  
    int index ) [protected]
```

Remove a deformation from the queue

Parameters

<i>index</i>	
--------------	--

5.43.2.7 ResetMesh()

```
void SG.SG_MeshDeform.ResetMesh ( )
```

Reset the points in the mesh to their original vertices.

5.43.2.8 ResetPoints()

```
void SG.SG_MeshDeform.ResetPoints (
    bool resetAll ) [protected]
```

Reset all (unique) vertices.

Parameters

<i>resetAll</i>	Set to true to reset all points, set to false to reset only the uniqueVertices (saves time)
-----------------	---

5.43.2.9 SameVertex()

```
bool SG.SG_MeshDeform.SameVertex (
    Vector3 v1,
    Vector3 v2 )
```

Check if one Vertex equals another

Parameters

<i>v1</i>	
<i>v2</i>	

Returns

5.43.2.10 SetDeform()

```
void SG.SG_MeshDeform.SetDeform (
    bool meshDeforms ) [protected]
```

Enable / Disable mesh deformation of this script. Default set to true.

Parameters

<i>meshDeforms</i>	
--------------------	--

5.43.2.11 UpdateMesh()

```
void SG.SG_MeshDeform.UpdateMesh ( ) [protected]
```

Apply all deformation in the Queue

5.43.2.12 UpdatePoint()

```
void SG.SG_MeshDeform.UpdatePoint (
    int uniqueVertIndex,
    Vector3 newPos ) [protected]
```

Update a vertex in the uniqueVertices array, and its associated sameVertices.

Parameters

<i>i</i>	
<i>newPos</i>	

5.43.3 Member Data Documentation

5.43.3.1 atRest

```
bool SG.SG_MeshDeform.atRest = true [protected]
```

Indicated that the Mesh should be defroming. No need to recalculate unless they are being touched by a Feedback Collider.

5.43.3.2 deformationQueue

```
List<SG.Materials.Deformation> SG.MeshDeform.deformationQueue = new List<SG.Materials.Deformation> ()  
[protected]
```

The queue of deformations that will be applied during the next update frame.

5.43.3.3 deforms

```
bool SG.MeshDeform.deforms = true [protected]
```

Used to enable/disable the mesh deformation.

5.43.3.4 deformVerts

```
Vector3 [] SG.MeshDeform.deformVerts [protected]
```

The deformed mesh vertices, which are used to update the Mesh

5.43.3.5 displaceType

```
SG.Materials.DisplaceType SG.MeshDeform.displaceType = SG.Materials.DisplaceType.Plane
```

Determines how the Vertices respond to the collider(s)

5.43.3.6 maxDisplacement

```
float SG.MeshDeform.maxDisplacement = 0.01f
```

The Maximum that a vertex can displace from its original position

5.43.3.7 meshFilter

```
MeshFilter SG.MeshDeform.meshFilter
```

Will be used to extract the Mesh variable without exposing it to other classes.

If no Mesh Filter is assigned via the inspector, the script will attempt to retrieve one from the GameObject it is attached to.

5.43.3.8 myMesh

`Mesh SG.SG_MeshDeform.myMesh [protected]`

The actual Mesh to manipulate.

5.43.3.9 sameVertices

`int [][] SG.SG_MeshDeform.sameVertices [protected]`

The points shared by the Vertices at each index of uniqueVertices.

5.43.3.10 uniqueVertices

`int [] SG.SG_MeshDeform.uniqueVertices [protected]`

The indices (in myMesh.vertices) that represent points that may be shared with others.

5.43.3.11 verts

`Vector3 [] SG.SG_MeshDeform.verts [protected]`

The original vertices of the mesh, used for Deformation Logic

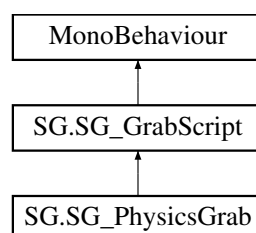
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Feedback/SG_MeshDeform.cs

5.44 SG.SG_PhysicsGrab Class Reference

A simplified version of the original SenseGlove_PhysGrab script; If an object is touched by finger-thumb or by palm-finger

Inheritance diagram for SG.SG_PhysicsGrab:



Public Member Functions

- override bool [CanInteract](#) ()
Retruns true if this GrabScript is ready to grab objects
- override bool [IsTouching](#) ()
Returns true if one of the fingers is touching an object
- override bool [Setup](#) ()
Called by [SG_GrabScript](#). Assign required variables
- override void [CheckForScripts](#) ()
Setup and check for connected scripts
- override void [UpdateGrabScript](#) ()
Grab new objects and release objects taht are no longer touched.
- [SG_Interactable\[\]](#) [GetMatching](#) (int finger1, int finger2)
Returns all grabables that both fingers are touching
- [SG_Interactable\[\]](#) [GetMatching](#) (int finger1, [SG_HoverCollider](#) touch)
Returns all grabables that both fingers are touchign
- void [CheckGestures](#) ()
Updates grab gestures specific to this script.
- List< [SG_Interactable](#) > [GetGrabables](#) ()
Returns a list of all Grabables that this script should be touching at this moment

Static Public Member Functions

- static bool [IsInside](#) ([SG_Interactable](#) heldObject, List< [SG_Interactable](#) > objectsToGrab)
Returns true if an [SG_Interactable](#) is inside a list of other [SG_Interactables](#)

Public Attributes

- [SG_HandModelInfo](#) handModel
The Hand Model info to which to link this script's colliders. If left unassigned, one needs to assign tracking to the colliders manually.
- [SG_HoverCollider](#) palmTouch
The Hand Palm collider, used when grabbing objects between the palm and finger (tool/handle grips)
- [SG_HoverCollider](#) thumbTouch
Thumb collider, used to determine finger/thumb collision
- [SG_HoverCollider](#) indexTouch
Index collider, used to determine finger/thumb and finger/palm collision
- [SG_HoverCollider](#) middleTouch
Index collider, used to determine finger/thumb and finger/palm collision

Protected Member Functions

- override bool [CanRelease](#) ([SG_Interactable](#) obj)
Returns true if this grabscript can release an object
- override void **Awake** ()

Protected Attributes

- `bool[] wantsGrab = new bool[3]`
Keeps track of the 'grabbing' pose of fingers
- `SG_HoverCollider[] touchScripts = new SG_HoverCollider[0]`
The touchscript collection that is easier to iterate through.

Static Protected Attributes

- `static float[] openHandThresholds = new float[5] { -20, -20, -20, -20, -90 }`
Above these flexions, the hand is considered 'open'
- `static float[] closedHandThresholds = new float[5] { -360, -360, -360, -360, -360 }`
below these flexions, the hand is considered 'open'

Properties

- override `bool DebugEnabled` [set]
Show / Hide the hover colliders of this script.

Additional Inherited Members

5.44.1 Detailed Description

A simplified version of the original SenseGlove_PhysGrab script; If an object is touched by finger-thumb or by palm-finger

5.44.2 Member Function Documentation

5.44.2.1 CanInteract()

```
override bool SG.SG_PhysicsGrab.CanInteract ( ) [virtual]
```

Retruns true if this GrabScript is ready to grab objects

Returns

Implements [SG.SG_GrabScript](#).

5.44.2.2 CanRelease()

```
override bool SG.SG_PhysicsGrab.CanRelease (
    SG\_Interactable obj ) [protected], [virtual]
```

Returns true if this grabscript can release an object

Parameters

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

Returns

Reimplemented from [SG.SG_GrabScript](#).

5.44.2.3 CheckForScripts()

```
override void SG.SG_PhysicsGrab.CheckForScripts ( ) [virtual]
```

Setup and check for connected scripts

Reimplemented from [SG.SG_GrabScript](#).

5.44.2.4 CheckGestures()

```
void SG.SG_PhysicsGrab.CheckGestures ( )
```

Updates grab gestures specific to this script.

5.44.2.5 GetGrabables()

```
List<SG\_Interactive> SG.SG_PhysicsGrab.GetGrabables ( )
```

Returns a list of all Grabables that this script should be touching at this moment

Returns

5.44.2.6 GetMatching() [1/2]

```
SG\_Interactive [ ] SG.SG_PhysicsGrab.GetMatching (
    int finger1,
    int finger2 )
```

Returns all grabables that both fingers are touching

Parameters

<i>finger1</i>	
<i>finger2</i>	

Returns

5.44.2.7 GetMatching() [2/2]

```
SG_Interactive [] SG.SG_PhysicsGrab.GetMatching (
    int finger1,
    SG_HoverCollider touch )
```

Returns all grabables that both fingers are touchign

Parameters

<i>finger1</i>	
<i>finger2</i>	

Returns

5.44.2.8 IsInside()

```
static bool SG.SG_PhysicsGrab.IsInside (
    SG_Interactive heldObject,
    List< SG_Interactive > objectsToGrab ) [static]
```

Returns true if an [SG_Interactive](#) is inside a list of other SG_Interactables

Parameters

<i>heldObject</i>	
<i>objectsToGrab</i>	

Returns

5.44.2.9 IsTouching()

```
override bool SG.SG_PhysicsGrab.IsTouching ( ) [virtual]
```

Returns true if one of the fingers is touching an object

Returns

Implements [SG.SG_GrabScript](#).

5.44.2.10 Setup()

```
override bool SG.SG_PhysicsGrab.Setup ( ) [virtual]
```

Called by [SG_GrabScript](#). Assign required variables

Returns

Implements [SG.SG_GrabScript](#).

5.44.2.11 UpdateGrabScript()

```
override void SG.SG_PhysicsGrab.UpdateGrabScript ( ) [virtual]
```

Grab new objects and release objects taht are no longer touched.

Implements [SG.SG_GrabScript](#).

5.44.3 Member Data Documentation

5.44.3.1 closedHandThresholds

```
float [ ] SG.SG_PhysicsGrab.closedHandThresholds = new float[5] { -360, -360, -360, -360, -360 } [static], [protected]
```

below these flexions, the hand is considered 'open'

5.44.3.2 handModel

`SG_HandModelInfo` `SG.SG_PhysicsGrab.handModel`

The Hand Model info to which to link this script's colliders. If left unassigned, one needs to assign tracking to the colliders manually.

5.44.3.3 indexTouch

`SG_HoverCollider` `SG.SG_PhysicsGrab.indexTouch`

Index collider, used to determine finger/thumb and finger/palm collision

5.44.3.4 middleTouch

`SG_HoverCollider` `SG.SG_PhysicsGrab.middleTouch`

Index collider, used to determine finger/thumb and finger/palm collision

5.44.3.5 openHandThresholds

```
float [ ] SG.SG_PhysicsGrab.openHandThresholds = new float[5] { -20, -20, -20, -20, -90 } [static],  
[protected]
```

Above these flexions, the hand is considered 'open'

5.44.3.6 palmTouch

`SG_HoverCollider` `SG.SG_PhysicsGrab.palmTouch`

The Hand Palm collider, used when grabbing objects between the palm and finger (tool/handle grips)

5.44.3.7 thumbTouch

`SG_HoverCollider` `SG.SG_PhysicsGrab.thumbTouch`

Thumb collider, used to determine finger/thumb collision

5.44.3.8 touchScripts

```
SG_HoverCollider [] SG.SG_PhysicsGrab.touchScripts = new SG_HoverCollider[0] [protected]
```

The touchscript collection that is easier to iterate through.

5.44.3.9 wantsGrab

```
bool [] SG.SG_PhysicsGrab.wantsGrab = new bool[3] [protected]
```

Keeps track of the 'grabbing' pose of fingers

5.44.4 Property Documentation

5.44.4.1 DebugEnabled

```
override bool SG.SG_PhysicsGrab.DebugEnabled [set]
```

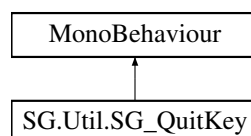
Show / Hide the hover colliders of this script.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Grabbing/SG_PhysicsGrab.cs

5.45 SG.Util.SG_QuitKey Class Reference

Inheritance diagram for SG.Util.SG_QuitKey:



Public Member Functions

- void **Quit** ()
- void **ResetScene** ()

Public Attributes

- KeyCode **exitKey** = KeyCode.None
- KeyCode **resetKey** = KeyCode.None

Private Member Functions

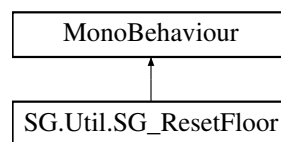
- void **Update** ()

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_QuitKey.cs

5.46 SG.Util.SG_ResetFloor Class Reference

Inheritance diagram for SG.Util.SG_ResetFloor:



Public Attributes

- string **resetTag** = "resetable"
- bool **resetEnabled** = true

Protected Member Functions

- void **CheckReset** (Collider other)

Private Member Functions

- void **Start** ()
- void **OnTriggerEnter** (Collider other)
- void **OnTriggerStay** (Collider other)

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_ResetFloor.cs

5.47 SG.SG_SenseGloveData Class Reference

Unity wrapper for the GloveData, which contains all a developer will need.

Public Member Functions

- [SG_SenseGloveData](#) (SenseGloveCs.GloveData data)
Extract right-handed coordinate system data from the SenseGlove DLL and convert it into Unity values.
- void [UpdateVariables](#) (SenseGloveCs.GloveData data)
Updates all variables that can change during the simulation.
- void [SetEmpty](#) ()
- Vector3[] [TotalGloveAngles](#) ()
Retrieve the total glove angles, used for gesture recognition (for each finger; pronation, abduction, flexion).
- float[] [GetFlexions](#) ()
- float[][] [GetFingerLengths](#) ()
Retrieve the finger lengths of this GloveData
- Vector3[] [GetJointPositions](#) ()
Retrieve the joint positions

Public Attributes

- bool [dataLoaded](#) = false
Determines if the glove-specific data has been loaded yet.
- [GloveSide](#) [gloveSide](#)
Check whether or not this is a left-handed or right-handed glove.
- string [deviceId](#)
The unique ID of this SenseGlove.
- string [gloveVersion](#)
The hardware version of this SenseGlove.
- float [firmwareVersion](#)
The version of the firmware that runs on this SenseGlove's Microcontroller
- float[][] [gloveValues](#)
The angles between glove segments, as calculated by the firmware. Sorted by finger, from proximal to distal.
- int [numberOfSensors](#)
The number of sensors on this Sense Glove.
- float[] [imuValues](#)
The raw x y z w values of the IMU within the SenseGlove.
- int[] [imuCalibration](#)
The IMU [Calibration](#) values for System, Gyro-, Accelero- and Magnetometer. These vary from -1 (N/A) and from 0 (not calibrated) to 3 (fully calibrated)
- int [packetsPerSecond](#) = 0
The amount of sensor packets the senseglove is sending to your system.
- Vector3 [commonOriginPos](#)
The position in mm of the common origin of the Hand and Glove, relative to the wrist.
- Quaternion [commonOriginRot](#)
The orientation of the common origin of the Hand and Glove, relative to the wrist.
- Vector3[][] [gloveAngles](#)
The euler angles between glove sections relative to its previous section, sorted by finger, from proximal to distal.
- Vector3[][] [gloveLengths](#)
The lengths of each glove section, sorted by finger, from proximal to distal.
- Vector3[][] [glovePositions](#)
The positions of the glove joints and thimble in mm, relative to the common origin. Sorted by finger, from proximal to distal.
- Quaternion[][] [gloveRotations](#)

The orientation of the glove joints and thimble, relative to the common origin. Sorted by finger, from proximal to distal.

- `Vector3[][]` [handAngles](#)
The euler angles [pronation/supination, abduction/adduction, flexion/extension] between finger joints relative to the previous bone, Sorted by finger, from proximal to distal.
- `Vector3[][]` [handLengths](#)
The lengths, in mm, of the finger phalanges. Sorted by finger, from proximal to distal.
- `Vector3[][]` [handPositions](#)
The positions of the hand joints fingertips, in mm, relative to the common origin. Sorted by finger, from proximal to distal.
- `Quaternion[][]` [handRotations](#)
The orientation of the hand joints and fingertips, relative to the common origin. Sorted by finger, from proximal to distal.
- `Quaternion` [absoluteWrist](#)
The absolute IMU orientation of the wrist.
- `Quaternion` [relativeWrist](#)
The wrist orientation relative to the foreArm.
- `Quaternion` [absoluteCalibratedWrist](#)
The absolute wrist angles, corrected with foreArm calibration.
- `int` [calibrationStep](#) = -1
The current step of the calibration algorithm.
- `int` [totalCalibrationSteps](#) = 0
The total number of steps of the calibration algorithm.

Protected Member Functions

- `SG_SenseGloveData ()`
Create an instance of [SG_SenseGloveData](#) with default values.

Static Protected Member Functions

- `static GloveSide GetSide (bool isRight)`
Retrieve the Glove Side of this Sense Glove.
- `static void GetChainVariables (ref SenseGloveCs.Kinematics.JointChain[] chains, ref Vector3[][] positions, ref Vector3[][] angles, ref Quaternion[][] rotations, ref Vector3[][] lengths)`
Fill a number of arrays with data from a single kinematic chain.
- `static void GetLinkVariables (ref SenseGloveCs.Kinematics.JointChain chain, ref Vector3[] positions, ref Vector3[] angles, ref Quaternion[] rotations, ref Vector3[] lengths)`
Fill the appropriate unity Quaternion and Vector3 arrays based on a single joining chain (finger or glove segment)

Properties

- `static SG_SenseGloveData Empty` `[get]`
Retrieve an unloaded set of data, which indicates that this glove has not been loaded yet.

5.47.1 Detailed Description

Unity wrapper for the GloveData, which contains all a developer will need.

5.47.2 Constructor & Destructor Documentation

5.47.2.1 SG_SenseGloveData() [1/2]

```
SG.SG_SenseGloveData.SG_SenseGloveData ( ) [protected]
```

Create an instance of [SG_SenseGloveData](#) with default values.

5.47.2.2 SG_SenseGloveData() [2/2]

```
SG.SG_SenseGloveData.SG_SenseGloveData (
    SenseGloveCs.GloveData data )
```

Extract right-handed coordinate system data from the SenseGlove DLL and convert it into Unity values.

Parameters

<i>data</i>	
<i>packets</i>	
<i>totalCSteps</i>	
<i>currCStep</i>	

5.47.3 Member Function Documentation

5.47.3.1 GetChainVariables()

```
static void SG.SG_SenseGloveData.GetChainVariables (
    ref SenseGloveCs.Kinematics.JointChain[] chains,
    ref Vector3 positions[][],
    ref Vector3 angles[][],
    ref Quaternion rotations[][],
    ref Vector3 lengths[][] ) [static], [protected]
```

Fill a number of arrays with data from a single kinematic chain.

Parameters

<i>chains</i>	
<i>positions</i>	
<i>angles</i>	
<i>rotations</i>	
<i>lengths</i>	

5.47.3.2 GetFingerLengths()

```
float [][] SG.SG_SenseGloveData.GetFingerLengths ( )
```

Retrieve the finger lengths of this GloveData

Returns

5.47.3.3 GetJointPositions()

```
Vector3 [] SG.SG_SenseGloveData.GetJointPositions ( )
```

Retrieve the joint positions

Returns

5.47.3.4 GetLinkVariables()

```
static void SG.SG_SenseGloveData.GetLinkVariables (
    ref SenseGloveCs.Kinematics.JointChain chain,
    ref Vector3[] positions,
    ref Vector3[] angles,
    ref Quaternion[] rotations,
    ref Vector3[] lengths ) [static], [protected]
```

Fill the appropriate unity Quaternion and Vector3 arrays based on a single joining chain (finger or glove segment)

Parameters

<i>chain</i>	
<i>positions</i>	
<i>angles</i>	
<i>rotations</i>	
<i>lengths</i>	

5.47.3.5 GetSide()

```
static GloveSide SG.SG_SenseGloveData.GetSide (
    bool isRight ) [static], [protected]
```

Retrieve the Glove Side of this Sense Glove.

Parameters

<i>isRight</i>	
----------------	--

Returns

5.47.3.6 TotalGloveAngles()

```
Vector3 [ ] SG.SG_SenseGloveData.TotalGloveAngles ( )
```

Retrieve the total glove angles, used for gesture recognition (for each finger; pronation, abduction, flexion).

Returns

5.47.3.7 UpdateVariables()

```
void SG.SG_SenseGloveData.UpdateVariables (
    SenseGloveCs.GloveData data )
```

Updates all variables that can change during the simulation.

Parameters

<i>data</i>	
-------------	--

5.47.4 Member Data Documentation

5.47.4.1 absoluteCalibratedWrist

```
Quaternion SG.SG_SenseGloveData.absoluteCalibratedWrist
```

The absolute wrist angles, corrected with foreArm calibration.

5.47.4.2 absoluteWrist

```
Quaternion SG.SG_SenseGloveData.absoluteWrist
```

The absolute IMU orientation of the wrist.

5.47.4.3 calibrationStep

```
int SG.SG_SenseGloveData.calibrationStep = -1
```

The current step of the calibration algorithm.

5.47.4.4 commonOriginPos

```
Vector3 SG.SG_SenseGloveData.commonOriginPos
```

The position in mm of the common origin of the Hand and Glove, relative to the wrist.

5.47.4.5 commonOriginRot

```
Quaternion SG.SG_SenseGloveData.commonOriginRot
```

The orientation of the common origin of the Hand and Glove, relative to the wrist.

5.47.4.6 dataLoaded

```
bool SG.SG_SenseGloveData.dataLoaded = false
```

Determines if the glove-specific data has been loaded yet.

5.47.4.7 deviceID

```
string SG.SG_SenseGloveData.deviceID
```

The unique ID of this SenseGlove.

5.47.4.8 firmwareVersion

```
float SG.SG_SenseGloveData.firmwareVersion
```

The version of the firmware that runs on this SenseGlove's Microcontroller

5.47.4.9 gloveAngles

```
Vector3 [][] SG.SG_SenseGloveData.gloveAngles
```

The euler angles between glove sections relative to its previous section, sorted by finger, from proximal to distal.

5.47.4.10 gloveLengths

```
Vector3 [][] SG.SG_SenseGloveData.gloveLengths
```

The lengths of each glove section, sorted by finger, from proximal to distal.

5.47.4.11 glovePositions

```
Vector3 [][] SG.SG_SenseGloveData.glovePositions
```

The positions of the glove joints and thimble in mm, relative to the common origin. Sorted by finger, from proximal to distal.

5.47.4.12 gloveRotations

```
Quaternion [][] SG.SG_SenseGloveData.gloveRotations
```

The orientation of the glove joints and thimble, relative to the common origin. Sorted by finger, from proximal to distal.

5.47.4.13 gloveSide

```
GloveSide SG.SG_SenseGloveData.gloveSide
```

Check whether or not this is a left-handed or right-handed glove.

5.47.4.14 gloveValues

```
float [][] SG.SG_SenseGloveData.gloveValues
```

The angles between glove segments, as calculated by the firmware. Sorted by finger, from proximal to distal.

5.47.4.15 gloveVersion

```
string SG.SG_SenseGloveData.gloveVersion
```

The hardware version of this SenseGlove.

5.47.4.16 handAngles

```
Vector3 [][] SG.SG_SenseGloveData.handAngles
```

The euler angles [pronation/supination, abduction/adduction, flexion/extension] between finger joints relative to the previous bone, Sorted by finger, from proximal to distal.

5.47.4.17 handLengths

```
Vector3 [][] SG.SG_SenseGloveData.handLengths
```

The lengths, in mm, of the finger phalanges. Sorted by finger, from proximal to distal.

5.47.4.18 handPositions

```
Vector3 [][] SG.SG_SenseGloveData.handPositions
```

The positions of the hand joints fingertips, in mm, relative to the common origin. Sorted by finger, from proximal to distal.

5.47.4.19 handRotations

```
Quaternion [][] SG.SG_SenseGloveData.handRotations
```

The orientation of the hand joints and fingertips, relative to the common origin. Sorted by finger, from proximal to distal.

5.47.4.20 imuCalibration

```
int [] SG.SG_SenseGloveData.imuCalibration
```

The IMU [Calibration](#) values for System, Gyro-, Accelerometer and Magnetometer. These vary from -1 (N/A) and from 0 (not calibrated) to 3 (fully calibrated)

5.47.4.21 imuValues

```
float [] SG.SG_SenseGloveData.imuValues
```

The raw x y z w values of the IMU within the SenseGlove.

5.47.4.22 numberOfSensors

```
int SG.SG_SenseGloveData.numberOfSensors
```

The number of sensors on this Sense Glove.

5.47.4.23 packetsPerSecond

```
int SG.SG_SenseGloveData.packetsPerSecond = 0
```

The amount of sensor packets the senseglove is sending to your system.

5.47.4.24 relativeWrist

```
Quaternion SG.SG_SenseGloveData.relativeWrist
```

The wrist orientation relative to the forearm.

5.47.4.25 totalCalibrationSteps

