

NEEDSIM Life simulation

Generated by Doxygen 1.8.10

Fri Nov 27 2015 13:08:24



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List	5
<b>4</b>	<b>Namespace Documentation</b>	<b>9</b>
4.1	NEEDSIM Namespace Reference	9
4.2	NEEDSIM3rdParty Namespace Reference	10
4.3	NEEDSIMEditor Namespace Reference	10
4.4	NEEDSIMSampleSceneScripts Namespace Reference	10
4.5	Simulation Namespace Reference	11
4.6	Simulation.Tests Namespace Reference	11
<b>5</b>	<b>Class Documentation</b>	<b>13</b>
5.1	NEEDSIM.Action Class Reference	13
5.1.1	Detailed Description	13
5.2	Simulation.Affordance Class Reference	13
5.2.1	Detailed Description	14
5.2.2	Member Function Documentation	14
5.2.2.1	AddInteraction(Interaction interaction)	14
5.2.2.2	ProlongLastInteraction()	14
5.2.2.3	StartInteraction(string name)	14
5.2.2.4	StartRandomInteraction()	14
5.2.3	Property Documentation	15
5.2.3.1	CurrentInteraction	15
5.2.3.2	HasInteraction	15
5.2.3.3	InteractionStartedThisFrame	15
5.3	Simulation.AffordanceTreeNode Class Reference	15
5.3.1	Detailed Description	16

5.3.2	Constructor & Destructor Documentation . . . . .	16
5.3.2.1	AffordanceTreeNode(AffordanceTreeNode parent, string name, string species↔ Name, Vector3 position) . . . . .	16
5.3.3	Member Function Documentation . . . . .	16
5.3.3.1	AcceptAuctionVictory(Slot slot) . . . . .	16
5.3.3.2	AddSlot(Vector3 worldPosition, Vector3 localPosition) . . . . .	17
5.3.3.3	AddSlot(Vector3 worldPosition, Vector3 localPosition, Vector3 lookAtTarget) . . . . .	17
5.3.3.4	AddSlot(Vector3 worldPosition, Vector3 localPosition, Vector3 lookAtTarget, bool isAuctionable) . . . . .	17
5.3.3.5	ApplyParentInteraction() . . . . .	17
5.3.3.6	AvailableSlot(bool consumeSlot) . . . . .	17
5.3.3.7	CountOfChildren() . . . . .	18
5.3.3.8	Printer(bool reportAffordances) . . . . .	18
5.3.3.9	Remove() . . . . .	18
5.3.3.10	setSpecies(string name) . . . . .	18
5.3.4	Property Documentation . . . . .	18
5.3.4.1	Affordance . . . . .	18
5.3.4.2	CurrentInteractionSatisfiesMostUrgentNeed . . . . .	18
5.3.4.3	Goal . . . . .	19
5.3.4.4	MostUrgentNeed . . . . .	19
5.3.4.5	Parent . . . . .	19
5.3.4.6	Root . . . . .	19
5.3.4.7	SatisfactionLevels . . . . .	19
5.3.4.8	Slots . . . . .	19
5.3.4.9	Species . . . . .	19
5.4	NEEDSIMSampleSceneScripts.Animal Class Reference . . . . .	19
5.4.1	Detailed Description . . . . .	19
5.5	NEEDSIM.Blackboard Class Reference . . . . .	20
5.5.1	Detailed Description . . . . .	20
5.5.2	Member Function Documentation . . . . .	20
5.5.2.1	AcceptSlot(Simulation.Slot slot) . . . . .	20
5.5.2.2	HasArrivedAtSlot() . . . . .	20
5.5.2.3	slotToAgentDistanceSmall(Vector3 agentPosition) . . . . .	20
5.6	NEEDSIMSampleSceneScripts.Bunny Class Reference . . . . .	21
5.6.1	Detailed Description . . . . .	21
5.7	NEEDSIM.ChaseSlot Class Reference . . . . .	21
5.7.1	Detailed Description . . . . .	22
5.7.2	Member Function Documentation . . . . .	22
5.7.2.1	Run() . . . . .	22
5.8	Simulation.DatabaseAsset Class Reference . . . . .	22

5.8.1	Detailed Description	23
5.8.2	Member Function Documentation	23
5.8.2.1	GetInteractionNames()	23
5.8.2.2	GetNeedNames()	23
5.8.2.3	GetSpeciesNames()	23
5.8.2.4	Init(string databaseName, List< NeedItem > needs, List< Species > species, List< InteractionData > interactions, bool isDefaultDatabase)	23
5.8.3	Member Data Documentation	23
5.8.3.1	DatabaseName	23
5.8.3.2	Interactions	23
5.8.3.3	isDefault	24
5.8.3.4	NeedsList	24
5.8.3.5	Species	24
5.9	NEEDSIM.DecideGoal Class Reference	24
5.9.1	Detailed Description	24
5.9.2	Member Function Documentation	24
5.9.2.1	Run()	24
5.10	NEEDSIM.DecideValue Class Reference	25
5.10.1	Detailed Description	25
5.10.2	Member Function Documentation	25
5.10.2.1	Run()	25
5.11	NEEDSIMSampleSceneScripts.Deer Class Reference	25
5.11.1	Detailed Description	26
5.12	NEEDSIMSampleSceneScripts.Fox Class Reference	26
5.12.1	Detailed Description	26
5.13	NEEDSIMSampleSceneScripts.FuchsalarmDemoScript Class Reference	26
5.13.1	Detailed Description	26
5.13.2	Member Function Documentation	26
5.13.2.1	SpawnTheFox()	26
5.14	Simulation.GeneralSettings Class Reference	27
5.14.1	Detailed Description	27
5.14.2	Property Documentation	27
5.14.2.1	AuctionableSlotColor	27
5.14.2.2	BlockedSlotColor	27
5.14.2.3	DefaultNeedChangePerSecondRate	27
5.14.2.4	DefaultNeedCriticalState	28
5.14.2.5	DefaultNeedMaxValue	28
5.14.2.6	DefaultNeedMinValue	28
5.14.2.7	DefaultNeedName	28
5.14.2.8	DefaultNeedSatisfiedState	28

5.14.2.9	ReadyCharacterSlotColor	28
5.14.2.10	ReservedSlotColor	28
5.14.2.11	SlotColor	28
5.14.2.12	SlotRepresentationHandleRadius	28
5.14.2.13	SlotRepresentationRadius	28
5.15	Simulation.Goal Class Reference	28
5.15.1	Detailed Description	29
5.15.2	Constructor & Destructor Documentation	29
5.15.2.1	Goal(string needToSatisfy, Needs.NeedSatisfactions satisfactionState)	29
5.15.3	Member Function Documentation	29
5.15.3.1	GoalAchieved(float satisfactionValue)	29
5.15.4	Property Documentation	29
5.15.4.1	HasBeenAchieved	29
5.15.4.2	NeedToSatisfy	29
5.15.4.3	SatisfactionState	30
5.16	NEEDSIMSampleSceneScripts.InputFieldRuntimeEditing Class Reference	30
5.16.1	Detailed Description	30
5.16.2	Member Function Documentation	30
5.16.2.1	EndInput(string newValue)	30
5.17	Simulation.Interaction Class Reference	30
5.17.1	Detailed Description	31
5.17.2	Constructor & Destructor Documentation	31
5.17.2.1	Interaction(string Name, Dictionary< string, float > SatisfactionRates, float Duration, bool hasPreconditions, Dictionary< string, bool > AllowedSpecies, Dictionary< Needs.NeedSatisfactions, bool > AllowedSatisfactions)	31
5.17.3	Member Function Documentation	31
5.17.3.1	CheckPreconditions(AffordanceTreeNode affordanceTreeNode)	31
5.17.4	Property Documentation	32
5.17.4.1	Duration	32
5.17.4.2	HasPreconditions	32
5.17.4.3	Name	32
5.17.4.4	Prolongable	32
5.17.4.5	SatisfactionRates	32
5.17.4.6	SpeciesAllowed	32
5.18	Simulation.InteractionData Class Reference	32
5.18.1	Detailed Description	33
5.18.2	Member Function Documentation	33
5.18.2.1	Init(string name, List< StringFloatPair > satisfactionRates)	33
5.18.3	Member Data Documentation	33
5.18.3.1	AtSatisfactionLevels	33

5.18.3.2	<a href="#">doesHavePreconditions</a>	33
5.18.3.3	<a href="#">duration</a>	33
5.18.3.4	<a href="#">interactionName</a>	33
5.18.3.5	<a href="#">satisfactions</a>	33
5.18.3.6	<a href="#">SpeciesAllowed</a>	33
5.19	<a href="#">NEEDSIM.InterruptionFuchsalarm Class Reference</a>	34
5.19.1	<a href="#">Detailed Description</a>	34
5.19.2	<a href="#">Member Function Documentation</a>	34
5.19.2.1	<a href="#">Run()</a>	34
5.20	<a href="#">NEEDSIMSampleSceneScripts.Lake Class Reference</a>	34
5.20.1	<a href="#">Detailed Description</a>	34
5.21	<a href="#">Simulation.Manager Class Reference</a>	34
5.21.1	<a href="#">Detailed Description</a>	35
5.21.2	<a href="#">Member Function Documentation</a>	35
5.21.2.1	<a href="#">SetAllNeedSatisfactionWeightsToOne()</a>	35
5.21.2.2	<a href="#">UpdateAffordanceTree()</a>	35
5.21.2.3	<a href="#">ZeroValuedNeeds(List&lt; string &gt; needNames)</a>	35
5.21.3	<a href="#">Property Documentation</a>	35
5.21.3.1	<a href="#">Data</a>	35
5.21.3.2	<a href="#">Instance</a>	36
5.22	<a href="#">NEEDSIM.MoveToSlot Class Reference</a>	36
5.22.1	<a href="#">Detailed Description</a>	36
5.22.2	<a href="#">Member Function Documentation</a>	36
5.22.2.1	<a href="#">Run()</a>	36
5.23	<a href="#">Simulation.NeedItem Class Reference</a>	36
5.23.1	<a href="#">Detailed Description</a>	37
5.23.2	<a href="#">Constructor &amp; Destructor Documentation</a>	37
5.23.2.1	<a href="#">NeedItem(string name, float changeRate, float critical, float satisfied)</a>	37
5.23.3	<a href="#">Member Data Documentation</a>	37
5.23.3.1	<a href="#">changePerSecond</a>	37
5.23.3.2	<a href="#">criticalState</a>	37
5.23.3.3	<a href="#">maxValue</a>	37
5.23.3.4	<a href="#">minValue</a>	38
5.23.3.5	<a href="#">needName</a>	38
5.23.3.6	<a href="#">satisfiedState</a>	38
5.24	<a href="#">Simulation.Needs Class Reference</a>	38
5.24.1	<a href="#">Detailed Description</a>	38
5.24.2	<a href="#">Member Enumeration Documentation</a>	39
5.24.2.1	<a href="#">NeedSatisfactions</a>	39
5.24.3	<a href="#">Constructor &amp; Destructor Documentation</a>	39

5.24.3.1	Needs(Species species)	39
5.24.4	Member Function Documentation	39
5.24.4.1	ApplyChangePerSecond()	39
5.24.4.2	ApplyChangeRates(Dictionary< string, float > input)	39
5.24.4.3	GetValue(string key)	39
5.24.4.4	GoalToSatisfyLowestNeed()	39
5.24.4.5	RandomizeValues()	40
5.24.4.6	Satisfactions()	40
5.24.4.7	SetSpecificSatisfactionLevels(Dictionary< string, float > needLevels)	40
5.25	NEEDSIM.NEEDSIMManager Class Reference	40
5.25.1	Detailed Description	40
5.26	NEEDSIM.NEEDSIMNode Class Reference	40
5.26.1	Detailed Description	41
5.26.2	Member Function Documentation	41
5.26.2.1	AcceptSlot()	41
5.26.2.2	ArrivalAtSlot(Slot slot)	41
5.26.2.3	BuildTreeBasedOnSceneHierarchy()	41
5.26.2.4	TryConsumingAnimationOrder(Animator animator)	41
5.27	NEEDSIM.NEEDSIMRoot Class Reference	42
5.27.1	Detailed Description	42
5.27.2	Constructor & Destructor Documentation	42
5.27.2.1	NEEDSIMRoot()	42
5.27.3	Member Function Documentation	42
5.27.3.1	BuildAffordanceTreeFromNode(NEEDSIMNode node)	42
5.27.3.2	BuildFlatAffordanceTreeFromScene()	44
5.28	NEEDSIMSampleSceneScripts.NeedsUI Class Reference	44
5.28.1	Detailed Description	44
5.29	NEEDSIM.PlanDemo Class Reference	44
5.29.1	Detailed Description	44
5.29.2	Member Function Documentation	44
5.29.2.1	Update()	44
5.30	NEEDSIMSampleSceneScripts.SampleCameraControl Class Reference	45
5.30.1	Detailed Description	45
5.31	NEEDSIM.SatisfyGoal Class Reference	45
5.31.1	Detailed Description	45
5.31.2	Member Function Documentation	45
5.31.2.1	Run()	45
5.32	NEEDSIM.SatisfyUrgentNeed Class Reference	46
5.32.1	Detailed Description	46
5.32.2	Member Function Documentation	46



5.32.2.1	Run()	46
5.33	NEEDSIMSampleSceneScripts.SceneSwitcher Class Reference	46
5.33.1	Detailed Description	47
5.34	NEEDSIMSampleSceneScripts.SimpleSpawn Class Reference	47
5.34.1	Detailed Description	47
5.34.2	Member Function Documentation	47
5.34.2.1	fillPopulation()	47
5.34.2.2	killAll()	47
5.35	Simulation.SimulationData Class Reference	47
5.35.1	Detailed Description	48
5.35.2	Constructor & Destructor Documentation	48
5.35.2.1	SimulationData(Dictionary< string, float > minValueDictionary, Dictionary< string, float > maxValueDictionary, Dictionary< string, float > changePerSecondDictionary, Dictionary< string, float > criticalStateDictionary, Dictionary< string, float > satisfiedStateDictionary, List< string > needNames, List< string > speciesNames, List< Species > species, List< Interaction > interactionList, Dictionary< string, Species > speciesByName)	48
5.35.3	Member Data Documentation	49
5.35.3.1	CriticalStateDictionary	49
5.35.3.2	InteractionByNameDictionary	49
5.35.3.3	NeedNames	49
5.35.3.4	SatisfiedStateDictionary	49
5.35.4	Property Documentation	49
5.35.4.1	ChangePerSecond	49
5.35.4.2	Root	49
5.35.4.3	WeightsForNeed	49
5.35.4.4	WeightsForNeedSatisfaction	49
5.36	NEEDSIM3rdParty.Singleton< T > Class Template Reference	50
5.36.1	Detailed Description	50
5.36.2	Member Function Documentation	50
5.36.2.1	OnDestroy()	50
5.37	Simulation.Slot Class Reference	50
5.37.1	Detailed Description	51
5.37.2	Member Enumeration Documentation	51
5.37.2.1	Result	51
5.37.2.2	SlotStates	52
5.37.3	Constructor & Destructor Documentation	52
5.37.3.1	Slot(Affordance affordance)	52
5.37.4	Member Function Documentation	52
5.37.4.1	AgentArrivalEasy(AffordanceTreeNode participant)	52
5.37.4.2	AgentDeparture()	52

5.37.4.3	CurrentInteractionDuration()	53
5.37.4.4	InterruptInteraction()	53
5.37.4.5	OfferSlot()	53
5.37.4.6	ReserveSlot()	53
5.37.4.7	SetSlotBlocked()	53
5.37.5	Property Documentation	53
5.37.5.1	currentInteraction	53
5.37.5.2	IsAuctionable	53
5.37.5.3	LocalLookAt	53
5.37.5.4	LocalPosition	54
5.37.5.5	LookAt	54
5.37.5.6	Position	54
5.37.5.7	SlotState	54
5.38	NEEDSIMSampleSceneScripts.SpawnBedsManager Class Reference	54
5.38.1	Detailed Description	54
5.38.2	Member Function Documentation	54
5.38.2.1	DestroyBed()	54
5.38.2.2	SpawnBed()	54
5.39	NEEDSIMSampleSceneScripts.SpawnUIRuntimeEditing Class Reference	55
5.39.1	Detailed Description	55
5.40	Simulation.Species Class Reference	55
5.40.1	Detailed Description	55
5.40.2	Constructor & Destructor Documentation	55
5.40.2.1	Species(string name, List< string > needs)	55
5.40.3	Member Data Documentation	55
5.40.3.1	needs	55
5.40.3.2	speciesName	56
5.41	Simulation.StringFloatPair Class Reference	56
5.41.1	Detailed Description	56
5.41.2	Constructor & Destructor Documentation	56
5.41.2.1	StringFloatPair(string key, float value)	56
5.41.3	Member Data Documentation	56
5.41.3.1	floatValue	56
5.41.3.2	stringValue	56
5.42	Simulation.Strings Class Reference	57
5.42.1	Detailed Description	59
5.42.2	Member Function Documentation	59
5.42.2.1	DefaultInteractionRelativeNeedName(string needName)	59
5.42.2.2	InteractionLabelToExtend(int extension)	60
5.42.2.3	NewHasBeenSet(string name)	60

5.42.2.4	SlotNumberLabel(int i)	60
5.42.3	Member Data Documentation	60
5.42.3.1	EditorExtensionPosition	60
5.42.3.2	InteractionDurationLabel	60
5.42.3.3	LabelCriticalState	61
5.42.3.4	LabelNeedChangeRate	61
5.42.3.5	LabelNeedName	61
5.42.3.6	LabelSatisfiedState	61
5.42.3.7	NeedAffectedLabel	61
5.42.3.8	NumberOfNeedsAffectedLabel	61
5.42.3.9	ProductDescription	62
5.42.3.10	ProductName	62
5.42.3.11	SatisfactionRateLabel	62
5.42.4	Property Documentation	62
5.42.4.1	AddInteractionButtonLabel	62
5.42.4.2	AddNeedButtonLabel	62
5.42.4.3	AddSpeciesButtonLabel	62
5.42.4.4	AdvancedDatabaseViewHeadline	62
5.42.4.5	AdvancedInstructionLine1	62
5.42.4.6	AdvancedInstructionLine2	62
5.42.4.7	AdvancedInstructionLine3	62
5.42.4.8	AdvancedInstructionLine4	62
5.42.4.9	AdvancedInstructionLine5	62
5.42.4.10	AdvancedSettings	62
5.42.4.11	Advice	62
5.42.4.12	AssetAppendix	62
5.42.4.13	AssetsPath	62
5.42.4.14	AssignNeedLabel	62
5.42.4.15	AssignSpeciesOptionToAgentLabel	62
5.42.4.16	Asterisk	62
5.42.4.17	AttachSpecificDBLabel	62
5.42.4.18	AuctionableBoolArray	62
5.42.4.19	BuildAffordanceTreeFromSceneLabel	62
5.42.4.20	ButtonAddDatabaseLabel	63
5.42.4.21	ButtonRemoveDatabaseLabel	63
5.42.4.22	Contact	63
5.42.4.23	currentDatabaseLoadedLabel	63
5.42.4.24	DataBasesAvailableLabel	63
5.42.4.25	DataFolder	63
5.42.4.26	DebugIngameLabel	63

5.42.4.27 DebugInspectorLabel . . . . .	63
5.42.4.28 DefaultDatabaseHeadline . . . . .	63
5.42.4.29 DefaultDatabaseName . . . . .	63
5.42.4.30 DefaultInteractionName . . . . .	63
5.42.4.31 DefaultNeedName . . . . .	63
5.42.4.32 DefaultSpecies . . . . .	63
5.42.4.33 defaultString . . . . .	63
5.42.4.34 DeleteDatabaseMessage . . . . .	63
5.42.4.35 DeleteDatabaseTitle . . . . .	63
5.42.4.36 DialogButtonExitPlay . . . . .	63
5.42.4.37 DialogCancel . . . . .	63
5.42.4.38 DialogMessageExitPlay . . . . .	63
5.42.4.39 DialogTitleExitPlay . . . . .	63
5.42.4.40 DialogYes . . . . .	63
5.42.4.41 DrawGizmosInGame . . . . .	63
5.42.4.42 EditorWindowTitle . . . . .	63
5.42.4.43 FolderName . . . . .	63
5.42.4.44 GeneralSettings . . . . .	63
5.42.4.45 GenericDatabaseName . . . . .	63
5.42.4.46 InteractionData . . . . .	64
5.42.4.47 InteractionDataArrayAccess . . . . .	64
5.42.4.48 InteractionNameLabel . . . . .	64
5.42.4.49 InteractionViewHeadline . . . . .	64
5.42.4.50 IsAgent . . . . .	64
5.42.4.51 IsAgentLabel . . . . .	64
5.42.4.52 IsAuctionableArrayAccess . . . . .	64
5.42.4.53 IsAuctionableArraySize . . . . .	64
5.42.4.54 IsAuctionableLabel . . . . .	64
5.42.4.55 LogSimulationLabel . . . . .	64
5.42.4.56 MarkCurrentDatabaseAsDefault . . . . .	64
5.42.4.57 ModifyLookAt . . . . .	64
5.42.4.58 MoreThanOneDefaultData . . . . .	64
5.42.4.59 NeedsDataBasePath . . . . .	64
5.42.4.60 NeedsInSpeciesLabel . . . . .	64
5.42.4.61 NoDatabaseFound . . . . .	64
5.42.4.62 NoDefaultDataFound . . . . .	64
5.42.4.63 None . . . . .	64
5.42.4.64 NumberOfSlotsLabel . . . . .	64
5.42.4.65 O_Space . . . . .	64
5.42.4.66 PreconditionDefaultAny . . . . .	65

