ICE CREATURE CONTROL

SPIRIT AND MOTION FOR VIRTUAL LIFE-FORMS

NPC CONTROLLER FOR UNITY

USER MANUAL

Version 1.1 (draft)

Pit Vetterick

1 INDEX

2 INTRODUCTION

2.1 THANK YOU FOR PURCHASING ICE CREATURE CONTROL

Congratulations on your choice to purchase our ICE Creature Control Package for UNITY 5! With your purchase you are supporting me to improve this product for you and in this spirit I would like to thank you and want you to enjoy your purchase to the fullest.

2.2 YOU ARE NOT ALONE

I spend countless hours to offer you a stable and feature-rich product and working hard to constantly improve it as well, but for a single person it's a mission impossible to testing all possible combinations to using the software and it could be that of all things your favourite feature will not work correctly, so please keep in mind that you are not alone and if you have any questions, problems or suggestions, please feel free to visit the Unity support forum or contact me directly.

In urgent cases you can contact me also directly via Skype and/or TeamViewer, but please consider that I'm an indie developer, without a large studio or support department behind me and therefore I can't be reachable 24/7, even though I really would like to it.

At this point I would like to thank all the community members who support me with inspiring ideas and suggestions but also by supporting other members with their experience and creative solutions. That's so great – thanks a lot for this!

http://forum.unity3d.com/threads/347147/

http://www.icecreaturecontrol.com

support@icecreaturecontrol.com

2.3 SPIRIT AND MOTION

The Unity Engine provides the perfect environment to handle virtual characters and especially the physical setup is done quite easily by using the given features. The problem is rather to realize an effective and flexible logic which have the ability to sense and react to the environment to finally control the virtual character autonomous and as natural as possible but for all that easy to use, combinable with additional components and reusable for nearly each kind of virtual character.

If you was looking for a plugin that fulfil all of these requirements, you can stop searching – I'm nearly sure you have found it!

ICECreatureControl is an unbelievable piece of software to breathe life into your virtual Characters without typing one single line of code. The software combines a complex behaviour system, animations and path finding techniques, a powerful character controller and even more features in one single, easy to use component to provide you the ultimate NPC Controller.

ICECreatureControl is a complex software with hundreds of settings and (if you want) millions of possibilities – something like Thors hammer in your hand! But please consider that these complexity have its price and it could be that you get a shock if you see the inspector panel for the first time, but don't panic, you will see, working with ICECreatureControl is really easy.

But nevertheless, before you start to discover all the features you should know a bit about the concept and philosophy behind the code, so it will be easier to you to understand what your creature will do during the runtime and I would like to recommend you to going the first steps in an empty project, even so if you have already advanced experience with an older version. ICECreatureControl v1.1 contains tons of improved and new features and there are also some structural changes to optimize and expand the code for further versions, therefore it will be in your own interest to learn the ropes and familiarise yourself with all the changes before jump in at the deep end.

2.4 PHILOSOPHY

The philosophy behind ICECreatureControl should be familiar to you, because it is an abstract copy of nature. Just imagine that your virtual character is a real life-form with all strength and weakness that comes with it and as his trainer you have to make sure that your protégé is healthy and ready for action, has the right motivation to wake up in the morning and all required skills to fulfil his life-task.

2.4.1 Fitness – The Mind-Body Connection

The 'medical check-up' of your creature should deal primarily with the physical fitness, which contains the functional body, incl. Model, Rig and Animations etc. - all fundamental properties to fulfil the physical requirements to act and looks healthy.

But as you know, wellbeing based generally on a fundamental link between physical and mental characteristics and the mental fitness of your creature will be finally responsible to control his physical capabilities in the right way.

ICECreatureControl will handle the mental part, which contains the logic to sense the environment and to make situation and behaviour based decisions, to finally control the physical body. But in order that your creature will behave as expected, you have link body and mind of your creature by adapt the Physics, Motion and Pathfinding settings of the ICECreatureControl and, as the case may be, also of additional components such as Collider, Rigidbody or NavMeshAgent.

If this is done your creature is a harmonic unit - a healthy mind in a healthy body - and you can start to motivate and teach your creature.

Btw. At this point you should also note that the ICECreatureControl is an unassumingly modest component, which can handle each object with a transform component and therefore each kind of GameObject in your scene, so it is not absolutely necessary to start with an full featured and animated model, you can assemble and configure your creature step by step.