```
int SG.SG_SenseGloveData.totalCalibrationSteps = 0
```

The total number of steps of the calibration algorithm.

5.47.5 Property Documentation

5.47.5.1 Empty

`SG_SenseGloveData SG.SG_SenseGloveData.Empty [static], [get]`

Retrieve an unloaded set of data, which indicates that this glove has not been loaded yet.

Allows access to the empty Constructor without exposing it.

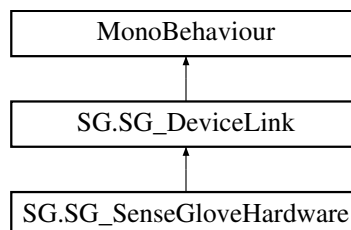
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_SenseGloveData.cs

5.48 SG.SG_SenseGloveHardware Class Reference

After being linked to a proper Sense Glove via the SenseGlove_DeviceManager, this script is responsible for updating `SG_SenseGloveData` every frame, and for exposing feedback - and calibration methods.

Inheritance diagram for SG.SG_SenseGloveHardware:



Classes

- class `BuzzCmd`
- class `GloveCalibrationArgs`

CalibrationArguments, containing both old and new finger lengths and joint positions.

Public Types

- enum `ConnectionMethod` { `ConnectionMethod.NextGlove` = 0, `ConnectionMethod.NextRightHand`, `ConnectionMethod.NextLeftHand` }
- enum `HapticSendMode` { `OnChange`, `OnFrame`, `Off`, `OnChangeRepeat` }

The way this object connects to SenseGlove_Objects detected on this system.

Public Member Functions

- delegate void [GloveEventHandler](#) (object source, System.EventArgs args)
Event delegate for the glove events event.
- delegate void [CalibrationFinishedEventHandler](#) (object source, [GloveCalibrationArgs](#) args)
Event delegate function for the CalibrateionFinished event.
- bool [HasFunction](#) (GloveFunctions function)
Verify if this SenseGlove has a particular functionality (buzz motors, haptic feedback, etc)
- bool [StopFeedback](#) ()
Stop all forms of feedback on this Sense Glove.
- bool [SendBrakeCmd](#) (int[] commands)
Send motor commands to the Sense Glove. summary>

Parameters

commands	
----------	--

Returns

- bool [SendBrakeCmd](#) (int thumbCmd, int indexCmd, int middleCmd, int ringCmd, int pinkyCmd)
Tell the Sense Glove to set its brakes at the desired magnitude [0..100%] for each finger until it receives a new command.
- bool [StopBrakes](#) ()
Release all brakes of the SenseGlove.
- bool [SendBuzzCmd](#) (bool[] fingers, int[] durations=null, int[] magnitudes=null, BuzzMotorPattern[] patterns=null)
Send a buzz-motor command to the Sense Glove, with optional parameters for each finger.
- bool [SendBuzzCmd](#) (bool[] fingers, int magnitude=100, int duration=400, BuzzMotorPattern pattern=BuzzMotorPattern.Constant)
Send one buzzmotor command to specific fingers, as indicated by the fingers array.
- bool [SendBuzzCmd](#) (int[] magnitudes, int duration)
Tell the Buzz motors to each vibrate on a different magnitude (0..100%) for a specific duration (ms)
- bool [SendBuzzCmd](#) (Finger finger, int magnitude, int duration, BuzzMotorPattern pattern=BuzzMotorPattern.Constant)
Send a buzzmotor command to a specific finger.
- bool [StopBuzzMotors](#) ()
Stop all vibration feedback on the Sense Glove.
- bool [SendThumperCmd](#) (SenseGloveCs.ThumperEffect effect)
Play an effect using the Thumper module on this glove (if it has any).
- void [SetHandParameters](#) (Vector3[] jointPositions, Vector3[][] handLengths)
Apply hand parameters to the Sense Glove internal model.
- void [ResetKinematics](#) ()
Reset the internal handmodel back to the default finger lengths and -positions
- void [SaveHandCalibration](#) ()
- bool [GetInterpolationProfile](#) (out SenseGloveCs.Kinematics.InterpolationSet_IMU set)
- bool [SetInterpolationProfile](#) (SenseGloveCs.Kinematics.InterpolationSet_IMU set)

Static Public Member Functions

- static bool [MatchesConnection](#) (bool rightHand, [ConnectionMethod](#) method)
Check whether or not a glove with a particular handed-ness matches a connection method.

Public Attributes

- [ConnectionMethod](#) [connectionMethod](#) = [ConnectionMethod.NextGlove](#)
The way with which the [SG_SenseGloveHardware](#) connects to a glove.
- bool [FFB_Enabled](#) = true
- bool [buzz_Enabled](#) = true
- [SGEvent](#) [OnGloveLoad](#)
Unity Event that fires when this script is assigned to a Sense Glove

Static Public Attributes

- static bool [deviceScannerPresent](#) = false
Saves setup time for multiple SenseGlove_Objects checking for DeviceManager's existence.

Protected Member Functions

- override bool **CanLinkTo** (IODevice device)
- override void **SetupDevice** ()
When linked, this function is run for first time setup.
- override void **DisposeDevice** ()
Unlink this glove from the manager.
- void **CheckForDeviceManager** ()
Check if a device manager is currently active within the Scene. If not, create an instance to manager our connection(s).
- void **UpdateGlove** ()
Updates this glove's data.
- void **CheckConnection** ()
Check if we should fire an of the connected events.
- void **OnGloveLink** ()
Used to call the GloveLoaded event.
- void **OnGloveUnLink** ()
Used to call the GloveLoaded event.
- void **OnGloveConnect** ()
Used to call the OnGloveLoaded event.
- void **OnGloveDisconnect** ()
Used to call the OnGloveLoaded event.
- void **FinishCalibration** (GloveCalibrationArgs calibrationArgs)
Used to call the OnCalibrationFinished event.
- void **FlushCmds** ()
Update and flush all commands for this Sense Glove.
- bool **WriteHaptics** (int[] brakeLvls, int[] buzzLvls, int thumperEffect)
- bool **AlreadySent** (int[] cmds, int[] lastCmd)
- bool **WriteBrakeCmd** (int[] commands)
Tell the Sense Glove to set its brakes at the desired magnitude [0..100%] for each finger until it recieves a new command.
- virtual void **ReadyCalibration** (GloveCalibrationArgs args, bool fromDLL)
The Calibration of the Sense Glove should be ready to fire.
- virtual void **CheckCalibration** ()
Check if we have any CalibrationComplete events queued, then send them.
- virtual void **Start** ()
- virtual void **Update** ()
- virtual void **LateUpdate** ()
- override void **OnDestroy** ()
- virtual void **OnApplicationQuit** ()

Protected Attributes

- bool **autoConnect** = true
Allows this Sense Glove_Object to manage its own connection status
- HapticSendMode **sendHaptics** = HapticSendMode.OnChangeRepeat
When Haptic Commands are sent to the SenseGlove Hardware.
- Solver **solver** = Solver.Interpolate4Sensors
The Solver used to calculate this Sense Glove's hand model each frame.
- bool **limitFingers** = true
Whether or not to apply natural limits to the fingers.
- bool **updateWrist** = true

- Whether or not to update the wrist model of the Sense Glove.*

 - SenseGlove `linkedGlove` = null

The Internal Sense Glove object that is linked to this monobehaviour Object
- `SG_SenseGloveData linkedGloveData` = `SG_SenseGloveData.Empty`

The last data from the linked glove.
- `List<GloveCalibrationArgs> calibrationArguments` = `new List<GloveCalibrationArgs>()`

Queued Calibration Command from the fingers, which will fire during Unity's next LateUpdate() (so as to allow acces to transforms)
- `bool wasConnected` = false

Whether or not the linked glove was connected the last time we checked.
- `List<int[]> brakeQueue` = `new List<int[]>()`

Command queue for the brakes, which is flushed at the end of every Update function.
- `int nextThump` = `(int)ThumperEffect.None`
- `int[] lastBrakeLvls` = `new int[5]`

The last sent brake command
- `List<BuzzCmd> buzzQueue` = `new List<BuzzCmd>()`
- `int[] lastBuzzLvls` = `new int[5]`
- `int lastThump` = `(int)ThumperEffect.None`
- `bool newLinkMade` = false
- `int cmdsSend` = 0
- `int maxCmdRepeat` = 2

Static Protected Attributes

- static `int maxBrakeCmds` = 10
 - static `int maxBuzzCmds` = 20
 - static `int thumpFFBThreshold` = 70
- If the average force-feedback levels are above this threshold, we should not fire Thumper Commands.*
- static `int thumpBuzzThreshold` = 70

Properties

- `bool IsRight` [get]
- If true, this Sense Glove is connected to a right hand. Otherwise, it is connected to a left hand.*
- `bool GloveReady` [get]
- Determines if this glove is ready and linked to the hardware.*
- override `bool IsConnected` [get]
- Check if the Sense Glove is connected.*
- `SG_SenseGloveData GloveData` [get]
- Retrieve Unity-Friendly Glove Data from the Sense Glove.*
- `bool? IsCalibrating` [get]
- Check if this glove is collection calibration points.*
- `GloveData InternalGloveData` [get]
- `float[][] FingerLengths` [get, set]
- The finger lengths used by this sense glove as a 5x3 array, which contains the Proximal-, Medial-, and Distal Phalange lengths for each finger, in that order, in mm.*
- `Vector3[] StartJointPositions` [get, set]
- The positions of the starting finger joints, the CMC or MCP joints, relative to the glove origin.*

Events

- [GloveEventHandler GloveLoaded](#)
Called when this script is assigned a Sense Glove via the SenseGlove_DeviceManager.
- [GloveEventHandler GloveUnLoaded](#)
Called when the SenseGlove_DeviceManager unlinks the Sense Glove from this object.
- [CalibrationFinishedEventHandler CalibrationFinished](#)
Occurs when the finger calibration is finished. Passes the old and new GloveData as arguments.

5.48.1 Detailed Description

After being linked to a proper Sense Glove via the SenseGlove_DeviceManager, this script is responsible for updating [SG_SenseGloveData](#) every frame, and for exposing feedback - and calibration methods.

5.48.2 Member Enumeration Documentation

5.48.2.1 ConnectionMethod

```
enum SG.SG_SenseGloveHardware.ConnectionMethod [strong]
```

The way this object connects to SenseGlove_Objects detected on this system.

Enumerator

NextGlove	Connect to the first unconnected SenseGlove on the system.
NextRightHand	Connect to the first unconnected Right Handed SenseGlove on the system.
NextLeftHand	Connect to the first unconnected Left Handed SenseGlove on the system.

5.48.3 Member Function Documentation

5.48.3.1 CalibrationFinishedEventHandler()

```
delegate void SG.SG_SenseGloveHardware.CalibrationFinishedEventHandler (
    object source,
    GloveCalibrationArgs args )
```

Event delegate function for the CalibrateionFinished event.

Parameters

<i>source</i>	
<i>args</i>	

5.48.3.2 CheckCalibration()

```
virtual void SG.SG_SenseGloveHardware.CheckCalibration ( ) [protected], [virtual]
```

Check if we have any CalibrationComplete events queued, then send them.

Placed indside a seprate method so we can call it during both Update and LateUpdate. Should only be fired from these

5.48.3.3 CheckConnection()

```
void SG.SG_SenseGloveHardware.CheckConnection ( ) [protected]
```

Check if we should fire an of the connected events.

While connection events also fire from the DLL, these are mostly from another worker thread. This is Unity-Safe.

5.48.3.4 CheckForDeviceManager()

```
void SG.SG_SenseGloveHardware.CheckForDeviceManager ( ) [protected]
```

Check if a device manager is currently active within the Scene. If not, create an instance to manager our connection(s).

5.48.3.5 DisposeDevice()

```
override void SG.SG_SenseGloveHardware.DisposeDevice ( ) [protected], [virtual]
```

Unlink this glove from the manager.

Reimplemented from [SG.SG_DeviceLink](#).

5.48.3.6 FinishCalibration()

```
void SG.SG_SenseGloveHardware.FinishCalibration (
    GloveCalibrationArgs calibrationArgs ) [protected]
```

Used to call the OnCalibrationFinished event.

Parameters

<i>calibrationArgs</i>	
------------------------	--

5.48.3.7 FlushCmds()

```
void SG.SG_SenseGloveHardware.FlushCmds ( ) [protected]
```

Update and flush all commands for this Sense Glove.

5.48.3.8 GloveEventHandler()

```
delegate void SG.SG_SenseGloveHardware.GloveEventHandler (
    object source,
    System.EventArgs args )
```

Event delegate for the glove events event.

Parameters

<i>source</i>	
<i>args</i>	

5.48.3.9 HasFunction()

```
bool SG.SG_SenseGloveHardware.HasFunction (
    GloveFunctions function )
```

Verify if this SenseGlove has a particular functionality (buzz motors, haptic feedback, etc)

Parameters

<i>function</i>	The function to test for
-----------------	--------------------------

Returns

5.48.3.10 MatchesConnection()

```
static bool SG.SG_SenseGloveHardware.MatchesConnection (
    bool rightHand,
    ConnectionMethod method ) [static]
```

Check whether or not a glove with a particular handed-ness mathces a connection method.

Parameters

<i>rightHand</i>	
<i>method</i>	

Returns**5.48.3.11 OnGloveConnect()**

```
void SG.SG_SenseGloveHardware.OnGloveConnect ( ) [protected]
```

Used to call the OnGloveLoaded event.

5.48.3.12 OnGloveDisconnect()

```
void SG.SG_SenseGloveHardware.OnGloveDisconnect ( ) [protected]
```

Used to call the OnGloveLoaded event.

5.48.3.13 OnGloveLink()

```
void SG.SG_SenseGloveHardware.OnGloveLink ( ) [protected]
```

Used to call the GloveLoaded event.

5.48.3.14 OnGloveUnLink()

```
void SG.SG_SenseGloveHardware.OnGloveUnLink ( ) [protected]
```

Used to call the GloveLoaded event.

5.48.3.15 ReadyCalibration()

```
virtual void SG.SG_SenseGloveHardware.ReadyCalibration (
    GloveCalibrationArgs args,
    bool fromDLL ) [protected], [virtual]
```

The [Calibration](#) of the Sense Glove should be ready to fire.

Parameters

<i>args</i>	
<i>fromDLL</i>	

5.48.3.16 ResetKinematics()

```
void SG.SG_SenseGloveHardware.ResetKinematics ( )
```

Reset the internal handmodel back to the default finger lengths and -positions

5.48.3.17 SendBrakeCmd()

```
bool SG.SG_SenseGloveHardware.SendBrakeCmd (
    int thumbCmd,
    int indexCmd,
    int middleCmd,
    int ringCmd,
    int pinkyCmd )
```

Tell the Sense Glove to set its brakes at the desired magnitude [0..100%] for each finger until it recieves a new command.

Parameters

<i>thumbCmd</i>	
<i>indexCmd</i>	
<i>middleCmd</i>	
<i>ringCmd</i>	
<i>pinkyCmd</i>	

Returns

Returns true if the command has been succesfully sent.

5.48.3.18 SendBuzzCmd() [1/4]

```
bool SG.SG_SenseGloveHardware.SendBuzzCmd (
    bool[] fingers,
    int magnitude = 100,
    int duration = 400,
    BuzzMotorPattern pattern = BuzzMotorPattern.Constant )
```

Send one buzzmotor command to specific fingers, as indicated by the fingers array.

Parameters

<i>fingers</i>	The fingers (from thumb to pinky) to which to actually apply the buzzMotor command.
<i>magn</i>	
<i>dur</i>	

Returns

5.48.3.19 SendBuzzCmd() [2/4]

```
bool SG_SG_SenseGloveHardware.SendBuzzCmd (
    bool[] fingers,
    int[] durations = null,
    int[] magnitudes = null,
    BuzzMotorPattern[] patterns = null )
```

Send a buzz-motor command to the Sense Glove, with optional parameters for each finger.

Parameters

<i>fingers</i>	
<i>durations</i>	
<i>magnitudes</i>	
<i>patterns</i>	

Returns

This is where the command is actually sent, with parameters for the fingers. All other SendBuzzCmd methods are wrappers.

5.48.3.20 SendBuzzCmd() [3/4]

```
bool SG_SG_SenseGloveHardware.SendBuzzCmd (
    Finger finger,
    int magnitude,
    int duration,
    BuzzMotorPattern pattern = BuzzMotorPattern.Constant )
```

Send a buzzmotor command to a specific finger.

Parameters

<i>finger</i>	
<i>magnitude</i>	
<i>duration</i>	
<i>pattern</i>	

Returns

5.48.3.21 SendBuzzCmd() [4/4]

```
bool SG.SG_SenseGloveHardware.SendBuzzCmd (
    int[] magnitudes,
    int duration )
```

Tell the Buzz motors to each vibrate on a different magnitude (0..100%) for a specific duration (ms)

Parameters

<i>magnitudes</i>	
<i>duration</i>	

Returns

5.48.3.22 SendThumperCmd()

```
bool SG.SG_SenseGloveHardware.SendThumperCmd (
    SenseGloveCs.ThumperEffect effect )
```

Play an effect using the Thumper module on this glove (if it has any).

Parameters

<i>effect</i>	
---------------	--

Returns

5.48.3.23 SetHandParameters()

```
void SG.SG_SenseGloveHardware.SetHandParameters (
    Vector3[] jointPositions,
    Vector3 handLengths[][] )
```

Apply hand parameters to the Sense Glove internal model.

Parameters

<i>jointPositions</i>	
<i>handLengths</i>	

5.48.3.24 SetupDevice()

```
override void SG.SG_SenseGloveHardware.SetupDevice ( ) [protected], [virtual]
```

When linked, this function is run for first time setup.

Reimplemented from [SG.SG_DeviceLink](#).

5.48.3.25 StopBrakes()

```
bool SG.SG_SenseGloveHardware.StopBrakes ( )
```

Release all brakes of the SenseGlove.

Returns**5.48.3.26 StopBuzzMotors()**

```
bool SG.SG_SenseGloveHardware.StopBuzzMotors ( )
```

Stop all vibration feedback on the Sense Glove.

Returns**5.48.3.27 StopFeedback()**

```
bool SG.SG_SenseGloveHardware.StopFeedback ( )
```

Stop all forms of feedback on this Sense Glove.

Returns

5.48.3.28 UpdateGlove()

```
void SG.SG_SenseGloveHardware.UpdateGlove ( ) [protected]
```

Updates this glove's data.

5.48.3.29 WriteBrakeCmd()

```
bool SG.SG_SenseGloveHardware.WriteBrakeCmd (
    int[] commands ) [protected]
```

Tell the Sense Glove to set its brakes at the desired magnitude [0..100%] for each finger until it receives a new command.

Parameters

<i>commands</i>	
-----------------	--

Returns

Returns true if the command has been successfully sent.

This is where the magic happens; where the actual command is sent. All other SendBrakeCmd methods are wrappers.

5.48.4 Member Data Documentation

5.48.4.1 autoConnect

```
bool SG.SG_SenseGloveHardware.autoConnect = true [protected]
```

Allows this Sense Glove_Object to manage its own connection status

5.48.4.2 brakeQueue

```
List<int[]> SG.SG_SenseGloveHardware.brakeQueue = new List<int[]>() [protected]
```

Command queue for the brakes, which is flushed at the end of every Update function.

5.48.4.3 buzz_Enabled

```
bool SG.SG_SenseGloveHardware.buzz_Enabled = true
```

5.48.4.4 calibrationArguments

```
List<GloveCalibrationArgs> SG.SG_SenseGloveHardware.calibrationArguments = new List<GloveCalibrationArgs>()  
[protected]
```

Queued [Calibration](#) Command from the fingers, which will fire during Unity's next LateUpdate() (so as to allow access to transforms)

5.48.4.5 connectionMethod

```
ConnectionMethod SG.SG_SenseGloveHardware.connectionMethod = ConnectionMethod.NextGlove
```

The way with which the [SG_SenseGloveHardware](#) connects to a glove.

5.48.4.6 deviceScannerPresent

```
bool SG.SG_SenseGloveHardware.deviceScannerPresent = false [static]
```

Saves setup time for multiple SenseGlove_Objects checking for DeviceManager's existence.

5.48.4.7 lastBrakeLvls

```
int [] SG.SG_SenseGloveHardware.lastBrakeLvls = new int[5] [protected]
```

The last sent brake command

5.48.4.8 limitFingers

```
bool SG.SG_SenseGloveHardware.limitFingers = true [protected]
```

Whether or not to apply natural limits to the fingers.

Marked as protected since these will likely always be true during normal use.

5.48.4.9 linkedGlove

```
SenseGlove SG.SG_SenseGloveHardware.linkedGlove = null [protected]
```

The Internal Sense Glove object that is linked to this monobehaviour Object

5.48.4.10 linkedGloveData

```
SG_SenseGloveData SG.SG_SenseGloveHardware.linkedGloveData = SG_SenseGloveData.Empty [protected]
```

The last data from the linked glove.

5.48.4.11 OnGloveLoad

```
SGEvent SG.SG_SenseGloveHardware.OnGloveLoad
```

Unity Even that fires when this script is assigned to a Sense Glove

5.48.4.12 sendHaptics

```
HapticSendMode SG.SG_SenseGloveHardware.sendHaptics = HapticSendMode.OnChangeRepeat [protected]
```

When Haptic Commands are sent to the SenseGlove Hardware.

5.48.4.13 solver

```
Solver SG.SG_SenseGloveHardware.solver = Solver.Interpolate4Sensors [protected]
```

The Solver used to calculate this Sense Glove's hand model each frame.

5.48.4.14 thumpFFBThreshold

```
int SG.SG_SenseGloveHardware.thumpFFBThreshold = 70 [static], [protected]
```

If the average force-feedback levels are above this threshold, we should not fire Thumper Commands.

5.48.4.15 updateWrist

```
bool SG.SG_SenseGloveHardware.updateWrist = true [protected]
```

Whether or not to update the wrist model of the Sense Glove.

We will always update it, but calibrate it at hand-model level.

5.48.4.16 wasConnected

```
bool SG.SG_SenseGloveHardware.wasConnected = false [protected]
```

Whether or not the linked glove was connected the last time we checked.

5.48.5 Property Documentation

5.48.5.1 FingerLengths

```
float [][] SG.SG_SenseGloveHardware.FingerLengths [get], [set]
```

The finger lengths used by this sense glove as a 5x3 array, which contains the Proximal-, Medial-, and Distal Phalange lengths for each finger, in that order, in mm.

5.48.5.2 GloveData

```
SG_SenseGloveData SG.SG_SenseGloveHardware.GloveData [get]
```

Retrieve Unity-Friendly Glove Data from the Sense Glove.

5.48.5.3 GloveReady

```
bool SG.SG_SenseGloveHardware.GloveReady [get]
```

Determines if this glove is ready and linked to the hardware.

5.48.5.4 IsCalibrating

```
bool? SG.SG_SenseGloveHardware.IsCalibrating [get]
```

Check if this glove is collection calibration points.

5.48.5.5 IsConnected

```
override bool SG.SG_SenseGloveHardware.IsConnected [get]
```

Check if the Sense Glove is connected.

5.48.5.6 IsRight

```
bool SG.SG_SenseGloveHardware.IsRight [get]
```

If true, this Sense Glove is connected to a right hand. Otherwise, it is connected to a left hand.

5.48.5.7 StartJointPositions

```
Vector3 [] SG.SG_SenseGloveHardware.StartJointPositions [get], [set]
```

The positions of the starting finger joints, the CMC or MCP joints, relative to the glove origin.

Returns

5.48.6 Event Documentation

5.48.6.1 CalibrationFinished

```
CalibrationFinishedEventHandler SG.SG_SenseGloveHardware.CalibrationFinished
```

Occurs when the finger calibration is finished. Passes the old and new GloveData as arguments.

5.48.6.2 GloveLoaded