5.42.4.67 PreconditionsHeadline . . . . .	65
5.42.4.68 PreconditionsSatisfactionLevelLabel . . . . .	65
5.42.4.69 PreconditionsSpecies . . . . .	65
5.42.4.70 PreconditionsSatisfactionLevelArray . . . . .	65
5.42.4.71 PreconditionsUnsatisfiedLabel . . . . .	65
5.42.4.72 PrintSimDebugLogLabel . . . . .	65
5.42.4.73 RandomStartLevelsLabel . . . . .	65
5.42.4.74 RemoveNeedButtonLabel . . . . .	65
5.42.4.75 SceneAppendix . . . . .	65
5.42.4.76 ShowDebugGizmosLabel . . . . .	65
5.42.4.77 ShowDebugInGame . . . . .	65
5.42.4.78 ShowDebugInInspector . . . . .	65
5.42.4.79 SimulationManagerConstructed . . . . .	65
5.42.4.80 SlotPositionLabel . . . . .	65
5.42.4.81 SlotPositionsArray . . . . .	65
5.42.4.82 SlotPositionsArrayAccess . . . . .	65
5.42.4.83 SlotPositionsArraySize . . . . .	65
5.42.4.84 SpeciesName . . . . .	65
5.42.4.85 SpeciesNameLabel . . . . .	66
5.42.4.86 SpeciesViewHeadline . . . . .	66
5.42.4.87 SupportMail . . . . .	66
5.42.4.88 TerritoryCenter . . . . .	66
5.42.4.89 TerritoryControlHeadline . . . . .	66
5.42.4.90 UnableToLoadDatabase . . . . .	66
5.42.4.91 UserDataResources . . . . .	66
5.42.4.92 VisitUsAt . . . . .	66
5.42.4.93 WebsiteURL . . . . .	66
5.42.4.94 WebsiteURLButtonText . . . . .	66
5.42.4.95 Welcome . . . . .	66
5.42.4.96 WindowSelectionNames . . . . .	66
5.42.4.97 YourDatabaseName . . . . .	66
5.43 NEEDSIMSampleSceneScripts.TimeSystem Class Reference . . . . .	66
5.43.1 Detailed Description . . . . .	66
<b>Index</b>	<b>67</b>



## Chapter 1

# Namespace Index

### 1.1 Packages

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

<a href="#">NEEDSIM</a>	9
<a href="#">NEEDSIM3rdParty</a>	10
<a href="#">NEEDSIMEditor</a>	10
<a href="#">NEEDSIMSampleSceneScripts</a>	10
<a href="#">Simulation</a>	11
<a href="#">Simulation.Tests</a>	11





## Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

NEEDSIM.Action . . . . .	13
NEEDSIM.ChaseSlot . . . . .	21
NEEDSIM.DecideGoal . . . . .	24
NEEDSIM.DecideValue . . . . .	25
NEEDSIM.InterruptionFuchsalarm . . . . .	34
NEEDSIM.MoveToSlot . . . . .	36
NEEDSIM.SatisfyGoal . . . . .	45
NEEDSIM.SatisfyUrgentNeed . . . . .	46
Simulation.Affordance . . . . .	13
Simulation.AffordanceTreeNode . . . . .	15
NEEDSIMSampleSceneScripts.Animal . . . . .	19
NEEDSIMSampleSceneScripts.Bunny . . . . .	21
NEEDSIMSampleSceneScripts.Deer . . . . .	25
NEEDSIMSampleSceneScripts.Fox . . . . .	26
NEEDSIM.Blackboard . . . . .	20
Simulation.DatabaseAsset . . . . .	22
NEEDSIMSampleSceneScripts.FuchsalarmDemoScript . . . . .	26
Simulation.GeneralSettings . . . . .	27
Simulation.Goal . . . . .	28
NEEDSIMSampleSceneScripts.InputFieldRuntimeEditing . . . . .	30
Simulation.Interaction . . . . .	30
Simulation.InteractionData . . . . .	32
NEEDSIMSampleSceneScripts.Lake . . . . .	34
Simulation.Manager . . . . .	34
Simulation.NeedItem . . . . .	36
Simulation.Needs . . . . .	38
NEEDSIM.NEEDSIMManager . . . . .	40
NEEDSIM.NEEDSIMNode . . . . .	40
NEEDSIMSampleSceneScripts.NeedsUI . . . . .	44
NEEDSIM.PlanDemo . . . . .	44
NEEDSIMSampleSceneScripts.SampleCameraControl . . . . .	45
NEEDSIMSampleSceneScripts.SceneSwitcher . . . . .	46
NEEDSIMSampleSceneScripts.SimpleSpawn . . . . .	47
Simulation.SimulationData . . . . .	47
NEEDSIM3rdParty.Singleton< T > . . . . .	50
NEEDSIM3rdParty.Singleton< NEEDSIMRoot > . . . . .	50
NEEDSIM.NEEDSIMRoot . . . . .	42

Simulation.Slot . . . . .	50
NEEDSIMSampleSceneScripts.SpawnBedsManager . . . . .	54
NEEDSIMSampleSceneScripts.SpawnUIRuntimeEditing . . . . .	55
Simulation.Species . . . . .	55
Simulation.StringFloatPair . . . . .	56
Simulation.Strings . . . . .	57
NEEDSIMSampleSceneScripts.TimeSystem . . . . .	66

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">NEEDSIM.Action</a>	We hope we wrote our example actions in a way that they can be integrated into your Finite State Machine, Behavior Tree, or Planner. We provide a sample use of our sample actions in the PlanDemo.cs class. . . . .	13
<a href="#">Simulation.Affordance</a>	An affordance is the opportunity granted by an object to an agent to perform an action. . . . .	13
<a href="#">Simulation.AffordanceTreeNode</a>	Each game object managed by the simulation has an <a href="#">Affordance</a> Tree Node. It manages scope, affordances, slots, and, if the affordance tree is an active part of the simulation, its goals and levels of satisfaction. . . . .	15
<a href="#">NEEDSIMSampleSceneScripts.Animal</a>	This example suggests an idea for playing animations based on the states of NEEDSIMNodes. . . . .	19
<a href="#">NEEDSIM.Blackboard</a>	This is used to store some values, and make some methods available in a place where they can be edited without affecting the other classes. . . . .	20
<a href="#">NEEDSIMSampleSceneScripts.Bunny</a>	This example script shows how the bunny can deal with the 'EatBunny' interaction, that can be performed by a fox at the slot provided by a bunny. . . . .	21
<a href="#">NEEDSIM.ChaseSlot</a>	This is an example for how a chasing behavior could be implemented in <a href="#">NEEDSIM</a> Life simulation. For a specific game better solutions might be desirable. . . . .	21
<a href="#">Simulation.DatabaseAsset</a>	Stores the data used by the <a href="#">NEEDSIM</a> Life simulation . . . . .	22
<a href="#">NEEDSIM.DecideGoal</a>	For goal oriented behaviors: Get a goal from the simulation, and try to get a slot where the goal can be satisfied. . . . .	24
<a href="#">NEEDSIM.DecideValue</a>	For Value-oriented behaviors: Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent). . . . .	25
<a href="#">NEEDSIMSampleSceneScripts.Deer</a>	The deer currently has no special features. Rather check out the fox and the bunny for now ;) . . . . .	25
<a href="#">NEEDSIMSampleSceneScripts.Fox</a>	Sets the fox to either run or walk . . . . .	26
<a href="#">NEEDSIMSampleSceneScripts.FuchsalarmDemoScript</a>	This script shows how the behaviors of the bunnies are interrupted when the fox is spawned. . . . .	26
<a href="#">Simulation.GeneralSettings</a>	A number of values used as configuration of the simulation and editor. . . . .	27

<a href="#">Simulation.Goal</a>	A goal is a satisfaction level of a need that an agent wants to achieve. . . . .	28
<a href="#">NEEDSIMSampleSceneScripts.InputFieldRuntimeEditing</a>	Helper to change need satisfaction rates of interactions and satisfaction change rates of needs at runtime in a UI. It provides a method to react to the user finishing his input to the InputField.	30
<a href="#">Simulation.Interaction</a>	An interaction at runtime . . . . .	30
<a href="#">Simulation.InteractionData</a>	From these data items used in the editor the interactions for the runtime will be generated. . .	32
<a href="#">NEEDSIM.InterruptionFuchsalarm</a>	This action demonstrates how interruption of typical <a href="#">NEEDSIM</a> behaviors could look like. . . .	34
<a href="#">NEEDSIMSampleSceneScripts.Lake</a>	This example suggests an idea for how a variety of animations can be played at an interactive object. . . . .	34
<a href="#">Simulation.Manager</a>	A singleton for managing the simulation at runtime . . . . .	34
<a href="#">NEEDSIM.MoveToSlot</a>	Moving to a slot. The best implementation for such a behavior might be different in your project, but this script offers a starting point. . . . .	36
<a href="#">Simulation.NeedItem</a>	A need, one of the core building blocks of <a href="#">NEEDSIMLifeSimulation</a> . . . . .	36
<a href="#">Simulation.Needs</a>	The <a href="#">Needs</a> class provides methods for interacting with the satisfaction levels of agents. . . . .	38
<a href="#">NEEDSIM.NEEDSIMManager</a>	This class stores the values that the <a href="#">NEEDSIMROOT</a> will use for running the simulation . . . . .	40
<a href="#">NEEDSIM.NEEDSIMNode</a>	Every object and agent in <a href="#">NEEDSIM</a> Life simulation has a <a href="#">NEEDSIMNode</a> : This is the essential component for using <a href="#">NEEDSIM</a> Life simulation. . . . .	40
<a href="#">NEEDSIM.NEEDSIMRoot</a>	Every scene should have one root node for the <a href="#">AFFORDANCE TREE</a> . This uses the settings of the <a href="#">NEEDSIM</a> Manager and controls the simulation. . . . .	42
<a href="#">NEEDSIMSampleSceneScripts.NeedsUI</a>	This class shows bars for need satisfaction. A full bar equals full satisfaction, an empty bar means the need is not satisfied. If the need is currently being satisfied an outline will be added.	44
<a href="#">NEEDSIM.PlanDemo</a>	A simple behavior control solution. We tried to write this in a way that makes it easy to use our code samples in Finite State Machines, Behavior Trees and Goal-oriented <a href="#">Action</a> Planning. The idea is that you can run our simulation from within a different solution, for example in case you want to have agents with fighting capabilities. . . . .	44
<a href="#">NEEDSIMSampleSceneScripts.SampleCameraControl</a>	A very simple scrolling camera for <a href="#">NEEDSIM</a> Life simulation example scenes. . . . .	45
<a href="#">NEEDSIM.SatisfyGoal</a>	Participate in a slot to satisfy a goal. . . . .	45
<a href="#">NEEDSIM.SatisfyUrgentNeed</a>	Participating a slot. The respective behavior for value/urgency oriented behaviors. . . . .	46
<a href="#">NEEDSIMSampleSceneScripts.SceneSwitcher</a>	public methods to switch scenes via a button click. You have to add the scenes to your build settings to use the prefab that uses this script. . . . .	46
<a href="#">NEEDSIMSampleSceneScripts.SimpleSpawn</a>	This is a spawning example script to maintain populations . . . . .	47
<a href="#">Simulation.SimulationData</a>	The data loaded in the simulation at runtime . . . . .	47
<a href="#">NEEDSIM3rdParty.Singleton&lt; T &gt;</a>	Be aware this will not prevent a non singleton constructor such as <code>T myT = new T();</code> To prevent that, add <code>protected T () {}</code> to your singleton class . . . . .	50
<a href="#">Simulation.Slot</a>	A slot is a position in the world where an agent can run the interactions provided by the object that offers the slot. . . . .	50

<a href="#">NEEDSIMSampleSceneScripts.SpawnBedsManager</a>	
This example shows how you could spawn all the objects and agents procedurally. . . . .	54
<a href="#">NEEDSIMSampleSceneScripts.SpawnUIRuntimeEditing</a>	
Spawns a UI Element for each need and each satisfaction rate of an interaction. . . . .	55
<a href="#">Simulation.Species</a>	
A species is a set of needs. For example zombies might only have the 'Hunger' need, whereas humans furthermore have a 'Social' need. . . . .	55
<a href="#">Simulation.StringFloatPair</a>	
A class that helps creating key value pairs. . . . .	56
<a href="#">Simulation.Strings</a>	
A centralized place for many of the strings used by the <a href="#">NEEDSIM</a> Life simulation. . . . .	57
<a href="#">NEEDSIMSampleSceneScripts.TimeSystem</a>	
This example uses arrays with 24 values each to modify how behaviors are evaluated at a specific time of day. This class works with Value Oriented behaviors, not with Goal Oriented behaviors, because it relies on the fact that all opportunities to satisfy needs are evaluated, not only the opportunities that can satisfy the need of the current goal. . . . .	66



## Chapter 4

# Namespace Documentation

### 4.1 NEEDSIM Namespace Reference

#### Classes

- class [Action](#)  
*We hope we wrote our example actions in a way that they can be integrated into your Finite State Machine, Behavior Tree, or Planner. We provide a sample use of our sample actions in the PlanDemo.cs class.*
- class [Blackboard](#)  
*This is used to store some values, and make some methods available in a place where they can be edited without affecting the other classes.*
- class [ChaseSlot](#)  
*This is an example for how a chasing behavior could be implemented in [NEEDSIM](#) Life simulation. For a specific game better solutions might be desirable.*
- class [DecideGoal](#)  
*For goal oriented behaviors: Get a goal from the simulation, and try to get a slot where the goal can be satisfied.*
- class [DecideValue](#)  
*For Value-oriented behaviors: Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent).*
- class [InterruptionFuchsalarm](#)  
*This action demonstrates how interruption of typical [NEEDSIM](#) behaviors could look like.*
- class [MoveToSlot](#)  
*Moving to a slot. The best implementation for such a behavior might be different in your project, but this script offers a starting point.*
- class [NEEDSIMManager](#)  
*This class stores the values that the NEEDSIMROOT will use for running the simulation*
- class [NEEDSIMNode](#)  
*Every object and agent in [NEEDSIM](#) Life simulation has a [NEEDSIMNode](#): This is the essential component for using [NEEDSIM](#) Life simulation.*
- class [NEEDSIMRoot](#)  
*Every scene should have one root node for the AFFORDANCE TREE. This uses the settings of the [NEEDSIM](#) Manager and controls the simulation.*
- class [PlanDemo](#)  
*A simple behavior control solution. We tried to write this in a way that makes it easy to use our code samples in Finite State Machines, Behavior Trees and Goal-oriented [Action](#) Planning. The idea is that you can run our simulation from within a different solution, for example in case you want to have agents with fighting capabilities.*
- class [SatisfyGoal](#)  
*Participate in a slot to satisfy a goal.*
- class [SatisfyUrgentNeed](#)  
*Participating a slot. The respective behavior for value/urgency oriented behaviors.*

## 4.2 NEEDSIM3rdParty Namespace Reference

### Classes

- class [Singleton](#)

*Be aware this will not prevent a non singleton constructor such as `T myT = new T();` To prevent that, add `protected T () {}` to your singleton class.*

## 4.3 NEEDSIMEditor Namespace Reference

## 4.4 NEEDSIMSampleSceneScripts Namespace Reference

### Classes

- class [Animal](#)

*This example suggests an idea for playing animations based on the states of [NEEDSIMNodes](#).*

- class [Bunny](#)

*This example script shows how the bunny can deal with the 'EatBunny' interaction, that can be performed by a fox at the slot provided by a bunny.*

- class [Deer](#)

*The deer currently has no special features. Rather check out the fox and the bunny for now ;)*

- class [Fox](#)

*Sets the fox to either run or walk*

- class [FuchsalarmDemoScript](#)

*This script shows how the behaviors of the bunnies are interrupted when the fox is spawned.*

- class [InputFieldRuntimeEditing](#)

*Helper to change need satisfaction rates of interactions and satisfaction change rates of needs at runtime in a UI. It provides a method to react to the user finishing his input to the [InputField](#).*

- class [Lake](#)

*This example suggests an idea for how a variety of animations can be played at an interactive object.*

- class [NeedsUI](#)

*This class shows bars for need satisfaction. A full bar equals full satisfaction, an empty bar means the need is not satisfied. If the need is currently being satisfied an outline will be added.*

- class [SampleCameraControl](#)

*A very simple scrolling camera for [NEEDSIM](#) Life simulation example scenes.*

- class [SceneSwitcher](#)

*public methods to switch scenes via a button click. You have to add the scenes to your build settings to use the prefab that uses this script.*

- class [SimpleSpawn](#)

*This is a spawning example script to maintain populations*

- class [SpawnBedsManager](#)

*This example shows how you could spawn all the objects and agents procedurally.*

- class [SpawnUIRuntimeEditing](#)

*Spawns a UI Element for each need and each satisfaction rate of an interaction.*

- class [TimeSystem](#)

*This example uses arrays with 24 values each to modify how behaviors are evaluated at a specific time of day. This class works with Value Oriented behaviors, not with Goal Oriented behaviors, because it relies on the fact that all opportunities to satisfy needs are evaluated, not only the opportunities that can satisfy the need of the current goal.*



## 4.5 Simulation Namespace Reference

### Namespaces

- namespace [Tests](#)

### Classes

- class [Affordance](#)  
*An affordance is the opportunity granted by an object to an agent to perform an action.*
- class [AffordanceTreeNode](#)  
*Each game object managed by the simulation has an [Affordance](#) Tree Node. It manages scope, affordances, slots, and, if the affordance tree is an active part of the simulation, its goals and levels of satisfaction.*
- class [DatabaseAsset](#)  
*Stores the data used by the [NEEDSIM](#) Life simulation*
- class [GeneralSettings](#)  
*A number of values used as configuration of the simulation and editor.*
- class [Goal](#)  
*A goal is a satisfaction level of a need that an agent wants to achieve.*
- class [Interaction](#)  
*An interaction at runtime*
- class [InteractionData](#)  
*From these data items used in the editor the interactions for the runtime will be generated.*
- class [Manager](#)  
*A singleton for managing the simulation at runtime*
- class [NeedItem](#)  
*A need, one of the core building blocks of [NEEDSIMLifeSimulation](#)*
- class [Needs](#)  
*The [Needs](#) class provides methods for interacting with the satisfaction levels of agents.*
- class [SimulationData](#)  
*The data loaded in the simulation at runtime*
- class [Slot](#)  
*A slot is a position in the world where an agent can run the interactions provided by the object that offers the slot.*
- class [Species](#)  
*A species is a set of needs. For example zombies might only have the 'Hunger' need, whereas humans furthermore have a 'Social' need.*
- class [StringFloatPair](#)  
*A class that helps creating key value pairs.*
- class [Strings](#)  
*A centralized place for many of the strings used by the [NEEDSIM](#) Life simulation.*

## 4.6 Simulation.Tests Namespace Reference



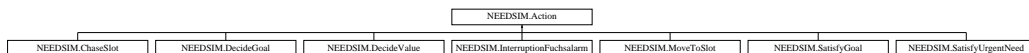
## Chapter 5

# Class Documentation

### 5.1 NEEDSIM.Action Class Reference

We hope we wrote our example actions in a way that they can be integrated into your Finite State Machine, Behavior Tree, or Planner. We provide a sample use of our sample actions in the PlanDemo.cs class.

Inheritance diagram for NEEDSIM.Action:



#### 5.1.1 Detailed Description

We hope we wrote our example actions in a way that they can be integrated into your Finite State Machine, Behavior Tree, or Planner. We provide a sample use of our sample actions in the PlanDemo.cs class.