2.4.2 Motivation – Sense of life

Now your creature is nearly ready for action, the only things that are missing now is the right motivation to act and the right way to do it.

Without motivation nothing would get done in this world and also your creature follows strictly the principle of cause and effect and will do nothing without cause, because it's governed by goal-orientated behaviour and the best (and only) motivation for your creature is a valuable goal - a target which your creature can reach.

ICECreatureControl provides you to determine several targets for your creature. Targets could be static GameObjects, such as simple Waypoints but also movable Objects, such as the Player Character or other NPCs as well. All Targets are potential interaction candidates, which could continuous affect the situation and provokes reactions.

(See also: Targets)

2.4.3 Qualification – Training is everything

Archimedes once demanded just a lever and an immovable point, to move the world. Give both to your creature and it couldn't even move itself because it would not know how to do it and therefore you have to teach your creature all relevant abilities before it will be useful for your project.

ICECreatureControl provides you a complex behaviour system to teach your creature. The Behaviour of your creature based on a collection of several BehaviourModes. Each Mode typify the guidelines for a specific task or situation and contains one or more BehaviourRules, which in turn contains the final instructions for animation, movements, influences, visual and audio effects etc.

BehaviourModes are flexible and reusable. You could define BehaviourModes, such as 'IDLE', 'RUN' or 'JUMP' as simple default instructions for several targets and situations, but you can also specify extensive sets with several customized rules to realize complex fighting scenes.

NOTE: Targets and Behaviours

Targets and Behaviours are the moving spirit of your creature and you can find their settings in nearly each module. ICECreatureControl provides your creature the ability to handle a large number of situations and reactions by specify Targets and Behaviours but please note, that everything your creature will (and can) do is finally just a reaction of a given situation and for this your virtual protégé should know all potential situations and all relevant rules of conduct as well.

3 USER INTERFACE – THE COCKPIT

The ICE Creature Control is a powerful framework with nearly endless possibilities but if you want also with hundreds of options and adjustments and exactly this could be a little bit tricky, especially if you have a lot of creatures with complex rules. But nevertheless I'm constantly strive to improve the user interface and to optimize the usability.

3.1 QUICK SELECTION

The Register Popup is directly connected to the Creature Register Component and allows you to switch quickly to the desired creature or to the inspector view of the register as well. Use the Register Popup in conjunction with the 'global' display options, to get a quick access to your focused settings sections, without searching in several foldouts.

See also: ICECreatureRegister

3.2 DISPLAY OPTIONS

Here you can choose your individual display options, dependent to your tasks and requirements. Hide the unneeded features and reduce the control to the relevant parts, so you will never lose the track.

Tip: If you activate the 'ALL GLOBAL' flag before switching to another creature the target inspector view will inherit your current display options. This feature will be helpful for you to adjust movements and behaviour details without searching the required section.

4 THREE THINGS YOU NEED TO KNOW - TARGETS, BEHAVIOURS, MOVEMENTS

4.1 TARGETS

Targets represents potential destinations and interaction objects and contains as fundamental elements all relevant information about motion and behaviour of your creature. Please note that the behaviour of your creature is target-driven, therefore it is fundamental that your creature have at least one reachable target.