```
GloveEventHandler SG.SG_SenseGloveHardware.GloveLoaded
```

Called when this script is assigned a Sense Glove via the SenseGlove_DeviceManager.

5.48.6.3 GloveUnLoaded

`GloveEventHandler` `SG.SG_SenseGloveHardware.GloveUnLoaded`

Called when the SenseGlove_DeviceManager unlinks the Sense Glove from this object.

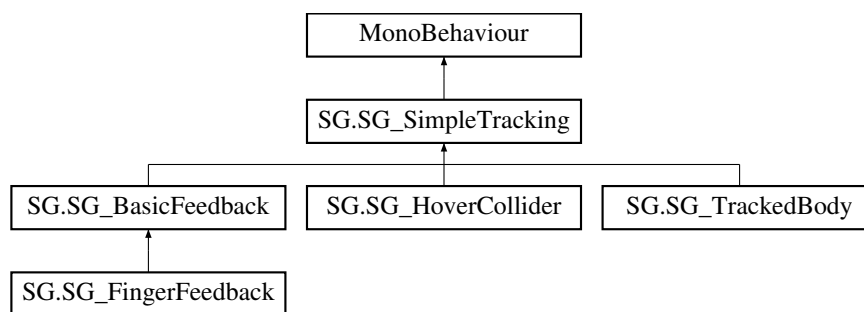
The documentation for this class was generated from the following file:

- `D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Devices/SG_SenseGloveHardware.cs`

5.49 SG.SG_SimpleTracking Class Reference

Attached to a GameObject to make it follow a 'target'

Inheritance diagram for SG.SG_SimpleTracking:



Public Types

- enum `UpdateDuring` { `LateUpdate`, `FixedUpdate`, `Update`, `Off` }
When the position of this GameObject is updated.

Public Member Functions

- virtual void `SetIgnoreCollision` (Collider col, bool ignoreCollision)
Ignore collision between this object and another collider
- virtual void `SetTrackingTarget` (Transform newTarget, bool calculateNewOffsets)
Set a new tracking target for this script, which also calculates new offsets

Public Attributes

- `UpdateDuring updateTime` = `UpdateDuring.LateUpdate`
Determines when an instance of this script updates its position.

Protected Member Functions

- virtual void [UpdatePosition](#) ()
Update the transform of this script to its TargetPosition and Rotation
- virtual void **Awake** ()
- virtual void **Update** ()
- virtual void **LateUpdate** ()
- virtual void **FixedUpdate** ()

Protected Attributes

- Transform [trackingTarget](#)
A transform to follow during the simulation. Offsets are determined during Start() of this script
- Vector3 [positionOffset](#) = Vector3.zero
Position offset between this object and the target transform
- Quaternion [rotationOffset](#) = Quaternion.identity
Rotation offset between this object and the target transform

Properties

- virtual bool [DebugEnabled](#) [get, set]
Enable/Disable the MeshRenderer connected to this script's GameObject
- Vector3 [TargetPosition](#) [get]
Returns the supposed, absolute position of this GameObject, based on its offsets.
- Quaternion [TargetRotation](#) [get]
Returns the supposed, absolute rotation of this GameObject, based on its offsets.
- bool [HasTarget](#) [get]
Returns true if this script has a target it can follow

5.49.1 Detailed Description

Attached to a GameObject to make it follow a 'target'

5.49.2 Member Enumeration Documentation

5.49.2.1 UpdateDuring

```
enum SG.SG_SimpleTracking.UpdateDuring [strong]
```

When the position of this GameObject is updated.

5.49.3 Member Function Documentation

5.49.3.1 SetIgnoreCollision()

```
virtual void SG.SG_SimpleTracking.SetIgnoreCollision (
    Collider col,
    bool ignoreCollision ) [virtual]
```

Ignore collision between this object and another collider

Parameters

<i>col</i>	
------------	--

5.49.3.2 SetTrackingTarget()

```
virtual void SG.SG_SimpleTracking.SetTrackingTarget (
    Transform newTarget,
    bool calculateNewOffsets ) [virtual]
```

Set a new tracking target for this script, which also calculates new offsets

Parameters

<i>newTarget</i>	
------------------	--

5.49.3.3 UpdatePosition()

```
virtual void SG.SG_SimpleTracking.UpdatePosition ( ) [protected], [virtual]
```

Update the transform of this script to its TragetPosition and Rotation

Reimplemented in [SG.SG_BasicFeedback](#), and [SG.SG_TrackedBody](#).

5.49.4 Member Data Documentation

5.49.4.1 positionOffset

```
Vector3 SG.SG_SimpleTracking.positionOffset = Vector3.zero [protected]
```

Position offset between this object and the target transform

5.49.4.2 rotationOffset

```
Quaternion SG.SG_SimpleTracking.rotationOffset = Quaternion.identity [protected]
```

Rotation offset between this object and the target transform

5.49.4.3 trackingTarget

Transform SG.SG_SimpleTracking.trackingTarget [protected]

A transform to follow during the simulation. Offsets are determined during Start() of this script

5.49.4.4 updateTime

UpdateDuring SG.SG_SimpleTracking.updateTime = UpdateDuring.LateUpdate

Determines when an instance of this script updates its position.

5.49.5 Property Documentation

5.49.5.1 DebugEnabled

virtual bool SG.SG_SimpleTracking.DebugEnabled [get], [set]

Enable/Disable the MeshRenderer connected to this script's GameObject

5.49.5.2 HasTarget

bool SG.SG_SimpleTracking.HasTarget [get]

Returns true if this script has a target it can follow

5.49.5.3 TargetPosition

Vector3 SG.SG_SimpleTracking.TargetPosition [get]

Returns the supposed, absolute position of this GameObject, based on its offsets.

5.49.5.4 TargetRotation

Quaternion SG.SG_SimpleTracking.TargetRotation [get]

Returns the supposed, absolute rotation of this GameObject, based on its offsets.

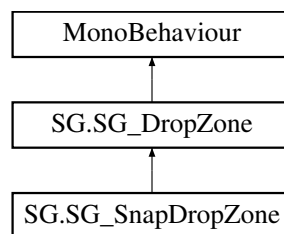
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_SimpleTracking.cs

5.50 SG.SG_SnapDropZone Class Reference

A DropZone that snaps a Grabable to a specific SnapPoint.

Inheritance diagram for SG.SG_SnapDropZone:



Classes

- class [SnapProps](#)
Contains parameters that assist in snapping/unsnapping to a SnapZone.

Public Types

- enum [SnapMethod](#) { [SnapMethod.ObjectDependent](#) = 0, [SnapMethod.Parent](#), [SnapMethod.FixedJoint](#) }
The way in which a SnapZone attaches objects to itself.

Public Member Functions

- override void [ValidateSettings](#) ()
Validates RB / Collider settings on initialization. Add another check for RigidBodyes.
- override void [AddObject](#) ([SG_Grabable](#) grabable)
Fires when an object first enters the zone. Record its snap-properties.
- bool [IsSnapped](#) ([SG_Grabable](#) grabable)
Returns true if this particular object has been detected and snapped within the SnapZone.
- void [ReleaseObject](#) ([SG_Grabable](#) grabable)
Release a specific object from the zone.

Public Attributes

- bool `disablesInteraction` = false
When set to true, this SnapZone automatically disables the interaction of the SenseGlove_Grabables that enter it.
- bool `takesFromHand` = false
If set to true, this SnapZone ends the interaction between the hand and the interactable.
- Transform `snapPoint`
The point to which the SenseGlove_Grabables will attempt to snap.
- SnapMethod `snapMethod` = SnapMethod.ObjectDependent
The way in which the SnapZone attaches objects to itself.

Protected Member Functions

- override void `CallObjectDetect` (SG_Grabable detectedObject)
Called when an Object is detected and its event is called. End interation if needed, then snap it
- override void `RemoveObject` (int index)
Fires when an object is removed from the zone. Unsubscribe from method(s).
- void `AttachObject` (SG_Grabable grabable)
Snaps an object to this Zone's snapPoint, based on the Grabable's grabType.
- void `ReleaseObject` (int index)
Released an obejct from physics, but not from detection
- virtual void `Start` ()

Protected Attributes

- List< SnapProps > `snapProperties` = new List<SnapProps>()
Contains properties for before/after snapping

Private Member Functions

- void `Grabable_InteractionBegan` (object source, SG_InteractArgs args)
Fires when an object is picked up from the Sense Glove. Disconnect it from this SnapZone.
- void `Grabable_InteractionEnded` (object source, SG_InteractArgs args)
Fires when one of my ObjectsToGet is released.
- void `Grabable_ObjectReset` (object source, System.EventArgs args)
Fires when an object is reset. Disconnect it from this SnapZone.

Additional Inherited Members

5.50.1 Detailed Description

A DropZone that snaps a Grabable to a specific SnapPoint.

5.50.2 Member Enumeration Documentation

5.50.2.1 SnapMethod

```
enum SG.SG_SnapDropZone.SnapMethod [strong]
```

The way in which a SnapZone attaches objects to itself.

Enumerator

ObjectDependent	The snapzone chooses which option to use, based on the Grabable's GrabType property
Parent	The Grabable becomes a child object of the dropzone. If it possesses a RigidBody, it is marked as kinematic.
FixedJoint	The DropZone creates a PhysicsJoint connection between this dropzone and the grabable rigidBody.

5.50.3 Member Function Documentation

5.50.3.1 AddObject()

```
override void SG.SG_SnapDropZone.AddObject (
    SG_Grabable grabable ) [virtual]
```

Fires when an object first enters the zone. Record its snap-properties.

Parameters

<i>grabable</i>	
-----------------	--

Reimplemented from [SG.SG_DropZone](#).

5.50.3.2 AttachObject()

```
void SG.SG_SnapDropZone.AttachObject (
    SG_Grabable grabable ) [protected]
```

Snaps an object to this Zone's snapPoint, based on the Grabable's grabType.

Parameters

<i>grabable</i>	
-----------------	--

5.50.3.3 CallObjectDetect()

```
override void SG.SG_SnapDropZone.CallObjectDetect (
    SG_Grabable detectedObject ) [protected], [virtual]
```

Called when an Object is detected and its event is called. End interaction if needed, then snap it

Parameters

<i>detectedObject</i>	
-----------------------	--

Reimplemented from [SG.SG_DropZone](#).

5.50.3.4 Grabable_InteractionBegan()

```
void SG.SG_SnapDropZone.Grabable_InteractionBegan (
    object source,
    SG_InteractArgs args ) [private]
```

Fires when an object is picked up from the Sense Glove. Disconnect it from this SnapZone.

Parameters

<i>source</i>	
<i>args</i>	

5.50.3.5 Grabable_InteractionEnded()

```
void SG.SG_SnapDropZone.Grabable_InteractionEnded (
    object source,
    SG_InteractArgs args ) [private]
```

Fires when one of my ObjectsToGet is released.

Should only be subscribed to when

Parameters

<i>source</i>	
<i>args</i>	

5.50.3.6 Grabable_ObjectReset()

```
void SG.SG_SnapDropZone.Grabable_ObjectReset (
    object source,
    System.EventArgs args ) [private]
```

Fires when an object is reset. Disconnect it from this SnapZone.

Parameters

<i>source</i>	
<i>args</i>	

5.50.3.7 IsSnapped()

```
bool SG.SG_SnapDropZone.IsSnapped (
    SG_Grabable grabable )
```

Returns true if this particular object has been detected and snapped within the SnapZone.

Parameters

<i>grabable</i>	
-----------------	--

Returns

5.50.3.8 ReleaseObject() [1/2]

```
void SG.SG_SnapDropZone.ReleaseObject (
    int index ) [protected]
```

Released an obejct from physics, but not from detection

Parameters

<i>index</i>	
--------------	--

5.50.3.9 ReleaseObject() [2/2]

```
void SG.SG_SnapDropZone.ReleaseObject (
    SG_Grabable grabable )
```

Release a specific object from the zone.

Parameters

<i>grabable</i>	
-----------------	--

5.50.3.10 RemoveObject()

```
override void SG.SG_SnapDropZone.RemoveObject (
    int index ) [protected], [virtual]
```

Fires when an object is removed from the zone. Unsubscribe from method(s).

Parameters

<i>index</i>	
--------------	--

Reimplemented from [SG.SG_DropZone](#).

5.50.3.11 ValidateSettings()

```
override void SG.SG_SnapDropZone.ValidateSettings ( ) [virtual]
```

Validates RB / Collider settings on initialization. Add another check for RigidBodies.

Reimplemented from [SG.SG_DropZone](#).

5.50.4 Member Data Documentation

5.50.4.1 disablesInteraction

```
bool SG.SG_SnapDropZone.disablesInteraction = false
```

When set to true, this SnapZone automatically disables the interaction of the SenseGlove_Grabables that enter it.

5.50.4.2 snapMethod

```
SnapMethod SG.SG_SnapDropZone.snapMethod = SnapMethod.ObjectDependent
```

The way in which the SnapZone attaches objects to itself.

5.50.4.3 snapPoint

`Transform SG.SG_SnapDropZone.snapPoint`

The point to which the SenseGlove_Grabables will attempt to snap.

If no Rigidbody is attached to this zone, we will attempt to look for one here.

5.50.4.4 snapProperties

`List<SnapProps> SG.SG_SnapDropZone.snapProperties = new List<SnapProps>() [protected]`

Contains properties for before/after snapping

5.50.4.5 takesFromHand

`bool SG.SG_SnapDropZone.takesFromHand = false`

If set to true, this SnapZone ends the interaction between the hand and the interactable.

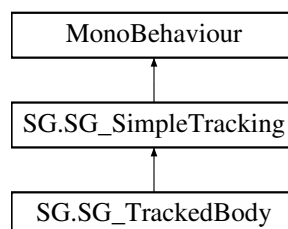
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_SnapDropZone.cs

5.51 SG.SG_TrackedBody Class Reference

A Rigidbody that tracks a transform by adding velocity to the body, rather than directly applying positions. It reverts back to simpleTrackign if no Rigidbody is present.

Inheritance diagram for SG.SG_TrackedBody:



Public Member Functions

- void [TryAddRB](#) (bool useGrav=false, bool isKinematic=false)
Try to add a rigidbody to this GameObject, if one isn't already present
- void [TryRemoveRB](#) ()
Remove the Rigidbody if one exists.

Public Attributes

- Rigidbody [physicsBody](#)
The Rigidbody to apply force-feedback to.

Protected Member Functions

- override void [UpdatePosition](#) ()
Update this object's transform by applying a velocity to the rigidbody
- override void **Awake** ()

Protected Attributes

- float [resetTimer](#) = 0
Timer to keep track of how long the collider has been away from its target transform

Static Protected Attributes

- static float [resetTime](#) = 3
Time after which the rigidbody will reset back to its targetposition if it is more than resetDistance away
- static float [resetDistance](#) = 1
Maximum distance between this script and it's target position before we assume the colliders are stuck somewhere.
- static float [rotationSpeed](#) = 25
Speed at which the rotation is matched

Properties

- bool [CollisionEnabled](#) [get, set]
Enable / Disable collision of this collider in general

Additional Inherited Members

5.51.1 Detailed Description

A Rigidbody that tracks a transform by adding velocity to the body, rather than directly applying positions. It reverts back to simpleTrackign if no Rigidbody is present.

5.51.2 Member Function Documentation

5.51.2.1 TryAddRB()

```
void SG.SG_TrackedBody.TryAddRB (
    bool useGrav = false,
    bool isKinematic = false )
```

Try to add a rigidbody to this GameObject, if one isn't already present

Parameters

<i>useGrav</i>	
<i>isKinematic</i>	

5.51.2.2 TryRemoveRB()

```
void SG.SG_TrackedBody.TryRemoveRB ( )
```

Remove the Rigidbody if one exists.

5.51.2.3 UpdatePosition()

```
override void SG.SG_TrackedBody.UpdatePosition ( ) [protected], [virtual]
```

Update this object's transform by applying a velocity to the rigidbody

Reimplemented from [SG.SG_SimpleTracking](#).

5.51.3 Member Data Documentation**5.51.3.1 physicsBody**

```
Rigidbody SG.SG_TrackedBody.physicsBody
```

The Rigidbody to apply force-feedback to.

5.51.3.2 resetDistance

```
float SG.SG_TrackedBody.resetDistance = 1 [static], [protected]
```

Maximum distance between this script and it's target position before we assume the colliders are stuck somewhere.

5.51.3.3 resetTime

```
float SG.SG_TrackedBody.resetTime = 3 [static], [protected]
```

Time after which the rigidbody will reset back to its targetposition if it is more than resetDistance away

5.51.3.4 resetTimer

```
float SG.SG_TrackedBody.resetTimer = 0 [protected]
```

Timer to keep track of how long the collider has been away from its target transform

5.51.3.5 rotationSpeed

```
float SG.SG_TrackedBody.rotationSpeed = 25 [static], [protected]
```

Speed at which the rotation is matched

5.51.4 Property Documentation

5.51.4.1 CollisionEnabled

```
bool SG.SG_TrackedBody.CollisionEnabled [get], [set]
```

Enable / Disable collision of this collider in general

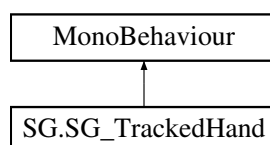
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_TrackedBody.cs

5.52 SG.SG_TrackedHand Class Reference

A hand model with different layers, that follows a GameObject with a configurable offset

Inheritance diagram for SG.SG_TrackedHand:



Public Types

- enum [TrackingHardware](#) { [TrackingHardware.Custom](#), [TrackingHardware.ViveTracker](#) }
The hardware this hand is tracked with. Used to calculate offsets.
- enum [TrackingMethod](#) { [TrackingMethod.Default](#), [TrackingMethod.PhysicsBased](#), [TrackingMethod.Disabled](#) }
The way the tracking is established.

Public Member Functions

- virtual void [SwapTracking](#) ([SG_TrackedHand](#) otherHand)
Swap the tracking targets between this hand and another one.
- void [UpdateTransformDefault](#) ()
Update this script's transform by applying a position and rotation directly.
- void [UpdateTransformPhysics](#) ()
Update this script's transform by applying a velocity to its rigidbody.
- void **OnCollisionEnter** (Collision collision)

Public Attributes

- [SG_SenseGloveHardware](#) hardware
The hand tracking hardware used to animate / link this TrackedHand.
- [TrackingHardware](#) trackingHardware = [TrackingHardware.ViveTracker](#)
The hardware that controls the trackedObject's position. Used to calculate offsets.
- [TrackingMethod](#) trackingMethod = [TrackingMethod.Default](#)
How the position of this TrackedHand is determined.
- [SG_HandModelInfo](#) handModel
Information of the 3D model of the hand this script represents.
- [SG_HandAnimator](#) handAnimation
The script that animates this trackedHand
- [SG_HandFeedback](#) feedbackScript
The script responsible for collecting force-feedback from objects to this hardware.
- [SG_GrabScript](#) grabScript
The script responsible for grabbing and manipulating objects.
- [SG_HandRigidBody](#) rigidBodyLayer
The script that allows this hand to push objects away.
- [SG_HandRigidBody](#) physicsTrackingLayer
The script that prevents this hand from passing through non-trigger colliders.

Protected Member Functions

- void [CheckForScripts](#) ()
Link relevant scripts to this trackedHand, if they have not been assigned yet.
- virtual void [SetupTracking](#) (Transform newTarget, [TrackingHardware](#) trackType, [TrackingMethod](#) trackMethod, bool rightHand)
Setup and/or change the tracking variables of this hand.
- virtual void **Awake** ()
- virtual void **Start** ()
- void **Update** ()
- void **FixedUpdate** ()

Protected Attributes

- Transform [trackedObject](#)
The object that this script will attempt to follow.
- bool [ignoreGrabables](#) = false
If set to true, this hand will ignore collisions with [SG_Interactive](#) objects that its rigidbody collides with.
- Vector3 [positionOffset](#) = Vector3.zero
The position offset between this trackedHand and its trackedObject.
- Quaternion [rotationOffset](#) = Quaternion.identity
The rotation offset between this trackedHand and its trackedObject.
- Rigidbody [handRB](#) = null
This object's Rigidbody, used when dealing with Physics-based tracking.

Static Protected Attributes

- static float [physRotationSpeed](#) = 25
The rotation speed of the Rigidbody, when using Physics-based tracking.

Properties

- Vector3? [TargetPosition](#) [get]
The position that this trackedHand should be in, based on its trackedObject and offsets.
- Quaternion? [TargetRotation](#) [get]
The rotation that this trackedHand should be in, based on its trackedObject and offsets.
- virtual bool [TracksRightHand](#) [get]
Returns true if this Script is set up to track a right hand.

5.52.1 Detailed Description

A hand model with different layers, that follows a GameObject with a configurable offset

5.52.2 Member Enumeration Documentation

5.52.2.1 TrackingHardware

```
enum SG.SG_TrackedHand.TrackingHardware [strong]
```

The hardware this hand is tracked with. Used to calculate offsets.

Enumerator

Custom	Custom tracking hardware is used, so offsets are calculated during Start().
ViveTracker	SenseGlove Vive Tracker Mount

5.52.2.2 TrackingMethod

```
enum SG.SG_TrackedHand.TrackingMethod [strong]
```

The way the tracking is established.

Enumerator

Default	The hand matches the trackedObject's position and rotations, with offsets.
PhysicsBased	The hand gets a rigidbody, which attempts to reach its targetRotation and -position
Disabled	This script does not handle any tracking. Use this when making the hand a child of your trackedObject.

5.52.3 Member Function Documentation

5.52.3.1 CheckForScripts()

```
void SG.SG_TrackedHand.CheckForScripts ( ) [protected]
```

Link relevant scripts to this trackedHand, if they have not been assigned yet.

5.52.3.2 SetupTracking()

```
virtual void SG.SG_TrackedHand.SetupTracking (
    Transform newTarget,
    TrackingHardware trackType,
    TrackingMethod trackMethod,
    bool rightHand ) [protected], [virtual]
```

Setup and/or change the tracking variables of this hand.

Parameters

<i>newTarget</i>	
<i>trackType</i>	
<i>trackMethod</i>	
<i>rightHand</i>	

5.52.3.3 SwapTracking()

```
virtual void SG.SG_TrackedHand.SwapTracking (
    SG_TrackedHand otherHand ) [virtual]
```

Swap the tracking targets between this hand an another one.

Parameters

<i>otherHand</i>	
------------------	--

5.52.3.4 UpdateTransformDefault()

```
void SG.SG_TrackedHand.UpdateTransformDefault ( )
```

Update this script's transform by applying a position and rotation directly.

5.52.3.5 UpdateTransformPhysics()

```
void SG.SG_TrackedHand.UpdateTransformPhysics ( )
```

Update this script's transform by applying a velocity to its rigidbody.

5.52.4 Member Data Documentation

5.52.4.1 feedbackScript

```
SG_HandFeedback SG.SG_TrackedHand.feedbackScript
```

The script responsible for collecting force-feedback from objects to this hardware.

5.52.4.2 grabScript

```
SG_GrabScript SG.SG_TrackedHand.grabScript
```

The script responsible for grabbing and manipulating objects.

5.52.4.3 handAnimation

[SG_HandAnimator](#) SG.SG_TrackedHand.handAnimation

The script that animates this trackedHand

5.52.4.4 handModel

[SG_HandModelInfo](#) SG.SG_TrackedHand.handModel

Information of the 3D model of the hand this script represents.

5.52.4.5 handRB

Rigidbody SG.SG_TrackedHand.handRB = null [protected]

This object's Rigidbody, used when dealing with Physics-based tracking.

5.52.4.6 hardware

[SG_SenseGloveHardware](#) SG.SG_TrackedHand.hardware

The hand tracking hardware used to animate / link this TrackedHand.

5.52.4.7 ignoreGrabables

bool SG.SG_TrackedHand.ignoreGrabables = false [protected]

If set to true, this hand will ignore collisions with [SG_Interactive](#) objects that its rigidbody collides with.

The PhysicsTrackingLayer bodies have no rigidbodies of their own, and so their OnCollisionEnter events fire here.

5.52.4.8 physicsTrackingLayer

[SG_HandRigidBodyes](#) SG.SG_TrackedHand.physicsTrackingLayer

The script that prevents this hand from passing through non-trigger colliders.

5.52.4.9 physRotationSpeed

```
float SG.SG_TrackedHand.physRotationSpeed = 25 [static], [protected]
```

The rotation speed of the Rigidbody, when using Physics-based tracking.

5.52.4.10 positionOffset

```
Vector3 SG.SG_TrackedHand.positionOffset = Vector3.zero [protected]
```

The position offset between this trackedHand and its trackedObject.

5.52.4.11 rigidBodyLayer

```
SG_HandRigidBodies SG.SG_TrackedHand.rigidBodyLayer
```

The script that allows this hand to push objects away.

5.52.4.12 rotationOffset

```
Quaternion SG.SG_TrackedHand.rotationOffset = Quaternion.identity [protected]
```

The rotation offset between this trackedHand and its trackedObject.

5.52.4.13 trackedObject

```
Transform SG.SG_TrackedHand.trackedObject [protected]
```

The object that this script will attempt to follow.

5.52.4.14 trackingHardware

```
TrackingHardware SG.SG_TrackedHand.trackingHardware = TrackingHardware.ViveTracker
```

The hardware that controls the trackedObject's position. Used to calculate offsets.

5.52.4.15 trackingMethod

```
TrackingMethod SG.SG_TrackedHand.trackingMethod = TrackingMethod.Default
```

How the position of this TrackedHand is determined.

5.52.5 Property Documentation

5.52.5.1 TargetPosition

```
Vector3? SG.SG_TrackedHand.TargetPosition [get]
```

The position that this trackedHand should be in, based on its trackedObject and offsets.

5.52.5.2 TargetRotation

```
Quaternion? SG.SG_TrackedHand.TargetRotation [get]
```

The rotation that this trackedHand should be in, based on its trackedObject and offsets.

5.52.5.3 TracksRightHand

```
virtual bool SG.SG_TrackedHand.TracksRightHand [get]
```

Returns true if this Script is set up to track a right hand.

Returns

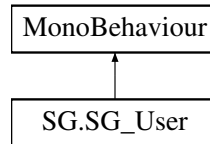
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Tracking/SG_TrackedHand.cs

5.53 SG.SG_User Class Reference

Utility Class to manage up to two SG_TrackedHands, and to swap their hands around.

Inheritance diagram for SG.SG_User:



Public Member Functions

- void [SetupHands](#) ()
Set up the collision of the hands
- void **SwapHandTracking** ()

Public Attributes

- [SG_TrackedHand](#) **leftHand**
- [SG_TrackedHand](#) **rightHand**
- KeyCode **swapHandsKey** = KeyCode.None

Private Member Functions

- void **Start** ()
- void **Update** ()

5.53.1 Detailed Description

Utility Class to manage up to two SG_TrackedHands, and to swap their hands around.

5.53.2 Member Function Documentation

5.53.2.1 SetupHands()