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

- Action.cs

### 5.2 Simulation.Affordance Class Reference

An affordance is the opportunity granted by an object to an agent to perform an action.

#### Public Member Functions

- [Slot.Result ProlongLastInteraction \(\)](#)  
*Prolong the last interaction. Only available if interaction already finished, will then restart without setting the frame the animation started in.*
- [Slot.Result StartInteraction \(string name\)](#)  
*Start [Interaction](#) by name.*
- [Slot.Result StartRandomInteraction \(\)](#)  
*From all interactions registered at this affordance, pick one randomly*
- bool [AddInteraction \(Interaction interaction\)](#)  
*Make an interaction available at this affordance*

## Properties

- bool [InteractionStartedThisFrame](#) [get]  
*Whether the interaction was started in the current frame*
- [Interaction](#) [CurrentInteraction](#) [get]  
*This will return null if the remaining duration is 0.*
- bool [HasInteraction](#) [get]  
*Whether an actual [Interaction](#) is available behind this affordance.*

### 5.2.1 Detailed Description

An affordance is the opportunity granted by an object to an agent to perform an action.

### 5.2.2 Member Function Documentation

#### 5.2.2.1 bool [Simulation.Affordance.AddInteraction](#) ( [Interaction](#) *interaction* )

Make an interaction available at this affordance

Parameters

<i>interaction</i>	The interaction you want to add.
--------------------	----------------------------------

#### 5.2.2.2 Slot.Result [Simulation.Affordance.ProlongLastInteraction](#) ( )

Prolong the last interaction. Only available if interaction already finished, will then restart without setting the frame the animation started in.

Returns

NoProlongableInteraction if none is found. The result of StartInteraction otherwise.

#### 5.2.2.3 Slot.Result [Simulation.Affordance.StartInteraction](#) ( string *name* )

Start [Interaction](#) by name.

Parameters

<i>name</i>	
-------------	--

Returns

#### 5.2.2.4 Slot.Result [Simulation.Affordance.StartRandomInteraction](#) ( )

From all interactions registered at this affordance, pick one randomly

Returns

### 5.2.3 Property Documentation

#### 5.2.3.1 Interaction Simulation.Affordance.CurrentInteraction [get]

This will return null if the remaining duration is 0.

#### 5.2.3.2 bool Simulation.Affordance.HasInteraction [get]

Whether an actual [Interaction](#) is available behind this affordance.

#### 5.2.3.3 bool Simulation.Affordance.InteractionStartedThisFrame [get]

Whether the interaction was started in the current frame

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

- Affordance.cs

## 5.3 Simulation.AffordanceTreeNode Class Reference

Each game object managed by the simulation has an [Affordance](#) Tree Node. It manages scope, affordances, slots, and, if the affordance tree is an active part of the simulation, its goals and levels of satisfaction.

### Public Member Functions

- [AffordanceTreeNode](#) ([AffordanceTreeNode](#) parent, string name, string speciesName, Vector3 position)  
*Create a new [AffordanceTreeNode](#) and attach it to the parent node*
- [Slot AvailableSlot](#) (bool consumeSlot)  
*Returns a slot that has been won in an auction. Can be consumed in the process of asking for it.*
- bool [AcceptAuctionVictory](#) ([Slot](#) slot)  
*Accepting a slot (for which a bid was previously placed).*
- bool [AddSlot](#) (Vector3 worldPosition, Vector3 localPosition)  
*Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.*
- bool [AddSlot](#) (Vector3 worldPosition, Vector3 localPosition, Vector3 lookAtTarget)  
*Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.*
- bool [AddSlot](#) (Vector3 worldPosition, Vector3 localPosition, Vector3 lookAtTarget, bool isAuctionable)  
*Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.*
- void [ApplyParentInteraction](#) ()  
*Apply the effects of the interaction that is running at the parent node to this node.*
- bool [setSpecies](#) (string name)  
*Set the species of this node to a specific kind.*
- void [Remove](#) ()  
*Remove/Delete this affordance tree node at an appropriate time*
- string [Printer](#) (bool reportAffordances)  
*Returns a string with debug information. Recursive (reports this node and its children).*
- int [CountOfChildren](#) ()  
*How many children ([Affordance](#) Tree nodes) this node has.*

### Properties

- [Affordance Affordance](#) [get]  
*The opportunities for interaction at this node, and an abstraction of the opportunities to interact with children nodes.*
- [Species Species](#) [get]  
*A species determines which needs an agent has. Only required for agents.*
- [Goal Goal](#) [get, set]  
*Goals are optional, and can be used to make decisions.*
- [Needs SatisfactionLevels](#) [get]  
*Each need has a value between its min and max value: its satisfaction level.*
- string [MostUrgentNeed](#) [get]  
*Which need is most urgent, taking weights as well as current satisfaction level into account.*
- bool [CurrentInteractionSatisfiesMostUrgentNeed](#) [get]  
*Whether the interaction this node is currently participating in (if any) can satisfy his most urgent need (taking weights as well as current satisfaction level into account)*
- List< [Slot](#) > [Slots](#) [get]  
*All the slots that belong to this [Affordance](#) Tree Node*
- [AffordanceTreeNode Parent](#) [get, set]  
*The parent of this node*
- [AffordanceTreeNode Root](#) [get]  
*The root of the [Affordance](#) Tree*

### 5.3.1 Detailed Description

Each game object managed by the simulation has an [Affordance](#) Tree Node. It manages scope, affordances, slots, and, if the affordance tree is an active part of the simulation, its goals and levels of satisfaction.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 Simulation.AffordanceTreeNode.AffordanceTreeNode ( [AffordanceTreeNode](#) parent, string name, string speciesName, Vector3 position )

Create a new [AffordanceTreeNode](#) and attach it to the parent node

#### Parameters

<i>parent</i>	The parent in the <a href="#">Affordance</a> Tree
<i>name</i>	The name of the node, useful for debugging
<i>speciesName</i>	If this is an agent, which species it is
<i>position</i>	World position

### 5.3.3 Member Function Documentation

#### 5.3.3.1 bool Simulation.AffordanceTreeNode.AcceptAuctionVictory ( [Slot](#) slot )

Accepting a slot (for which a bid was previously placed).

#### Parameters

<i>slot</i>	The slot this affordance tree node should consume (usually resulting in going to the slot and running an interaction)
-------------	---

#### Returns

Whether a slot was made available for consumption

### 5.3.3.2 bool Simulation.AffordanceTreeNode.AddSlot ( Vector3 *worldPosition*, Vector3 *localPosition* )

Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.

#### Parameters

<i>worldPosition</i>	The absolute position in the game world
<i>localPosition</i>	The position relative to the parent transform

#### Returns

Whether the slot was successfully added

### 5.3.3.3 bool Simulation.AffordanceTreeNode.AddSlot ( Vector3 *worldPosition*, Vector3 *localPosition*, Vector3 *lookAtTarget* )

Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.

#### Parameters

<i>worldPosition</i>	The absolute position in the game world
<i>localPosition</i>	The position relative to the parent transform
<i>lookAtTarget</i>	A position that characters can be oriented towards for better animation results

#### Returns

Whether the slot was successfully added

### 5.3.3.4 bool Simulation.AffordanceTreeNode.AddSlot ( Vector3 *worldPosition*, Vector3 *localPosition*, Vector3 *lookAtTarget*, bool *isAuctionable* )

Adds a slot (a location to run interactions) to this [Affordance](#) Tree Node.

#### Parameters

<i>worldPosition</i>	The absolute position in the game world
<i>localPosition</i>	The position relative to the parent transform
<i>lookAtTarget</i>	A position that characters can be oriented towards for better animation results
<i>isAuctionable</i>	If the slot is not auctionable it will not be offered to agents.

#### Returns

Whether the slot was successfully added

### 5.3.3.5 void Simulation.AffordanceTreeNode.ApplyParentInteraction ( )

Apply the effects of the interaction that is running at the parent node to this node.

### 5.3.3.6 Slot Simulation.AffordanceTreeNode.AvailableSlot ( bool *consumeSlot* )

Returns a slot that has been won in an auction. Can be consumed in the process of asking for it.

## Parameters

<i>consumeSlot</i>	Whether or not the slot should be consumed
--------------------	--

## Returns

The slot that can be consumed

### 5.3.3.7 int Simulation.AffordanceTreeNode.CountOfChildren ( )

How many children ([Affordance](#) Tree nodes) this node has.

## Returns

### 5.3.3.8 string Simulation.AffordanceTreeNode.Printer ( bool *reportAffordances* )

Returns a string with debug information. Recursive (reports this node and its children).

## Parameters

<i>report↔ Affordances</i>	Adds detailed information about the affordances to the debug info.
--------------------------------	--

## Returns

String with debug information

### 5.3.3.9 void Simulation.AffordanceTreeNode.Remove ( )

Remove/Delete this affordance tree node at an appropriate time

### 5.3.3.10 bool Simulation.AffordanceTreeNode.setSpecies ( string *name* )

Set the species of this node to a specific kind.

## Parameters

<i>name</i>	The name (identifier) of the species
-------------	--------------------------------------

## Returns

Whether the species was successfully set.

## 5.3.4 Property Documentation

### 5.3.4.1 Affordance Simulation.AffordanceTreeNode.Affordance [get]

The opportunities for interaction at this node, and an abstraction of the opportunities to interact with children nodes.

### 5.3.4.2 bool Simulation.AffordanceTreeNode.CurrentInteractionSatisfiesMostUrgentNeed [get]

Whether the interaction this node is currently participating in (if any) can satisfy his most urgent need (taking weights as well as current satisfaction level into account)



5.3.4.3 **Goal** `Simulation.AffordanceTreeNode.Goal` [get], [set]

Goals are optional, and can be used to make decisions.

5.3.4.4 **string** `Simulation.AffordanceTreeNode.MostUrgentNeed` [get]

Which need is most urgent, taking weights as well as current satisfaction level into account.

5.3.4.5 **AffordanceTreeNode** `Simulation.AffordanceTreeNode.Parent` [get], [set]

The parent of this node

5.3.4.6 **AffordanceTreeNode** `Simulation.AffordanceTreeNode.Root` [get]

The root of the [Affordance](#) Tree

5.3.4.7 **Needs** `Simulation.AffordanceTreeNode.SatisfactionLevels` [get]

Each need has a value between its min and max value: its satisfaction level.

5.3.4.8 **List<Slot>** `Simulation.AffordanceTreeNode.Slots` [get]

All the slots that belong to this [Affordance](#) Tree Node

5.3.4.9 **Species** `Simulation.AffordanceTreeNode.Species` [get]

A species determines which needs an agent has. Only required for agents.

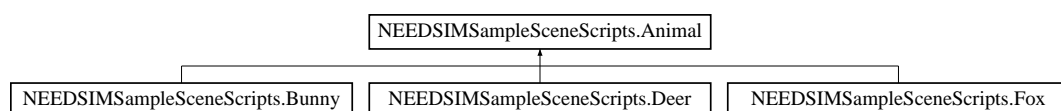
The documentation for this class was generated from the following file:

- AffordanceTreeNode.cs

## 5.4 NEEDSIMSampleSceneScripts.Animal Class Reference

This example suggests an idea for playing animations based on the states of NEEDSIMNodes.

Inheritance diagram for NEEDSIMSampleSceneScripts.Animal:



### 5.4.1 Detailed Description

This example suggests an idea for playing animations based on the states of NEEDSIMNodes.

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

- Animal.cs

## 5.5 NEEDSIM.Blackboard Class Reference

This is used to store some values, and make some methods available in a place where they can be edited without affecting the other classes.

### Public Member Functions

- bool [HasArrivedAtSlot](#) ()

*Depending on your game and your NavMesh you might have to change the conditions here.*

- bool [slotToAgentDistanceSmall](#) (Vector3 agentPosition)

*Whether the slot and the agent are as close as defined in the smallDistance value*

- bool [AcceptSlot](#) (Simulation.Slot slot)

*Do the necessary step to follow up on accepting a slot. Set destination on nav mesh, and put agent into the correct state*

### 5.5.1 Detailed Description

This is used to store some values, and make some methods available in a place where they can be edited without affecting the other classes.

### 5.5.2 Member Function Documentation

#### 5.5.2.1 bool NEEDSIM.Blackboard.AcceptSlot ( Simulation.Slot slot )

Do the necessary step to follow up on accepting a slot. Set destination on nav mesh, and put agent into the correct state

##### Parameters

<i>slot</i>	
-------------	--

##### Returns

#### 5.5.2.2 bool NEEDSIM.Blackboard.HasArrivedAtSlot ( )

Depending on your game and your NavMesh you might have to change the conditions here.

##### Returns

Whether the NavMeshAgent has arrived at the slot.

#### 5.5.2.3 bool NEEDSIM.Blackboard.slotToAgentDistanceSmall ( Vector3 agentPosition )

Whether the slot and the agent are as close as defined in the smallDistance value

##### Parameters

<i>agentPosition</i>	
----------------------	--

#### Returns

true if agent is closer than small distance to target

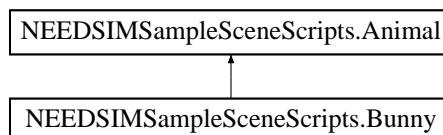
The documentation for this class was generated from the following file:

- Blackboard.cs

## 5.6 NEEDSIMSampleSceneScripts.Bunny Class Reference

This example script shows how the bunny can deal with the 'EatBunny' interaction, that can be performed by a fox at the slot provided by a bunny.

Inheritance diagram for NEEDSIMSampleSceneScripts.Bunny:



### 5.6.1 Detailed Description

This example script shows how the bunny can deal with the 'EatBunny' interaction, that can be performed by a fox at the slot provided by a bunny.

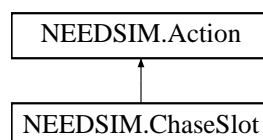
The documentation for this class was generated from the following file:

- Bunny.cs

## 5.7 NEEDSIM.ChaseSlot Class Reference

This is an example for how a chasing behavior could be implemented in [NEEDSIM](#) Life simulation. For a specific game better solutions might be desirable.

Inheritance diagram for NEEDSIM.ChaseSlot:



### Public Member Functions

- override Action.Result [Run](#) ()

*Go to the slot that has been given to the agent.*

### 5.7.1 Detailed Description

This is an example for how a chasing behavior could be implemented in [NEEDSIM](#) Life simulation. For a specific game better solutions might be desirable.

### 5.7.2 Member Function Documentation

#### 5.7.2.1 override Action.Result NEEDSIM.ChaseSlot.Run ( ) [virtual]

Go to the slot that has been given to the agent.

#### Returns

Running as long as on the way. Success upon arrival.

Implements [NEEDSIM.Action](#).

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

- ChaseSlot.cs

## 5.8 Simulation.DatabaseAsset Class Reference

Stores the data used by the [NEEDSIM](#) Life simulation

Inherits ScriptableObject.

### Public Member Functions

- void [Init](#) (string databaseName, List< [NeedItem](#) > needs, List< [Species](#) > species, List< [InteractionData](#) > interactions, bool isDefaultDatabase)  
*Initializes the database asset with values*
- string[] [GetNeedNames](#) ()  
*Constructs a new array with the names of all needs.*
- string[] [GetSpeciesNames](#) ()  
*Constructs a new array with the names of all species.*
- string[] [GetInteractionNames](#) ()  
*Constructs a new array with the names of all interactions.*

### Public Attributes

- string [DatabaseName](#)  
*The name of the database*
- bool [isDefault](#)  
*Whether this is the one default database.*
- List< [NeedItem](#) > [NeedsList](#)  
*All the needs in the database*
- List< [Species](#) > [Species](#)  
*All the species in the database*
- List< [InteractionData](#) > [Interactions](#)  
*All the interactions in the database.*

### 5.8.1 Detailed Description

Stores the data used by the [NEEDSIM](#) Life simulation

### 5.8.2 Member Function Documentation

#### 5.8.2.1 `string [] Simulation.DatabaseAsset.GetInteractionNames ( )`

Constructs a new array with the names of all interactions.

##### Returns

A new array with the name of all interactions.

#### 5.8.2.2 `string [] Simulation.DatabaseAsset.GetNeedNames ( )`

Constructs a new array with the names of all needs.

##### Returns

A new array with the name of all needs.

#### 5.8.2.3 `string [] Simulation.DatabaseAsset.GetSpeciesNames ( )`

Constructs a new array with the names of all species.

##### Returns

A new array with the name of all species.

#### 5.8.2.4 `void Simulation.DatabaseAsset.Init ( string databaseName, List< NeedItem > needs, List< Species > species, List< InteractionData > interactions, bool isDefaultDatabase )`

Initializes the database asset with values

##### Parameters

<i>databaseName</i>	The name of the database
<i>needs</i>	The needs in the database
<i>species</i>	The species in the databse
<i>interactions</i>	The interactions in the database
<i>isDefaultDatabase</i>	Whether this is the one default database.

### 5.8.3 Member Data Documentation

#### 5.8.3.1 `string Simulation.DatabaseAsset.DatabaseName`

The name of the database

#### 5.8.3.2 `List<InteractionData> Simulation.DatabaseAsset.Interactions`

All the interactions in the database.

#### 5.8.3.3 bool Simulation.DatabaseAsset.isDefault

Whether this is the one default database.

#### 5.8.3.4 List<NeedItem> Simulation.DatabaseAsset.NeedsList

All the needs in the database

#### 5.8.3.5 List<Species> Simulation.DatabaseAsset.Species

All the species in the database

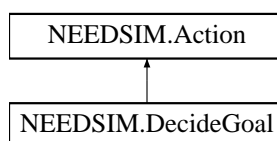
The documentation for this class was generated from the following file:

- DatabaseAsset.cs

## 5.9 NEEDSIM.DecideGoal Class Reference

For goal oriented behaviors: Get a goal from the simulation, and try to get a slot where the goal can be satisfied.

Inheritance diagram for NEEDSIM.DecideGoal:



### Public Member Functions

- override Action.Result [Run](#) ()  
*Get a goal from the simulation, and try to get a slot where the goal can be satisfied.*

#### 5.9.1 Detailed Description

For goal oriented behaviors: Get a goal from the simulation, and try to get a slot where the goal can be satisfied.

#### 5.9.2 Member Function Documentation

##### 5.9.2.1 override Action.Result NEEDSIM.DecideGoal.Run ( ) [virtual]

Get a goal from the simulation, and try to get a slot where the goal can be satisfied.

#### Returns

Success if a slot has been distributed to the agent.

Implements [NEEDSIM.Action](#).

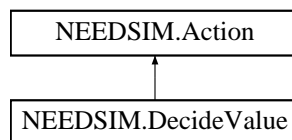
The documentation for this class was generated from the following file:

- DecideGoal.cs

## 5.10 NEEDSIM.DecideValue Class Reference

For Value-oriented behaviors: Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent).

Inheritance diagram for NEEDSIM.DecideValue:



### Public Member Functions

- override Action.Result [Run](#) ()

*Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent).*

### 5.10.1 Detailed Description

For Value-oriented behaviors: Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent).

### 5.10.2 Member Function Documentation

#### 5.10.2.1 override Action.Result NEEDSIM.DecideValue.Run ( ) [virtual]

Try to get a slot based on utility of all available slots (relative to the current satisfaction level of the needs of the agent).

#### Returns

Success if a slot has been distributed to the agent.

Implements [NEEDSIM.Action](#).

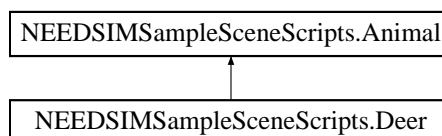
The documentation for this class was generated from the following file:

- DecideValue.cs

## 5.11 NEEDSIMSampleSceneScripts.Deer Class Reference

The deer currently has no special features. Rather check out the fox and the bunny for now ;)

Inheritance diagram for NEEDSIMSampleSceneScripts.Deer:



### 5.11.1 Detailed Description

The deer currently has no special features. Rather check out the fox and the bunny for now ;)

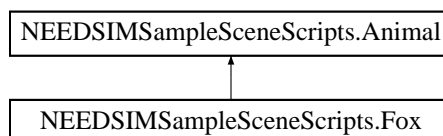
The documentation for this class was generated from the following file:

- Deer.cs

## 5.12 NEEDSIMSampleSceneScripts.Fox Class Reference

Sets the fox to either run or walk

Inheritance diagram for NEEDSIMSampleSceneScripts.Fox:



### 5.12.1 Detailed Description

Sets the fox to either run or walk

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

- Fox.cs

## 5.13 NEEDSIMSampleSceneScripts.FuchsalarmDemoScript Class Reference

This script shows how the behaviors of the bunnies are interrupted when the fox is spawned.

Inherits MonoBehaviour.