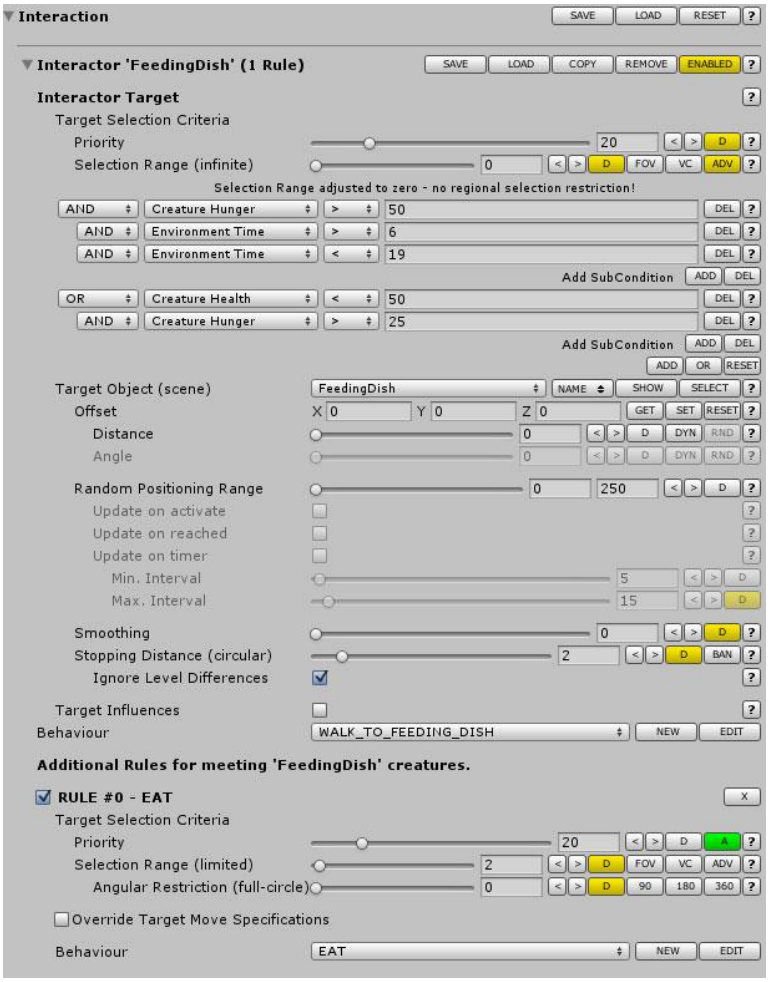
4.1.1 Target Object

Basically a Target Object can be each static or movable GameObject in your scene or a Prefab as well. The only requirement is here that the given position should be reachable for your creature and please consider also the typical characteristics of scene objects and prefabs (e.g. nested prefabs etc.)

4.1.2 Target Selection Criteria

Your creature could have several targets at the same time in such cases it can use a set of selection criteria to evaluate the most suitable target related to the given situation. Here you can define the priority and relevance of a target.

* Please consider, the HOME target should normally have the lowest priority, because it should be rather a side show than the main event, a secluded place where your creature can spawn or become modified invisible to the player.



4.1.2.1 Selection Priority

In cases that your creature will have several valid targets at the same time, the priority determines the relevance of the targets and the creature will select the target with the highest priority. If there are two or more targets with the same priority, the selection will be randomized.

4.1.2.2 Selection Range

The Selection Range defines the maximum distance in which the creature could detect the target. If the Selection Range is adjusted to zero, the Selection Range will be ignored and the condition will be always true.

4.1.2.2.1 Field Of View (FOV)

While Field Of View (FOV) is active the target must be inside the defined view angle of the creature. Please note, that FOV verifies only the position of the creature and not the real visibility, that's will be done if the Visibility Check is active as well.

4.1.2.2.2 Visibility Check (VC)

While Visibility Check is active the target must be visible for creature, which means that the visual axis between creature and target may not be intersected by another collider. Please note, that VC verifies the sighting line without consideration of current viewing direction, that's will be done if the Field Of View is active as well.

4.1.2.3 Selection Angle

While the Field Of View defines the view angle of the creature, the Selection Angle deals with a notional view angle of the target, in which the creature must be inside to fulfil the condition. To adjust this angle to zero will have the same effect as an adjustment of 360 degrees, in both cases the selection angle will be ignored and the condition will be true.

4.1.3 Advanced Settings

The advanced Target Selection Criteria provides you to define multiple selectors with customized conditions.