```
void SG.SG_User.SetupHands ( )
```

Set up the collision of the hands

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_User.cs

5.54 SG.SG_Util Class Reference

Contains methods that make the SenseGloveCs library work with Unity.

Public Types

- enum **MoveAxis** {
X = 0, **Y**, **Z**, **NegativeX**,
NegativeY, **NegativeZ** }

Static Public Member Functions

- static string [ToString](#) (Vector3 V)
Convert a Unity Vector3 to a string with a greater precision than its default method.
- static string [ToString](#) (Quaternion Q)
Convert a Unity Quaternion to a string with a greater precision than its default method.
- static string [ToString](#) (float[] V)
Convert a float[] to a string with a greater precision than its default Unity(?) method.
- static string [ToString](#) (int[] V)
Convert an int[] to a string with a greater precision than its default Unity(?) method.
- static Vector3 [ToUnityPosition](#) (SenseGloveCs.Kinematics.Vect3D pos)
Convert a float[3] position taken from the DLL into a Unity Position.
- static Vector3[] [ToUnityPosition](#) (SenseGloveCs.Kinematics.Vect3D[] pos)
Convert an array of float[3] positions taken from the DLL into a Vector3[].
- static SenseGloveCs.Kinematics.Vect3D [ToPosition](#) (Vector3 pos)
Convert from a unity vector3 to a float[3] used in the DLL.
- static SenseGloveCs.Kinematics.Vect3D[] [ToPosition](#) (Vector3[] pos)
Convert an array of unity positions back into an array used by the DLL.
- static Quaternion [ToUnityQuaternion](#) (SenseGloveCs.Kinematics.Quat quat)
Convert a float[4] quaternion taken from the DLL into a Unity Quaternion.
- static SenseGloveCs.Kinematics.Quat [ToQuaternion](#) (Quaternion Q)
Convert a unity Quaternion into a float[4] used in the DLL.
- static SenseGloveCs.Kinematics.Vect3D [ToEuler](#) (Vector3 euler)
Convert a unity eulerAngles notation into one used by the DLL.
- static Vector3 [ToUnityEuler](#) (SenseGloveCs.Kinematics.Vect3D euler)
Convert a set of euler angles from the DLL into the Unity notation.
- static float [NormalizeAngle](#) (float angle)
Normalize an angle (in degrees) such that it is within the -180...180 range.
- static float [NormalizeAngle](#) (float angle, float minAngle, float maxAngle)
Normalize an angle (in degrees) such that it is within the -180...180 range.
- static Vector3 [NormalizeAngles](#) (Vector3 angles)
Normalize a set of (euler) angles to fall within a -180... 180 range.
- static float [Map](#) (float value, float inMin, float inMax, float outMin, float outMax)
Map a value from one range to another.
- static Vector3 [Average](#) (List< Vector3 > values)
Calculates the average between a list of Vector3 values
- static int [Average](#) (int[] values)
Calculates the average between a list of integer values
- static Vector3 [CalculateAngularVelocity](#) (Quaternion currentRot, Quaternion previousRot, float deltaTime)

- Calculate the angular velocity of a GameObject, using its current rotation and that of the previous frame.*
- static void [CalculateOffsets](#) (Transform obj, Transform reference, out Vector3 posOffset, out Quaternion rotOffset)
- Calculate a position and rotation difference between two transforms.*
- static void [TransformRigidBody](#) (ref Rigidbody obj, Vector3 targetPosition, Quaternion targetRotation, float rotationSpeed)
- Add a velocity / angularVelocity to a rigidbody to move towards a targetPosition and rotation*
- static Rigidbody [TryAddRB](#) (GameObject obj, bool useGrav=false, bool isKinematic=false)
- Add a rigidbody to a GameObject if one does not exist yet and apply the desired parameters.*
- static void [TryRemoveRB](#) (GameObject obj)
- Remove the rigidbody from a gameObject, if one exists.*
- static void [CheckForHandInfo](#) (Transform obj, ref [SG_HandModelInfo](#) info)
- Check if an object has a [SG_HandModelInfo](#) component and assign it to the info parameter.*
- static [SG_TrackedHand](#) [CheckForTrackedHand](#) (Transform obj)
- Try to get a [SG_TrackedHand](#) that this script is attached to.*
- static void [SetChildren](#) (Transform obj, bool active)
- Set all the children of the following Transform to active/inactive*
- static Vector3 [GetAxis](#) ([MovementAxis](#) axis)
- Returns a unit vector representing the chosen movement axis.*
- static bool [IsNegative](#) ([MoveAxis](#) axis)
- Returns true if this axis is negative*
- static int [AxisIndex](#) ([MoveAxis](#) axis)
- Returns an index (0, 1, 2) to access a Vector3*
- static Vector3 [GetVector](#) ([MoveAxis](#) axis)
- Returns a normalized Vector representing this axis in 3D space.*
- static GameObject [SpawnSphere](#) (float worldDiameter, Transform parent, bool withCollider=true)
- Spawn a sphere and make it a child of parent.*
- static void [AppendButtonText](#) (UnityEngine.UI.Button button, string addedText)
- Append texts to existing button text (used to add hotkey info to buttons)*

Static Public Attributes

- static bool **keyBindsEnabled** = true

Properties

- static string **SenseGloveDir** [get]

5.54.1 Detailed Description

Contains methods that make the SenseGloveCs library work with Unity.

5.54.2 Member Function Documentation

5.54.2.1 AppendButtonText()

```
static void SG.SG_Util.AppendButtonText (
    UnityEngine.UI.Button button,
    string addedText ) [static]
```

Append texts to existing button text (used to add hotkey info to buttons)

Parameters

<i>button</i>	
<i>addedText</i>	

5.54.2.2 Average() [1/2]

```
static int SG.SG_Util.Average (  
    int[] values ) [static]
```

Calculates the average between a list of integer values

Parameters

<i>values</i>	
---------------	--

Returns**5.54.2.3 Average()** [2/2]

```
static Vector3 SG.SG_Util.Average (  
    List< Vector3 > values ) [static]
```

Calculates the average between a list of Vector3 values

Parameters

<i>values</i>	
---------------	--

Returns**5.54.2.4 AxisIndex()**

```
static int SG.SG_Util.AxisIndex (  
    MoveAxis axis ) [static]
```

Returns an index (0, 1, 2) to access a Vector3

Parameters

<i>axis</i>	
-------------	--

Returns

5.54.2.5 CalculateAngularVelocity()

```
static Vector3 SG.SG_Util.CalculateAngularVelocity (
    Quaternion currentRot,
    Quaternion previousRot,
    float deltaTime ) [static]
```

Calculate the angular velocity of a GameObject, using its current rotation and that of the previous frame.

Parameters

<i>currentRot</i>	
<i>previousRot</i>	

Placed here because it may be used by other scripts as well.

Returns

5.54.2.6 CalculateOffsets()

```
static void SG.SG_Util.CalculateOffsets (
    Transform obj,
    Transform reference,
    out Vector3 posOffset,
    out Quaternion rotOffset ) [static]
```

Calculate a position and rotation difference between two transforms.

Parameters

<i>obj</i>	
<i>reference</i>	
<i>posOffset</i>	
<i>rotOffset</i>	

5.54.2.7 CheckForHandInfo()

```
static void SG.SG_Util.CheckForHandInfo (
    Transform obj,
    ref SG\_HandModelInfo info ) [static]
```

Check if an object has a [SG_HandModelInfo](#) component and assign it to the info parameter.

Parameters

<i>obj</i>	
<i>info</i>	

5.54.2.8 CheckForTrackedHand()

```
static SG\_TrackedHand SG.SG_Util.CheckForTrackedHand (
    Transform obj ) [static]
```

Try to get a [SG_TrackedHand](#) that this script is attached to.

Parameters

<i>obj</i>	
<i>handScript</i>	

5.54.2.9 GetAxis()

```
static Vector3 SG.SG_Util.GetAxis (
    MovementAxis axis ) [static]
```

Returns a unit vector representing the chosen movement axis.

Parameters

<i>axis</i>	
-------------	--

Returns

5.54.2.10 GetVector()

```
static Vector3 SG.SG_Util.GetVector (
    MoveAxis axis ) [static]
```

Returns a normalized Vector representing this axis in 3D space.

Parameters

<i>axis</i>	
-------------	--

Returns

5.54.2.11 IsNegative()

```
static bool SG.SG_Util.IsNegative (
    MoveAxis axis ) [static]
```

Returns true if this axis is negative

Parameters

<i>axis</i>	
-------------	--

Returns

5.54.2.12 Map()

```
static float SG.SG_Util.Map (
    float value,
    float inMin,
    float inMax,
    float outMin,
    float outMax ) [static]
```

Map a value from one range to another.

Parameters

<i>value</i>	
<i>inMin</i>	
<i>inMax</i>	
<i>outMin</i>	
<i>outMax</i>	

Returns

5.54.2.13 NormalizeAngle() [1/2]

```
static float SG.SG_Util.NormalizeAngle (  
    float angle ) [static]
```

Normalize an angle (in degrees) such that it is within the -180...180 range.

Parameters

<i>angle</i>	
--------------	--

Returns

5.54.2.14 NormalizeAngle() [2/2]

```
static float SG.SG_Util.NormalizeAngle (  
    float angle,  
    float minAngle,  
    float maxAngle ) [static]
```

Normalize an angle (in degrees) such that it is within the -180...180 range.

Parameters

<i>angle</i>	
--------------	--

Returns

5.54.2.15 NormalizeAngles()

```
static Vector3 SG.SG_Util.NormalizeAngles (  
    Vector3 angles ) [static]
```

Normalize a set of (euler) angles to fall within a -180... 180 range.

Parameters

<i>angles</i>	
---------------	--

Returns

5.54.2.16 SetChildren()

```
static void SG.SG_Util.SetChildren (
    Transform obj,
    bool active ) [static]
```

Set all the children of the following Transform to active/inactive

Parameters

<i>obj</i>	
<i>active</i>	

5.54.2.17 SpawnSphere()

```
static GameObject SG.SG_Util.SpawnSphere (
    float worldDiameter,
    Transform parent,
    bool withCollider = true ) [static]
```

Spawn a sphere and make it a child of parent.

Parameters

<i>worldDiameter</i>	
<i>parent</i>	
<i>withCollider</i>	

Returns

5.54.2.18 ToEuler()

```
static SenseGloveCs.Kinematics.Vect3D SG.SG_Util.ToEuler (
    Vector3 euler ) [static]
```

Convert a unity eulerAngles notation into one used by the DLL.

Parameters

<i>euler</i>	
--------------	--

Returns

5.54.2.19 ToPosition() [1/2]

```
static SenseGloveCs.Kinematics.Vect3D SG.SG_Util.ToPosition (
    Vector3 pos ) [static]
```

Convert from a unity vector3 to a float[3] used in the DLL.

Parameters

<i>pos</i>	
------------	--

Returns

5.54.2.20 ToPosition() [2/2]

```
static SenseGloveCs.Kinematics.Vect3D [ ] SG.SG_Util.ToPosition (
    Vector3[] pos ) [static]
```

Convert an array of unity positions back into an array used by the DLL

Parameters

<i>pos</i>	
------------	--

Returns

5.54.2.21 ToQuaternion()

```
static SenseGloveCs.Kinematics.Quat SG.SG_Util.ToQuaternion (
    Quaternion Q ) [static]
```

Convert a unity Quaternion into a float[4] used in the DLL.

Parameters

Q	
-----	--

Returns

5.54.2.22 ToString() [1/4]

```
static string SG.SG_Util.ToString (
    float[] V ) [static]
```

Convert a float[] to a string with a greater precision than it default Unity(?) method.

Parameters

V	
-----	--

Returns

5.54.2.23 ToString() [2/4]

```
static string SG.SG_Util.ToString (
    int[] V ) [static]
```

Convert an int[] to a string with a greater precision than it default Unity(?) method.

Parameters

V	
-----	--

Returns

5.54.2.24 ToString() [3/4]

```
static string SG.SG_Util.ToString (
    Quaternion Q ) [static]
```

Convert a Unity Quaternion to a string with a greater precision than its default method.

Parameters

Q	
-----	--

Returns

5.54.2.25 ToString() [4/4]

```
static string SG.SG_Util.ToString (
    Vector3 V ) [static]
```

Convert a Unity Vector3 to a string with a greater precision than its default method.

Parameters

V	
-----	--

Returns

5.54.2.26 ToUnityEuler()

```
static Vector3 SG.SG_Util.ToUnityEuler (
    SenseGloveCs.Kinematics.Vect3D euler ) [static]
```

Convert a set of euler angles from the DLL into the Unity notation.

Parameters

<i>euler</i>	
--------------	--

Returns

5.54.2.27 ToUnityPosition() [1/2]

```
static Vector3 SG.SG_Util.ToUnityPosition (
    SenseGloveCs.Kinematics.Vect3D pos ) [static]
```

Convert a float[3] position taken from the DLL into a Unity Position.

Parameters

<i>pos</i>	
------------	--

Returns

5.54.2.28 ToUnityPosition() [2/2]

```
static Vector3 [ ] SG.SG_Util.ToUnityPosition (
    SenseGloveCs.Kinematics.Vect3D [ ] pos ) [static]
```

Convert an array of float[3] positions taken from the DLL into a Vector3[].

Parameters

<i>pos</i>	
------------	--

Returns

5.54.2.29 ToUnityQuaternion()

```
static Quaternion SG.SG_Util.ToUnityQuaternion (
    SenseGloveCs.Kinematics.Quat quat ) [static]
```

Convert a float[4] quaternion taken from the DLL into a Unity Quaternion.

Parameters

<i>quat</i>	
-------------	--

Returns

5.54.2.30 TransformRigidBody()

```
static void SG.SG_Util.TransformRigidBody (
    ref Rigidbody obj,
    Vector3 targetPosition,
    Quaternion targetRotation,
    float rotationSpeed ) [static]
```

Add a velocity / angularVelocity to a rigidbody to move towards a targetPosition and rotation

Parameters

<i>obj</i>	
<i>targetPosition</i>	
<i>targetRotation</i>	
<i>rotationSpeed</i>	

5.54.2.31 TryAddRB()

```
static Rigidbody SG.SG_Util.TryAddRB (
    GameObject obj,
    bool useGrav = false,
    bool isKinematic = false ) [static]
```

Add a rigidbody to a GameObject if one does not exist yet and apply the desired parameters.

Parameters

<i>obj</i>	
<i>useGrav</i>	
<i>isKinematic</i>	

Returns

5.54.2.32 TryRemoveRB()

```
static void SG.SG_Util.TryRemoveRB (
    GameObject obj ) [static]
```

Remove the rigidbody from a gameObject, if one exists.

Parameters

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

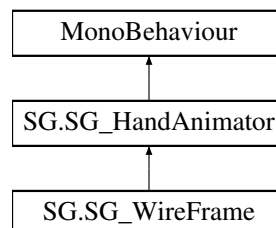
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_Util.cs

5.55 SG.SG_WireFrame Class Reference

Type of [SG_HandAnimator](#) to debug hardware- and software models.

Inheritance diagram for SG.SG_WireFrame:



Public Member Functions

- override void [UpdateHand](#) ([SG_SenseGloveData](#) data)
(Manually) Update the hand and glove model of the wireframe.
- override void [ResizeHand](#) (float[][] newLengths)
Resizes the (white) cylinders that connect to the hand.
- void [SetGlove](#) (bool active)
Enable / Disable the drawing of the Glove.
- void [SetHand](#) (bool active)
Enable / Disable the drawing of the Hand Model.

Public Attributes

- GameObject [gloveBase](#)
The GameObject that will contain the glove sections.
- GameObject [gloveSectionModel](#)
A GameObject with four children: Three cylinders representing dX dY dZ, and a sphere representing the point itself.
- GameObject [handBase](#)
The GameObject that will contain the finger sections (phalange models).
- GameObject [phalangeModel](#)
A GameObject with two children, One Cylinder representing the Phalange Lengths, and a sphere representing the joint.
- GameObject [handPalmModel](#)
A simple model representing the hand palm, to help make the model less abstract.
- GameObject [previewGroup](#)
A group of models that represent a preview of the hand, which will be deleted upon the glove connecting.
- KeyCode [toggleHandKey](#) = KeyCode.None
Key Code to manually toggle hand model rendering.
- KeyCode [toggleGloveKey](#) = KeyCode.None
Key Code to manually toggle glove model rendering.

Protected Member Functions

- override void [CollectFingerJoints](#) ()
Collect the finger joints. If these do not exist yet, try again.
- override void [SenseGlove_OnGloveLoaded](#) (object source, EventArgs args)
Override the OnGloveLoaded event so we may create the glove model.
- override void [SenseGlove_OnCalibrationFinished](#) (object source, [SG_SenseGloveHardware.GloveCalibrationArgs](#) args)
Keeps the glove index position on the same location, while shifting the other glove fingers back or forth.

Private Member Functions

- void [SetupGlove](#) ([SG_SenseGloveData](#) gloveData)
Create a new glove section based on the parameters sent from the Sense Glove.
- void [SetupFingers](#) ([SG_SenseGloveData](#) gloveData)
Create all the individual finger sections based on the glove's handModel.
- void [SetupHandPalm](#) (bool right)
Assign the proper name to the hand palm model and mirror it if nessecary.

Private Attributes

- bool [setupComplete](#) = false
Do not run the setups more than once.
- Transform[][] [gloveJoints](#)
Glove joints to which the gloveAngles are applied.

Additional Inherited Members

5.55.1 Detailed Description

Type of [SG_HandAnimator](#) to debug hardware- and software models.

5.55.2 Member Function Documentation

5.55.2.1 CollectFingerJoints()

```
override void SG.SG_WireFrame.CollectFingerJoints ( ) [protected], [virtual]
```

Collect the finger joints. If these do not exist yet, try again.

Implements [SG.SG_HandAnimator](#).

5.55.2.2 ResizeHand()

```
override void SG.SG_WireFrame.ResizeHand (
    float newLengths[][] ) [virtual]
```

Resizes the (white) cylinders that connect to the hand.

Parameters

<i>newLengths</i>	
-------------------	--

Reimplemented from [SG.SG_HandAnimator](#).

5.55.2.3 SenseGlove_OnCalibrationFinished()

```
override void SG.SG_WireFrame.SenseGlove_OnCalibrationFinished (
    object source,
    SG_SenseGloveHardware.GloveCalibrationArgs args ) [protected], [virtual]
```

Keeps the glove index position on the same location, while shifting the other glove fingers back or forth.

Parameters

<i>source</i>	
<i>args</i>	

Reimplemented from [SG.SG_HandAnimator](#).

5.55.2.4 SenseGlove_OnGloveLoaded()

```
override void SG.SG_WireFrame.SenseGlove_OnGloveLoaded (
    object source,
    EventArgs args ) [protected]
```

Override the OnGloveLoaded event so we may create the glove model.

Parameters

<i>source</i>	
<i>args</i>	

5.55.2.5 SetGlove()

```
void SG.SG_WireFrame.SetGlove (
    bool active )
```

Enable / Disable the drawing of the Glove.

Parameters

<i>active</i>	
---------------	--

5.55.2.6 SetHand()

```
void SG.SG_WireFrame.SetHand (
    bool active )
```

Enable / Disable the drawing of the Hand Model.

Parameters

<i>active</i>	
---------------	--

5.55.2.7 SetupFingers()

```
void SG.SG_WireFrame.SetupFingers (
```

```
SG_SenseGloveData gloveData ) [private]
```

Create all the individual finger sections based on the glove's handModel.

Parameters

<i>gloveData</i>	
------------------	--

5.55.2.8 SetupGlove()

```
void SG.SG_WireFrame.SetupGlove (  
    SG_SenseGloveData gloveData ) [private]
```

Create a new glove section based on the parameters sent from the Sense Glove.

Parameters

<i>gloveData</i>	
------------------	--

5.55.2.9 SetupHandPalm()

```
void SG.SG_WireFrame.SetupHandPalm (  
    bool right ) [private]
```

Assign the proper name to the hand palm model and mirror it if nessecary.

Parameters

<i>right</i>	
--------------	--

5.55.2.10 UpdateHand()

```
override void SG.SG_WireFrame.UpdateHand (  
    SG_SenseGloveData data ) [virtual]
```

(Manually) Update the hand and glove model of the wireframe.

Parameters

<i>data</i>	
-------------	--

Reimplemented from [SG.SG_HandAnimator](#).

5.55.3 Member Data Documentation

5.55.3.1 gloveBase

`GameObject SG.SG_WireFrame.gloveBase`

The GameObject that will contain the glove sections.

5.55.3.2 gloveJoints

`Transform [][] SG.SG_WireFrame.gloveJoints [private]`

Glove joints to which the gloveAngles are applied.

5.55.3.3 gloveSectionModel

`GameObject SG.SG_WireFrame.gloveSectionModel`

A GameObject with four children: Three cylinders representing dX dY dZ, and a sphere representing the point itself.

5.55.3.4 handBase

`GameObject SG.SG_WireFrame.handBase`

The GameObject that will contain the finger sections (phalange models).

5.55.3.5 handPalmModel

`GameObject SG.SG_WireFrame.handPalmModel`

A simple model representing the hand palm, to help make the model less abstract.

5.55.3.6 phalangeModel

```
GameObject SG.SG_WireFrame.phalangeModel
```

A GameObject with two children, One Cylinder representing the Phalange Lengths, and a sphere representing the joint.

5.55.3.7 previewGroup

```
GameObject SG.SG_WireFrame.previewGroup
```

A group of models that represent a preview of the hand, which will be deleted upon the glove connecting.

5.55.3.8 setupComplete

```
bool SG.SG_WireFrame.setupComplete = false [private]
```

Do not run the setups more than once.

5.55.3.9 toggleGloveKey

```
KeyCode SG.SG_WireFrame.toggleGloveKey = KeyCode.None
```

Key Code to manually toggle glove model rendering.

5.55.3.10 toggleHandKey