### Public Member Functions

- void [SpawnTheFox](#) ()  
*Call this to create a fox and interrupt all the bunnies.*

### 5.13.1 Detailed Description

This script shows how the behaviors of the bunnies are interrupted when the fox is spawned.

### 5.13.2 Member Function Documentation

#### 5.13.2.1 void NEEDSIMSampleSceneScripts.FuchsalarmDemoScript.SpawnTheFox ( )

Call this to create a fox and interrupt all the bunnies.

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

- FuchsalarmDemoScript.cs



## 5.14 Simulation.GeneralSettings Class Reference

A number of values used as configuration of the simulation and editor.

### Properties

- static Color [SlotColor](#) [get]  
*The standard color of a slot being drawn in the scene view.*
- static Color [AuctionableSlotColor](#) [get]  
*If a slot can be auctioned, that is offered to agents, this color is available.*
- static Color [BlockedSlotColor](#) [get]  
*If the slot is blocked it is drawn in this color.*
- static Color [ReservedSlotColor](#) [get]  
*The color of a currently reserved slot.*
- static Color [ReadyCharacterSlotColor](#) [get]  
*Whether a character is at the slot and ready to participate.*
- static float [SlotRepresentationRadius](#) [get]  
*The radius of the circle or sphere around a slot postion.*
- static float [SlotRepresentationHandleRadius](#) [get]  
*The radius/size of the handle of a slot.*
- static string [DefaultNeedName](#) [get]  
*Default value for newly created needs. The name of the need.*
- static float [DefaultNeedMinValue](#) [get]  
*Default value for newly created needs. Need satisfaction can not be below this value.*
- static float [DefaultNeedMaxValue](#) [get]  
*Default value for newly created needs. Need satisfaction can not be above this value.*
- static float [DefaultNeedCriticalState](#) [get]  
*Default value for newly created needs. Below this value the need is considered to be in a critical state.*
- static float [DefaultNeedSatisfiedState](#) [get]  
*Default value for newly created needs. Above this value the need is considered to be in a satisfied state.*
- static float [DefaultNeedChangePerSecondRate](#) [get]  
*Default value for newly created needs. At which rate needs become unsatisfied over time*

### 5.14.1 Detailed Description

A number of values used as configuration of the simulation and editor.

### 5.14.2 Property Documentation

#### 5.14.2.1 Color [Simulation.GeneralSettings.AuctionableSlotColor](#) [static], [get]

If a slot can be auctioned, that is offered to agents, this color is available.

#### 5.14.2.2 Color [Simulation.GeneralSettings.BlockedSlotColor](#) [static], [get]

If the slot is blocked it is drawn in this color.

#### 5.14.2.3 float [Simulation.GeneralSettings.DefaultNeedChangePerSecondRate](#) [static], [get]

Default value for newly created needs. At which rate needs become unsatisfied over time

5.14.2.4 float `Simulation.GeneralSettings.DefaultNeedCriticalState` [static], [get]

Default value for newly created needs. Below this value the need is considered to be in a critical state.

5.14.2.5 float `Simulation.GeneralSettings.DefaultNeedMaxValue` [static], [get]

Default value for newly created needs. Need satisfaction can not be above this value.

5.14.2.6 float `Simulation.GeneralSettings.DefaultNeedMinValue` [static], [get]

Default value for newly created needs. Need satisfaction can not be below this value.

5.14.2.7 string `Simulation.GeneralSettings.DefaultNeedName` [static], [get]

Default value for newly created needs. The name of the need.

5.14.2.8 float `Simulation.GeneralSettings.DefaultNeedSatisfiedState` [static], [get]

Default value for newly created needs. Above this value the need is considered to be in a satisfied state.

5.14.2.9 Color `Simulation.GeneralSettings.ReadyCharacterSlotColor` [static], [get]

Whether a character is at the slot and ready to participate.

5.14.2.10 Color `Simulation.GeneralSettings.ReservedSlotColor` [static], [get]

The color of a currently reserved slot.

5.14.2.11 Color `Simulation.GeneralSettings.SlotColor` [static], [get]

The standard color of a slot being drawn in the scene view.

5.14.2.12 float `Simulation.GeneralSettings.SlotRepresentationHandleRadius` [static], [get]

The radius/size of the handle of a slot.

5.14.2.13 float `Simulation.GeneralSettings.SlotRepresentationRadius` [static], [get]

The radius of the circle or sphere around a slot position.

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

- GeneralSettings.cs

## 5.15 Simulation.Goal Class Reference

A goal is a satisfaction level of a need that an agent wants to achieve.

## Public Member Functions

- [Goal](#) (string needToSatisfy, [Needs.NeedSatisfactions](#) satisfactionState)  
*Create a new goal to achieve a specific [Needs.NeedSatisfactions](#) state for a specific need.*
- bool [GoalAchieved](#) (float satisfactionValue)  
*True, if the specific NeedSatisfaction [Goal](#) is achieved. False otherwise.*

## Properties

- string [NeedToSatisfy](#) [get]  
*Which need should be satisfied by this goal.*
- [Needs.NeedSatisfactions](#) [SatisfactionState](#) [get]  
*What is the current [Needs.NeedSatisfactions](#) state of the need this goal is concerned with.*
- bool [HasBeenAchieved](#) [get, set]  
*Whether the goal has been achieved.*

### 5.15.1 Detailed Description

A goal is a satisfaction level of a need that an agent wants to achieve.

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 `Simulation.Goal.Goal ( string needToSatisfy, Needs.NeedSatisfactions satisfactionState )`

Create a new goal to achieve a specific [Needs.NeedSatisfactions](#) state for a specific need.

Parameters

<i>needToSatisfy</i>	The name of the need
<i>satisfactionState</i>	The state that is desired

### 5.15.3 Member Function Documentation

#### 5.15.3.1 `bool Simulation.Goal.GoalAchieved ( float satisfactionValue )`

True, if the specific NeedSatisfaction [Goal](#) is achieved. False otherwise.

Parameters

<i>satisfactionValue</i>	
--------------------------	--

Returns

### 5.15.4 Property Documentation

#### 5.15.4.1 `bool Simulation.Goal.HasBeenAchieved [get], [set]`

Whether the goal has been achieved.

#### 5.15.4.2 `string Simulation.Goal.NeedToSatisfy [get]`

Which need should be satisfied by this goal.

#### 5.15.4.3 Needs.NeedSatisfactions Simulation.Goal.SatisfactionState [get]

What is the current [Needs.NeedSatisfactions](#) state of the need this goal is concerned with.

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

- Goal.cs

## 5.16 NEEDSIMSampleSceneScripts.InputFieldRuntimeEditing Class Reference

Helper to change need satisfaction rates of interactions and satisfaction change rates of needs at runtime in a UI. It provides a method to react to the user finishing his input to the InputField.

Inherits MonoBehaviour.

### Public Member Functions

- void [EndInput](#) (string newValue)  
*Update the value if the user has ended his input.*

#### 5.16.1 Detailed Description

Helper to change need satisfaction rates of interactions and satisfaction change rates of needs at runtime in a UI. It provides a method to react to the user finishing his input to the InputField.

#### 5.16.2 Member Function Documentation

##### 5.16.2.1 void NEEDSIMSampleSceneScripts.InputFieldRuntimeEditing.EndInput ( string newValue )

Update the value if the user has ended his input.

Parameters

newValue	
----------	--

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

- InputFieldRuntimeEditing.cs

## 5.17 Simulation.Interaction Class Reference

An interaction at runtime

### Public Member Functions

- [Interaction](#) (string Name, Dictionary< string, float > [SatisfactionRates](#), float [Duration](#), bool hasPreconditions, Dictionary< string, bool > AllowedSpecies, Dictionary< [Needs.NeedSatisfactions](#), bool > AllowedSatisfactions)  
*Create a new interaction instance that will be available at simulation runtime.*
- bool [CheckPreconditions](#) ([AffordanceTreeNode](#) affordanceTreeNode)  
*For some interactions to be used conditions need to be fulfilled.*

## Properties

- string [Name](#) [get]  
*The unique name of the need. Identifier.*
- Dictionary< string, float > [SatisfactionRates](#) [get]  
*The rate at which each need is decayed or satisfied per second whilst the interaction is performed.*
- float [Duration](#) [get]  
*The duration in seconds of the interaction.*
- bool [Prolongable](#) [get]  
*Whether the interaction can be prolonged for another duration intervall*
- bool [HasPreconditions](#) [get]  
*Whether this interaction has preconditions. Setting this to false can speed up the simulation at the cost of not evaluating preconditions.*
- Dictionary< string, bool > [SpeciesAllowed](#) [get]  
*Which species are allowed to participate this interaction (precondition)*

### 5.17.1 Detailed Description

An interaction at runtime

### 5.17.2 Constructor & Destructor Documentation

5.17.2.1 `Simulation.Interaction.Interaction ( string Name, Dictionary< string, float > SatisfactionRates, float Duration, bool hasPreconditions, Dictionary< string, bool > AllowedSpecies, Dictionary< Needs.NeedSatisfactions, bool > AllowedSatisfactions )`

Create a new interaction instance that will be available at simulation runtime.

Parameters

<i>Name</i>	
<i>Satisfaction↔ Rates</i>	
<i>Duration</i>	
<i>has↔ Preconditions</i>	
<i>AllowedSpecies</i>	
<i>Allowed↔ Satisfactions</i>	

### 5.17.3 Member Function Documentation

5.17.3.1 `bool Simulation.Interaction.CheckPreconditions ( AffordanceTreeNode affordanceTreeNode )`

For some interactions to be used conditions need to be fulfilled.

Parameters

<i>affordance↔ TreeNode</i>	The <a href="#">AffordanceTreeNode</a> that wants to use the <a href="#">Interaction</a>
---------------------------------	--

Returns

Whether all preconditions are fulfilled.

## 5.17.4 Property Documentation

### 5.17.4.1 float Simulation.Interaction.Duration [get]

The duration in seconds of the interaction.

### 5.17.4.2 bool Simulation.Interaction.HasPreconditions [get]

Whether this interaction has preconditions. Setting this to false can speed up the simulation at the cost of not evaluating preconditions.

### 5.17.4.3 string Simulation.Interaction.Name [get]

The unique name of the need. Identifier.

### 5.17.4.4 bool Simulation.Interaction.Prolongable [get]

Whether the interaction can be prolonged for another duration intervall

### 5.17.4.5 Dictionary<string, float> Simulation.Interaction.SatisfactionRates [get]

The rate at which each need is decayed or satisfied per second whilst the interaction is performed.

### 5.17.4.6 Dictionary<string, bool> Simulation.Interaction.SpeciesAllowed [get]

Which species are allowed to participate this interaction (precondition)

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

- Interaction.cs

## 5.18 Simulation.InteractionData Class Reference

From these data items used in the editor the interactions for the runtime will be generated.

### Public Member Functions

- void [Init](#) (string name, List< [StringFloatPair](#) > satisfactionRates)  
*Create the data for a new interaction.*

### Public Attributes

- string [interactionName](#)  
*The unique name of the need. Identifier.*
- List< [StringFloatPair](#) > [satisfactions](#)  
*The rate at which each need is decayed or satisfied per second whilst the interaction is performed.*
- float [duration](#)  
*The duration in seconds of the interaction.*
- bool [doesHavePreconditions](#)

Whether this interaction has preconditions. Setting this to false can speed up the simulation at the cost of not evaluating preconditions.

- List< string > [SpeciesAllowed](#)

Which species are allowed to participate this interaction (precondition)

- List< string > [AtSatisfactionLevels](#)

NOT YET IMPLEMENTED

### 5.18.1 Detailed Description

From these data items used in the editor the interactions for the runtime will be generated.

### 5.18.2 Member Function Documentation

#### 5.18.2.1 void Simulation.InteractionData.Init ( string name, List< StringFloatPair > satisfactionRates )

Create the data for a new interaction.

Parameters

<i>name</i>	The unique name of the need. Identifier.
<i>satisfactionRates</i>	The rate at which the interaction satisfies or decays each need per second.

### 5.18.3 Member Data Documentation

#### 5.18.3.1 List<string> Simulation.InteractionData.AtSatisfactionLevels

NOT YET IMPLEMENTED

#### 5.18.3.2 bool Simulation.InteractionData.doesHavePreconditions

Whether this interaction has preconditions. Setting this to false can speed up the simulation at the cost of not evaluating preconditions.

#### 5.18.3.3 float Simulation.InteractionData.duration

The duration in seconds of the interaction.

#### 5.18.3.4 string Simulation.InteractionData.interactionName

The unique name of the need. Identifier.

#### 5.18.3.5 List<StringFloatPair> Simulation.InteractionData.satisfactions

The rate at which each need is decayed or satisfied per second whilst the interaction is performed.

#### 5.18.3.6 List<string> Simulation.InteractionData.SpeciesAllowed

Which species are allowed to participate this interaction (precondition)

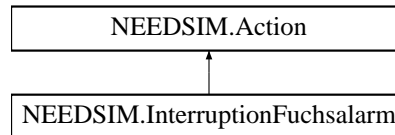
The documentation for this class was generated from the following file:

- InteractionData.cs

## 5.19 NEEDSIM.InterruptionFuchsalarm Class Reference

This action demonstrates how interruption of typical [NEEDSIM](#) behaviors could look like.

Inheritance diagram for NEEDSIM.InterruptionFuchsalarm:



### Public Member Functions

- override Action.Result [Run](#) ()  
*Satisfying a goal at an AffordanceTree node.*

#### 5.19.1 Detailed Description

This action demonstrates how interruption of typical [NEEDSIM](#) behaviors could look like.

#### 5.19.2 Member Function Documentation

##### 5.19.2.1 override Action.Result NEEDSIM.InterruptionFuchsalarm.Run ( ) [virtual]

Satisfying a goal at an AffordanceTree node.

##### Returns

Success if need satsifaction goal was achieved, running whilst it is being satisfied.

Implements [NEEDSIM.Action](#).

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

- InterruptionFuchsalarm.cs

## 5.20 NEEDSIMSampleSceneScripts.Lake Class Reference

This example suggests an idea for how a variety of animations can be played at an interactive object.

Inherits MonoBehaviour.

#### 5.20.1 Detailed Description

This example suggests an idea for how a variety of animations can be played at an interactive object.

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

- Lake.cs

## 5.21 Simulation.Manager Class Reference

A singleton for managing the simulation at runtime



## Public Member Functions

- Dictionary< string, float > [ZeroValuedNeeds](#) (List< string > needNames)  
*Get a new Dictionary with the respective needs values at 0.0f*
- bool [UpdateAffordanceTree](#) ()  
*Update this [Affordance](#) Tree Node, and its children (recursive). Necessary to call for the simulation to work*
- bool [SetAllNeedSatisfactionWeightsToOne](#) ()  
*This essentially turns of the WeightsForNeedSatisfaction, as if they are all one it means when they are multiplied the original values are not changed.*

## Properties

- static [Manager Instance](#) [get]  
*This is a singleton*
- [SimulationData Data](#) [get, set]  
*The data loaded into the simulation.*

### 5.21.1 Detailed Description

A singleton for managing the simulation at runtime

### 5.21.2 Member Function Documentation

#### 5.21.2.1 bool Simulation.Manager.SetAllNeedSatisfactionWeightsToOne ( )

This essentially turns of the WeightsForNeedSatisfaction, as if they are all one it means when they are multiplied the original values are not changed.

#### Returns

true, if there was no issue.

#### 5.21.2.2 bool Simulation.Manager.UpdateAffordanceTree ( )

Update this [Affordance](#) Tree Node, and its children (recursive). Necessary to call for the simulation to work

#### 5.21.2.3 Dictionary<string, float> Simulation.Manager.ZeroValuedNeeds ( List< string > needNames )

Get a new Dictionary with the respective needs values at 0.0f

#### Parameters

<i>needNames</i>	The needs you want to construct the dictionary for
------------------	--

#### Returns

a dictionary with the value 0.0f for each need.

### 5.21.3 Property Documentation

#### 5.21.3.1 SimulationData Simulation.Manager.Data [get], [set]

The data loaded into the simulation.

#### 5.21.3.2 Manager Simulation.Manager.Instance [static],[get]

This is a singleton

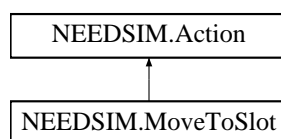
The documentation for this class was generated from the following file:

- Manager.cs

## 5.22 NEEDSIM.MoveToSlot Class Reference

Moving to a slot. The best implementation for such a behavior might be different in your project, but this script offers a starting point.

Inheritance diagram for NEEDSIM.MoveToSlot:



### Public Member Functions

- override Action.Result [Run](#) ()  
*Go to the slot that has been given to the agent.*

#### 5.22.1 Detailed Description

Moving to a slot. The best implementation for such a behavior might be different in your project, but this script offers a starting point.

#### 5.22.2 Member Function Documentation

##### 5.22.2.1 override Action.Result NEEDSIM.MoveToSlot.Run ( ) [virtual]

Go to the slot that has been given to the agent.

#### Returns

Running as long as on the way. Success upon arrival.

Implements [NEEDSIM.Action](#).

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

- MoveToSlot.cs

## 5.23 Simulation.NeedItem Class Reference

A need, one of the core building blocks of NEEDSIMLifeSimulation

## Public Member Functions

- [NeedItem](#) (string name, float changeRate, float critical, float satisfied)

*Create a new need.*

## Public Attributes

- string [needName](#)  
*The unique name of the need. Identifier.*
- float [minValue](#)  
*The limit below which the need satisfaction level is capped.*
- float [maxValue](#)  
*The limit above which the the need satisfaction level is capped.*
- float [changePerSecond](#)  
*How much the need changes per second. For example how hungry a character gets over time*
- float [criticalState](#)  
*The limit below which the state of the need satisfcation is considered critical.*
- float [satisfiedState](#)  
*The limit above which the state of the need satisfaction is considered satisfied.*

### 5.23.1 Detailed Description

A need, one of the core building blocks of NEEDSIMLifeSimulation

### 5.23.2 Constructor & Destructor Documentation

#### 5.23.2.1 Simulation.NeedItem.NeedItem ( string name, float changeRate, float critical, float satisfied )

Create a new need.

##### Parameters

<i>name</i>	The unique name of the need. Identifier.
<i>changeRate</i>	How much the need changes per second. For example how hungry a character gets over time
<i>critical</i>	The limit below which the state of the need satisfcation is considered critical.
<i>satisfied</i>	The limit above which the state of the need satisfaction is considered satisfied.

### 5.23.3 Member Data Documentation

#### 5.23.3.1 float Simulation.NeedItem.changePerSecond

How much the need changes per second. For example how hungry a character gets over time

#### 5.23.3.2 float Simulation.NeedItem.criticalState

The limit below which the state of the need satisfcation is considered critical.

#### 5.23.3.3 float Simulation.NeedItem.maxValue

The limit above which the the need satisfaction level is capped.

#### 5.23.3.4 float Simulation.NeedItem.minValue

The limit below which the need satisfaction level is capped.

#### 5.23.3.5 string Simulation.NeedItem.needName

The unique name of the need. Identifier.

#### 5.23.3.6 float Simulation.NeedItem.satisfiedState

The limit above which the state of the need satisfaction is considered satisfied.

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

- NeedItem.cs

## 5.24 Simulation.Needs Class Reference

The [Needs](#) class provides methods for interacting with the satisfaction levels of agents.

### Public Types

- enum [NeedSatisfactions](#) {  
[NeedSatisfactions.Unvalued](#) = 0, [NeedSatisfactions.Maximized](#), [NeedSatisfactions.Satisfied](#), [NeedSatisfactions.Uncritical](#),  
[NeedSatisfactions.Critical](#) }  
*Discrete states of satisfaction levels.*

### Public Member Functions

- [Needs](#) ([Species](#) species)  
*Constructor*
- void [SetSpecificSatisfactionLevels](#) (Dictionary< string, float > needLevels)  
*Set the level of each need satisfaction to a particular value*
- void [ApplyChangeRates](#) (Dictionary< string, float > input)  
*Apply a specific set of change rates, for example the change rates provided by an interaction.*
- void [ApplyChangePerSecond](#) ()  
*Apply the change per second rates, that is decay (or increase) each need with its default global change per second rate*
- [Goal](#) [GoalToSatisfyLowestNeed](#) ()  
*Compute a goal to satisfy the numerically lowest need.*
- void [RandomizeValues](#) ()  
*Set a random value for each need satisfaction*
- Dictionary< string, [Needs.NeedSatisfactions](#) > [Satisfactions](#) ()  
*The discrete satisfaction states for each need*
- float [GetValue](#) (string key)

### 5.24.1 Detailed Description

The [Needs](#) class provides methods for interacting with the satisfaction levels of agents.