4.1.3.1 Selected Creature Parameter

4.1.3.1.1 OwnGameObject

GameObject of the creature

4.1.3.1.2 OwnActiveTarget

GameObject of the active target

4.1.3.1.3 OwnActiveTargetTime

Time in seconds since the current target is active

4.1.3.1.4 OwnActiveTargetTimeTotal

Total time in seconds of the active target

4.1.3.1.5 OwnBehaviour

Current behaviour of the creature

4.1.3.1.6 OwnCommand

Last incoming command of your creature

4.1.3.1.7 OwnAge

Age of your creature

4.1.3.1.8 OwnOdour

Current odour of your creature

4.1.3.1.9 OwnOdourIntensity

Current odour intensity of your creature

4.1.3.1.10 OwnOdourRange

Current odour range of your creature

4.1.3.1.11 OwnEnvTemperatureDeviation

Current temperature deviation in relation to the defined comfort temperature

4.1.3.1.12 OwnFitness

Current fitness value of your creature

4.1.3.1.13 OwnHealth

Current health value of your creature

4.1.3.1.14 OwnStamina

Current stamina value of your creature

4.1.3.1.15 OwnPower

Current power value of your creature

4.1.3.1.16 OwnDamage

Current damage value of your creature

4.1.3.1.17 OwnStress

Current stress value of your creature

4.1.3.1.18 OwnDebility

Current debility value of your creature

4.1.3.1.19 OwnHunger

Current hunger value of your creature

4.1.3.1.20 OwnThirst

Current thirst value of your creature

4.1.3.1.21 OwnAggressivity

Current aggressivity value of your creature

4.1.3.1.22 OwnExperience

Current experience value of your creature

4.1.3.1.23 OwnAnxiety
Current anxiety value of your creature

4.1.3.1.24 OwnNosiness
Current nosiness value of your creature

4.1.3.1.25 OwnVisualSense
Current visual sense value of your creature

4.1.3.1.26 OwnAuditorySense
Current auditory sense value of your creature

4.1.3.1.27 OwnOlfactorySense
Current olfactory sense value of your creature

4.1.3.1.28 OwnGustatorySense
Current gustary sense value of your creature

4.1.3.1.29 OwnTactileSense
Current tactile sense value of your creature

4.1.3.1.30 OwnPosition
Current position values of your creature

4.1.3.1.31 OwnIsDead
Current alive status

4.1.3.2 Selected Target Parameter

4.1.3.2.1 TargetGameObject
Current target GameObject

4.1.3.2.2 TargetDistance
Current target transform position distance to the selected creature

4.1.3.2.3 TargetOffsetPositionDistance
Current target offset position distance to the selected creature

4.1.3.2.4 TargetMovePositionDistance
Current target move position distance to the selected creature

4.1.3.2.5 TargetReferenceType
Current target type (CREATURE, PLAYER, ITEM, LOCATION, WAYPOINT, MARKER)

4.1.3.3 Selected Target Creature Parameter (only available if the target is a CREATURE)

4.1.3.3.1 CreatureGameObject
GameObject of the creature

4.1.3.3.2 CreatureActiveTarget
GameObject of the active target of the target creature (target must be a creature)

4.1.3.3.3 CreatureActiveTargetTime
Time in seconds since the current target is active

4.1.3.3.4 CreatureActiveTargetTimeTotal
Total time in seconds of the active target

4.1.3.3.5 CreatureBehaviour
Current behaviour of the target creature (target must be a creature)

4.1.3.3.6 CreatureCommand
Last incoming command of the target creature (target must be a creature)

4.1.3.3.7 CreatureAge
Age of the target creature (target must be a creature)

4.1.3.3.8 CreatureOdour
Current odour of the target creature (target must be a creature)

4.1.3.3.9 CreatureOdourIntensity
Current odour intensity of the target creature (target must be a creature)

4.1.3.3.10 CreatureOdourIntensityNet
Current net odour intensity of the target creature. Calculated value based on the distance and the own olfactory sense value. (target must be a creature)

4.1.3.3.11 CreatureOdourIntensityByDistance
Current odour intensity of the target creature. Calculated value based on the distance. (target must be a creature)