```
KeyCode SG.SG_WireFrame.toggleHandKey = KeyCode.None
```

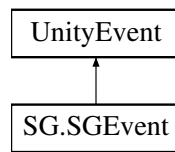
Key Code to manually toggle hand model rendering.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_WireFrame.cs

5.56 SG.SGEvent Class Reference

Inheritance diagram for SG.SGEvent:



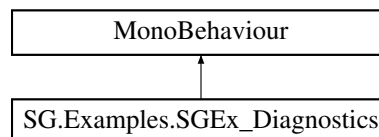
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Util/SG_Util.cs

5.57 SG.Examples.SGEx_Diagnostics Class Reference

Allows one to access certain Sense Glove functions using the keys on the keyboard.

Inheritance diagram for SG.Examples.SGEx_Diagnostics:



Public Member Functions

- void **CalibrateWrist** ()
- void **SetFFB** (int finger, int level)
- void **SetFFB** (int[] levels)
- void **ToggleFFB** ()
- void **SetBuzz** (int finger, int level)
- void **SetBuzz** (int[] levels)
- void **ToggleBuzz** ()
- void **BeginTestThumper** (bool loops)
- void **EndTestThumper** ()
- void **SetBrakeBuzz** (int[] ffb, int[] buzz)
- void **EngageAllFeedback** ()
- void **EndAllFeedback** ()
- void **ToggleAllFeedback** ()

Public Attributes

- [SG_SenseGloveHardware](#) **senseGlove**
- Text **instructText**
- bool **hotKeysEnabled** = true
- KeyCode **toggleAllFFBKey** = KeyCode.Return
- KeyCode **toggleAllBuzzKey** = KeyCode.B
- KeyCode **testThumperKey** = KeyCode.T
- KeyCode **fullLoadKey** = KeyCode.F
- KeyCode **calibrateWristKey** = KeyCode.P
- GameObject[] **disableUntilFound** = new GameObject[0]

Properties

- string **Instructions** [set]
- bool **CanTestFFB** [get]
- int[] **FFBLvls** [get, private set]
- bool **AllFFBOn** [get]
- bool **CanTestBuzzMotors** [get]
- int[] **BuzzMotorLvls** [get, private set]
- bool **AllBuzzOn** [get]
- bool **CanTestThumper** [get]
- bool **ThumperOn** [get, private set]
- bool **AllFeedbackOn** [get]

Private Member Functions

- void **UpdateDiagnostics** ()
- void **SendThumperCmd** (ThumperEffect effect)
- void **UpdateThumper** ()
- void **Awake** ()
- void **Start** ()
- void **Update** ()

Private Attributes

- bool **firstLink** = false
- float **thumperTimer** = 0
- float **thumpTime** = 1.2f
- SenseGloveCs.ThumperEffect **thumperToTest** = SenseGloveCs.ThumperEffect.Impact_Thump_100
- string **baseInst** = ""

5.57.1 Detailed Description

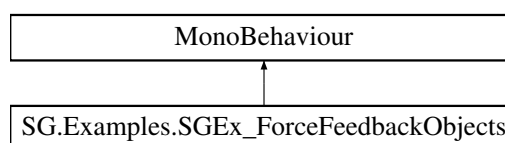
Allows one to access certain Sense Glove fuctions using the keys on the keyboard.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↵
Diagnostics.cs

5.58 SG.Examples.SGEx_ForceFeedbackObjects Class Reference

Inheritance diagram for SG.Examples.SGEx_ForceFeedbackObjects:



Public Member Functions

- void **CalibrateWrist** ()
- void **NextObject** ()
- void **PreviousObject** ()

Static Public Member Functions

- static bool **CheckHandOpen** ([SG_TrackedHand](#) hand)

Public Attributes

- [SG_TrackedHand](#) **leftHand**
- KeyCode **nextObjKey** = KeyCode.D
- KeyCode **prevObjKey** = KeyCode.A
- KeyCode **calibrateWristKey** = KeyCode.P
- Button **nextButton**
- Button **wristButton**
- GameObject[] **ffbObjects** = new GameObject[0]
- Text **objectText**

Protected Member Functions

- void **SetRelevantScripts** ([SG_TrackedHand](#) hand, bool active)
- void **SetObject** (int index, bool active)
- int **WrapIndex** (int newIndex)

Protected Attributes

- [SG_TrackedHand](#) **activeHand** = null
- [SG_Breakable](#)[] **breakables** = new [SG_Breakable](#)[0]
- int **objIndex** = -1
- bool **allowedSwap** = false
- float **openTime** = 0.2f
- float **openedTimer** = 0
- float **breakableResetTime** = 1.0f
- float **breakableTimer** = 1.0f

Properties

- bool **ButtonsActive** [get, set]
- bool **ButtonsInteractable** [get, set]

Private Member Functions

- void **ConnectObjects** ([SG_TrackedHand](#) hand)
- void **Awake** ()
- void **Start** ()
- void **Update** ()

Private Attributes

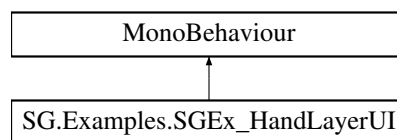
- [SG_TrackedHand](#) **rightHand**
- Button **previousButton**

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↔
ForceFeedbackObjects.cs

5.59 SG.Examples.SGEx_HandLayerUI Class Reference

Inheritance diagram for SG.Examples.SGEx_HandLayerUI:



Public Types

- enum **ShowingLayer** {
None, HandModelLayer, AnimationLayer, FeedbackLayer,
GrabLayer, RigidbodyLayer, PhysicsLayer, All }

Public Member Functions

- void **NextStep** ()
- void **PreviousStep** ()
- void **GoToStep** (int index)
- void **SetInstructions** (int index)
- void **SetLayer** (int index)
- void **CalibrateWrist** ()
- void **SetLayerObjects** (int L, bool active)
- void **SetAllObjects** (bool active)
- void **ShowLayer** (ShowingLayer layer)
- void **UpdateOverview** (ShowingLayer layer)

Public Attributes

- Text **instructionsUI**
- Button **prevBtn**
- [SG_TrackedHand](#) **leftHand**
- [SG_TrackedHand](#) **rightHand**
- KeyCode **nextKey** = KeyCode.D
- KeyCode **prevKey** = KeyCode.A
- KeyCode **wristKey** = KeyCode.P
- int **currStep** = -1
- int **mustConnectStep** = 6
- Button **nextButton**
- Text[] **overviewTexts** = new Text[0]
- Color **textHLColor** = Color.white
- Color **textDisabledColor** = Color.gray
- GameObject[] **feedbackObjects** = new GameObject[0]
- GameObject[] **grabLayerObjects** = new GameObject[0]
- GameObject[] **rigidBodyObjects** = new GameObject[0]
- GameObject[] **physicsObjects** = new GameObject[0]

Protected Attributes

- ShowingLayer **showing** = ShowingLayer.All
- [SG_TrackedHand](#) **activeHand** = null
- string[] **instructionTexts**
- ShowingLayer[] **linkedLayers**

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

- Button **nextBtn**
- Button **calibrateWristBtn**
- Button **prevButton**
- GameObject[][] **layerObjects** = new GameObject[0][]

5.59.1 Member Data Documentation

5.59.1.1 instructionTexts

```
string [] SG.Examples.SGEx_HandLayerUI.instructionTexts [protected]
```

Initial value:

```
= new string[]
{
    "This example will run you through the SenseGlove hand prefab and its different 'layers'",
    "The SenseGlove hand consists of 7 layers: A HandModel, Animator, Feedback Layer, Grab Layer, Rigidbody Layer and PhysicsTracking layer.",
    "Each of these layers can be enabled/disabled by turning their gameobjects on/off, either code or through the inspector. Nearly all of them can be safely deleted in their entirety if their functionality is not required.",
    "The TrackedHand script, attached to the root of the prefab, is your main access point to all layers. It can be set to follow a specific GameObject, with preprogrammed offsets for certain tracking hardware.",
    "The HandModel layer contains the 3D assets to draw and position the hand. The SG_HandInfo script tells the other SenseGlove Scripts where the joints are located.",

    "One can swap out the hand model for another by replacing the HandModel's children, and assigning the proper transforms in the SG_HandInfo script via code or the inspector.",

    "Unless you want to manually set Tracking targets for the colliders of the other layers, the SG_HandModelInfo script is the only one that should not be deleted.",
    "The animation layer is responsible for animating the hand using the SG_HandAnimation script. It can be disabled if you wish to animate the hand model yourself.",
    "The Feedback layer contains colliders that respond to impacts and to SenseGlove_Material Scripts. Each frame, the SG_HandFeedback script collects the appropriate forces and sends these to the SenseGlove.",
    "The Grab Layer allows one to pick up and manipulate objects with SG_Interactive scripts. If you already have manipulation scripts (such as through VRTK), you can disable this layer and replace it with your own.",
    "The Rigidbody layer adds rigidbodies that allow one to push and hold other rigidbody objects. This gameobject and its children can be placed on their own layer, or be told to ignore certain colliders.",
    "The PhysicsTracking layer contains non-trigger colliders that prevent the SG_TrackedHand from passing through non-grabable objects, provided that its 'trackingMethod' property is set to be 'PhysicsBased'.",
    "This separation of layers allows for a hand model that can be adjusted to your needs, and which allows different physics behaviours without touching the actual 3D Model.",
}
```

5.59.1.2 linkedLayers

```
ShowingLayer [] SG.Examples.SGEx_HandLayerUI.linkedLayers [protected]
```

Initial value:

```
= new ShowingLayer[]
{
    ShowingLayer.None,
    ShowingLayer.None,
    ShowingLayer.None,
    ShowingLayer.None,
    ShowingLayer.HandModelLayer,
    ShowingLayer.HandModelLayer,
    ShowingLayer.HandModelLayer,
    ShowingLayer.AnimationLayer,
    ShowingLayer.FeedbackLayer,
    ShowingLayer.GrabLayer,
    ShowingLayer.RigidbodyLayer,
    ShowingLayer.PhysicsLayer,
    ShowingLayer.All,
}
```

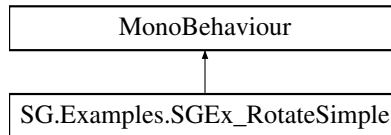
The documentation for this class was generated from the following file:

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↵HandLayerUI.cs

5.60 SG.Examples.SGEx_RotateSimple Class Reference

A script to rotate an object around a specified axis

Inheritance diagram for SG.Examples.SGEx_RotateSimple:



Public Member Functions

- void **ResetRotation** ()

Public Attributes

- [MovementAxis](#) **moveAround** = MovementAxis.Y
- float **rotationSpeed** = 10f
- bool **resetOnEnable** = false

Properties

- Quaternion **OriginalRotation** [get, protected set]

Private Member Functions

- void **Awake** ()
- void **OnEnable** ()
- void **Update** ()

5.60.1 Detailed Description

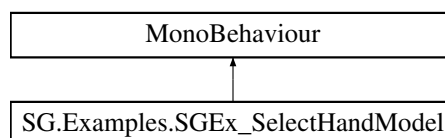
A script to rotate an object around a specified axis

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↔
RotateSimple.cs

5.61 SG.Examples.SGEx_SelectHandModel Class Reference

Inheritance diagram for SG.Examples.SGEx_SelectHandModel:



Public Member Functions

- void **SetSolver** (SenseGloveCs.Solver solv)
- void **SetModels** (bool left, bool right)

Public Attributes

- [SG_SenseGloveHardware](#) **leftGlove**
- KeyCode **swapHandsKey** = KeyCode.Return

Private Member Functions

- void **Start** ()
- void **Update** ()

Private Attributes

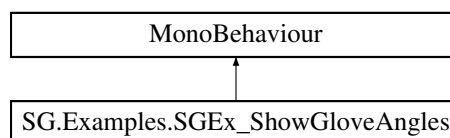
- [SG_SenseGloveHardware](#) **rightGlove**
- GameObject **leftHandModel**
- GameObject **rightHandModel**
- bool **leftReady** = false
- bool **rightReady** = false

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↔
SelectHandModel.cs

5.62 SG.Examples.SGEx_ShowGloveAngles Class Reference

Inheritance diagram for SG.Examples.SGEx_ShowGloveAngles:



Public Attributes

- [SG_SenseGloveHardware](#) **senseGlove**
- GridLayoutGroup **angleCanvas**

Private Member Functions

- Text **CreateTextBox** (string textString, Font font, Transform parent, string objName="textBox")
- void **SetupAngleUI** ()
- void **UpdateAngleUI** (float[][] sensors)
- void **Update** ()

Private Attributes

- Text[][] **angleBoxes**
- bool **setup** = false

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Examples/Resources/SGEx_↵
ShowGloveAngles.cs

5.63 SG.SG_SnapDropZone.SnapProps Class Reference

Contains parameters that assist in snapping/unsnapping to a SnapZone.

Public Member Functions

- [SnapProps](#) (SG_Grabable grabable)
Create a new instance of [SnapProps](#), based on a single Grabable's properties.
- void [RestoreProperties](#) (SG_Grabable grabable)
Restore properties back to their original state(s).
- void [CreateJoint](#) (SG_Grabable grabable, Rigidbody snapZoneBody, float breakForce)
Create a Physics Joint between a grabable and a snapZone.
- void [BreakJoint](#) ()
Destroy the PhysicsJoint if it was created in the past.

Public Attributes

- bool [wasInteractable](#)
Determines if this object was Interactable before it snapped to this zone.
- bool [wasKinematic](#)
Determines if the Rigidbody was Kinematic before it snapped.
- bool [usedGravity](#)
Determines if the Rigidbody used Gravity before it snapped.
- Joint [myJoint](#)
Optional PhysicsJoint that is created if the Object is picked up using FixedJoints.
- Transform [oldParent](#) = null
The old parent of the object
- bool [isSnapped](#) = false
Lets the zone know if this object has snapped yet. False by default.

5.63.1 Detailed Description

Contains parameters that assist in snapping/unsnapping to a SnapZone.

Placed inside a class to reduce the amount of List<> parameters.

5.63.2 Constructor & Destructor Documentation

5.63.2.1 SnapProps()

```
SG.SG_SnapDropZone.SnapProps.SnapProps (
    SG_Grabable grabable )
```

Create a new instance of [SnapProps](#), based on a single Grabable's properties.

Parameters

<i>grabable</i>	
-----------------	--

5.63.3 Member Function Documentation

5.63.3.1 BreakJoint()

```
void SG.SG_SnapDropZone.SnapProps.BreakJoint ( )
```

Destroy the PhysicsJoint if it was created in the past.

5.63.3.2 CreateJoint()

```
void SG.SG_SnapDropZone.SnapProps.CreateJoint (
    SG_Grabable grabable,
    Rigidbody snapZoneBody,
    float breakForce )
```

Create a Physics Joint between a grabable and a snapZone.

Parameters

<i>grabable</i>	
<i>snapZoneBody</i>	
<i>breakForce</i>	

5.63.3.3 RestoreProperties()

```
void SG.SG_SnapDropZone.SnapProps.RestoreProperties (
    SG_Grabable grabable )
```

Restore properties back to their original state(s).

Parameters

<i>grabable</i>	
-----------------	--

5.63.4 Member Data Documentation

5.63.4.1 isSnapped

```
bool SG.SG_SnapDropZone.SnapProps.isSnapped = false
```

Lets the zone know if this object has snapped yet. False by default.

5.63.4.2 myJoint

```
Joint SG.SG_SnapDropZone.SnapProps.myJoint
```

Optional PhysicsJoint that is created if the Object is picked up using FixedJoints.

5.63.4.3 oldParent

```
Transform SG.SG_SnapDropZone.SnapProps.oldParent = null
```

The old parent of the object

5.63.4.4 usedGravity

```
bool SG.SG_SnapDropZone.SnapProps.usedGravity
```

Determines if the Rigidbody used Gravity before it snapped.

5.63.4.5 wasInteractable

```
bool SG.SG_SnapDropZone.SnapProps.wasInteractable
```

Determines if this object was Interactable before it snapped to this zone.

5.63.4.6 wasKinematic