## 5.24.2 Member Enumeration Documentation

### 5.24.2.1 enum Simulation.Needs.NeedSatisfactions [strong]

Discrete states of satisfaction levels.

Enumerator

**Unvalued** Unfortunately no discrete value of the need satisfaction levels is available.

**Maximized** The satisfaction level of the need is maximized.

**Satisfied** The need is satisfied, but not yet maximized.

**Uncritical** The need is below the satisfied limit, but not yet critical

**Critical** The need satisfaction level is below or equal to the target value for critical satisfaction.

## 5.24.3 Constructor & Destructor Documentation

### 5.24.3.1 Simulation.Needs.Needs ( Species *species* )

Constructor

Parameters

<i>species</i>	The species that the agent owning this set of needs belongs to
----------------	--

## 5.24.4 Member Function Documentation

### 5.24.4.1 void Simulation.Needs.ApplyChangePerSecond ( )

Apply the change per second rates, that is decay (or increase) each need with its default global change per second rate

### 5.24.4.2 void Simulation.Needs.ApplyChangeRates ( Dictionary< string, float > *input* )

Apply a specific set of change rates, for example the change rates provided by an interaction.

Parameters

<i>input</i>	
--------------	--

### 5.24.4.3 float Simulation.Needs.GetValue ( string *key* )

Parameters

<i>key</i>	The name of the need.
------------	-----------------------

Returns

The satisfaction level for the specified need. NaN if the key is invalid.

### 5.24.4.4 Goal Simulation.Needs.GoalToSatisfyLowestNeed ( )

Compute a goal to satisfy the numerically lowest need.

Returns

#### 5.24.4.5 void Simulation.Needs.RandomizeValues ( )

Set a random value for each need satisfaction

#### 5.24.4.6 Dictionary<string, Needs.NeedSatisfactions> Simulation.Needs.Satisfactions ( )

The discrete satisfaction states for each need

#### Returns

A dictionary which has the discrete states as values for each need.

#### 5.24.4.7 void Simulation.Needs.SetSpecficSatisfactionLevels ( Dictionary< string, float > needLevels )

Set the level of each need satisfaction to a particular value

#### Parameters

<i>needLevels</i>	The new satisfaction levels
-------------------	-----------------------------

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

- Needs.cs

## 5.25 NEEDSIM.NEEDSIMManager Class Reference

This class stores the values that the NEEDSIMROOT will use for running the simulation

Inherits MonoBehaviour.

### 5.25.1 Detailed Description

This class stores the values that the NEEDSIMROOT will use for running the simulation

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

- NEEDSIMManager.cs

## 5.26 NEEDSIM.NEEDSIMNode Class Reference

Every object and agent in [NEEDSIM](#) Life simulation has a [NEEDSIMNode](#): This is the essential component for using [NEEDSIM](#) Life simulation.

Inherits MonoBehaviour.

### Public Member Functions

- void [BuildTreeBasedOnSceneHierarchy](#) ()  
*Recursively build an Affordance Tree from the scene hierarchy. This method will not work for intermediate objects in the hierarchy - there is only deeper search if a direct ancestor is a [NEEDSIMNode](#).*
- bool [AcceptSlot](#) ()  
*If the simulation distributed a slot to this agent try to accept it.*
- bool [ArrivalAtSlot](#) (Slot slot)

*This method tries to call the AgentArrivalEasy() method at the slot it is passed to.*

- bool [TryConsumingAnimationOrder](#) (Animator animator)

*This method assumes that in the animator a trigger named "Movement" exists, and that for each interaction a trigger with the same name exists in the animator, e.g. that for the interaction "Eat" a trigger named "Eat" is in the animation and is used to transition into a state or sub-state-machine that plays animation(s) for eating.*

### 5.26.1 Detailed Description

Every object and agent in [NEEDSIM](#) Life simulation has a [NEEDSIMNode](#): This is the essential component for using [NEEDSIM](#) Life simulation.

### 5.26.2 Member Function Documentation

#### 5.26.2.1 bool NEEDSIM.NEEDSIMNode.AcceptSlot ( )

If the simulation distributed a slot to this agent try to accept it.

##### Returns

Whether the slot was accepted

#### 5.26.2.2 bool NEEDSIM.NEEDSIMNode.ArrivalAtSlot ( Slot slot )

This method tries to call the AgentArrivalEasy() method at the slot it is passed to.

##### Parameters

<i>slot</i>	The slot this agent arrives to
-------------	--------------------------------

##### Returns

Whether the arrival at a slot by this agent was successful

#### 5.26.2.3 void NEEDSIM.NEEDSIMNode.BuildTreeBasedOnSceneHierarchy ( )

Recursively build an Affordance Tree from the scene hierarchy. This method will not work for intermediate objects in the hierarchy - there is only deeper search if a direct ancestor is a [NEEDSIMNode](#).

#### 5.26.2.4 bool NEEDSIM.NEEDSIMNode.TryConsumingAnimationOrder ( Animator animator )

This method assumes that in the animator a trigger named "Movement" exists, and that for each interaction a trigger with the same name exists in the animator, e.g. that for the interaction "Eat" a trigger named "Eat" is in the animation and is used to transition into a state or sub-state-machine that plays animation(s) for eating.

##### Parameters

<i>animator</i>	
-----------------	--

##### Returns

Whether the most recent animation order was consumed.

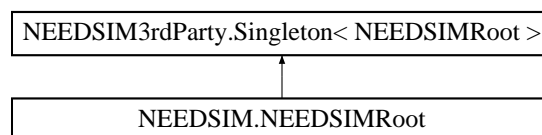
The documentation for this class was generated from the following file:

- NEEDSIMNode.cs

## 5.27 NEEDSIM.NEEDSIMRoot Class Reference

Every scene should have one root node for the AFFORDANCE TREE. This uses the settings of the [NEEDSIM](#) Manager and controls the simulation.

Inheritance diagram for NEEDSIM.NEEDSIMRoot:



### Public Member Functions

- void [BuildFlatAffordanceTreeFromScene](#) ()  
Goes through the scene and adds all [NEEDSIMNodes](#) as children of the root. This is best if you don't have complexity issues to deal with. Otherwise building the affordance tree with more depth is advised.
- void [BuildAffordanceTreeFromNode](#) ([NEEDSIMNode](#) node)  
To use this please pass the node you want to use as root node to this method. It is assumed that in the scene hierarchy this node is the root node to all other [NEEDSIMNodes](#). So if you have a village with houses and objects in each house you should, in the scene view, make the village parent of the houses, and the houses object of the village, and each game object should have a [NEEDSIMNode](#). You can leave the interactions set to none for the village and the houses if they are just used to calculate the abstraction for the Affordance Tree. The method [BuildTreeBasedOnSceneHierarchy\(\)](#) that is called here will not work for intermediate objects in the hierarchy - there is only deeper search if a direct ancestor is a [NEEDSIMNode](#).

### Protected Member Functions

- [NEEDSIMRoot](#) ()  
guarantee this will be always a singleton only - can't use the constructor!

#### 5.27.1 Detailed Description

Every scene should have one root node for the AFFORDANCE TREE. This uses the settings of the [NEEDSIM](#) Manager and controls the simulation.

#### 5.27.2 Constructor & Destructor Documentation

##### 5.27.2.1 NEEDSIM.NEEDSIMRoot.NEEDSIMRoot ( ) [protected]

guarantee this will be always a singleton only - can't use the constructor!

#### 5.27.3 Member Function Documentation

##### 5.27.3.1 void NEEDSIM.NEEDSIMRoot.BuildAffordanceTreeFromNode ( [NEEDSIMNode](#) node )

To use this please pass the node you want to use as root node to this method. It is assumed that in the scene hierarchy this node is the root node to all other [NEEDSIMNodes](#). So if you have a village with houses and objects in each house you should, in the scene view, make the village parent of the houses, and the houses object of the village, and each game object should have a [NEEDSIMNode](#). You can leave the interactions set to none for the village and the houses if they are just used to calculate the abstraction for the Affordance Tree. The method



BuildTreeBasedOnSceneHierarchy() that is called here will not work for intermediate objects in the hierarchy - there is only deeper search if a direct ancestor is a [NEEDSIMNode](#).

## Parameters

<i>node</i>	
-------------	--

## 5.27.3.2 void NEEDSIM.NEEDSIMRoot.BuildFlatAffordanceTreeFromScene ( )

Goes through the scene and adds all NEEDSIMNodes as children of the root. This is best if you don't have complexity issues to deal with. Otherwise building the affordance tree with more depth is advised.

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

- NEEDSIMRoot.cs

## 5.28 NEEDSIMSampleSceneScripts.NeedsUI Class Reference

This class shows bars for need satisfaction. A full bar equals full satisfaction, an empty bar means the need is not satisfied. If the need is currently being satisfied an outline will be added.

Inherits MonoBehaviour.

### 5.28.1 Detailed Description

This class shows bars for need satisfaction. A full bar equals full satisfaction, an empty bar means the need is not satisfied. If the need is currently being satisfied an outline will be added.

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

- NeedsUI.cs

## 5.29 NEEDSIM.PlanDemo Class Reference

A simple behavior control solution. We tried to write this in a way that makes it easy to use our code samples in Finite State Machines, Behavior Trees and Goal-oriented [Action](#) Planning. The idea is that you can run our simulation from within a different solution, for example in case you want to have agents with fighting capabilities.

### Public Member Functions

- void [Update](#) ()

*Update runs the plan.*

### 5.29.1 Detailed Description

A simple behavior control solution. We tried to write this in a way that makes it easy to use our code samples in Finite State Machines, Behavior Trees and Goal-oriented [Action](#) Planning. The idea is that you can run our simulation from within a different solution, for example in case you want to have agents with fighting capabilities.

### 5.29.2 Member Function Documentation

#### 5.29.2.1 void NEEDSIM.PlanDemo.Update ( )

Update runs the plan.

If the currently running action returns, upon evaluation, Result.Running, we keep on running that action. If Result.Failure is returned we start a new sequence. If Result.Success is returned we go to the next step in the current sequence, or, if at the last step, start a new sequence.

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

- PlanDemo.cs

## 5.30 NEEDSIMSampleSceneScripts.SampleCameraControl Class Reference

A very simple scrolling camera for [NEEDSIM](#) Life simulation example scenes.

Inherits MonoBehaviour.

### 5.30.1 Detailed Description

A very simple scrolling camera for [NEEDSIM](#) Life simulation example scenes.

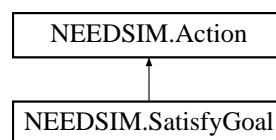
The documentation for this class was generated from the following file:

- SampleCameraControl.cs

## 5.31 NEEDSIM.SatisfyGoal Class Reference

Participate in a slot to satisfy a goal.

Inheritance diagram for NEEDSIM.SatisfyGoal:



### Public Member Functions

- override Action.Result [Run](#) ()  
*Satisfying a goal at an AffordanceTree node.*

### 5.31.1 Detailed Description

Participate in a slot to satisfy a goal.

### 5.31.2 Member Function Documentation

#### 5.31.2.1 override Action.Result NEEDSIM.SatisfyGoal.Run ( ) [virtual]

Satisfying a goal at an AffordanceTree node.

#### Returns

Success if need satisfaction goal was achieved, running whilst it is being satisfied.

Implements [NEEDSIM.Action](#).

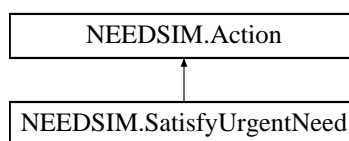
The documentation for this class was generated from the following file:

- SatisfyGoal.cs

## 5.32 NEEDSIM.SatisfyUrgentNeed Class Reference

Participating a slot. The respective behavior for value/urgency oriented behaviors.

Inheritance diagram for NEEDSIM.SatisfyUrgentNeed:



### Public Member Functions

- override Action.Result [Run](#) ()

*Participating a slot. The respective behavior for value/urgency oriented behaviors.*

### 5.32.1 Detailed Description

Participating a slot. The respective behavior for value/urgency oriented behaviors.

### 5.32.2 Member Function Documentation

#### 5.32.2.1 override Action.Result NEEDSIM.SatisfyUrgentNeed.Run ( ) [virtual]

Participating a slot. The respective behavior for value/urgency oriented behaviors.

#### Returns

TODO

Implements [NEEDSIM.Action](#).

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

- SatisfyUrgentNeed.cs

## 5.33 NEEDSIMSampleSceneScripts.SceneSwitcher Class Reference

public methods to switch scenes via a button click. You have to add the scenes to your build settings to use the prefab that uses this script.

Inherits MonoBehaviour.

### 5.33.1 Detailed Description

public methods to switch scenes via a button click. You have to add the scenes to your build settings to use the prefab that uses this script.

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

- SceneSwitcher.cs

## 5.34 NEEDSIMSampleSceneScripts.SimpleSpawn Class Reference

This is a spawning example script to maintain populations

Inherits MonoBehaviour.

### Public Member Functions

- void [fillPopulation](#) ()  
*Spawn an instance of the prefab for each spawning point at which the previously (if any) spawned instance is dead (null)*
- void [killAll](#) ()  
*Remove all agents from the slots they currently participate in and delete them.*

### 5.34.1 Detailed Description

This is a spawning example script to maintain populations

### 5.34.2 Member Function Documentation

#### 5.34.2.1 void NEEDSIMSampleSceneScripts.SimpleSpawn.fillPopulation ( )

Spawn an instance of the prefab for each spawning point at which the previously (if any) spawned instance is dead (null)

#### 5.34.2.2 void NEEDSIMSampleSceneScripts.SimpleSpawn.killAll ( )

Remove all agents from the slots they currently participate in and delete them.

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

- SimpleSpawn.cs

## 5.35 Simulation.SimulationData Class Reference

The data loaded in the simulation at runtime

### Public Member Functions

- [SimulationData](#) (Dictionary< string, float > minValuePairDictionary, Dictionary< string, float > maxValuePairDictionary, Dictionary< string, float > changePerSecondDictionary, Dictionary< string, float > criticalStateDictionary, Dictionary< string, float > satisfiedStateDictionary, List< string > needNames, List< string >

speciesNames, List< [Species](#) > species, List< [Interaction](#) > interactionList, Dictionary< string, [Species](#) > speciesByName)

*Constructs new simulation data.*

### Public Attributes

- readonly Dictionary< string, float > [CriticalStateDictionary](#)  
*For each need, what is the value below or equal to which it is considered critical.*
- readonly Dictionary< string, float > [SatisfiedStateDictionary](#)  
*For each need, what is the value above which it is considered satisfied.*
- readonly List< string > [NeedNames](#)  
*The names of all the needs.*
- readonly Dictionary< string, [Interaction](#) > [InteractionByNameDictionary](#)  
*An instance of each interaction by its name*

### Properties

- Dictionary< string, float > [ChangePerSecond](#) [get]  
*How much each need changes over time, for example at which rate a character becomes more hungry.*
- [AffordanceTreeNode Root](#) [get, set]  
*The root of the AFFORDANCE TREE. A lot of management of the simulation can be done from this point.*
- Dictionary< string, float > [WeightsForNeed](#) [get, set]  
*A general and global weight that affects the utility agent see in each need. Best to use values between 0.0 and 1.0.*
- Dictionary< [Needs.NeedSatisfactions](#), float > [WeightsForNeedSatisfaction](#) [get, set]  
*How each of the [Needs.NeedSatisfactions](#) states is valued - use this for example to prioritize critical needs. Best to use values between 0.0 and 1.0*

#### 5.35.1 Detailed Description

The data loaded in the simulation at runtime

#### 5.35.2 Constructor & Destructor Documentation

- 5.35.2.1 Simulation.SimulationData.SimulationData ( Dictionary< string, float > *minValueDictionary*, Dictionary< string, float > *maxValueDictionary*, Dictionary< string, float > *changePerSecondDictionary*, Dictionary< string, float > *criticalStateDictionary*, Dictionary< string, float > *satisfiedStateDictionary*, List< string > *needNames*, List< string > *speciesNames*, List< [Species](#) > *species*, List< [Interaction](#) > *interactionList*, Dictionary< string, [Species](#) > *speciesByName* )

Constructs new simulation data.

##### Parameters

<i>minValueDictionary</i>	The lowest value a need can have at runtime.
<i>maxValueDictionary</i>	The highest value a need can have at runtime.
<i>changePerSecondDictionary</i>	For each need, how much it changes over time, for example a decay of the need 'Hunger' would mean characters get hungry over time

<i>criticalState</i> ↔ <i>Dictionary</i>	The values below or equal to which a need is considered to be in critical state.
<i>satisfiedState</i> ↔ <i>Dictionary</i>	The value above which a need is considered to be in satisfied state.
<i>needNames</i>	The names of all needs
<i>speciesNames</i>	The names of all species
<i>species</i>	All species
<i>interactionList</i>	All interactions
<i>speciesByName</i>	

### 5.35.3 Member Data Documentation

#### 5.35.3.1 readonly Dictionary<string, float> Simulation.SimulationData.CriticalStateDictionary

For each need, what is the value below or equal to which it is considered critical.

#### 5.35.3.2 readonly Dictionary<string, Interaction> Simulation.SimulationData.InteractionByNameDictionary

An instance of each interaction by its name

#### 5.35.3.3 readonly List<string> Simulation.SimulationData.NeedNames

The names of all the needs.

#### 5.35.3.4 readonly Dictionary<string, float> Simulation.SimulationData.SatisfiedStateDictionary

For each need, what is the value above which it is considered satisfied.

### 5.35.4 Property Documentation

#### 5.35.4.1 Dictionary<string, float> Simulation.SimulationData.ChangePerSecond [get]

How much each need changes over time, for example at which rate a character becomes more hungry.

#### 5.35.4.2 AffordanceTreeNode Simulation.SimulationData.Root [get], [set]

The root of the AFFORDANCE TREE. A lot of management of the simulation can be done from this point.

#### 5.35.4.3 Dictionary<string, float> Simulation.SimulationData.WeightsForNeed [get], [set]

A general and global weight that affects the utility agent see in each need. Best to use values between 0.0 and 1.0.

#### 5.35.4.4 Dictionary<Needs.NeedSatisfactions, float> Simulation.SimulationData.WeightsForNeedSatisfaction [get], [set]

How each of the [Needs.NeedSatisfactions](#) states is valued - use this for example to prioritize critical needs. Best to use values between 0.0 and 1.0

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

- SimulationData.cs

## 5.36 NEEDSIM3rdParty.Singleton< T > Class Template Reference

Be aware this will not prevent a non singleton constructor such as `T myT = new T();` To prevent that, add `protected T () {}` to your singleton class.

Inherits MonoBehaviour.

### Public Member Functions

- void [OnDestroy](#) ()

*When Unity quits, it destroys objects in a random order. In principle, a [Singleton](#) is only destroyed when application quits. If any script calls Instance after it have been destroyed, it will create a buggy ghost object that will stay on the Editor scene even after stopping playing the Application. Really bad! So, this was made to be sure we're not creating that buggy ghost object.*

### 5.36.1 Detailed Description

Be aware this will not prevent a non singleton constructor such as `T myT = new T();` To prevent that, add `protected T () {}` to your singleton class.

As a note, this is made as MonoBehaviour because we need Coroutines.

#### Type Constraints

***T : MonoBehaviour***

### 5.36.2 Member Function Documentation

#### 5.36.2.1 void NEEDSIM3rdParty.Singleton< T >.OnDestroy ( )

When Unity quits, it destroys objects in a random order. In principle, a [Singleton](#) is only destroyed when application quits. If any script calls Instance after it have been destroyed, it will create a buggy ghost object that will stay on the Editor scene even after stopping playing the Application. Really bad! So, this was made to be sure we're not creating that buggy ghost object.

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

- Singleton.cs

## 5.37 Simulation.Slot Class Reference

A slot is a position in the world where an agent can run the interactions provided by the object that offers the slot.