```
bool SG.SG_SnapDropZone.SnapProps.wasKinematic
```

Determines if the Rigidbody was Kinematic before it snapped.

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

- D:/Gitlab/SenseGloveAPI/Unity/SG_UnityPlugin_v1/Assets/SenseGlove/Scripts/Controls/SG_SnapDrop↔
Zone.cs

Index

- [_grabReference](#)
 - [SG.SG_Dial, 81](#)
 - [_grabScript](#)
 - [SG.SG_Interactable, 191](#)
- [abd](#)
 - [SG.Calibration.CalibrationPose, 24](#)
- [absDeformPosition](#)
 - [SG.Materials.Deformation, 26](#)
- [absEntryVector](#)
 - [SG.Materials.Deformation, 26](#)
- [absoluteCalibratedWrist](#)
 - [SG.SG_SenseGloveData, 223](#)
- [absoluteWrist](#)
 - [SG.SG_SenseGloveData, 224](#)
- [activationThreshold](#)
 - [SG.SG_HandDetector, 150](#)
- [activationTime](#)
 - [SG.SG_HandDetector, 150](#)
- [actualMoveDirection](#)
 - [SG.SG_Drawer, 90](#)
- [AddDeform](#)
 - [SG.SG_MeshDeform, 204](#)
- [AddDeformation](#)
 - [SG.SG_MeshDeform, 204](#)
- [AddEntry](#)
 - [SG.SG_HandDetector, 147](#)
- [AddObject](#)
 - [SG.SG_DropZone, 94](#)
 - [SG.SG_SnapDropZone, 252](#)
- [AddRigidBodies](#)
 - [SG.SG_HandRigidBodies, 162](#)
- [AddTarget](#)
 - [SG.SG_DropZone, 94](#)
- [AddToList](#)
 - [SG.SG_HoverCollider, 180](#)
- [AllObjectsDetected](#)
 - [SG.SG_DropZone, 101](#)
- [AngleIndex](#)
 - [SG.SG_Dial, 78](#)
- [angleIndex](#)
 - [SG.SG_Dial, 81](#)
- [anglOffset](#)
 - [SG.SG_Dial, 81](#)
- [angularVelocities](#)
 - [SG.SG_GrabScript, 129](#)
- [AppendButtonText](#)
 - [SG.SG_Util, 269](#)
- [atRest](#)
 - [SG.SG_MeshDeform, 208](#)
- [attachMethod](#)
 - [SG.SG_Grabable, 117](#)
- [AttachObject](#)
 - [SG.SG_SnapDropZone, 252](#)
- [AttachScript](#)
 - [SG.SG_FingerFeedback, 104](#)
- [AttachType](#)
 - [SG, 11](#)
- [audioToPlay](#)
 - [SG.SG_HandTrigger, 169](#)
- [autoConnect](#)
 - [SG.SG_SenseGloveHardware, 241](#)
- [autoSetup](#)
 - [SG.SG_Hinge, 176](#)
- [Average](#)
 - [SG.SG_Util, 270](#)
- [AxisIndex](#)
 - [SG.SG_Util, 270](#)
- [baseGrabAngles](#)
 - [SG.SG_GestureGrabScript, 111](#)
- [baseReleaseAngles](#)
 - [SG.SG_GestureGrabScript, 111](#)
- [baseTxt](#)
 - [SG.Calibration.SG_CalibrationSequence, 62](#)
- [BeginInteraction](#)
 - [SG.SG_Interactable, 186](#)
- [brakeQueue](#)
 - [SG.SG_SenseGloveHardware, 241](#)
- [Break](#)
 - [SG.SG_Breakable, 48](#)
 - [SG.SG_BreakableContainer, 54](#)
- [breakable](#)
 - [SG.SG_Material, 200](#)
- [BreakJoint](#)
 - [SG.SG_Grabable, 114](#)
 - [SG.SG_SnapDropZone.SnapProps, 298](#)
- [breakParticles](#)
 - [SG.SG_Breakable, 50](#)
- [breakSound](#)
 - [SG.SG_Breakable, 50](#)
- [brokenBy](#)
 - [SG.SG_Material, 200](#)
- [brokenDeform](#)
 - [SG.SG_Breakable, 50](#)
- [brokenMaterial](#)
 - [SG.SG_Breakable, 50](#)
- [brokenObject](#)
 - [SG.SG_Breakable, 51](#)
- [brokenShards](#)

- SG.SG_BreakableContainer, 56
- buzz_CMD_Time
 - SG.SG_HandTrigger, 169
- buzz_Enabled
 - SG.SG_SenseGloveHardware, 241
- buzzTimer
 - SG.SG_HandTrigger, 169
- calbrUsing
 - SG.Calibration.CalibrationPose, 24
- CalculateAngularVelocity
 - SG.SG_Util, 271
- CalculateForce
 - SG.SG_Material, 198
- CalculateHaptics
 - SG.SG_Material, 198
- CalculateOffsets
 - SG.SG_Util, 271
- CalculateResponseForce
 - SG.SG_Material, 198
- CalibrateCurrentStep
 - SG.Calibration.SG_CalibrationSequence, 60
- CalibrateParameters
 - SG.Calibration.CalibrationPose, 20
- CalibratePose
 - SG.Calibration.SG_CalibrationSequence, 60
- CalibrateWrist
 - SG.SG_HandAnimator, 138
- calibrationArguments
 - SG.SG_SenseGloveHardware, 242
- CalibrationFinished
 - SG.SG_SenseGloveHardware, 245
- CalibrationFinishedEventHandler
 - SG.SG_SenseGloveHardware, 233
- CalibrationPose
 - SG.Calibration.CalibrationPose, 19, 20
- calibrationStep
 - SG.SG_SenseGloveData, 224
- calibrDir
 - SG.Calibration.SG_CalibrationStorage, 68
- CallObjectDetect
 - SG.SG_DropZone, 95
 - SG.SG_SnapDropZone, 252
- CallObjectRemoved
 - SG.SG_DropZone, 95
- CalPose
 - SG.Calibration.SG_CalibrationSequence, 59
- CalStage
 - SG.Calibration.SG_CalibrationSequence, 60
- CanImpact
 - SG.SG_BasicFeedback, 45
- CanInteract
 - SG.SG_GestureGrabScript, 110
 - SG.SG_GrabScript, 123
 - SG.SG_Interactive, 186
 - SG.SG_PhysicsGrab, 212
- CanRelease
 - SG.SG_GrabScript, 123
 - SG.SG_PhysicsGrab, 212
- canTransfer
 - SG.SG_Grabable, 117
- changes
 - SG.Calibration.SG_CalibrationSequence, 62
- CheckCalibration
 - SG.SG_SenseGloveHardware, 234
- CheckConnection
 - SG.SG_SenseGloveHardware, 234
- CheckConnections
 - SG.SG_DeviceManager, 75
- CheckDetectionTimes
 - SG.SG_DropZone, 95
- CheckForDeviceManager
 - SG.SG_SenseGloveHardware, 234
- CheckForHandInfo
 - SG.SG_Util, 272
- CheckForScripts
 - SG.SG_AutoHandAnimation, 39
 - SG.SG_GrabScript, 124
 - SG.SG_HandAnimator, 138
 - SG.SG_HandFeedback, 154
 - SG.SG_HandRigidBodies, 162
 - SG.SG_PhysicsGrab, 213
 - SG.SG_TrackedHand, 262
- CheckForTrackedHand
 - SG.SG_Util, 272
- CheckGestures
 - SG.SG_PhysicsGrab, 213
- CheckLimits
 - SG.SG_Hinge, 173
- CheckObjectEnter
 - SG.SG_DropZone, 95
- CheckPickupRef
 - SG.SG_Grabable, 114
- checkStayTime
 - SG.SG_DropZone, 99
- checkStayTimer
 - SG.SG_DropZone, 99
- checkTime
 - SG.SG_Breakable, 51
- CheckUnbreak
 - SG.SG_Breakable, 48
- ClearConnections
 - SG.SG_DeviceManager, 75
- ClearDeformations
 - SG.SG_MeshDeform, 205
- ClearHeldObjects
 - SG.SG_GrabScript, 124
- ClearObjects
 - SG.SG_DropZone, 96
- ClearTouchedObjects
 - SG.SG_HoverCollider, 180
- closedHandThresholds
 - SG.SG_PhysicsGrab, 215
- CollectCorrections
 - SG.SG_HandAnimator, 138
- CollectFingerJoints
 - SG.SG_AutoHandAnimation, 39

- SG.SG_HandAnimator, [138](#)
- SG.SG_WireFrame, [284](#)
- CollectHandParameters
 - SG.SG_HandAnimator, [138](#)
- CollectMeshData
 - SG.SG_MeshDeform, [205](#)
- ColliderDistances
 - SG.SG_HandFeedback, [156](#)
- collidersInside
 - SG.SG_HoverCollider, [183](#)
- CollisionEnabled
 - SG.SG_TrackedBody, [259](#)
- CollisionsEnabled
 - SG.SG_HandRigidBodies, [165](#)
- commonOriginPos
 - SG.SG_SenseGloveData, [224](#)
- commonOriginRot
 - SG.SG_SenseGloveData, [224](#)
- connectedGlove
 - SG.SG_HandFeedback, [156](#)
- connectedTo
 - SG.SG_GrabZone, [135](#)
- ConnectionIndex
 - SG.SG_GrabZone, [133](#)
- ConnectionMethod
 - SG.SG_SenseGloveHardware, [233](#)
- connectionMethod
 - SG.SG_SenseGloveHardware, [242](#)
- ConnectJoint
 - SG.SG_Grabable, [115](#)
- ConnectTo
 - SG.SG_GrabZone, [133](#)
- ContainsSenseGlove
 - SG.SG_HandDetector, [147](#)
- contentPositions
 - SG.SG_BreakableContainer, [56](#)
- contentRotations
 - SG.SG_BreakableContainer, [56](#)
- contents
 - SG.SG_BreakableContainer, [56](#)
- contentsContainer
 - SG.SG_BreakableContainer, [56](#)
- cooldownTimer
 - SG.SG_BasicFeedback, [42](#)
- CreateJoint
 - SG.SG_SnapDropZone.SnapProps, [298](#)
- currAngle
 - SG.SG_Dial, [81](#)
- Custom
 - SG.Materials, [15](#)
 - SG.SG_TrackedHand, [261](#)
- dataLoaded
 - SG.SG_SenseGloveData, [224](#)
- debugDirections
 - SG.SG_FingerFeedback, [107](#)
- DebugEnabled
 - SG.SG_BasicFeedback, [45](#)
 - SG.SG_GrabScript, [131](#)
- SG.SG_HandFeedback, [157](#)
- SG.SG_HandModelInfo, [160](#)
- SG.SG_HandRigidBodies, [165](#)
- SG.SG_PhysicsGrab, [217](#)
- SG.SG_SimpleTracking, [249](#)
- debugGroup
 - SG.SG_HandAnimator, [142](#)
- debugText
 - SG.SG_HandAnimator, [142](#)
- Default
 - SG, [11](#), [13](#)
 - SG.Materials.MaterialProps, [36](#)
 - SG.SG_TrackedHand, [262](#)
- Deformation
 - SG.Materials.Deformation, [26](#)
- deformationQueue
 - SG.SG_MeshDeform, [208](#)
- DeformMesh
 - SG.SG_MeshDeform, [205](#)
- deforms
 - SG.SG_MeshDeform, [209](#)
- deformScript
 - SG.SG_Material, [200](#)
- deformVerts
 - SG.SG_MeshDeform, [209](#)
- DetachScript
 - SG.SG_FingerFeedback, [104](#)
- detectedColliders
 - SG.SG_HandDetector, [150](#)
- detectedGloves
 - SG.SG_HandDetector, [150](#)
- detectHeldObjects
 - SG.SG_DropZone, [99](#)
- detectIndex
 - SG.SG_HandDetector, [150](#)
- detectionMethod
 - SG.SG_HandDetector, [151](#)
- detectionTime
 - SG.SG_DropZone, [99](#)
- detectionTimes
 - SG.SG_HandDetector, [151](#)
- DetectionType
 - SG.SG_HandDetector, [147](#)
- detectMiddle
 - SG.SG_HandDetector, [151](#)
- detectPinky
 - SG.SG_HandDetector, [151](#)
- detectRing
 - SG.SG_HandDetector, [151](#)
- detectThumb
 - SG.SG_HandDetector, [151](#)
- DeviceDetectedArgs
 - SG.SG_DeviceManager.DeviceDetectedArgs, [27](#)
- DeviceID
 - SG.SG_DeviceManager.DeviceDetectedArgs, [28](#)
- deviceId
 - SG.SG_SenseGloveData, [224](#)
- DeviceIndex

- SG.SG_DeviceManager.DeviceDetectedArgs, 28
- deviceScannerPresent
 - SG.SG_SenseGloveHardware, 242
- devicesToLink
 - SG.SG_DeviceManager, 76
- DifferenceFromWrist
 - SG.SG_HandAnimator, 138
- Disabled
 - SG.SG_TrackedHand, 262
- disablesInteraction
 - SG.SG_SnapDropZone, 255
- DisplaceType
 - SG.Materials, 15
- displaceType
 - SG.SG_MeshDeform, 209
- DisposeDevice
 - SG.SG_DeviceLink, 73
 - SG.SG_SenseGloveHardware, 234
- distance
 - SG.Materials.Deformation, 26
- DistanceInCollider
 - SG.SG_FingerFeedback, 108
- DLL_debugLevel
 - SG.SG_Debugger, 70
- DoorClosed
 - SG.SG_Door, 84
- DoorOpened
 - SG.SG_Door, 84
- DrawerClosed
 - SG.SG_Drawer, 91
- DrawerOpened
 - SG.SG_Drawer, 91
- dropProperties
 - SG.SG_DropZone, 99
- DropZoneArgs
 - SG.SG_DropZone.DropZoneArgs, 29
- DropZoneEventHandler
 - SG.SG_DropZone, 96
- effectToShow
 - SG.SG_HandTrigger, 169
- Egg
 - SG.Materials, 15
- elapsedTime
 - SG.SG_GrabScript, 129
- Empty
 - SG.SG_SenseGloveData, 228
- EndGlobal
 - SG.Calibration.SG_CalibrationSequence, 60
- EndInteractAllowed
 - SG.SG_Interactive, 186
- EndInteraction
 - SG.SG_GrabScript, 124
 - SG.SG_Interactive, 187
- endPopup
 - SG.Calibration.SG_CalibrationSequence, 63
- entryOrigin
 - SG.SG_FingerFeedback, 107
- entryPoint
 - SG.SG_FingerFeedback, 107
- eventFired
 - SG.SG_HandDetector, 152
- feedbackScript
 - SG.SG_TrackedHand, 263
- FindForceDirection
 - SG.SG_FingerFeedback, 104
- fingerCorrection
 - SG.SG_HandAnimator, 142
- fingerDebug
 - SG.SG_HandModelInfo, 159
- fingerFeedbackScripts
 - SG.SG_HandFeedback, 156
- FingerJoints
 - SG.SG_HandModelInfo, 160
- fingerJoints
 - SG.SG_HandAnimator, 142
- FingerLengths
 - SG.SG_SenseGloveHardware, 244
- fingerObjs
 - SG.SG_HandRigidBodies, 164
- fingerPalm
 - SG.SG_Interactive, 191
- fingerThumb
 - SG.SG_Interactive, 191
- FinishCalibration
 - SG.SG_SenseGloveHardware, 234
- FireDetectEvent
 - SG.SG_HandDetector, 148
 - SG.SG_HandTrigger, 167
- FireHapticFeedback
 - SG.SG_HandTrigger, 167
- FireRemoveEvent
 - SG.SG_HandDetector, 148
 - SG.SG_HandTrigger, 168
- firmwareVersion
 - SG.SG_SenseGloveData, 224
- FixedJoint
 - SG, 13
 - SG.SG_SnapDropZone, 252
- FlushCmds
 - SG.SG_SenseGloveHardware, 235
- Follow
 - SG, 13
- ForceClosed
 - SG.SG_Drawer, 87
- ForceLevel
 - SG.SG_FingerFeedback, 108
- ForceOpen
 - SG.SG_Drawer, 87
- foreArmTransform
 - SG.SG_HandModelInfo, 159
- foreArmTransfrom
 - SG.SG_HandAnimator, 142
- FunctionCall
 - SG, 13
- generalButtonText

- SG.Calibration.SG_CalibrationSequence, 63
- GetAngle
 - SG.SG_Dial, 78
 - SG.SG_Hinge, 173
- GetAngularVelocity
 - SG.SG_GrabScript, 124
- GetAxis
 - SG.SG_Dial, 79
 - SG.SG_Util, 272
- GetCalibrationValues
 - SG.Calibration.SG_CalibrationSequence, 60
- GetChainVariables
 - SG.SG_SenseGloveData, 221
- GetFilename
 - SG.Calibration.SG_CalibrationStorage, 66
- GetFingerLengths
 - SG.SG_SenseGloveData, 222
- GetFingerTip
 - SG.SG_HandModelInfo, 158
- GetFist
 - SG.Calibration.CalibrationPose, 20
- GetFullFist
 - SG.Calibration.CalibrationPose, 21
- GetFullOpen
 - SG.Calibration.CalibrationPose, 21
- GetGrabables
 - SG.SG_PhysicsGrab, 213
- GetHardware
 - SG.SG_GrabScript, 125
 - SG.SG_HandAnimator, 139
 - SG.SG_HandFeedback, 154
 - SG.SG_HandRigidBody, 162
- GetHingeAngle
 - SG.SG_Hinge, 173
- GetInteractableScript
 - SG.SG_HoverCollider, 180
- GetJointPositions
 - SG.SG_SenseGloveData, 222
- GetLinkVariables
 - SG.SG_SenseGloveData, 222
- GetMatching
 - SG.SG_PhysicsGrab, 213, 214
- GetMaterialScript
 - SG.SG_FingerFeedback, 105
- GetOpenHand
 - SG.Calibration.CalibrationPose, 21
- GetSide
 - SG.SG_SenseGloveData, 222
- GetThumbAbd
 - SG.Calibration.CalibrationPose, 22
- GetThumbFlexed
 - SG.Calibration.CalibrationPose, 22
- GetThumbNoAbd
 - SG.Calibration.CalibrationPose, 23
- GetThumbsUp
 - SG.Calibration.CalibrationPose, 23
- GetTouchIndex
 - SG.SG_Interactable, 187
- GetVector
 - SG.SG_Util, 272
- GetVelocity
 - SG.SG_GrabScript, 125
- gloveAngles
 - SG.SG_SenseGloveData, 225
- gloveBase
 - SG.SG_WireFrame, 287
- GloveCalibrationArgs
 - SG.SG_SenseGloveHardware.GloveCalibrationArgs, 32
- GloveData
 - SG.SG_SenseGloveHardware, 244
- GloveDetected
 - SG.SG_HandDetector, 153
- GloveDetectionArgs
 - SG.SG_HandDetector.GloveDetectionArgs, 34
- GloveEventHandler
 - SG.SG_SenseGloveHardware, 235
- gloveJoints
 - SG.SG_WireFrame, 287
- gloveLengths
 - SG.SG_SenseGloveData, 225
- GloveLoaded
 - SG.SG_SenseGloveHardware, 245
- glovePositions
 - SG.SG_SenseGloveData, 225
- GloveReady
 - SG.SG_SenseGloveHardware, 244
- GloveRemoved
 - SG.SG_HandDetector, 153
- gloveRotations
 - SG.SG_SenseGloveData, 225
- gloveSectionModel
 - SG.SG_WireFrame, 287
- GloveSide
 - SG, 11
- gloveSide
 - SG.SG_SenseGloveData, 225
- GlovesInside
 - SG.SG_HandDetector, 148
- GloveUnLoaded
 - SG.SG_SenseGloveHardware, 245
- gloveValues
 - SG.SG_SenseGloveData, 225
- gloveVersion
 - SG.SG_SenseGloveData, 226
- GoToMainStage
 - SG.Calibration.SG_CalibrationSequence, 61
- grabable
 - SG.SG_DropZone.DropZoneArgs, 30
- Grabable_InteractionBegun
 - SG.SG_SnapDropZone, 253
- Grabable_InteractionEnded
 - SG.SG_SnapDropZone, 253
- Grabable_ObjectReset
 - SG.SG_SnapDropZone, 253
- grabAnchor

- SG.SG_GrabScript, 129
- grabbing
 - SG.SG_GestureGrabScript, 111
- GrabEventHandler
 - SG.SG_GrabScript, 125
- grabOffset
 - SG.SG_Drawer, 90
 - SG.SG_Grabable, 118
- grabReference
 - SG.SG_Drawer, 90
 - SG.SG_Grabable, 118
 - SG.SG_GrabScript, 129
 - SG.SG_Hinge, 176
- grabRotation
 - SG.SG_Grabable, 118
- GrabScript
 - SG.SG_Interactive, 193
- grabScript
 - SG.SG_TrackedHand, 263
- GrabType
 - SG, 13
- Hand
 - SG.SG_GrabScript, 131
 - SG.SG_HandAnimator, 144
 - SG.SG_HandFeedback, 157
 - SG.SG_HandRigidBody, 165
- handAngles
 - SG.SG_SenseGloveData, 226
- handAnimation
 - SG.SG_TrackedHand, 263
- handBase
 - SG.SG_WireFrame, 287
- handLengths
 - SG.SG_SenseGloveData, 226
- handles
 - SG.SG_Drawer, 90
 - SG.SG_Hinge, 176
- handLocation
 - SG.SG_BasicFeedback, 42
- handModel
 - SG.SG_HandDetector.GloveDetectionArgs, 34
 - SG.SG_HandFeedback, 156
 - SG.SG_HandRigidBody, 164
 - SG.SG_PhysicsGrab, 215
 - SG.SG_TrackedHand, 264
- HandModelIndex
 - SG.SG_HandDetector, 148
- handModelInfo
 - SG.SG_AutoHandAnimation, 39
- handPalmModel
 - SG.SG_WireFrame, 287
- handPositions
 - SG.SG_SenseGloveData, 226
- handRB
 - SG.SG_TrackedHand, 264
- handRotations
 - SG.SG_SenseGloveData, 226
- hapticDur
 - SG.Materials.MaterialProps, 37
- hapticDuration
 - SG.SG_HandTrigger, 169
 - SG.SG_Material, 200
- hapticFeedback
 - SG.SG_HandTrigger, 170
 - SG.SG_Material, 200
- hapticForce
 - SG.Materials.MaterialProps, 37
 - SG.SG_HandTrigger, 170
- hapticMagnitude
 - SG.SG_Material, 200
- hardware
 - SG.SG_GrabScript, 129
 - SG.SG_TrackedHand, 264
- HardwareReady
 - SG.SG_GrabScript, 131
 - SG.SG_HandAnimator, 144
 - SG.SG_HandFeedback, 157
 - SG.SG_HandRigidBody, 165
- HasFunction
 - SG.SG_SenseGloveHardware, 235
- HasTarget
 - SG.SG_SimpleTracking, 249
- HeldObjects
 - SG.SG_GrabScript, 126
- heldObjects
 - SG.SG_GrabScript, 130
- hiddenBeforeStart
 - SG.Calibration.SG_CalibrationSequence, 63
- highLight
 - SG.SG_HandDetector, 152
- highLighters
 - SG.SG_DropZone, 100
- hinge
 - SG.SG_Dial, 81
- hingeAxis
 - SG.SG_Dial, 82
 - SG.SG_Hinge, 177
- hingePoint
 - SG.SG_Dial, 82
 - SG.SG_Hinge, 177
- HingeRatio
 - SG.SG_Hinge, 174
- hoverCollider
 - SG.SG_GestureGrabScript, 111
- ignoreGrabables
 - SG.SG_TrackedHand, 264
- impactCooldown
 - SG.SG_BasicFeedback, 43
- impactFeedbackEnabled
 - SG.SG_BasicFeedback, 43
- impactProfile
 - SG.SG_BasicFeedback, 43
- imuCalibration
 - SG.SG_SenseGloveData, 226
- imuValues
 - SG.SG_SenseGloveData, 227

- indexJoins
 - SG.SG_HandModelInfo, [159](#)
- indexTouch
 - SG.SG_PhysicsGrab, [216](#)
- Instance_DebugMessageRecieved
 - SG.SG_Debugger, [69](#)
- instrText
 - SG.Calibration.SG_CalibrationSequence, [63](#)
- instructionTexts
 - SG.Examples.SGEx_HandLayerUI, [293](#)
- Interactable
 - SG.SG_GrabScript.SG_GrabEventArgs, [120](#)
- interactablesTouched
 - SG.SG_HoverCollider, [183](#)
- InteractingWith
 - SG.SG_Interactable, [188](#)
- InteractionBegin
 - SG.SG_DetectGrab, [71](#)
 - SG.SG_Dial, [79](#)
 - SG.SG_Drawer, [87](#)
 - SG.SG_Grabable, [115](#)
 - SG.SG_GrabZone, [134](#)
 - SG.SG_Hinge, [174](#)
 - SG.SG_Interactable, [188](#)
- InteractionBegun
 - SG.SG_Interactable, [193](#)
- InteractionEnd
 - SG.SG_DetectGrab, [72](#)
 - SG.SG_Dial, [79](#)
 - SG.SG_Drawer, [87](#)
 - SG.SG_Grabable, [115](#)
 - SG.SG_GrabZone, [134](#)
 - SG.SG_Hinge, [174](#)
 - SG.SG_Interactable, [188](#)
- InteractionEnded
 - SG.SG_Interactable, [193](#)
- interpolator
 - SG.Calibration.SG_CalibrationSequence, [63](#)
- InUse
 - SG.SG_HandTrigger, [168](#)
- inUse
 - SG.SG_HandTrigger, [170](#)
- IsBroken
 - SG.SG_Breakable, [48](#)
 - SG.SG_Material, [198](#)
- isBroken
 - SG.SG_Material, [201](#)
- IsCalibrating
 - SG.SG_SenseGloveHardware, [244](#)
- IsClosed
 - SG.SG_Drawer, [88](#)
- IsConnected
 - SG.SG_SenseGloveHardware, [244](#)
- IsDetected
 - SG.SG_DropZone, [96](#)
- IsGrabbed
 - SG.SG_Grabable, [116](#)
- IsGrabbing
 - SG.SG_GrabScript, [126](#)
- IsInside
 - SG.SG_PhysicsGrab, [214](#)
- isInteractable
 - SG.SG_Interactable, [191](#)
- IsInteracting
 - SG.SG_Interactable, [189](#)
- IsNegative
 - SG.SG_Util, [273](#)
- IsOpen
 - SG.SG_Drawer, [88](#)
- IsRight
 - SG.SG_SenseGloveHardware, [245](#)
- IsSnapped
 - SG.SG_SnapDropZone, [254](#)
- isSnapped
 - SG.SG_SnapDropZone.SnapProps, [299](#)
- IsTarget
 - SG.SG_DropZone, [97](#)
- IsTouching
 - SG.SG_FingerFeedback, [105](#), [106](#)
 - SG.SG_GestureGrabScript, [110](#)
 - SG.SG_GrabScript, [126](#)
 - SG.SG_HoverCollider, [180](#), [181](#)
 - SG.SG_PhysicsGrab, [214](#)
- joint
 - SG.SG_Hinge, [177](#)
- lastAngles
 - SG.SG_GestureGrabScript, [112](#)
- lastBrakeLvls
 - SG.SG_SenseGloveHardware, [242](#)
- lastPosition
 - SG.SG_BasicFeedback, [43](#)
 - SG.SG_GrabScript, [130](#)
- lastRotation
 - SG.SG_GrabScript, [130](#)
- leftAnimation
 - SG.Calibration.SG_CalibrationSequence, [63](#)
- LeftHand
 - SG, [13](#)
- limitFingers
 - SG.SG_SenseGloveHardware, [242](#)
- linkedGlove
 - SG.SG_BasicFeedback, [43](#)
 - SG.SG_SenseGloveHardware, [242](#)
- linkedGloveData
 - SG.SG_SenseGloveHardware, [243](#)
- linkedLayers
 - SG.Examples.SGEx_HandLayerUI, [294](#)
- ListIndex
 - SG.SG_DeviceManager, [75](#)
 - SG.SG_DropZone, [97](#)
 - SG.SG_HoverCollider, [181](#)
- LoadInterpolation
 - SG.Calibration.SG_CalibrationStorage, [66](#)
- LoadMaterialProps
 - SG.SG_Material, [199](#)

- LoadProfiles
 - SG.Calibration.SG_CalibrationSequence, 61
- Log
 - SG.SG_Debugger, 69
- LogError
 - SG.SG_Debugger, 70
- LogWarning
 - SG.SG_Debugger, 70
- loop
 - SG.SG_HandTrigger, 170
- ManualRelease
 - SG.SG_GrabScript, 126
- Map
 - SG.SG_Util, 273
- MatchesConnection
 - SG.SG_SenseGloveHardware, 235
- MatchingObjects
 - SG.SG_HoverCollider, 182
- material
 - SG.SG_Material, 201
- MaterialBreaks
 - SG.SG_Material, 202
- MaterialProps
 - SG.Materials.MaterialProps, 35
- maxAngle
 - SG.SG_Dial, 82
 - SG.SG_Hinge, 177
- maxBuzzLevel
 - SG.SG_BasicFeedback, 43
- maxDataPoints
 - SG.SG_GrabScript, 130
- maxDisplacement
 - SG.SG_MeshDeform, 209
- maxDrawerDist
 - SG.SG_Drawer, 90
- maxForce
 - SG.Materials.MaterialProps, 37
 - SG.SG_Material, 201
- maxForceDist
 - SG.Materials.MaterialProps, 37
 - SG.SG_Material, 201
- maxImpactSpeed
 - SG.SG_BasicFeedback, 44
- maxVelocityPoints
 - SG.SG_BasicFeedback, 44
- meshFilter
 - SG.SG_MeshDeform, 209
- middleJoints
 - SG.SG_HandModelInfo, 159
- middleTouch
 - SG.SG_PhysicsGrab, 216
- minAngle
 - SG.SG_Dial, 82
 - SG.SG_Hinge, 177
- minBuzzLevel
 - SG.SG_BasicFeedback, 44
- minDrawerDist
 - SG.SG_Drawer, 91
- minImpactSpeed
 - SG.SG_BasicFeedback, 44
- minimumFingers
 - SG.SG_Material, 201
- MoveAxis
 - SG.SG_Drawer, 88
- moveAxis
 - SG.SG_Drawer, 91
- moveDirection
 - SG.SG_Drawer, 91
- MovementAxis
 - SG, 13
- mustBeGrabbed
 - SG.SG_Material, 201
- MustBeReleased
 - SG.SG_Interactive, 189
- MustOpenHand
 - SG, 13
- myCollider
 - SG.SG_HandDetector, 152
- myInteractive
 - SG.SG_Material, 202
- myJoint
 - SG.SG_SnapDropZone.SnapProps, 299
- myMesh
 - SG.SG_MeshDeform, 209
- myRigidbody
 - SG.SG_HandDetector, 152
- newData
 - SG.SG_SenseGloveHardware.GloveCalibrationArgs, 33
- NextGlove
 - SG.SG_SenseGloveHardware, 233
- NextLeftHand
 - SG.SG_SenseGloveHardware, 233
- NextRightHand
 - SG.SG_SenseGloveHardware, 233
- nextStepKey
 - SG.Calibration.SG_CalibrationSequence, 64
- None
 - SG.SG_Breakable, 48
- NormalizeAngle
 - SG.SG_Util, 274
- NormalizeAngles
 - SG.SG_Util, 274
- NumberOfObjects
 - SG.SG_DropZone, 101
- numberOfSensors
 - SG.SG_SenseGloveData, 227
- ObjectBreaks
 - SG.SG_Breakable, 52
- ObjectBrokenEventHandler
 - SG.SG_Breakable, 48
- ObjectDependent
 - SG.SG_SnapDropZone, 252
- ObjectDetected
 - SG.SG_DropZone, 102

- ObjectDisabled
 - SG.SG_FingerFeedback, [106](#)
- ObjectGrabbed
 - SG.SG_GrabScript, [132](#)
- ObjectReleased
 - SG.SG_GrabScript, [132](#)
- ObjectRemoved
 - SG.SG_DropZone, [102](#)
- ObjectReset
 - SG.SG_Interactive, [193](#)
- ObjectsInside
 - SG.SG_DropZone, [101](#)
- objectsInside
 - SG.SG_DropZone, [100](#)
- objectsToGet
 - SG.SG_DropZone, [100](#)
- ObjectUnBreaks
 - SG.SG_Breakable, [52](#)
- ObjectUnBrokenEventHandler
 - SG.SG_Breakable, [49](#)
- offsetAngle
 - SG.SG_Hinge, [177](#)
- oldData
 - SG.SG_SenseGloveHardware.GloveCalibrationArgs, [33](#)
- oldParent
 - SG.SG_SnapDropZone.SnapProps, [299](#)
- OnDisable
 - SG.SG_Material, [199](#)
- OnDoorClosed
 - SG.SG_Door, [84](#)
- OnDoorOpened
 - SG.SG_Door, [84](#)
- OnGloveConnect
 - SG.SG_SenseGloveHardware, [236](#)
- OnGloveDisconnect
 - SG.SG_SenseGloveHardware, [236](#)
- OnGloveLink
 - SG.SG_SenseGloveHardware, [236](#)
- OnGloveLoad
 - SG.SG_SenseGloveHardware, [243](#)
- OnGloveUnLink
 - SG.SG_SenseGloveHardware, [236](#)
- OnGrabbedObject
 - SG.SG_GrabScript, [127](#)
- OnObjectBreaks
 - SG.SG_Breakable, [49](#)
- OnObjectDetected
 - SG.SG_DropZone, [100](#)
- OnObjectRemoved
 - SG.SG_DropZone, [100](#)
- OnObjectUnBreaks
 - SG.SG_Breakable, [49](#)
- OnReleasedObject
 - SG.SG_GrabScript, [127](#)
- openEventFired
 - SG.SG_Drawer, [91](#)
- openHandEx
 - SG.Calibration.SG_CalibrationSequence, [64](#)
- openHandThresholds
 - SG.SG_PhysicsGrab, [216](#)
- originalDist
 - SG.SG_Interactive, [192](#)
- OriginalParent
 - SG.SG_Grabable, [119](#)
- originalPos
 - SG.SG_Interactive, [192](#)
- originalRot
 - SG.SG_Interactive, [192](#)
- packetsPerSecond
 - SG.SG_SenseGloveData, [227](#)
- palmTouch
 - SG.SG_PhysicsGrab, [216](#)
- Parent
 - SG, [13](#)
 - SG.SG_SnapDropZone, [252](#)
- Parse
 - SG.Materials.MaterialProps, [36](#)
- particlesToPlay
 - SG.SG_HandTrigger, [170](#)
- patterns
 - SG.SG_SenseGloveHardware.BuzzCmd, [17](#)
- paused
 - SG.SG_GrabScript, [130](#)
- pauseTime
 - SG.SG_GrabScript, [130](#)
- phalangeModel
 - SG.SG_WireFrame, [287](#)
- PhysicsBased
 - SG.SG_TrackedHand, [262](#)
- physicsBody
 - SG.SG_DropZone, [100](#)
 - SG.SG_Grabable, [118](#)
 - SG.SG_Hinge, [178](#)
 - SG.SG_TrackedBody, [258](#)
- physicsTrackingLayer
 - SG.SG_TrackedHand, [264](#)
- physRotationSpeed
 - SG.SG_TrackedHand, [264](#)
- pickupMethod
 - SG.SG_Grabable, [118](#)
- pickupReference
 - SG.SG_Grabable, [118](#)
- pinkyJoints
 - SG.SG_HandModelInfo, [159](#)
- Plane
 - SG.Materials, [15](#)
- poses
 - SG.Calibration.SG_CalibrationSequence, [64](#)
- positionOffset
 - SG.SG_SimpleTracking, [248](#)
 - SG.SG_TrackedHand, [265](#)
- previewGroup
 - SG.SG_WireFrame, [288](#)
- qBase

- SG.SG_Dial, [82](#)
- raisedBreak
 - SG.SG_Material, [202](#)
- ReadTxtFile
 - SG.Util.FileIO, [30](#)
- ReadyCalibration
 - SG.SG_SenseGloveHardware, [236](#)
- refinementMenu
 - SG.Calibration.SG_CalibrationSequence, [64](#)
- RegularExit
 - SG.Calibration.SG_CalibrationSequence, [61](#)
- RelativeWrist
 - SG.SG_HandAnimator, [144](#)
- relativeWrist
 - SG.SG_SenseGloveData, [227](#)
- releaseDistance
 - SG.SG_Interactive, [192](#)
- ReleaseMethod
 - SG, [13](#)
- releaseMethod
 - SG.SG_Interactive, [192](#)
- ReleaseObject
 - SG.SG_SnapDropZone, [254](#)
- ReleaseObjectAt
 - SG.SG_GrabScript, [127](#)
- RemoveDeform
 - SG.SG_MeshDeform, [205](#)
- RemoveEntry
 - SG.SG_HandDetector, [149](#)
- RemoveFromList
 - SG.SG_HoverCollider, [182](#)
- RemoveObject
 - SG.SG_DropZone, [97](#), [98](#)
 - SG.SG_SnapDropZone, [255](#)
- RemoveRigidBodies
 - SG.SG_HandRigidBodies, [163](#)
- ReportConnections
 - SG.SG_DeviceManager, [76](#)
- requiresThumb
 - SG.SG_Material, [202](#)
- Reset
 - SG.SG_Breakable, [48](#)
- ResetCalibration
 - SG.Calibration.SG_CalibrationSequence, [61](#)
- ResetContents
 - SG.SG_BreakableContainer, [54](#)
- resetDistance
 - SG.SG_TrackedBody, [258](#)
- ResetForces
 - SG.SG_FingerFeedback, [106](#)
- ResetKinematics
 - SG.SG_SenseGloveHardware, [237](#)
- ResetMesh
 - SG.SG_MeshDeform, [207](#)
- ResetObject
 - SG.SG_Breakable, [49](#)
 - SG.SG_BreakableContainer, [54](#)
 - SG.SG_Drawer, [88](#)
 - SG.SG_Grabable, [116](#)
 - SG.SG_GrabZone, [134](#)
 - SG.SG_Interactive, [189](#)
- ResetPoints
 - SG.SG_MeshDeform, [207](#)
- ResetShards
 - SG.SG_BreakableContainer, [54](#)
- resetTime
 - SG.SG_Breakable, [51](#)
 - SG.SG_TrackedBody, [258](#)
- resetTimer
 - SG.SG_TrackedBody, [259](#)
- ResetZoneAndObjects
 - SG.SG_DropZone, [98](#)
- resizeFingers
 - SG.SG_HandAnimator, [142](#)
- ResizeHand
 - SG.SG_HandAnimator, [139](#)
 - SG.SG_WireFrame, [284](#)
- RestoreProperties
 - SG.SG_SnapDropZone.SnapProps, [299](#)
- RightHand
 - SG, [13](#)
- rightHands
 - SG.Calibration.SG_CalibrationSequence, [64](#)
- rigidBodyLayer
 - SG.SG_TrackedHand, [265](#)
- ringJoints
 - SG.SG_HandModelInfo, [160](#)
- RotationAxis
 - SG.SG_Hinge, [175](#)
- rotationOffset
 - SG.SG_SimpleTracking, [248](#)
 - SG.SG_TrackedHand, [265](#)
- rotationSpeed
 - SG.SG_TrackedBody, [259](#)
- rotOffset
 - SG.SG_Dial, [82](#)
- Rubber
 - SG.Materials, [15](#)
- SameScript
 - SG.SG_FingerFeedback, [106](#)
 - SG.SG_HoverCollider, [182](#)
- SameVertex
 - SG.SG_MeshDeform, [207](#)
- sameVertices
 - SG.SG_MeshDeform, [210](#)
- SaveAndExit
 - SG.Calibration.SG_CalibrationSequence, [62](#)
- saveButton
 - SG.Calibration.SG_CalibrationSequence, [64](#)
- SaveCalibration
 - SG.Calibration.SG_CalibrationSequence, [62](#)
- SaveRBParameters
 - SG.SG_Grabable, [116](#)
- SaveTransform
 - SG.SG_Drawer, [89](#)
 - SG.SG_Grabable, [116](#)

- SG.SG_GrabZone, [135](#)
- SG.SG_Interactive, [189](#)
- SaveTxtFile
 - SG.Util.FileIO, [31](#)
- SendBrakeCmd
 - SG.SG_SenseGloveHardware, [237](#)
- SendBuzzCmd
 - SG.SG_SenseGloveHardware, [237–239](#)
- sendHaptics
 - SG.SG_SenseGloveHardware, [243](#)
- SendImpactFeedback
 - SG.SG_BasicFeedback, [42](#)
- SendThumperCmd
 - SG.SG_SenseGloveHardware, [239](#)
- SenseGlove_OnCalibrationFinished
 - SG.SG_HandAnimator, [139](#)
 - SG.SG_WireFrame, [284](#)
- SenseGlove_OnGloveLoaded
 - SG.SG_HandAnimator, [141](#)
 - SG.SG_WireFrame, [285](#)
- SetAngle
 - SG.SG_Dial, [80](#)
 - SG.SG_Hinge, [175](#)
- SetChildren
 - SG.SG_Util, [275](#)
- SetColliders
 - SG.SG_BreakableContainer, [54](#)
- SetCollision
 - SG.SG_Grabable, [116](#)
- SetDeform
 - SG.SG_MeshDeform, [207](#)
- SetEffectObject
 - SG.SG_HandTrigger, [168](#)
- SetGlove
 - SG.SG_WireFrame, [285](#)
- SetHand
 - SG.SG_WireFrame, [285](#)
- SetHandParameters
 - SG.SG_SenseGloveHardware, [239](#)
- SetHighLight
 - SG.SG_DropZone, [98](#)
 - SG.SG_HandDetector, [149](#)
- SetIgnoreCollision
 - SG.SG_HandFeedback, [155](#)
 - SG.SG_HandRigidBodies, [163, 164](#)
 - SG.SG_SimpleTracking, [247](#)
- SetInteractive
 - SG.SG_Drawer, [89](#)
 - SG.SG_Hinge, [175](#)
 - SG.SG_Interactive, [190](#)
- SetMoveAxis
 - SG.SG_Drawer, [89](#)
- SetParticles
 - SG.SG_HandTrigger, [168](#)
- SetRB
 - SG.SG_BreakableContainer, [55](#)
- SetTrackingTarget
 - SG.SG_SimpleTracking, [248](#)
- Setup
 - SG.SG_GestureGrabScript, [110](#)
 - SG.SG_GrabScript, [128](#)
 - SG.SG_PhysicsGrab, [215](#)
- setup
 - SG.SG_DropZone, [101](#)
- SetupArray
 - SG.Calibration.CalibrationPose, [23](#)
- setupComplete
 - SG.SG_WireFrame, [288](#)
- SetupDevice
 - SG.SG_DeviceLink, [73](#)
 - SG.SG_SenseGloveHardware, [240](#)
- SetupFingers
 - SG.SG_WireFrame, [285](#)
- setupFinished
 - SG.SG_GrabScript, [131](#)
- SetupFloatArray
 - SG.Calibration.CalibrationPose, [24](#)
- SetupGlove
 - SG.SG_WireFrame, [286](#)
- SetupHandPalm
 - SG.SG_WireFrame, [286](#)
- SetupHands
 - SG.SG_User, [267](#)
- SetupHinge
 - SG.SG_Hinge, [176](#)
- SetupScripts
 - SG.SG_HandFeedback, [155](#)
- SetupSelf
 - SG.SG_BasicFeedback, [42](#)
 - SG.SG_FingerFeedback, [107](#)
 - SG.SG_HandRigidBodies, [164](#)
- SetupTracking
 - SG.SG_TrackedHand, [262](#)
- SG, [9](#)
 - AttachType, [11](#)
 - Default, [11, 13](#)
 - FixedJoint, [13](#)
 - Follow, [13](#)
 - FunctionCall, [13](#)
 - GloveSide, [11](#)
 - GrabType, [13](#)
 - LeftHand, [13](#)
 - MovementAxis, [13](#)
 - MustOpenHand, [13](#)
 - Parent, [13](#)
 - ReleaseMethod, [13](#)
 - RightHand, [13](#)
 - SG_HandSection, [13](#)
 - SnapToAnchor, [11](#)
 - Unknown, [13](#)
- SG.Calibration, [14](#)
- SG.Calibration.CalibrationPose, [18](#)
 - abd, [24](#)
 - calbrUsing, [24](#)
 - CalibrateParameters, [20](#)
 - CalibrationPose, [19, 20](#)

- GetFist, [20](#)
- GetFullFist, [21](#)
- GetFullOpen, [21](#)
- GetOpenHand, [21](#)
- GetThumbAbd, [22](#)
- GetThumbFlexed, [22](#)
- GetThumbNoAbd, [23](#)
- GetThumbsUp, [23](#)
- SetupArray, [23](#)
- SetupFloatArray, [24](#)
- x0, [24](#)
- xAffect, [24](#)
- yAffect, [25](#)
- yValue, [25](#)
- SG.Calibration.SG_CalibrationSequence, [57](#)
 - baseTxt, [62](#)
 - CalibrateCurrentStep, [60](#)
 - CalibratePose, [60](#)
 - CalPose, [59](#)
 - CalStage, [60](#)
 - changes, [62](#)
 - EndGlobal, [60](#)
 - endPopup, [63](#)
 - generalButtonTxt, [63](#)
 - GetCalibrationValues, [60](#)
 - GoToMainStage, [61](#)
 - hiddenBeforeStart, [63](#)
 - instrText, [63](#)
 - interpolator, [63](#)
 - leftAnimation, [63](#)
 - LoadProfiles, [61](#)
 - nextStepKey, [64](#)
 - openHandEx, [64](#)
 - poses, [64](#)
 - refinementMenu, [64](#)
 - RegularExit, [61](#)
 - ResetCalibration, [61](#)
 - rightHands, [64](#)
 - SaveAndExit, [62](#)
 - saveButton, [64](#)
 - SaveCalibration, [62](#)
 - skipButton, [65](#)
 - SkipStep, [62](#)
 - stage, [65](#)
 - StartGlobal, [62](#)
 - subStage, [65](#)
 - wireFrame, [65](#)
- SG.Calibration.SG_CalibrationStorage, [65](#)
 - calibrDir, [68](#)
 - GetFilename, [66](#)
 - LoadInterpolation, [66](#)
 - StoreInterpolation, [67](#)
- SG.Examples, [14](#)
- SG.Examples.SGEx_Diagnostics, [289](#)
- SG.Examples.SGEx_ForceFeedbackObjects, [290](#)
- SG.Examples.SGEx_HandLayerUI, [292](#)
 - instructionTexts, [293](#)
 - linkedLayers, [294](#)
- SG.Examples.SGEx_RotateSimple, [295](#)
- SG.Examples.SGEx_SelectHandModel, [295](#)
- SG.Examples.SGEx_ShowGloveAngles, [296](#)
- SG.Materials, [14](#)
 - Custom, [15](#)
 - DisplaceType, [15](#)
 - Egg, [15](#)
 - Plane, [15](#)
 - Rubber, [15](#)
 - Steel, [15](#)
 - VirtualMaterial, [15](#)
- SG.Materials.Deformation, [25](#)
 - absDeformPosition, [26](#)
 - absEntryVector, [26](#)
 - Deformation, [26](#)
 - distance, [26](#)
- SG.Materials.MaterialProps, [34](#)
 - Default, [36](#)
 - hapticDur, [37](#)
 - hapticForce, [37](#)
 - MaterialProps, [35](#)
 - maxForce, [37](#)
 - maxForceDist, [37](#)
 - Parse, [36](#)
 - TryGetFloat, [36](#)
 - TryGetRawValue, [37](#)
 - yieldDist, [38](#)
- SG.SG_AutoHandAnimation, [38](#)
 - CheckForScripts, [39](#)
 - CollectFingerJoints, [39](#)
 - handModelInfo, [39](#)
 - Start, [39](#)
- SG.SG_BasicFeedback, [40](#)
 - CanImpact, [45](#)
 - cooldownTimer, [42](#)
 - DebugEnabled, [45](#)
 - handLocation, [42](#)
 - impactCooldown, [43](#)
 - impactFeedbackEnabled, [43](#)
 - impactProfile, [43](#)
 - lastPosition, [43](#)
 - linkedGlove, [43](#)
 - maxBuzzLevel, [43](#)
 - maxImpactSpeed, [44](#)
 - maxVelocityPoints, [44](#)
 - minBuzzLevel, [44](#)
 - minImpactSpeed, [44](#)
 - SendImpactFeedback, [42](#)
 - SetupSelf, [42](#)
 - SmoothedVelocity, [45](#)
 - UpdatePosition, [42](#)
 - velocities, [44](#)
 - vibrationTime, [44](#)
- SG.SG_Breakable, [46](#)
 - Break, [48](#)
 - breakParticles, [50](#)
 - breakSound, [50](#)
 - brokenDeform, [50](#)

- brokenMaterial, 50
- brokenObject, 51
- checkTime, 51
- CheckUnbreak, 48
- IsBroken, 48
- None, 48
- ObjectBreaks, 52
- ObjectBrokenEventHandler, 48
- ObjectUnBreaks, 52
- ObjectUnBrokenEventHandler, 49
- OnObjectBreaks, 49
- OnObjectUnBreaks, 49
- Reset, 48
- ResetObject, 49
- resetTime, 51
- UnBreak, 49
- Unbreak, 48
- unbreakMethod, 51
- UnbreakType, 47
- wholeDeform, 51
- wholeMaterial, 51
- WholeMaterial_MaterialBreaks, 50
- wholeObject, 52
- SG.SG_BreakableContainer, 52
 - Break, 54
 - brokenShards, 56
 - contentPositions, 56
 - contentRotations, 56
 - contents, 56
 - contentsContainer, 56
 - ResetContents, 54
 - ResetObject, 54
 - ResetShards, 54
 - SetColliders, 54
 - SetRB, 55
 - shardContainer, 56
 - shardPositions, 57
 - shardRotations, 57
 - SpawnContents, 55
 - SpawnShards, 55
 - UnBreak, 55
 - unbreakWithContents, 57
- SG.SG_Debugger, 68
 - DLL_debugLevel, 70
 - Instance_DebugMessageRecieved, 69
 - Log, 69
 - LogError, 70
 - LogWarning, 70
 - unityEnabled, 70
 - unityEnabled_S, 70
- SG.SG_DetectGrab, 71
 - InteractionBegin, 71
 - InteractionEnd, 72
- SG.SG_DeviceLink, 72
 - DisposeDevice, 73
 - SetupDevice, 73
- SG.SG_DeviceManager, 74
 - CheckConnections, 75
 - ClearConnections, 75
 - devicesToLink, 76
 - ListIndex, 75
 - ReportConnections, 76
 - UnlinkAll, 76
- SG.SG_DeviceManager.DeviceDetectedArgs, 27
 - DeviceDetectedArgs, 27
 - DeviceID, 28
 - DeviceIndex, 28
 - Type, 28
- SG.SG_Dial, 77
 - _grabReference, 81
 - AngleIndex, 78
 - angleIndex, 81
 - anglOffset, 81
 - currAngle, 81
 - GetAngle, 78
 - GetAxis, 79
 - hinge, 81
 - hingeAxis, 82
 - hingePoint, 82
 - InteractionBegin, 79
 - InteractionEnd, 79
 - maxAngle, 82
 - minAngle, 82
 - qBase, 82
 - rotOffset, 82
 - SetAngle, 80
 - UpdateAngle, 80
 - UpdateInteraction, 80
 - useLimits, 83
 - ValidateAngle, 80
- SG.SG_Door, 83
 - DoorClosed, 84
 - DoorOpened, 84
 - OnDoorClosed, 84
 - OnDoorOpened, 84
- SG.SG_Drawer, 85
 - actualMoveDirection, 90
 - DrawerClosed, 91
 - DrawerOpened, 91
 - ForceClosed, 87
 - ForceOpen, 87
 - grabOffset, 90
 - grabReference, 90
 - handles, 90
 - InteractionBegin, 87
 - InteractionEnd, 87
 - IsClosed, 88
 - IsOpen, 88
 - maxDrawerDist, 90
 - minDrawerDist, 91
 - MoveAxis, 88
 - moveAxis, 91
 - moveDirection, 91
 - openEventFired, 91
 - ResetObject, 88
 - SaveTransform, 89

- SetInteractable, [89](#)
- SetMoveAxis, [89](#)
- UpdateInteraction, [89](#)
- SG.SG_DropZone, [92](#)
 - AddObject, [94](#)
 - AddTarget, [94](#)
 - AllObjectsDetected, [101](#)
 - CallObjectDetect, [95](#)
 - CallObjectRemoved, [95](#)
 - CheckDetectionTimes, [95](#)
 - CheckObjectEnter, [95](#)
 - checkStayTime, [99](#)
 - checkStayTimer, [99](#)
 - ClearObjects, [96](#)
 - detectHeldObjects, [99](#)
 - detectionTime, [99](#)
 - dropProperties, [99](#)
 - DropZoneEventHandler, [96](#)
 - highLighters, [100](#)
 - IsDetected, [96](#)
 - IsTarget, [97](#)
 - ListIndex, [97](#)
 - NumberOfObjects, [101](#)
 - ObjectDetected, [102](#)
 - ObjectRemoved, [102](#)
 - ObjectsInside, [101](#)
 - objectsInside, [100](#)
 - objectsToGet, [100](#)
 - OnObjectDetected, [100](#)
 - OnObjectRemoved, [100](#)
 - physicsBody, [100](#)
 - RemoveObject, [97](#), [98](#)
 - ResetZoneAndObjects, [98](#)
 - SetHighLight, [98](#)
 - setup, [101](#)
 - ValidateRB, [98](#)
 - ValidateSettings, [98](#)
- SG.SG_DropZone.DropProps, [28](#)
- SG.SG_DropZone.DropZoneArgs, [29](#)
 - DropZoneArgs, [29](#)
 - grabable, [30](#)
- SG.SG_FingerFeedback, [102](#)
 - AttachScript, [104](#)
 - debugDirections, [107](#)
 - DetachScript, [104](#)
 - DistanceInCollider, [108](#)
 - entryOrigin, [107](#)
 - entryPoint, [107](#)
 - FindForceDirection, [104](#)
 - ForceLevel, [108](#)
 - GetMaterialScript, [105](#)
 - IsTouching, [105](#), [106](#)
 - ObjectDisabled, [106](#)
 - ResetForces, [106](#)
 - SameScript, [106](#)
 - SetupSelf, [107](#)
 - TouchedCollider, [108](#)
 - TouchedDeformScript, [108](#)
 - TouchedMaterialScript, [108](#)
 - TouchedObject, [109](#)
 - UpdateFeedback, [107](#)
- SG.SG_GestureGrabScript, [109](#)
 - baseGrabAngles, [111](#)
 - baseReleaseAngles, [111](#)
 - CanInteract, [110](#)
 - grabbing, [111](#)
 - hoverCollider, [111](#)
 - IsTouching, [110](#)
 - lastAngles, [112](#)
 - Setup, [110](#)
 - UpdateGrabScript, [111](#)
 - wantedGrab, [112](#)
- SG.SG_Grabable, [112](#)
 - attachMethod, [117](#)
 - BreakJoint, [114](#)
 - canTransfer, [117](#)
 - CheckPickupRef, [114](#)
 - ConnectJoint, [115](#)
 - grabOffset, [118](#)
 - grabReference, [118](#)
 - grabRotation, [118](#)
 - InteractionBegin, [115](#)
 - InteractionEnd, [115](#)
 - IsGrabbed, [116](#)
 - OriginalParent, [119](#)
 - physicsBody, [118](#)
 - pickupMethod, [118](#)
 - pickupReference, [118](#)
 - ResetObject, [116](#)
 - SaveRBParameters, [116](#)
 - SaveTransform, [116](#)
 - SetCollision, [116](#)
 - SnapMeTo, [117](#)
 - snapReference, [119](#)
 - UpdateInteraction, [117](#)
 - UsedGravity, [119](#)
 - usedGravity, [119](#)
 - WasKinematic, [119](#)
 - wasKinematic, [119](#)
 - ZeroVelocity, [117](#)
- SG.SG_GrabScript, [121](#)
 - angularVelocities, [129](#)
 - CanInteract, [123](#)
 - CanRelease, [123](#)
 - CheckForScripts, [124](#)
 - ClearHeldObjects, [124](#)
 - DebugEnabled, [131](#)
 - elapsedTime, [129](#)
 - EndInteraction, [124](#)
 - GetAngularVelocity, [124](#)
 - GetHardware, [125](#)
 - GetVelocity, [125](#)
 - grabAnchor, [129](#)
 - GrabEventHandler, [125](#)
 - grabReference, [129](#)
 - Hand, [131](#)

- hardware, 129
- HardwareReady, 131
- HeldObjects, 126
- heldObjects, 130
- IsGrabbing, 126
- IsTouching, 126
- lastPosition, 130
- lastRotation, 130
- ManualRelease, 126
- maxDataPoints, 130
- ObjectGrabbed, 132
- ObjectReleased, 132
- OnGrabbedObject, 127
- OnReleasedObject, 127
- paused, 130
- pauseTime, 130
- ReleaseObjectAt, 127
- Setup, 128
- setupFinished, 131
- TryGrabObject, 128
- UpdateDynamics, 128
- UpdateGrabScript, 128
- velocities, 131
- SG.SG_GrabScript.SG_GrabEventArgs, 120
 - Interactable, 120
- SG.SG_GrabZone, 132
 - connectedTo, 135
 - ConnectionIndex, 133
 - ConnectTo, 133
 - InteractionBegin, 134
 - InteractionEnd, 134
 - ResetObject, 134
 - SaveTransform, 135
 - UpdateInteraction, 135
- SG.SG_HandAnimator, 136
 - CalibrateWrist, 138
 - CheckForScripts, 138
 - CollectCorrections, 138
 - CollectFingerJoints, 138
 - CollectHandParameters, 138
 - debugGroup, 142
 - debugText, 142
 - DifferenceFromWrist, 138
 - fingerCorrection, 142
 - fingerJoints, 142
 - foreArmTransform, 142
 - GetHardware, 139
 - Hand, 144
 - HardwareReady, 144
 - RelativeWrist, 144