### Public Types

- enum [SlotStates](#) {  
[SlotStates.Blocked](#), [SlotStates.Reserved](#), [SlotStates.ReadyCharacter](#), [SlotStates.ReadyForAuction](#),  
[SlotStates.CurrentlyOnAuction](#) }

*A slot is in one of these states at any time.*

- enum [Result](#) {  
[Result.Success](#), [Result.NoAffordance](#), [Result.NoInteraction](#), [Result.UnclearFailure](#),  
[Result.InteractionAlreadyRunning](#), [Result.NoProlongableInteraction](#) }

*The result of an agent trying to use this slot*



### Public Member Functions

- [Slot](#) ([Affordance](#) affordance)  
*Create a new slot*
- bool [InterruptInteraction](#) ()  
*Interrupt the interaction currently running at this slot*
- float [CurrentInteractionDuration](#) ()  
*Get the remaining duration of the currently running interaction.*
- bool [ReserveSlot](#) ()  
*Set the state of the slot to reserved.*
- [Result](#) [AgentArrivalEasy](#) ([AffordanceTreeNode](#) participant)  
*This tries to start a random interaction if no interaction is running and sets slot state to ReadyCharacter.*
- bool [AgentDeparture](#) ()  
*Frees the slot for being offered to other agents.*
- bool [OfferSlot](#) ()  
*Sets the slot state to currently being on auction*
- bool [SetSlotBlocked](#) ()  
*Set the slot state to being blocked.*

### Properties

- [SlotStates](#) [SlotState](#) [get]  
*The current state of the slot.*
- Vector3 [Position](#) [get, set]  
*World space position of the slot.*
- Vector3 [LocalPosition](#) [get, set]  
*Please set the position relative to the slot's parent transform here*
- Vector3 [LookAt](#) [get, set]  
*In world coordinates, where the agent should orient him/herself to*
- Vector3 [LocalLookAt](#) [get, set]  
*The local position of he look at, where the agent should orient him/herself to, relative to the parents transform.*
- bool [IsAuctionable](#) [get, set]  
*Whether the auction system is allowed to offer this slot to agents.*
- [Interaction](#) [currentInteraction](#) [get]  
*Which interaction is currently running at this slot.*

#### 5.37.1 Detailed Description

A slot is a position in the world where an agent can run the interactions provided by the object that offers the slot.

#### 5.37.2 Member Enumeration Documentation

##### 5.37.2.1 enum [Simulation.Slot.Result](#) [strong]

The result of an agent trying to use this slot

#### Enumerator

- Success** No problem was determined as is slot used.
- NoAffordance** There as a problem with initialization and the affordance this slot belongs to is null.
- NoInteraction** There is no interaction available at the affordace this slot belongs to.

**UnclearFailure** There is some issue that lacks proper description

**InteractionAlreadyRunning** There is already an interaction running. This is not necessarily a problem, and can in many circumstances be considered a success.

**NoProlongableInteraction** When trying to prolong an interaction there was no such option.

## 5.37.2.2 enum `Simulation.Slot.SlotStates` [`strong`]

A slot is in one of these states at any time.

### Enumerator

**Blocked** A slot that is blocked for whatever reason can not be used by agents.

**Reserved** An agent reserved this slot for her/himself, and it is assumed that the agent is on the way to slot.

**ReadyCharacter** An agent is ready to interact with the slot, or interacting with the slot.

**ReadyForAuction** This slot can be auctioned, but is not currently auctioned

**CurrentlyOnAuction** Currently this slot is available for auctions.

## 5.37.3 Constructor & Destructor Documentation

### 5.37.3.1 `Simulation.Slot.Slot ( Affordance affordance )`

Create a new slot

#### Parameters

<i>affordance</i>	The affordance this slot will belong to.
-------------------	--

## 5.37.4 Member Function Documentation

### 5.37.4.1 `Result Simulation.Slot.AgentArrivalEasy ( AffordanceTreeNode participant )`

This tries to start a random interaction if no interaction is running and sets slot state to ReadyCharacter.

#### Parameters

<i>participant</i>	The agent that will be participating in the interactions of this slot.
--------------------	--

#### Returns

The result of trying to start an interaction

### 5.37.4.2 `bool Simulation.Slot.AgentDeparture ( )`

Frees the slot for being offered to other agents.

#### Returns

### 5.37.4.3 float Simulation.Slot.CurrentInteractionDuration ( )

Get the remaining duration of the currently running interaction.

Returns

### 5.37.4.4 bool Simulation.Slot.InterruptInteraction ( )

Interrupt the interaction currently running at this slot

Returns

### 5.37.4.5 bool Simulation.Slot.OfferSlot ( )

Sets the slot state to currently being on auction

Returns

### 5.37.4.6 bool Simulation.Slot.ReserveSlot ( )

Set the state of the slot to reserved.

Returns

Will fail if the slot is blocked.

### 5.37.4.7 bool Simulation.Slot.SetSlotBlocked ( )

Set the slot state to being blocked.

Returns

## 5.37.5 Property Documentation

### 5.37.5.1 Interaction Simulation.Slot.currentInteraction [get]

Which interaction is currently running at this slot.

### 5.37.5.2 bool Simulation.Slot.IsAuctionable [get], [set]

Whether the auction system is allowed to offer this slot to agents.

### 5.37.5.3 Vector3 Simulation.Slot.LocalLookAt [get], [set]

The local position of he look at, where the agent should orient him/herself to, relative to the parents transform.

#### 5.37.5.4 Vector3 Simulation.Slot.LocalPosition [get], [set]

Please set the position relative to the slot's parent transform here

#### 5.37.5.5 Vector3 Simulation.Slot.LookAt [get], [set]

In world coordinates, where the agent should orient him/herself to

#### 5.37.5.6 Vector3 Simulation.Slot.Position [get], [set]

World space position of the slot.

#### 5.37.5.7 SlotStates Simulation.Slot.SlotState [get]

The current state of the slot.

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

- Slot.cs

## 5.38 NEEDSIMSampleSceneScripts.SpawnBedsManager Class Reference

This example shows how you could spawn all the objects and agents procedurally.

Inherits MonoBehaviour.

### Public Member Functions

- void [SpawnBed](#) ()  
*Create an instance of a bed, add it to the simulation, and translate it to the right position.*
- void [DestroyBed](#) ()  
*Destroy an instance of a bed.*

#### 5.38.1 Detailed Description

This example shows how you could spawn all the objects and agents procedurally.

#### 5.38.2 Member Function Documentation

##### 5.38.2.1 void NEEDSIMSampleSceneScripts.SpawnBedsManager.DestroyBed ( )

Destroy an instance of a bed.

##### 5.38.2.2 void NEEDSIMSampleSceneScripts.SpawnBedsManager.SpawnBed ( )

Create an instance of a bed, add it to the simulation, and translate it to the right position.

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

- SpawnBedsManager.cs

## 5.39 NEEDSIMSampleSceneScripts.SpawnUIRuntimeEditing Class Reference

Spawns a UI Element for each need and each satisfaction rate of an interaction.

Inherits MonoBehaviour.

### 5.39.1 Detailed Description

Spawns a UI Element for each need and each satisfaction rate of an interaction.

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

- SpawnUIRuntimeEditing.cs

## 5.40 Simulation.Species Class Reference

A species is a set of needs. For example zombies might only have the 'Hunger' need, whereas humans furthermore have a 'Social' need.

### Public Member Functions

- [Species](#) (string name, List< string > [needs](#))  
*Construct a new species*

### Public Attributes

- string [speciesName](#)  
*The unique name of the need. Identifier.*
- List< string > [needs](#)  
*The set of needs that defines this species.*

### 5.40.1 Detailed Description

A species is a set of needs. For example zombies might only have the 'Hunger' need, whereas humans furthermore have a 'Social' need.

### 5.40.2 Constructor & Destructor Documentation

#### 5.40.2.1 Simulation.Species.Species ( string name, List< string > needs )

Construct a new species

#### Parameters

<i>name</i>	The unique name of the need. Identifier.
<i>needs</i>	The set of needs that defines this species

### 5.40.3 Member Data Documentation

#### 5.40.3.1 List<string> Simulation.Species.needs

The set of needs that defines this species.

#### 5.40.3.2 string Simulation.Species.speciesName

The unique name of the need. Identifier.

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

- Species.cs

## 5.41 Simulation.StringFloatPair Class Reference

A class that helps creating key value pairs.

### Public Member Functions

- [StringFloatPair](#) (string key, float value)  
*Create a new helper to later on create key value pairs*

### Public Attributes

- string [stringValue](#)  
*The key/name/identifier*
- float [floatValue](#)  
*The value*

#### 5.41.1 Detailed Description

A class that helps creating key value pairs.

#### 5.41.2 Constructor & Destructor Documentation

##### 5.41.2.1 Simulation.StringFloatPair.StringFloatPair ( string key, float value )

Create a new helper to later on create key value pairs

Parameters

<i>key</i>	This will be the key
<i>value</i>	This will be the value

#### 5.41.3 Member Data Documentation

##### 5.41.3.1 float Simulation.StringFloatPair.floatValue

The value

##### 5.41.3.2 string Simulation.StringFloatPair.stringValue

The key/name/identifier

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

- StringFloatPair.cs

### 5.42 Simulation.Strings Class Reference

A centralized place for many of the strings used by the [NEEDSIM](#) Life simulation.

#### Static Public Member Functions

- static string [NewHasBeenSet](#) (string name)
- static string [DefaultInteractionRelativeNeedName](#) (string needName)
- static string [InteractionLabelToExtend](#) (int extension)
- static string [SlotNumberLabel](#) (int i)

#### Public Attributes

- const string [EditorExtensionPosition](#) = "Window/" + [ProductName](#) + " - " + [ProductDescription](#)
- const string [ProductName](#) = "NEEDSIM"
- const string [ProductDescription](#) = "Life Simulation"

#### Static Public Attributes

- static GUIContent [LabelNeedName](#)
- static GUIContent [LabelNeedChangeRate](#)
- static GUIContent [LabelCriticalState](#)
- static GUIContent [LabelSatisfiedState](#)
- static GUIContent [InteractionDurationLabel](#)
- static GUIContent [NeedAffectedLabel](#)
- static GUIContent [NumberOfNeedsAffectedLabel](#)
- static GUIContent [SatisfactionRateLabel](#)

#### Properties

- static string [SimulationManagerConstructed](#) [get]
- static string [FolderName](#) [get]
- static string [EditorWindowTitle](#) [get]
- static string [NoDefaultDataFound](#) [get]
- static string [MoreThanOneDefaultData](#) [get]
- static string [Advice](#) [get]
- static string [NoDatabaseFound](#) [get]
- static string [BuildAffordanceTreeFromSceneLabel](#) [get]
- static string [DialogTitleExitPlay](#) [get]
- static string [DialogMessageExitPlay](#) [get]
- static string [DialogButtonExitPlay](#) [get]
- static string [Welcome](#) [get]
- static string [SupportMail](#) [get]
- static string [WebsiteURL](#) [get]
- static string [WebsiteURLButtonText](#) [get]
- static string [VisitUsAt](#) [get]
- static string [Contact](#) [get]
- static string [AdvancedInstructionLine1](#) [get]
- static string [AdvancedInstructionLine2](#) [get]
- static string [AdvancedInstructionLine3](#) [get]
- static string [AdvancedInstructionLine4](#) [get]
- static string [AdvancedInstructionLine5](#) [get]

- static string [MarkCurrentDatabaseAsDefault](#) [get]
- static string [UnableToLoadDatabase](#) [get]
- static string [GenericDatabaseName](#) [get]
- static string [defaultString](#) [get]
- static string [currentDatabaseLoadedLabel](#) [get]
- static string [DefaultDatabaseName](#) [get]
- static string [AssetsPath](#) [get]
- static string [NeedsDataBasePath](#) [get]
- static string [DataFolder](#) [get]
- static string [UserDataResources](#) [get]
- static string [Asterisk](#) [get]
- static string [AssetAppendix](#) [get]
- static string [SceneAppendix](#) [get]
- static string [DefaultNeedName](#) [get]
- static string [DefaultSpecies](#) [get]
- static string [SpeciesNameLabel](#) [get]
- static string [NeedsInSpeciesLabel](#) [get]
- static string [AddNeedButtonLabel](#) [get]
- static string [AddSpeciesButtonLabel](#) [get]
- static string [AddInteractionButtonLabel](#) [get]
- static string [RemoveNeedButtonLabel](#) [get]
- static string [DefaultDatabaseHeadline](#) [get]
- static string [SpeciesViewHeadline](#) [get]
- static string [AdvancedDatabaseViewHeadline](#) [get]
- static string [DataBasesAvailableLabel](#) [get]
- static string [ButtonAddDatabaseLabel](#) [get]
- static string [ButtonRemoveDatabaseLabel](#) [get]
- static string[] [WindowSelectionNames](#) [get]
- static string [InteractionViewHeadline](#) [get]
- static string [None](#) [get]
- static string [DefaultInteractionName](#) [get]
- static string [AssignNeedLabel](#) [get]
- static string [InteractionNameLabel](#) [get]
- static string [PreconditionsHeadline](#) [get]
- static string [PreconditionsSatisfactionLevelLabel](#) [get]
- static string[] [PreconditionsSatisfactionLevelArray](#) [get]
- static string [PreconditionsSpecies](#) [get]
- static string [PreconditionDefaultAny](#) [get]
- static string [PreconditionsUnsatisfiedLabel](#) [get]
- static string [IsAgent](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [DrawGizmosInGame](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [O\\_Space](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [TerritoryCenter](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [SpeciesName](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [ShowDebugInGame](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*

- static string [ShowDebugInInspector](#) [get]

*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*



- static string [InteractionData](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [SlotPositionsArray](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [AuctionableBoolArray](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [InteractionDataArrayAccess](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [SlotPositionsArraySize](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [SlotPositionsArrayAccess](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [IsAuctionableArraySize](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [IsAuctionableArrayAccess](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [IsAgentLabel](#) [get]
- static string [ModifyLookAt](#) [get]  
*This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs*
- static string [RandomStartLevelsLabel](#) [get]
- static string [ShowDebugGizmosLabel](#) [get]
- static string [TerritoryControlHeadline](#) [get]
- static string [NumberOfSlotsLabel](#) [get]
- static string [SlotPositionLabel](#) [get]
- static string [IsAuctionableLabel](#) [get]
- static string [AssignSpeciesOptionToAgentLabel](#) [get]
- static string [DebugIngameLabel](#) [get]
- static string [DebugInspectorLabel](#) [get]
- static string [DeleteDatabaseTitle](#) [get]
- static string [DeleteDatabaseMessage](#) [get]
- static string [DialogYes](#) [get]
- static string [DialogCancel](#) [get]
- static string [YourDatabaseName](#) [get]
- static string [GeneralSettings](#) [get]
- static string [AdvancedSettings](#) [get]
- static string [LogSimulationLabel](#) [get]
- static string [PrintSimDebugLogLabel](#) [get]
- static string [AttachSpecificDBLabel](#) [get]

### 5.42.1 Detailed Description

A centralized place for many of the strings used by the [NEEDSIM](#) Life simulation.

### 5.42.2 Member Function Documentation

- 5.42.2.1 static string [Simulation.Strings.DefaultInteractionRelativeNeedName](#) ( string *needName* ) [static]

## Parameters

<i>needName</i>	
-----------------	--

## Returns

5.42.2.2 static string Simulation.Strings.InteractionLabelToExtend ( int *extension* ) [static]

## Parameters

<i>extension</i>	
------------------	--

## Returns

5.42.2.3 static string Simulation.Strings.NewHasBeenSet ( string *name* ) [static]

## Parameters

<i>name</i>	
-------------	--

## Returns

5.42.2.4 static string Simulation.Strings.SlotNumberLabel ( int *i* ) [static]

## Parameters

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

## Returns

### 5.42.3 Member Data Documentation

5.42.3.1 const string Simulation.Strings.EditorExtensionPosition = "Window/" + **ProductName** + " - " + **ProductDescription**

5.42.3.2 GUIContent Simulation.Strings.InteractionDurationLabel [static]

## Initial value:

```
= new GUIContent(
    "Duration",
    "The duration of the interaction in seconds. Often a good idea is to use the duration of
    the respective animations.")
```

#### 5.42.3.3 GUIContent Simulation.Strings.LabelCriticalState [static]

**Initial value:**

```
= new GUIContent(
    "Critical below:",
    "A need does not only have a numeric value, but we define some numeric ranges to be more
    abstract states of how satisfied a need is - Critical, Uncritical, Unsatisfied, Satisfied and Maximized. This can
    be used in behavior authored on top of the core NEEDSIM Life simulation."
)
```

#### 5.42.3.4 GUIContent Simulation.Strings.LabelNeedChangeRate [static]

**Initial value:**

```
= new GUIContent(
    "+/- per second",
    "How satisfied a need is changes over time - for example we used a negative rate to model how
    characters get hungry in a simulation where we assume a value of 100 for 'Hunger' means the agent has a full
    belly, and zero means he is starving."
)
```

#### 5.42.3.5 GUIContent Simulation.Strings.LabelNeedName [static]

**Initial value:**

```
= new GUIContent(
    "Need name",
    "Please chose a unique name for this need. Creating a model of needs and interactions to satisfy
    them is the core idea of NEEDSIM Life simulation.")
```

#### 5.42.3.6 GUIContent Simulation.Strings.LabelSatisfiedState [static]

**Initial value:**

```
= new GUIContent(
    "Satisfied above:",
    "A need does not only have a numeric value, but we define some numeric ranges to be more
    abstract states of how satisfied a need is - Critical, Uncritical, Unsatisfied, Satisfied and Maximized. This can
    be used in behavior authored on top of the core NEEDSIM Life simulation."
)
```

#### 5.42.3.7 GUIContent Simulation.Strings.NeedAffectedLabel [static]

**Initial value:**

```
= new GUIContent(
    "Need affected:",
    "The interaction changes the satisfaction level of the need selected from the drop down menu."
)
```

#### 5.42.3.8 GUIContent Simulation.Strings.NumberOfNeedsAffectedLabel [static]

**Initial value:**

```
= new GUIContent(
    "No. of Needs affected:",
    "One ore more needs can be affected by the same interaction. For example the interaction
    sunbathing might reduce the need to relax, but could increase thirst."
)
```

5.42.3.9 `const string Simulation.Strings.ProductDescription = "Life Simulation"`

5.42.3.10 `const string Simulation.Strings.ProductName = "NEEDSIM"`

5.42.3.11 `GUIContent Simulation.Strings.SatisfactionRateLabel [static]`

**Initial value:**

```
= new GUIContent(
    "Satisfaction +/-",
    "The rate at which the specified need is decayed or satisfied per second. Standard ranges for
    needs are from 0 to 100, so values should be in between -100 to 100."
)
```

## 5.42.4 Property Documentation

5.42.4.1 `string Simulation.Strings.AddInteractionButtonLabel [static], [get]`

5.42.4.2 `string Simulation.Strings.AddNeedButtonLabel [static], [get]`

5.42.4.3 `string Simulation.Strings.AddSpeciesButtonLabel [static], [get]`

5.42.4.4 `string Simulation.Strings.AdvancedDatabaseViewHeadline [static], [get]`

5.42.4.5 `string Simulation.Strings.AdvancedInstructionLine1 [static], [get]`

5.42.4.6 `string Simulation.Strings.AdvancedInstructionLine2 [static], [get]`

5.42.4.7 `string Simulation.Strings.AdvancedInstructionLine3 [static], [get]`

5.42.4.8 `string Simulation.Strings.AdvancedInstructionLine4 [static], [get]`

5.42.4.9 `string Simulation.Strings.AdvancedInstructionLine5 [static], [get]`

5.42.4.10 `string Simulation.Strings.AdvancedSettings [static], [get]`

5.42.4.11 `string Simulation.Strings.Advice [static], [get]`

5.42.4.12 `string Simulation.Strings.AssetAppendix [static], [get]`

5.42.4.13 `string Simulation.Strings.AssetsPath [static], [get]`

5.42.4.14 `string Simulation.Strings.AssignNeedLabel [static], [get]`

5.42.4.15 `string Simulation.Strings.AssignSpeciesOptionToAgentLabel [static], [get]`

5.42.4.16 `string Simulation.Strings.Asterisk [static], [get]`

5.42.4.17 `string Simulation.Strings.AttachSpecificDBLabel [static], [get]`

5.42.4.18 `string Simulation.Strings.AuctionableBoolArray [static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.19 `string Simulation.Strings.BuildAffordanceTreeFromSceneLabel [static], [get]`