 - resizeFingers, 142
 - ResizeHand, 139
 - SenseGlove_OnCalibrationFinished, 139
 - SenseGlove_OnGloveLoaded, 141
 - updateFingers, 143
 - UpdateHand, 141
 - UpdateWrist, 141
 - updateWrist, 143
- WristAngles, 144
- wristAngles, 143
- wristCalibration, 143
- wristCorrection, 143
- wristTransform, 143
- SG.SG_HandDetector, 145
 - activationThreshold, 150
 - activationTime, 150
 - AddEntry, 147
 - ContainsSenseGlove, 147
 - detectedColliders, 150
 - detectedGloves, 150
 - detectIndex, 150
 - detectionMethod, 151
 - detectionTimes, 151
 - DetectionType, 147
 - detectMiddle, 151
 - detectPinky, 151
 - detectRing, 151
 - detectThumb, 151
 - eventFired, 152
 - FireDetectEvent, 148
 - FireRemoveEvent, 148
 - GloveDetected, 153
 - GloveRemoved, 153
 - GlovesInside, 148
 - HandModelIndex, 148
 - highLight, 152
 - myCollider, 152
 - myRigidbody, 152
 - RemoveEntry, 149
 - SetHighLight, 149
 - singleGlove, 152
 - ValidScript, 149
- SG.SG_HandDetector.GloveDetectionArgs, 33
 - GloveDetectionArgs, 34
 - handModel, 34
- SG.SG_HandFeedback, 153
 - CheckForScripts, 154
 - ColliderDistances, 156
 - connectedGlove, 156
 - DebugEnabled, 157
 - fingerFeedbackScripts, 156
 - GetHardware, 154
 - Hand, 157
 - handModel, 156
 - HardwareReady, 157
 - SetIgnoreCollision, 155
 - SetupScripts, 155
 - TouchingMaterial, 155
 - UpdateForces, 155
 - wristFeedbackScript, 156
- SG.SG_HandModelInfo, 157
 - DebugEnabled, 160
 - fingerDebug, 159
 - FingerJoints, 160
 - foreArmTransform, 159
 - GetFingerTip, 158

- indexJoints, 159
- middleJoints, 159
- pinkyJoints, 159
- ringJoints, 160
- thumbJoints, 160
- wristDebug, 160
- wristTransform, 160
- SG.SG_HandRigidBodies, 161
 - AddRigidBodies, 162
 - CheckForScripts, 162
 - CollisionsEnabled, 165
 - DebugEnabled, 165
 - fingerObjs, 164
 - GetHardware, 162
 - Hand, 165
 - handModel, 164
 - HardwareReady, 165
 - RemoveRigidBodies, 163
 - SetIgnoreCollision, 163, 164
 - SetupSelf, 164
 - wristObj, 164
- SG.SG_HandTrigger, 166
 - audioToPlay, 169
 - buzz_CMD_Time, 169
 - buzzTimer, 169
 - effectToShow, 169
 - FireDetectEvent, 167
 - FireHapticFeedback, 167
 - FireRemoveEvent, 168
 - hapticDuration, 169
 - hapticFeedback, 170
 - hapticForce, 170
 - InUse, 168
 - inUse, 170
 - loop, 170
 - particlesToPlay, 170
 - SetEffectObject, 168
 - SetParticles, 168
 - whichFingers, 170
- SG.SG_Hinge, 171
 - autoSetup, 176
 - CheckLimits, 173
 - GetAngle, 173
 - GetHingeAngle, 173
 - grabReference, 176
 - handles, 176
 - hingeAxis, 177
 - hingePoint, 177
 - HingeRatio, 174
 - InteractionBegin, 174
 - InteractionEnd, 174
 - joint, 177
 - maxAngle, 177
 - minAngle, 177
 - offsetAngle, 177
 - physicsBody, 178
 - RotationAxis, 175
 - SetAngle, 175
 - SetInteractable, 175
 - SetupHinge, 176
 - StopPhysicsBody, 176
 - UpdateInteraction, 176
 - usedGravity, 178
 - wasKinematic, 178
- SG.SG_HoverCollider, 178
 - AddToList, 180
 - ClearTouchedObjects, 180
 - collidersInside, 183
 - GetInteractableScript, 180
 - interactablesTouched, 183
 - IsTouching, 180, 181
 - ListIndex, 181
 - MatchingObjects, 182
 - RemoveFromList, 182
 - SameScript, 182
 - TouchedObjects, 183
- SG.SG_Interactive, 184
 - _grabScript, 191
 - BeginInteraction, 186
 - CanInteract, 186
 - EndInteractAllowed, 186
 - EndInteraction, 187
 - fingerPalm, 191
 - fingerThumb, 191
 - GetTouchIndex, 187
 - GrabScript, 193
 - InteractingWith, 188
 - InteractionBegin, 188
 - InteractionBegun, 193
 - InteractionEnd, 188
 - InteractionEnded, 193
 - isInteractable, 191
 - IsInteracting, 189
 - MustBeReleased, 189
 - ObjectReset, 193
 - originalDist, 192
 - originalPos, 192
 - originalRot, 192
 - releaseDistance, 192
 - releaseMethod, 192
 - ResetObject, 189
 - SaveTransform, 189
 - SetInteractable, 190
 - Touched, 194
 - TouchedBy, 190
 - touchedColliders, 192
 - touchedScripts, 193
 - Untouched, 194
 - UntouchedBy, 190
 - UpdateInteraction, 191
 - WithinBounds, 191
- SG.SG_InteractArgs, 194
- SG.SG_KeyBinds, 195
- SG.SG_Material, 196
 - breakable, 200
 - brokenBy, 200

- CalculateForce, [198](#)
- CalculateHaptics, [198](#)
- CalculateResponseForce, [198](#)
- deformScript, [200](#)
- hapticDuration, [200](#)
- hapticFeedback, [200](#)
- hapticMagnitude, [200](#)
- IsBroken, [198](#)
- isBroken, [201](#)
- LoadMaterialProps, [199](#)
- material, [201](#)
- MaterialBreaks, [202](#)
- maxForce, [201](#)
- maxForceDist, [201](#)
- minimumFingers, [201](#)
- mustBeGrabbed, [201](#)
- myInteractable, [202](#)
- OnDisable, [199](#)
- raisedBreak, [202](#)
- requiresThumb, [202](#)
- UnBreak, [199](#)
- yieldDistance, [202](#)
- SG.SG_MeshDeform, [203](#)
 - AddDeform, [204](#)
 - AddDeformation, [204](#)
 - atRest, [208](#)
 - ClearDeformations, [205](#)
 - CollectMeshData, [205](#)
 - deformationQueue, [208](#)
 - DeformMesh, [205](#)
 - deforms, [209](#)
 - deformVerts, [209](#)
 - displaceType, [209](#)
 - maxDisplacement, [209](#)
 - meshFilter, [209](#)
 - myMesh, [209](#)
 - RemoveDeform, [205](#)
 - ResetMesh, [207](#)
 - ResetPoints, [207](#)
 - SameVertex, [207](#)
 - sameVertices, [210](#)
 - SetDeform, [207](#)
 - uniqueVertices, [210](#)
 - UpdateMesh, [208](#)
 - UpdatePoint, [208](#)
 - verts, [210](#)
- SG.SG_PhysicsGrab, [210](#)
 - CanInteract, [212](#)
 - CanRelease, [212](#)
 - CheckForScripts, [213](#)
 - CheckGestures, [213](#)
 - closedHandThresholds, [215](#)
 - DebugEnabled, [217](#)
 - GetGrabables, [213](#)
 - GetMatching, [213](#), [214](#)
 - handModel, [215](#)
 - indexTouch, [216](#)
 - IsInside, [214](#)
 - IsTouching, [214](#)
 - middleTouch, [216](#)
 - openHandThresholds, [216](#)
 - palmTouch, [216](#)
 - Setup, [215](#)
 - thumbTouch, [216](#)
 - touchScripts, [216](#)
 - UpdateGrabScript, [215](#)
 - wantsGrab, [217](#)
- SG.SG_SenseGloveData, [218](#)
 - absoluteCalibratedWrist, [223](#)
 - absoluteWrist, [224](#)
 - calibrationStep, [224](#)
 - commonOriginPos, [224](#)
 - commonOriginRot, [224](#)
 - dataLoaded, [224](#)
 - deviceId, [224](#)
 - Empty, [228](#)
 - firmwareVersion, [224](#)
 - GetChainVariables, [221](#)
 - GetFingerLengths, [222](#)
 - GetJointPositions, [222](#)
 - GetLinkVariables, [222](#)
 - GetSide, [222](#)
 - gloveAngles, [225](#)
 - gloveLengths, [225](#)
 - glovePositions, [225](#)
 - gloveRotations, [225](#)
 - gloveSide, [225](#)
 - gloveValues, [225](#)
 - gloveVersion, [226](#)
 - handAngles, [226](#)
 - handLengths, [226](#)
 - handPositions, [226](#)
 - handRotations, [226](#)
 - imuCalibration, [226](#)
 - imuValues, [227](#)
 - numberOfSensors, [227](#)
 - packetsPerSecond, [227](#)
 - relativeWrist, [227](#)
 - SG_SenseGloveData, [221](#)
 - totalCalibrationSteps, [227](#)
 - TotalGloveAngles, [223](#)
 - UpdateVariables, [223](#)
- SG.SG_SenseGloveHardware, [228](#)
 - autoConnect, [241](#)
 - brakeQueue, [241](#)
 - buzz_Enabled, [241](#)
 - calibrationArguments, [242](#)
 - CalibrationFinished, [245](#)
 - CalibrationFinishedEventHandler, [233](#)
 - CheckCalibration, [234](#)
 - CheckConnection, [234](#)
 - CheckForDeviceManager, [234](#)
 - ConnectionMethod, [233](#)
 - connectionMethod, [242](#)
 - deviceScannerPresent, [242](#)
 - DisposeDevice, [234](#)

- FingerLengths, [244](#)
- FinishCalibration, [234](#)
- FlushCmds, [235](#)
- GloveData, [244](#)
- GloveEventHandler, [235](#)
- GloveLoaded, [245](#)
- GloveReady, [244](#)
- GloveUnLoaded, [245](#)
- HasFunction, [235](#)
- IsCalibrating, [244](#)
- IsConnected, [244](#)
- IsRight, [245](#)
- lastBrakeLvls, [242](#)
- limitFingers, [242](#)
- linkedGlove, [242](#)
- linkedGloveData, [243](#)
- MatchesConnection, [235](#)
- NextGlove, [233](#)
- NextLeftHand, [233](#)
- NextRightHand, [233](#)
- OnGloveConnect, [236](#)
- OnGloveDisconnect, [236](#)
- OnGloveLink, [236](#)
- OnGloveLoad, [243](#)
- OnGloveUnLink, [236](#)
- ReadyCalibration, [236](#)
- ResetKinematics, [237](#)
- SendBrakeCmd, [237](#)
- SendBuzzCmd, [237–239](#)
- sendHaptics, [243](#)
- SendThumperCmd, [239](#)
- SetHandParameters, [239](#)
- SetupDevice, [240](#)
- solver, [243](#)
- StartJointPositions, [245](#)
- StopBrakes, [240](#)
- StopBuzzMotors, [240](#)
- StopFeedback, [240](#)
- thumpFFBThreshold, [243](#)
- UpdateGlove, [240](#)
- updateWrist, [243](#)
- wasConnected, [244](#)
- WriteBrakeCmd, [241](#)
- SG.SG_SenseGloveHardware.BuzzCmd, [17](#)
- patterns, [17](#)
- SG.SG_SenseGloveHardware.GloveCalibrationArgs, [31](#)
- GloveCalibrationArgs, [32](#)
- newData, [33](#)
- oldData, [33](#)
- SG.SG_SimpleTracking, [246](#)
- DebugEnabled, [249](#)
- HasTarget, [249](#)
- positionOffset, [248](#)
- rotationOffset, [248](#)
- SetIgnoreCollision, [247](#)
- SetTrackingTarget, [248](#)
- TargetPosition, [249](#)
- TargetRotation, [249](#)
- trackingTarget, [248](#)
- UpdateDuring, [247](#)
- UpdatePosition, [248](#)
- updateTime, [249](#)
- SG.SG_SnapDropZone, [250](#)
- AddObject, [252](#)
- AttachObject, [252](#)
- CallObjectDetect, [252](#)
- disablesInteraction, [255](#)
- FixedJoint, [252](#)
- Grabable_InteractionBegan, [253](#)
- Grabable_InteractionEnded, [253](#)
- Grabable_ObjectReset, [253](#)
- IsSnapped, [254](#)
- ObjectDependent, [252](#)
- Parent, [252](#)
- ReleaseObject, [254](#)
- RemoveObject, [255](#)
- SnapMethod, [251](#)
- snapMethod, [255](#)
- snapPoint, [255](#)
- snapProperties, [256](#)
- takesFromHand, [256](#)
- ValidateSettings, [255](#)
- SG.SG_SnapDropZone.SnapProps, [297](#)
- BreakJoint, [298](#)
- CreateJoint, [298](#)
- isSnapped, [299](#)
- myJoint, [299](#)
- oldParent, [299](#)
- RestoreProperties, [299](#)
- SnapProps, [298](#)
- usedGravity, [299](#)
- wasInteractable, [299](#)
- wasKinematic, [300](#)
- SG.SG_TrackedBody, [256](#)
- CollisionEnabled, [259](#)
- physicsBody, [258](#)
- resetDistance, [258](#)
- resetTime, [258](#)
- resetTimer, [259](#)
- rotationSpeed, [259](#)
- TryAddRB, [257](#)
- TryRemoveRB, [258](#)
- UpdatePosition, [258](#)
- SG.SG_TrackedHand, [259](#)
- CheckForScripts, [262](#)
- Custom, [261](#)
- Default, [262](#)
- Disabled, [262](#)
- feedbackScript, [263](#)
- grabScript, [263](#)
- handAnimation, [263](#)
- handModel, [264](#)
- handRB, [264](#)
- hardware, [264](#)
- ignoreGrabables, [264](#)
- PhysicsBased, [262](#)

- physicsTrackingLayer, [264](#)
- physRotationSpeed, [264](#)
- positionOffset, [265](#)
- rigidBodyLayer, [265](#)
- rotationOffset, [265](#)
- SetupTracking, [262](#)
- SwapTracking, [262](#)
- TargetPosition, [266](#)
- TargetRotation, [266](#)
- trackedObject, [265](#)
- TrackingHardware, [261](#)
- trackingHardware, [265](#)
- TrackingMethod, [262](#)
- trackingMethod, [265](#)
- TracksRightHand, [266](#)
- UpdateTransformDefault, [263](#)
- UpdateTransformPhysics, [263](#)
- ViveTracker, [261](#)
- SG.SG_User, [267](#)
 - SetupHands, [267](#)
- SG.SG_Util, [268](#)
 - AppendButtonText, [269](#)
 - Average, [270](#)
 - AxisIndex, [270](#)
 - CalculateAngularVelocity, [271](#)
 - CalculateOffsets, [271](#)
 - CheckForHandInfo, [272](#)
 - CheckForTrackedHand, [272](#)
 - GetAxis, [272](#)
 - GetVector, [272](#)
 - IsNegative, [273](#)
 - Map, [273](#)
 - NormalizeAngle, [274](#)
 - NormalizeAngles, [274](#)
 - SetChildren, [275](#)
 - SpawnSphere, [275](#)
 - ToEuler, [275](#)
 - ToPosition, [276](#)
 - ToQuaternion, [277](#)
 - ToString, [277](#), [278](#)
 - ToUnityEuler, [278](#)
 - ToUnityPosition, [279](#)
 - ToUnityQuaternion, [279](#)
 - TransformRigidBody, [281](#)
 - TryAddRB, [281](#)
 - TryRemoveRB, [281](#)
- SG.SG_WireFrame, [282](#)
 - CollectFingerJoints, [284](#)
 - gloveBase, [287](#)
 - gloveJoints, [287](#)
 - gloveSectionModel, [287](#)
 - handBase, [287](#)
 - handPalmModel, [287](#)
 - phalangeModel, [287](#)
 - previewGroup, [288](#)
 - ResizeHand, [284](#)
 - SenseGlove_OnCalibrationFinished, [284](#)
 - SenseGlove_OnGloveLoaded, [285](#)
 - SetGlove, [285](#)
 - SetHand, [285](#)
 - setupComplete, [288](#)
 - SetupFingers, [285](#)
 - SetupGlove, [286](#)
 - SetupHandPalm, [286](#)
 - toggleGloveKey, [288](#)
 - toggleHandKey, [288](#)
 - UpdateHand, [286](#)
- SG.SGEvent, [289](#)
- SG.Util, [15](#)
- SG.Util.FileIO, [30](#)
 - ReadTxtFile, [30](#)
 - SaveTxtFile, [31](#)
- SG.Util.SG_QuitKey, [217](#)
- SG.Util.SG_ResetFloor, [218](#)
- SG_HandSection
 - SG, [13](#)
- SG_SenseGloveData
 - SG.SG_SenseGloveData, [221](#)
- shardContainer
 - SG.SG_BreakableContainer, [56](#)
- shardPositions
 - SG.SG_BreakableContainer, [57](#)
- shardRotations
 - SG.SG_BreakableContainer, [57](#)
- singleGlove
 - SG.SG_HandDetector, [152](#)
- skipButton
 - SG.Calibration.SG_CalibrationSequence, [65](#)
- SkipStep
 - SG.Calibration.SG_CalibrationSequence, [62](#)
- SmoothedVelocity
 - SG.SG_BasicFeedback, [45](#)
- SnapMethod
 - SG.SG_SnapDropZone, [251](#)
- snapMethod
 - SG.SG_SnapDropZone, [255](#)
- SnapMeTo
 - SG.SG_Grabable, [117](#)
- snapPoint
 - SG.SG_SnapDropZone, [255](#)
- snapProperties
 - SG.SG_SnapDropZone, [256](#)
- SnapProps
 - SG.SG_SnapDropZone.SnapProps, [298](#)
- snapReference
 - SG.SG_Grabable, [119](#)
- SnapToAnchor
 - SG, [11](#)
- solver
 - SG.SG_SenseGloveHardware, [243](#)
- SpawnContents
 - SG.SG_BreakableContainer, [55](#)
- SpawnShards
 - SG.SG_BreakableContainer, [55](#)
- SpawnSphere
 - SG.SG_Util, [275](#)

- stage
 - SG.Calibration.SG_CalibrationSequence, 65
- Start
 - SG.SG_AutoHandAnimation, 39
- StartGlobal
 - SG.Calibration.SG_CalibrationSequence, 62
- StartJointPositions
 - SG.SG_SenseGloveHardware, 245
- Steel
 - SG.Materials, 15
- StopBrakes
 - SG.SG_SenseGloveHardware, 240
- StopBuzzMotors
 - SG.SG_SenseGloveHardware, 240
- StopFeedback
 - SG.SG_SenseGloveHardware, 240
- StopPhysicsBody
 - SG.SG_Hinge, 176
- StoreInterpolation
 - SG.Calibration.SG_CalibrationStorage, 67
- subStage
 - SG.Calibration.SG_CalibrationSequence, 65
- SwapTracking
 - SG.SG_TrackedHand, 262
- takesFromHand
 - SG.SG_SnapDropZone, 256
- TargetPosition
 - SG.SG_SimpleTracking, 249
 - SG.SG_TrackedHand, 266
- TargetRotation
 - SG.SG_SimpleTracking, 249
 - SG.SG_TrackedHand, 266
- thumbJoints
 - SG.SG_HandModelInfo, 160
- thumbTouch
 - SG.SG_PhysicsGrab, 216
- thumpFFBThreshold
 - SG.SG_SenseGloveHardware, 243
- ToEuler
 - SG.SG_Util, 275
- toggleGloveKey
 - SG.SG_WireFrame, 288
- toggleHandKey
 - SG.SG_WireFrame, 288
- ToPosition
 - SG.SG_Util, 276
- ToQuaternion
 - SG.SG_Util, 277
- ToString
 - SG.SG_Util, 277, 278
- totalCalibrationSteps
 - SG.SG_SenseGloveData, 227
- TotalGloveAngles
 - SG.SG_SenseGloveData, 223
- Touched
 - SG.SG_Interactive, 194
- TouchedBy
 - SG.SG_Interactive, 190
- TouchedCollider
 - SG.SG_FingerFeedback, 108
- touchedColliders
 - SG.SG_Interactive, 192
- TouchedDeformScript
 - SG.SG_FingerFeedback, 108
- TouchedMaterialScript
 - SG.SG_FingerFeedback, 108
- TouchedObject
 - SG.SG_FingerFeedback, 109
- TouchedObjects
 - SG.SG_HoverCollider, 183
- touchedScripts
 - SG.SG_Interactive, 193
- TouchingMaterial
 - SG.SG_HandFeedback, 155
- touchScripts
 - SG.SG_PhysicsGrab, 216
- ToUnityEuler
 - SG.SG_Util, 278
- ToUnityPosition
 - SG.SG_Util, 279
- ToUnityQuaternion
 - SG.SG_Util, 279
- trackedObject
 - SG.SG_TrackedHand, 265
- TrackingHardware
 - SG.SG_TrackedHand, 261
- trackingHardware
 - SG.SG_TrackedHand, 265
- TrackingMethod
 - SG.SG_TrackedHand, 262
- trackingMethod
 - SG.SG_TrackedHand, 265
- trackingTarget
 - SG.SG_SimpleTracking, 248
- TracksRightHand
 - SG.SG_TrackedHand, 266
- TransformRigidBody
 - SG.SG_Util, 281
- TryAddRB
 - SG.SG_TrackedBody, 257
 - SG.SG_Util, 281
- TryGetFloat
 - SG.Materials.MaterialProps, 36
- TryGetRawValue
 - SG.Materials.MaterialProps, 37
- TryGrabObject
 - SG.SG_GrabScript, 128
- TryRemoveRB
 - SG.SG_TrackedBody, 258
 - SG.SG_Util, 281
- Type
 - SG.SG_DeviceManager.DeviceDetectedArgs, 28
- UnBreak
 - SG.SG_Breakable, 49
 - SG.SG_BreakableContainer, 55
 - SG.SG_Material, 199

- Unbreak
 - SG.SG_Breakable, [48](#)
- unbreakMethod
 - SG.SG_Breakable, [51](#)
- UnbreakType
 - SG.SG_Breakable, [47](#)
- unbreakWithContents
 - SG.SG_BreakableContainer, [57](#)
- uniqueVertices
 - SG.SG_MeshDeform, [210](#)
- unityEnabled
 - SG.SG_Debugger, [70](#)
- unityEnabled_S
 - SG.SG_Debugger, [70](#)
- Unknown
 - SG, [13](#)
- UnlinkAll
 - SG.SG_DeviceManager, [76](#)
- UnTouched
 - SG.SG_Interactable, [194](#)
- UnTouchedBy
 - SG.SG_Interactable, [190](#)
- UpdateAngle
 - SG.SG_Dial, [80](#)
- UpdateDuring
 - SG.SG_SimpleTracking, [247](#)
- UpdateDynamics
 - SG.SG_GrabScript, [128](#)
- UpdateFeedback
 - SG.SG_FingerFeedback, [107](#)
- updateFingers
 - SG.SG_HandAnimator, [143](#)
- UpdateForces
 - SG.SG_HandFeedback, [155](#)
- UpdateGlove
 - SG.SG_SenseGloveHardware, [240](#)
- UpdateGrabScript
 - SG.SG_GestureGrabScript, [111](#)
 - SG.SG_GrabScript, [128](#)
 - SG.SG_PhysicsGrab, [215](#)
- UpdateHand
 - SG.SG_HandAnimator, [141](#)
 - SG.SG_WireFrame, [286](#)
- UpdateInteraction
 - SG.SG_Dial, [80](#)
 - SG.SG_Drawer, [89](#)
 - SG.SG_Grabable, [117](#)
 - SG.SG_GrabZone, [135](#)
 - SG.SG_Hinge, [176](#)
 - SG.SG_Interactable, [191](#)
- UpdateMesh
 - SG.SG_MeshDeform, [208](#)
- UpdatePoint
 - SG.SG_MeshDeform, [208](#)
- UpdatePosition
 - SG.SG_BasicFeedback, [42](#)
 - SG.SG_SimpleTracking, [248](#)
 - SG.SG_TrackedBody, [258](#)
- updateTime
 - SG.SG_SimpleTracking, [249](#)
- UpdateTransformDefault
 - SG.SG_TrackedHand, [263](#)
- UpdateTransformPhysics
 - SG.SG_TrackedHand, [263](#)
- UpdateVariables
 - SG.SG_SenseGloveData, [223](#)
- UpdateWrist
 - SG.SG_HandAnimator, [141](#)
- updateWrist
 - SG.SG_HandAnimator, [143](#)
 - SG.SG_SenseGloveHardware, [243](#)
- UsedGravity
 - SG.SG_Grabable, [119](#)
- usedGravity
 - SG.SG_Grabable, [119](#)
 - SG.SG_Hinge, [178](#)
 - SG.SG_SnapDropZone.SnapProps, [299](#)
- useLimits
 - SG.SG_Dial, [83](#)
- ValidateAngle
 - SG.SG_Dial, [80](#)
- ValidateRB
 - SG.SG_DropZone, [98](#)
- ValidateSettings
 - SG.SG_DropZone, [98](#)
 - SG.SG_SnapDropZone, [255](#)
- ValidScript
 - SG.SG_HandDetector, [149](#)
- velocities
 - SG.SG_BasicFeedback, [44](#)
 - SG.SG_GrabScript, [131](#)
- verts
 - SG.SG_MeshDeform, [210](#)
- vibrationTime
 - SG.SG_BasicFeedback, [44](#)
- VirtualMaterial
 - SG.Materials, [15](#)
- ViveTracker
 - SG.SG_TrackedHand, [261](#)
- wantedGrab
 - SG.SG_GestureGrabScript, [112](#)
- wantsGrab
 - SG.SG_PhysicsGrab, [217](#)
- wasConnected
 - SG.SG_SenseGloveHardware, [244](#)
- wasInteractable
 - SG.SG_SnapDropZone.SnapProps, [299](#)
- WasKinematic
 - SG.SG_Grabable, [119](#)
- wasKinematic
 - SG.SG_Grabable, [119](#)
 - SG.SG_Hinge, [178](#)
 - SG.SG_SnapDropZone.SnapProps, [300](#)
- whichFingers
 - SG.SG_HandTrigger, [170](#)

- wholeDeform
 - SG.SG_Breakable, [51](#)
- wholeMaterial
 - SG.SG_Breakable, [51](#)
- WholeMaterial_MaterialBreaks
 - SG.SG_Breakable, [50](#)
- wholeObject
 - SG.SG_Breakable, [52](#)
- wireFrame
 - SG.Calibration.SG_CalibrationSequence, [65](#)
- WithinBounds
 - SG.SG_Interactable, [191](#)
- WristAngles
 - SG.SG_HandAnimator, [144](#)
- wristAngles
 - SG.SG_HandAnimator, [143](#)
- wristCalibration
 - SG.SG_HandAnimator, [143](#)
- wristCorrection
 - SG.SG_HandAnimator, [143](#)
- wristDebug
 - SG.SG_HandModelInfo, [160](#)
- wristFeedbackScript
 - SG.SG_HandFeedback, [156](#)
- wristObj
 - SG.SG_HandRigidBodies, [164](#)
- wristTransform
 - SG.SG_HandModelInfo, [160](#)
- wristTransfrom
 - SG.SG_HandAnimator, [143](#)
- WriteBrakeCmd
 - SG.SG_SenseGloveHardware, [241](#)
- x0
 - SG.Calibration.CalibrationPose, [24](#)
- xAffect
 - SG.Calibration.CalibrationPose, [24](#)
- yAffect
 - SG.Calibration.CalibrationPose, [25](#)
- yieldDist
 - SG.Materials.MaterialProps, [38](#)
- yieldDistance
 - SG.SG_Material, [202](#)
- yValue
 - SG.Calibration.CalibrationPose, [25](#)
- ZeroVelocity
 - SG.SG_Grabable, [117](#)