- 5.42.4.20 string Simulation.Strings.ButtonAddDatabaseLabel [static], [get]
- 5.42.4.21 string Simulation.Strings.ButtonRemoveDatabaseLabel [static], [get]
- 5.42.4.22 string Simulation.Strings.Contact [static], [get]
- 5.42.4.23 string Simulation.Strings.currentDatabaseLoadedLabel [static], [get]
- 5.42.4.24 string Simulation.Strings.DataBasesAvailableLabel [static], [get]
- 5.42.4.25 string Simulation.Strings.DataFolder [static], [get]
- 5.42.4.26 string Simulation.Strings.DebugIngameLabel [static], [get]
- 5.42.4.27 string Simulation.Strings.DebugInSpectorLabel [static], [get]
- 5.42.4.28 string Simulation.Strings.DefaultDatabaseHeadline [static], [get]
- 5.42.4.29 string Simulation.Strings.DefaultDatabaseName [static], [get]
- 5.42.4.30 string Simulation.Strings.DefaultInteractionName [static], [get]
- 5.42.4.31 string Simulation.Strings.DefaultNeedName [static], [get]
- 5.42.4.32 string Simulation.Strings.DefaultSpecies [static], [get]
- 5.42.4.33 string Simulation.Strings.defaultString [static], [get]
- 5.42.4.34 string Simulation.Strings.DeleteDatabaseMessage [static], [get]
- 5.42.4.35 string Simulation.Strings.DeleteDatabaseTitle [static], [get]
- 5.42.4.36 string Simulation.Strings.DialogButtonExitPlay [static], [get]
- 5.42.4.37 string Simulation.Strings.DialogCancel [static], [get]
- 5.42.4.38 string Simulation.Strings.DialogMessageExitPlay [static], [get]
- 5.42.4.39 string Simulation.Strings.DialogTitleExitPlay [static], [get]
- 5.42.4.40 string Simulation.Strings.DialogYes [static], [get]
- 5.42.4.41 string Simulation.Strings.DrawGizmosInGame [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

- 5.42.4.42 string Simulation.Strings.EditorWindowTitle [static], [get]
- 5.42.4.43 string Simulation.Strings.FolderName [static], [get]
- 5.42.4.44 string Simulation.Strings.GeneralSettings [static], [get]
- 5.42.4.45 string Simulation.Strings.GenericDatabaseName [static], [get]

5.42.4.46 `string Simulation.Strings.InteractionData` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.47 `string Simulation.Strings.InteractionDataArrayAccess` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.48 `string Simulation.Strings.InteractionNameLabel` `[static], [get]`

5.42.4.49 `string Simulation.Strings.InteractionViewHeadline` `[static], [get]`

5.42.4.50 `string Simulation.Strings.IsAgent` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.51 `string Simulation.Strings.IsAgentLabel` `[static], [get]`

5.42.4.52 `string Simulation.Strings.IsAuctionableArrayAccess` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.53 `string Simulation.Strings.IsAuctionableArraySize` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.54 `string Simulation.Strings.IsAuctionableLabel` `[static], [get]`

5.42.4.55 `string Simulation.Strings.LogSimulationLabel` `[static], [get]`

5.42.4.56 `string Simulation.Strings.MarkCurrentDatabaseAsDefault` `[static], [get]`

5.42.4.57 `string Simulation.Strings.ModifyLookAt` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.58 `string Simulation.Strings.MoreThanOneDefaultData` `[static], [get]`

5.42.4.59 `string Simulation.Strings.NeedsDataBasePath` `[static], [get]`

5.42.4.60 `string Simulation.Strings.NeedsInSpeciesLabel` `[static], [get]`

5.42.4.61 `string Simulation.Strings.NoDatabaseFound` `[static], [get]`

5.42.4.62 `string Simulation.Strings.NoDefaultDataFound` `[static], [get]`

5.42.4.63 `string Simulation.Strings.None` `[static], [get]`

5.42.4.64 `string Simulation.Strings.NumberOfSlotsLabel` `[static], [get]`

5.42.4.65 `string Simulation.Strings.O_Space` `[static], [get]`

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.66 string Simulation.Strings.PreconditionDefaultAny [static], [get]

5.42.4.67 string Simulation.Strings.PreconditionsHeadline [static], [get]

5.42.4.68 string Simulation.Strings.PreconditionsSatisfactionLevelLabel [static], [get]

5.42.4.69 string Simulation.Strings.PreconditionsSpecies [static], [get]

5.42.4.70 string [] Simulation.Strings.PreconditionsSatisfactionLevelArray [static], [get]

5.42.4.71 string Simulation.Strings.PreconditionsUnsatisfiedLabel [static], [get]

5.42.4.72 string Simulation.Strings.PrintSimDebugLogLabel [static], [get]

5.42.4.73 string Simulation.Strings.RandomStartLevelsLabel [static], [get]

5.42.4.74 string Simulation.Strings.RemoveNeedButtonLabel [static], [get]

5.42.4.75 string Simulation.Strings.SceneAppendix [static], [get]

5.42.4.76 string Simulation.Strings.ShowDebugGizmosLabel [static], [get]

5.42.4.77 string Simulation.Strings.ShowDebugInGame [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.78 string Simulation.Strings.ShowDebugInInspector [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.79 string Simulation.Strings.SimulationManagerConstructed [static], [get]

5.42.4.80 string Simulation.Strings.SlotPositionLabel [static], [get]

5.42.4.81 string Simulation.Strings.SlotPositionsArray [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.82 string Simulation.Strings.SlotPositionsArrayAccess [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.83 string Simulation.Strings.SlotPositionsArraySize [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

5.42.4.84 string Simulation.Strings.SpeciesName [static], [get]

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

```
5.42.4.85  string Simulation.Strings.SpeciesNameLabel  [static],[get]
5.42.4.86  string Simulation.Strings.SpeciesViewHeadline  [static],[get]
5.42.4.87  string Simulation.Strings.SupportMail  [static],[get]
5.42.4.88  string Simulation.Strings.TerritoryCenter  [static],[get]
```

This has to be changed if changes to the respective field name are made in NEEDSIMNode.cs

```
5.42.4.89  string Simulation.Strings.TerritoryControlHeadline  [static],[get]
5.42.4.90  string Simulation.Strings.UnableToLoadDatabase  [static],[get]
5.42.4.91  string Simulation.Strings.UserDataResources  [static],[get]
5.42.4.92  string Simulation.Strings.VisitUsAt  [static],[get]
5.42.4.93  string Simulation.Strings.WebsiteURL  [static],[get]
5.42.4.94  string Simulation.Strings.WebsiteURLButtonText  [static],[get]
5.42.4.95  string Simulation.Strings.Welcome  [static],[get]
5.42.4.96  string [] Simulation.Strings.WindowSelectionNames  [static],[get]
5.42.4.97  string Simulation.Strings.YourDatabaseName  [static],[get]
```

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

- Strings.cs

## 5.43 NEEDSIMSampleSceneScripts.TimeSystem Class Reference

This example uses arrays with 24 values each to modify how behaviors are evaluated at a specific time of day. This class works with Value Oriented behaviors, not with Goal Oriented behaviors, because it relies on the fact that all opportunities to satisfy needs are evaluated, not only the opportunities that can satisfy the need of the current goal.

Inherits MonoBehaviour.

### 5.43.1 Detailed Description

This example uses arrays with 24 values each to modify how behaviors are evaluated at a specific time of day. This class works with Value Oriented behaviors, not with Goal Oriented behaviors, because it relies on the fact that all opportunities to satisfy needs are evaluated, not only the opportunities that can satisfy the need of the current goal.

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

- TimeSystem.cs



## Index

- AcceptAuctionVictory
  - Simulation::AffordanceTreeNode, [16](#)
- AcceptSlot
  - NEEDSIM::Blackboard, [20](#)
  - NEEDSIM::NEEDSIMNode, [41](#)
- AddInteraction
  - Simulation::Affordance, [14](#)
- AddInteractionButtonLabel
  - Simulation::Strings, [62](#)
- AddNeedButtonLabel
  - Simulation::Strings, [62](#)
- AddSlot
  - Simulation::AffordanceTreeNode, [16](#), [17](#)
- AddSpeciesButtonLabel
  - Simulation::Strings, [62](#)
- AdvancedDatabaseViewHeadline
  - Simulation::Strings, [62](#)
- AdvancedInstructionLine1
  - Simulation::Strings, [62](#)
- AdvancedInstructionLine2
  - Simulation::Strings, [62](#)
- AdvancedInstructionLine3
  - Simulation::Strings, [62](#)
- AdvancedInstructionLine4
  - Simulation::Strings, [62](#)
- AdvancedInstructionLine5
  - Simulation::Strings, [62](#)
- AdvancedSettings
  - Simulation::Strings, [62](#)
- Advice
  - Simulation::Strings, [62](#)
- Affordance
  - Simulation::AffordanceTreeNode, [18](#)
- AffordanceTreeNode
  - Simulation::AffordanceTreeNode, [16](#)
- AgentArrivalEasy
  - Simulation::Slot, [52](#)
- AgentDeparture
  - Simulation::Slot, [52](#)
- ApplyChangePerSecond
  - Simulation::Needs, [39](#)
- ApplyChangeRates
  - Simulation::Needs, [39](#)
- ApplyParentInteraction
  - Simulation::AffordanceTreeNode, [17](#)
- ArrivalAtSlot
  - NEEDSIM::NEEDSIMNode, [41](#)
- AssetAppendix
  - Simulation::Strings, [62](#)
- AssetsPath
  - Simulation::Strings, [62](#)
- AssignNeedLabel
  - Simulation::Strings, [62](#)
- AssignSpeciesOptionToAgentLabel
  - Simulation::Strings, [62](#)
- Asterisk
  - Simulation::Strings, [62](#)
- AtSatisfactionLevels
  - Simulation::InteractionData, [33](#)
- AttachSpecificDBLabel
  - Simulation::Strings, [62](#)
- AuctionableBoolArray
  - Simulation::Strings, [62](#)
- AuctionableSlotColor
  - Simulation::GeneralSettings, [27](#)
- AvailableSlot
  - Simulation::AffordanceTreeNode, [17](#)
- Blocked
  - Simulation::Slot, [52](#)
- BlockedSlotColor
  - Simulation::GeneralSettings, [27](#)
- BuildAffordanceTreeFromNode
  - NEEDSIM::NEEDSIMRoot, [42](#)
- BuildAffordanceTreeFromSceneLabel
  - Simulation::Strings, [62](#)
- BuildFlatAffordanceTreeFromScene
  - NEEDSIM::NEEDSIMRoot, [44](#)
- BuildTreeBasedOnSceneHierarchy
  - NEEDSIM::NEEDSIMNode, [41](#)
- ButtonAddDatabaseLabel
  - Simulation::Strings, [62](#)
- ButtonRemoveDatabaseLabel
  - Simulation::Strings, [63](#)
- ChangePerSecond
  - Simulation::SimulationData, [49](#)
- changePerSecond
  - Simulation::NeedItem, [37](#)
- CheckPreconditions
  - Simulation::Interaction, [31](#)
- Contact
  - Simulation::Strings, [63](#)
- CountOfChildren
  - Simulation::AffordanceTreeNode, [18](#)
- Critical
  - Simulation::Needs, [39](#)
- criticalState
  - Simulation::NeedItem, [37](#)

CriticalStateDictionary  
     Simulation::SimulationData, [49](#)  
 currentDatabaseLoadedLabel  
     Simulation::Strings, [63](#)  
 CurrentInteraction  
     Simulation::Affordance, [15](#)  
 currentInteraction  
     Simulation::Slot, [53](#)  
 CurrentInteractionDuration  
     Simulation::Slot, [52](#)  
 CurrentInteractionSatisfiesMostUrgentNeed  
     Simulation::AffordanceTreeNode, [18](#)  
 CurrentlyOnAuction  
     Simulation::Slot, [52](#)  
  
 Data  
     Simulation::Manager, [35](#)  
 DatabasesAvailableLabel  
     Simulation::Strings, [63](#)  
 DataFolder  
     Simulation::Strings, [63](#)  
 DatabaseName  
     Simulation::DatabaseAsset, [23](#)  
 DebugInSpectorLabel  
     Simulation::Strings, [63](#)  
 DebugIngameLabel  
     Simulation::Strings, [63](#)  
 DefaultDatabaseHeadline  
     Simulation::Strings, [63](#)  
 DefaultDatabaseName  
     Simulation::Strings, [63](#)  
 DefaultInteractionName  
     Simulation::Strings, [63](#)  
 DefaultInteractionRelativeNeedName  
     Simulation::Strings, [59](#)  
 DefaultNeedChangePerSecondRate  
     Simulation::GeneralSettings, [27](#)  
 DefaultNeedCriticalState  
     Simulation::GeneralSettings, [27](#)  
 DefaultNeedMaxValue  
     Simulation::GeneralSettings, [28](#)  
 DefaultNeedMinValue  
     Simulation::GeneralSettings, [28](#)  
 DefaultNeedName  
     Simulation::GeneralSettings, [28](#)  
     Simulation::Strings, [63](#)  
 DefaultNeedSatisfiedState  
     Simulation::GeneralSettings, [28](#)  
 DefaultSpecies  
     Simulation::Strings, [63](#)  
 defaultString  
     Simulation::Strings, [63](#)  
 DeleteDatabaseMessage  
     Simulation::Strings, [63](#)  
 DeleteDatabaseTitle  
     Simulation::Strings, [63](#)  
 DestroyBed  
     NEEDSIMSampleSceneScripts::SpawnBeds↔  
         Manager, [54](#)  
  
 DialogButtonExitPlay  
     Simulation::Strings, [63](#)  
 DialogCancel  
     Simulation::Strings, [63](#)  
 DialogMessageExitPlay  
     Simulation::Strings, [63](#)  
 DialogTitleExitPlay  
     Simulation::Strings, [63](#)  
 DialogYes  
     Simulation::Strings, [63](#)  
 doesHavePreconditions  
     Simulation::InteractionData, [33](#)  
 DrawGizmosInGame  
     Simulation::Strings, [63](#)  
 Duration  
     Simulation::Interaction, [32](#)  
 duration  
     Simulation::InteractionData, [33](#)  
  
 EditorExtensionPosition  
     Simulation::Strings, [60](#)  
 EditorWindowTitle  
     Simulation::Strings, [63](#)  
 EndInput  
     NEEDSIMSampleSceneScripts::InputField↔  
         RuntimeEditing, [30](#)  
  
 fillPopulation  
     NEEDSIMSampleSceneScripts::SimpleSpawn, [47](#)  
 floatValue  
     Simulation::StringFloatPair, [56](#)  
 FolderName  
     Simulation::Strings, [63](#)  
  
 GeneralSettings  
     Simulation::Strings, [63](#)  
 GenericDatabaseName  
     Simulation::Strings, [63](#)  
 GetInteractionNames  
     Simulation::DatabaseAsset, [23](#)  
 GetNeedNames  
     Simulation::DatabaseAsset, [23](#)  
 GetSpeciesNames  
     Simulation::DatabaseAsset, [23](#)  
 GetValue  
     Simulation::Needs, [39](#)  
 Goal  
     Simulation::AffordanceTreeNode, [18](#)  
     Simulation::Goal, [29](#)  
 GoalAchieved  
     Simulation::Goal, [29](#)  
 GoalToSatisfyLowestNeed  
     Simulation::Needs, [39](#)  
  
 HasArrivedAtSlot  
     NEEDSIM::Blackboard, [20](#)  
 HasBeenAchieved  
     Simulation::Goal, [29](#)  
 HasInteraction

- Simulation::Affordance, [15](#)
- HasPreconditions
  - Simulation::Interaction, [32](#)
- Init
  - Simulation::DatabaseAsset, [23](#)
  - Simulation::InteractionData, [33](#)
- Instance
  - Simulation::Manager, [35](#)
- Interaction
  - Simulation::Interaction, [31](#)
- InteractionAlreadyRunning
  - Simulation::Slot, [52](#)
- InteractionByNameDictionary
  - Simulation::SimulationData, [49](#)
- InteractionData
  - Simulation::Strings, [63](#)
- InteractionDataArrayAccess
  - Simulation::Strings, [64](#)
- InteractionDurationLabel
  - Simulation::Strings, [60](#)
- InteractionLabelToExtend
  - Simulation::Strings, [60](#)
- interactionName
  - Simulation::InteractionData, [33](#)
- InteractionNameLabel
  - Simulation::Strings, [64](#)
- InteractionStartedThisFrame
  - Simulation::Affordance, [15](#)
- InteractionViewHeadline
  - Simulation::Strings, [64](#)
- Interactions
  - Simulation::DatabaseAsset, [23](#)
- InterruptInteraction
  - Simulation::Slot, [53](#)
- IsAgent
  - Simulation::Strings, [64](#)
- IsAgentLabel
  - Simulation::Strings, [64](#)
- IsAuctionable
  - Simulation::Slot, [53](#)
- IsAuctionableArrayAccess
  - Simulation::Strings, [64](#)
- IsAuctionableArraySize
  - Simulation::Strings, [64](#)
- IsAuctionableLabel
  - Simulation::Strings, [64](#)
- isDefault
  - Simulation::DatabaseAsset, [23](#)
- killAll
  - NEEDSIMSampleSceneScripts::SimpleSpawn, [47](#)
- LabelCriticalState
  - Simulation::Strings, [60](#)
- LabelNeedChangeRate
  - Simulation::Strings, [61](#)
- LabelNeedName
  - Simulation::Strings, [61](#)
- LabelSatisfiedState
  - Simulation::Strings, [61](#)
- LocalLookAt
  - Simulation::Slot, [53](#)
- LocalPosition
  - Simulation::Slot, [53](#)
- LogSimulationLabel
  - Simulation::Strings, [64](#)
- LookAt
  - Simulation::Slot, [54](#)
- MarkCurrentDatabaseAsDefault
  - Simulation::Strings, [64](#)
- maxValue
  - Simulation::NeedItem, [37](#)
- Maximized
  - Simulation::Needs, [39](#)
- minValue
  - Simulation::NeedItem, [37](#)
- ModifyLookAt
  - Simulation::Strings, [64](#)
- MoreThanOneDefaultData
  - Simulation::Strings, [64](#)
- MostUrgentNeed
  - Simulation::AffordanceTreeNode, [19](#)
- NEEDSIM, [9](#)
- NEEDSIM.Action, [13](#)
- NEEDSIM.Blackboard, [20](#)
- NEEDSIM.ChaseSlot, [21](#)
- NEEDSIM.DecideGoal, [24](#)
- NEEDSIM.DecideValue, [25](#)
- NEEDSIM.InterruptionFuchsalarm, [34](#)
- NEEDSIM.MoveToSlot, [36](#)
- NEEDSIM.NEEDSIMManager, [40](#)
- NEEDSIM.NEEDSIMNode, [40](#)
- NEEDSIM.NEEDSIMRoot, [42](#)
- NEEDSIM.PlanDemo, [44](#)
- NEEDSIM.SatisfyGoal, [45](#)
- NEEDSIM.SatisfyUrgentNeed, [46](#)
- NEEDSIM3rdParty, [10](#)
- NEEDSIM3rdParty.Singleton< T >, [50](#)
- NEEDSIM3rdParty::Singleton
  - OnDestroy, [50](#)
- NEEDSIM::Blackboard
  - AcceptSlot, [20](#)
  - HasArrivedAtSlot, [20](#)
  - slotToAgentDistanceSmall, [20](#)
- NEEDSIM::ChaseSlot
  - Run, [22](#)
- NEEDSIM::DecideGoal
  - Run, [24](#)
- NEEDSIM::DecideValue
  - Run, [25](#)
- NEEDSIM::InterruptionFuchsalarm
  - Run, [34](#)
- NEEDSIM::MoveToSlot
  - Run, [36](#)
- NEEDSIM::NEEDSIMNode

- AcceptSlot, [41](#)
- ArrivalAtSlot, [41](#)
- BuildTreeBasedOnSceneHierarchy, [41](#)
- TryConsumingAnimationOrder, [41](#)
- NEEDSIM::NEEDSIMRoot
  - BuildAffordanceTreeFromNode, [42](#)
  - BuildFlatAffordanceTreeFromScene, [44](#)
  - NEEDSIMRoot, [42](#)
- NEEDSIM::PlanDemo
  - Update, [44](#)
- NEEDSIM::SatisfyGoal
  - Run, [45](#)
- NEEDSIM::SatisfyUrgentNeed
  - Run, [46](#)
- NEEDSIMEditor, [10](#)
- NEEDSIMRoot
  - NEEDSIM::NEEDSIMRoot, [42](#)
- NEEDSIMSampleSceneScripts, [10](#)
- NEEDSIMSampleSceneScripts.Animal, [19](#)
- NEEDSIMSampleSceneScripts.Bunny, [21](#)
- NEEDSIMSampleSceneScripts.Deer, [25](#)
- NEEDSIMSampleSceneScripts.Fox, [26](#)
- NEEDSIMSampleSceneScripts.FuchsalarmDemo↔
  - Script, [26](#)
- NEEDSIMSampleSceneScripts.InputFieldRuntime↔
  - Editing, [30](#)
- NEEDSIMSampleSceneScripts.Lake, [34](#)
- NEEDSIMSampleSceneScripts.NeedsUI, [44](#)
- NEEDSIMSampleSceneScripts.SampleCameraControl, [45](#)
- NEEDSIMSampleSceneScripts.SceneSwitcher, [46](#)
- NEEDSIMSampleSceneScripts.SimpleSpawn, [47](#)
- NEEDSIMSampleSceneScripts.SpawnBedsManager, [54](#)
- NEEDSIMSampleSceneScripts.SpawnUIRuntime↔
  - Editing, [55](#)
- NEEDSIMSampleSceneScripts.TimeSystem, [66](#)
- NEEDSIMSampleSceneScripts::FuchsalarmDemo↔
  - Script
    - SpawnTheFox, [26](#)
- NEEDSIMSampleSceneScripts::InputFieldRuntime↔
  - Editing
    - EndInput, [30](#)
- NEEDSIMSampleSceneScripts::SimpleSpawn
  - fillPopulation, [47](#)
  - killAll, [47](#)
- NEEDSIMSampleSceneScripts::SpawnBedsManager
  - DestroyBed, [54](#)
  - SpawnBed, [54](#)
- Name
  - Simulation::Interaction, [32](#)
- NeedAffectedLabel
  - Simulation::Strings, [61](#)
- NeedItem
  - Simulation::NeedItem, [37](#)
- needName
  - Simulation::NeedItem, [38](#)
- NeedNames
  - Simulation::SimulationData, [49](#)
- NeedSatisfactions
  - Simulation::Needs, [39](#)
- NeedToSatisfy
  - Simulation::Goal, [29](#)
- Needs
  - Simulation::Needs, [39](#)
- needs
  - Simulation::Species, [55](#)
- NeedsDataBasePath
  - Simulation::Strings, [64](#)
- NeedsInSpeciesLabel
  - Simulation::Strings, [64](#)
- NeedsList
  - Simulation::DatabaseAsset, [24](#)
- NewHasBeenSet
  - Simulation::Strings, [60](#)
- NoAffordance
  - Simulation::Slot, [51](#)
- NoDatabaseFound
  - Simulation::Strings, [64](#)
- NoDefaultDataFound
  - Simulation::Strings, [64](#)
- NoInteraction
  - Simulation::Slot, [51](#)
- NoProlongableInteraction
  - Simulation::Slot, [52](#)
- None
  - Simulation::Strings, [64](#)
- NumberOfNeedsAffectedLabel
  - Simulation::Strings, [61](#)
- NumberOfSlotsLabel
  - Simulation::Strings, [64](#)
- O\_Space
  - Simulation::Strings, [64](#)
- OfferSlot
  - Simulation::Slot, [53](#)
- OnDestroy
  - NEEDSIM3rdParty::Singleton, [50](#)
- Parent
  - Simulation::AffordanceTreeNode, [19](#)
- Position
  - Simulation::Slot, [54](#)
- PreconditionDefaultAny
  - Simulation::Strings, [64](#)
- PreconditionsHeadline
  - Simulation::Strings, [65](#)
- PreconditionsSatisfactionLevelLabel
  - Simulation::Strings, [65](#)
- PreconditionsSpecies
  - Simulation::Strings, [65](#)
- PreconditionsSatisfactionLevelArray
  - Simulation::Strings, [65](#)
- PreconditionsUnsatisfiedLabel
  - Simulation::Strings, [65](#)
- PrintSimDebugLogLabel
  - Simulation::Strings, [65](#)

- Printer
  - Simulation::AffordanceTreeNode, [18](#)
- ProductDescription
  - Simulation::Strings, [61](#)
- ProductName
  - Simulation::Strings, [62](#)
- ProlongLastInteraction
  - Simulation::Affordance, [14](#)
- Prolongable
  - Simulation::Interaction, [32](#)
- RandomStartLevelsLabel
  - Simulation::Strings, [65](#)
- RandomizeValues
  - Simulation::Needs, [39](#)
- ReadyCharacter
  - Simulation::Slot, [52](#)
- ReadyCharacterSlotColor
  - Simulation::GeneralSettings, [28](#)
- ReadyForAuction
  - Simulation::Slot, [52](#)
- Remove
  - Simulation::AffordanceTreeNode, [18](#)
- RemoveNeedButtonLabel
  - Simulation::Strings, [65](#)
- ReserveSlot
  - Simulation::Slot, [53](#)
- Reserved
  - Simulation::Slot, [52](#)
- ReservedSlotColor
  - Simulation::GeneralSettings, [28](#)
- Result
  - Simulation::Slot, [51](#)
- Root
  - Simulation::AffordanceTreeNode, [19](#)
  - Simulation::SimulationData, [49](#)
- Run
  - NEEDSIM::ChaseSlot, [22](#)
  - NEEDSIM::DecideGoal, [24](#)
  - NEEDSIM::DecideValue, [25](#)
  - NEEDSIM::InterruptionFuchsalarm, [34](#)
  - NEEDSIM::MoveToSlot, [36](#)
  - NEEDSIM::SatisfyGoal, [45](#)
  - NEEDSIM::SatisfyUrgentNeed, [46](#)
- SatisfactionLevels
  - Simulation::AffordanceTreeNode, [19](#)
- SatisfactionRateLabel
  - Simulation::Strings, [62](#)
- SatisfactionRates
  - Simulation::Interaction, [32](#)
- SatisfactionState
  - Simulation::Goal, [29](#)
- Satisfactions
  - Simulation::Needs, [40](#)
- satisfactions
  - Simulation::InteractionData, [33](#)
- Satisfied
  - Simulation::Needs, [39](#)
- satisfiedState
  - Simulation::NeedItem, [38](#)
- SatisfiedStateDictionary
  - Simulation::SimulationData, [49](#)
- SceneAppendix
  - Simulation::Strings, [65](#)
- SetAllNeedSatisfactionWeightsToOne
  - Simulation::Manager, [35](#)
- SetSlotBlocked
  - Simulation::Slot, [53](#)
- SetSpecficSatisfactionLevels
  - Simulation::Needs, [40](#)
- setSpecies
  - Simulation::AffordanceTreeNode, [18](#)
- ShowDebugGizmosLabel
  - Simulation::Strings, [65](#)
- ShowDebugInGame
  - Simulation::Strings, [65](#)
- ShowDebugInInspector
  - Simulation::Strings, [65](#)
- Simulation, [11](#)
- Simulation.Affordance, [13](#)
- Simulation.AffordanceTreeNode, [15](#)
- Simulation.DatabaseAsset, [22](#)
- Simulation.GeneralSettings, [27](#)
- Simulation.Goal, [28](#)
- Simulation.Interaction, [30](#)
- Simulation.InteractionData, [32](#)
- Simulation.Manager, [34](#)
- Simulation.NeedItem, [36](#)
- Simulation.Needs, [38](#)
- Simulation.SimulationData, [47](#)
- Simulation.Slot, [50](#)
- Simulation.Species, [55](#)
- Simulation.StringFloatPair, [56](#)
- Simulation.Strings, [57](#)
- Simulation.Tests, [11](#)
- Simulation::Affordance
  - AddInteraction, [14](#)
  - CurrentInteraction, [15](#)
  - HasInteraction, [15](#)
  - InteractionStartedThisFrame, [15](#)
  - ProlongLastInteraction, [14](#)
  - StartInteraction, [14](#)
  - StartRandomInteraction, [14](#)
- Simulation::AffordanceTreeNode
  - AcceptAuctionVictory, [16](#)
  - AddSlot, [16](#), [17](#)
  - Affordance, [18](#)
  - AffordanceTreeNode, [16](#)
  - ApplyParentInteraction, [17](#)
  - AvailableSlot, [17](#)
  - CountOfChildren, [18](#)
  - CurrentInteractionSatisfiesMostUrgentNeed, [18](#)
  - Goal, [18](#)
  - MostUrgentNeed, [19](#)
  - Parent, [19](#)
  - Printer, [18](#)



- Remove, [18](#)
- Root, [19](#)
- SatisfactionLevels, [19](#)
- setSpecies, [18](#)
- Slots, [19](#)
- Species, [19](#)
- Simulation::DatabaseAsset
  - DatabaseName, [23](#)
  - GetInteractionNames, [23](#)
  - GetNeedNames, [23](#)
  - GetSpeciesNames, [23](#)
  - Init, [23](#)
  - Interactions, [23](#)
  - isDefault, [23](#)
  - NeedsList, [24](#)
  - Species, [24](#)
- Simulation::GeneralSettings
  - AuctionableSlotColor, [27](#)
  - BlockedSlotColor, [27](#)
  - DefaultNeedChangePerSecondRate, [27](#)
  - DefaultNeedCriticalState, [27](#)
  - DefaultNeedMaxValue, [28](#)
  - DefaultNeedMinValue, [28](#)
  - DefaultNeedName, [28](#)
  - DefaultNeedSatisfiedState, [28](#)
  - ReadyCharacterSlotColor, [28](#)
  - ReservedSlotColor, [28](#)
  - SlotColor, [28](#)
  - SlotRepresentationHandleRadius, [28](#)
  - SlotRepresentationRadius, [28](#)
- Simulation::Goal
  - Goal, [29](#)
  - GoalAchieved, [29](#)
  - HasBeenAchieved, [29](#)
  - NeedToSatisfy, [29](#)
  - SatisfactionState, [29](#)
- Simulation::Interaction
  - CheckPreconditions, [31](#)
  - Duration, [32](#)
  - HasPreconditions, [32](#)
  - Interaction, [31](#)
  - Name, [32](#)
  - Prolongable, [32](#)
  - SatisfactionRates, [32](#)
  - SpeciesAllowed, [32](#)
- Simulation::InteractionData
  - AtSatisfactionLevels, [33](#)
  - doesHavePreconditions, [33](#)
  - duration, [33](#)
  - Init, [33](#)
  - interactionName, [33](#)
  - satisfactions, [33](#)
  - SpeciesAllowed, [33](#)
- Simulation::Manager
  - Data, [35](#)
  - Instance, [35](#)
  - SetAllNeedSatisfactionWeightsToOne, [35](#)
  - UpdateAffordanceTree, [35](#)
  - ZeroValuedNeeds, [35](#)
- Simulation::NeedItem
  - changePerSecond, [37](#)
  - criticalState, [37](#)
  - maxValue, [37](#)
  - minValue, [37](#)
  - NeedItem, [37](#)
  - needName, [38](#)
  - satisfiedState, [38](#)
- Simulation::Needs
  - ApplyChangePerSecond, [39](#)
  - ApplyChangeRates, [39](#)
  - Critical, [39](#)
  - GetValue, [39](#)
  - GoalToSatisfyLowestNeed, [39](#)
  - Maximized, [39](#)
  - NeedSatisfactions, [39](#)
  - Needs, [39](#)
  - RandomizeValues, [39](#)
  - Satisfactions, [40](#)
  - Satisfied, [39](#)
  - SetSpecificSatisfactionLevels, [40](#)
  - Uncritical, [39](#)
  - Unvalued, [39](#)
- Simulation::SimulationData
  - ChangePerSecond, [49](#)
  - CriticalStateDictionary, [49](#)
  - InteractionByNameDictionary, [49](#)
  - NeedNames, [49](#)
  - Root, [49](#)
  - SatisfiedStateDictionary, [49](#)
  - SimulationData, [48](#)
  - WeightsForNeed, [49](#)
  - WeightsForNeedSatisfaction, [49](#)
- Simulation::Slot
  - AgentArrivalEasy, [52](#)
  - AgentDeparture, [52](#)
  - Blocked, [52](#)
  - currentInteraction, [53](#)
  - CurrentInteractionDuration, [52](#)
  - CurrentlyOnAuction, [52](#)
  - InteractionAlreadyRunning, [52](#)
  - InterruptInteraction, [53](#)
  - IsAuctionable, [53](#)
  - LocalLookAt, [53](#)
  - LocalPosition, [53](#)
  - LookAt, [54](#)
  - NoAffordance, [51](#)
  - NoInteraction, [51](#)
  - NoProlongableInteraction, [52](#)
  - OfferSlot, [53](#)
  - Position, [54](#)
  - ReadyCharacter, [52](#)
  - ReadyForAuction, [52](#)
  - ReserveSlot, [53](#)
  - Reserved, [52](#)
  - Result, [51](#)
  - SetSlotBlocked, [53](#)

- Slot, [52](#)
- SlotState, [54](#)
- SlotStates, [52](#)
- Success, [51](#)
- UnclearFailure, [51](#)
- Simulation::Species
  - needs, [55](#)
  - Species, [55](#)
  - speciesName, [55](#)
- Simulation::StringFloatPair
  - floatValue, [56](#)
  - StringFloatPair, [56](#)
  - stringValue, [56](#)
- Simulation::Strings
  - AddInteractionButtonLabel, [62](#)
  - AddNeedButtonLabel, [62](#)
  - AddSpeciesButtonLabel, [62](#)
  - AdvancedDatabaseViewHeadline, [62](#)
  - AdvancedInstructionLine1, [62](#)
  - AdvancedInstructionLine2, [62](#)
  - AdvancedInstructionLine3, [62](#)
  - AdvancedInstructionLine4, [62](#)
  - AdvancedInstructionLine5, [62](#)
  - AdvancedSettings, [62](#)
  - Advice, [62](#)
  - AssetAppendix, [62](#)
  - AssetsPath, [62](#)
  - AssignNeedLabel, [62](#)
  - AssignSpeciesOptionToAgentLabel, [62](#)
  - Asterisk, [62](#)
  - AttachSpecificDBLabel, [62](#)
  - AuctionableBoolArray, [62](#)
  - BuildAffordanceTreeFromSceneLabel, [62](#)
  - ButtonAddDatabaseLabel, [62](#)
  - ButtonRemoveDatabaseLabel, [63](#)
  - Contact, [63](#)
  - currentDatabaseLoadedLabel, [63](#)
  - DataBasesAvailableLabel, [63](#)
  - DataFolder, [63](#)
  - DebugInSpectorLabel, [63](#)
  - DebugIngameLabel, [63](#)
  - DefaultDatabaseHeadline, [63](#)
  - DefaultDatabaseName, [63](#)
  - DefaultInteractionName, [63](#)
  - DefaultInteractionRelativeNeedName, [59](#)
  - DefaultNeedName, [63](#)
  - DefaultSpecies, [63](#)
  - defaultString, [63](#)
  - DeleteDatabaseMessage, [63](#)
  - DeleteDatabaseTitle, [63](#)
  - DialogButtonExitPlay, [63](#)
  - DialogCancel, [63](#)
  - DialogMessageExitPlay, [63](#)
  - DialogTitleExitPlay, [63](#)
  - DialogYes, [63](#)
  - DrawGizmosInGame, [63](#)
  - EditorExtensionPosition, [60](#)
  - EditorWindowTitle, [63](#)
  - FolderName, [63](#)
  - GeneralSettings, [63](#)
  - GenericDatabaseName, [63](#)
  - InteractionData, [63](#)
  - InteractionDataArrayAccess, [64](#)
  - InteractionDurationLabel, [60](#)
  - InteractionLabelToExtend, [60](#)
  - InteractionNameLabel, [64](#)
  - InteractionViewHeadline, [64](#)
  - IsAgent, [64](#)
  - IsAgentLabel, [64](#)
  - IsAuctionableArrayAccess, [64](#)
  - IsAuctionableArraySize, [64](#)
  - IsAuctionableLabel, [64](#)
  - LabelCriticalState, [60](#)
  - LabelNeedChangeRate, [61](#)
  - LabelNeedName, [61](#)
  - LabelSatisfiedState, [61](#)
  - LogSimulationLabel, [64](#)
  - MarkCurrentDatabaseAsDefault, [64](#)
  - ModifyLookAt, [64](#)
  - MoreThanOneDefaultData, [64](#)
  - NeedAffectedLabel, [61](#)
  - NeedsDataBasePath, [64](#)
  - NeedsInSpeciesLabel, [64](#)
  - NewHasBeenSet, [60](#)
  - NoDatabaseFound, [64](#)
  - NoDefaultDataFound, [64](#)
  - None, [64](#)
  - NumberOfNeedsAffectedLabel, [61](#)
  - NumberOfSlotsLabel, [64](#)
  - O\_Space, [64](#)
  - PreconditionDefaultAny, [64](#)
  - PreconditionsHeadline, [65](#)
  - PreconditionsSatisfactionLevelLabel, [65](#)
  - PreconditionsSpecies, [65](#)
  - PreconditionsSatisfactionLevelArray, [65](#)
  - PreconditionsUnsatisfiedLabel, [65](#)
  - PrintSimDebugLogLabel, [65](#)
  - ProductDescription, [61](#)
  - ProductName, [62](#)
  - RandomStartLevelsLabel, [65](#)
  - RemoveNeedButtonLabel, [65](#)
  - SatisfactionRateLabel, [62](#)
  - SceneAppendix, [65](#)
  - ShowDebugGizmosLabel, [65](#)
  - ShowDebugInGame, [65](#)
  - ShowDebugInInspector, [65](#)
  - SimulationManagerConstructed, [65](#)
  - SlotNumberLabel, [60](#)
  - SlotPositionLabel, [65](#)
  - SlotPositionsArray, [65](#)
  - SlotPositionsArrayAccess, [65](#)
  - SlotPositionsArraySize, [65](#)
  - SpeciesName, [65](#)
  - SpeciesNameLabel, [65](#)
  - SpeciesViewHeadline, [66](#)
  - SupportMail, [66](#)

- TerritoryCenter, 66
- TerritoryControlHeadline, 66
- UnableToLoadDatabase, 66
- UserDataResources, 66
- VisitUsAt, 66
- WebsiteURL, 66
- WebsiteURLButtonText, 66
- Welcome, 66
- WindowSelectionNames, 66
- YourDatabaseName, 66
- SimulationData
  - Simulation::SimulationData, 48
- SimulationManagerConstructed
  - Simulation::Strings, 65
- Slot
  - Simulation::Slot, 52
- SlotColor
  - Simulation::GeneralSettings, 28
- SlotNumberLabel
  - Simulation::Strings, 60
- SlotPositionLabel
  - Simulation::Strings, 65
- SlotPositionsArray
  - Simulation::Strings, 65
- SlotPositionsArrayAccess
  - Simulation::Strings, 65
- SlotPositionsArraySize
  - Simulation::Strings, 65
- SlotRepresentationHandleRadius
  - Simulation::GeneralSettings, 28
- SlotRepresentationRadius
  - Simulation::GeneralSettings, 28
- SlotState
  - Simulation::Slot, 54
- SlotStates
  - Simulation::Slot, 52
- slotToAgentDistanceSmall
  - NEEDSIM::Blackboard, 20
- Slots
  - Simulation::AffordanceTreeNode, 19
- SpawnBed
  - NEEDSIMSampleSceneScripts::SpawnBeds↔
    - Manager, 54
- SpawnTheFox
  - NEEDSIMSampleSceneScripts::Fuchsalarm↔
    - DemoScript, 26
- Species
  - Simulation::AffordanceTreeNode, 19
  - Simulation::DatabaseAsset, 24
  - Simulation::Species, 55
- SpeciesAllowed
  - Simulation::Interaction, 32
  - Simulation::InteractionData, 33
- SpeciesName
  - Simulation::Strings, 65
- speciesName
  - Simulation::Species, 55
- SpeciesNameLabel
  - Simulation::Strings, 65
- SpeciesViewHeadline
  - Simulation::Strings, 66
- StartInteraction
  - Simulation::Affordance, 14
- StartRandomInteraction
  - Simulation::Affordance, 14
- StringFloatPair
  - Simulation::StringFloatPair, 56
- stringValue
  - Simulation::StringFloatPair, 56
- Success
  - Simulation::Slot, 51
- SupportMail
  - Simulation::Strings, 66
- TerritoryCenter
  - Simulation::Strings, 66
- TerritoryControlHeadline
  - Simulation::Strings, 66
- TryConsumingAnimationOrder
  - NEEDSIM::NEEDSIMNode, 41
- UnableToLoadDatabase
  - Simulation::Strings, 66
- UnclearFailure
  - Simulation::Slot, 51
- Uncritical
  - Simulation::Needs, 39
- Unvalued
  - Simulation::Needs, 39
- Update
  - NEEDSIM::PlanDemo, 44
- UpdateAffordanceTree
  - Simulation::Manager, 35
- UserDataResources
  - Simulation::Strings, 66
- VisitUsAt
  - Simulation::Strings, 66
- WebsiteURL
  - Simulation::Strings, 66
- WebsiteURLButtonText
  - Simulation::Strings, 66
- WeightsForNeed
  - Simulation::SimulationData, 49
- WeightsForNeedSatisfaction
  - Simulation::SimulationData, 49
- Welcome
  - Simulation::Strings, 66
- WindowSelectionNames
  - Simulation::Strings, 66
- YourDatabaseName
  - Simulation::Strings, 66
- ZeroValuedNeeds
  - Simulation::Manager, 35