4.1.3.3.12 CreatureOdourRange
Current odour range of the target creature (target must be a creature)

4.1.3.3.13 CreatureEnvTemperatureDeviation
Current temperature deviation in relation to the defined comfort temperature

4.1.3.3.14 CreatureFitness
Current fitness value of the target creature (target must be a creature)

4.1.3.3.15 CreatureHealth
Current health value of the target creature (target must be a creature)

4.1.3.3.16 CreatureStamina
Current stamina value of the target creature (target must be a creature)

4.1.3.3.17 CreaturePower
Current power value of the target creature (target must be a creature)

4.1.3.3.18 CreatureDamage
Current damage value of the target creature (target must be a creature)

4.1.3.3.19 CreatureStress
Current stress value of the target creature (target must be a creature)

4.1.3.3.20 CreatureDebility
Current debility value of the target creature (target must be a creature)

4.1.3.3.21 CreatureHunger
Current hunger value of the target creature (target must be a creature)

4.1.3.3.22 CreatureThirst
Current thirst value of the target creature (target must be a creature)

4.1.3.3.23 CreatureAggressivity
Current aggressivity value of the target creature (target must be a creature)

4.1.3.3.24 CreatureExperience
Current experience value of the target creature (target must be a creature)

4.1.3.3.25 CreatureAnxiety
Current anxiety value of the target creature (target must be a creature)

4.1.3.3.26 CreatureNosiness
Current nosiness value of the target creature (target must be a creature)

4.1.3.3.27 CreatureVisualSense
Current visual sense value of the target creature (target must be a creature)

4.1.3.3.28 CreatureAuditorySense
Current auditory sense value of the target creature (target must be a creature)

4.1.3.3.29 CreatureOlfactorySense
Current olfactory sense value of the target creature (target must be a creature)

4.1.3.3.30 CreatureGustatorySense
Current gustary sense value of the target creature (target must be a creature)

4.1.3.3.31 CreatureTactileSense
Current tactile sense value of the target creature (target must be a creature)

4.1.3.3.32 CreaturePosition
Current position values of the target creature (target must be a creature)

4.1.3.3.33 CreatureIsDead
Current alive status of the target creature (target must be a creature)

4.1.3.4 Selected Target Player Parameter (only available if the target is a PLAYER)

4.1.3.4.1 Player
GameObject of the player

(More parameter planed)

4.1.3.5 Selected Target Location Parameter (only available if the target is a LOCATION)

4.1.3.5.1 Location

GameObject of the player

(More parameter planed)

4.1.3.6 Selected Target Waypoint Parameter (only available if the target is a WAYPOINT)

4.1.3.6.1 Waypoint

GameObject of the player

(More parameter planed)

4.1.3.7 Selected Target Item Parameter (only available if the target is a ITEM)

4.1.3.7.1 Item

GameObject of the player

(More parameter planed)

4.1.3.8 Selected Target Marker Parameter (only available if the target is a MARKER)

4.1.3.8.1 Marker

GameObject of the player

4.1.3.8.2 MarkerOdour

Current odour of the target marker (target must be a marker)

4.1.3.8.3 MarkerOdourIntensity

Current odour intensity of the target marker (target must be a marker)

4.1.3.8.4 MarkerOdourIntensityNet

Current net odour intensity of the target marker. Calculated value based on the distance and the own olfactory sense value. (target must be a marker)

4.1.3.8.5 MarkerOdourIntensityByDistance

Current odour intensity of the target marker. Calculated value based on the distance. (target must be a marker)

4.1.3.8.6 MarkerOdourRange

Current odour range of the target marker (target must be a marker)

4.1.3.9 PreviousTargetType

4.1.3.10 PreviousTargetName

4.1.3.11 PreviousTargetTag

4.1.3.12 Environment Parameter

4.1.3.12.1 EnvironmentTimeHour

4.1.3.12.2 EnvironmentTimeMinute

4.1.3.12.3 EnvironmentTimeSecond

4.1.3.12.4 EnvironmentDateYear

4.1.3.12.5 EnvironmentDateMonth

4.1.3.12.6 EnvironmentDateDay

4.1.3.12.7 EnvironmentTemperature

4.1.3.12.8 EnvironmentWeather

4.1.3.13 Input Parameter

4.1.3.13.1 InputKey

4.1.4 Target Move Specifications

Your creature always try to reach the TargetMovePosition of the given target object. By default the Target Move Position will be the transform position of a target, but in the majority of cases the transform position will be suboptimal or simply non-practical as access point and therefore the Target Move Specifications provides several settings to adapt the TargetMovePosition as desired and allows you to define a fixed point related to the target or a dynamic and randomized position as well.

4.1.4.1 Target Offset

The Offset values specifies a local position related to the transform position of the target. The offset settings are optional and allows you to adapt the target position if the original transform position of an object is not reachable or in another way suboptimal or not usable. The TargetOffsetPosition contains the world coordinates of the local offset, which will used as centre for the randomized positioning and finally as TargetMovePosition as well.

4.1.4.1.1 Target Offset Distance

Additional to adapt the offset position by enter the coordinates, you can use distance and angle to define the desired position. Distance defines the offset distance related to the transform position of the target.

4.1.4.1.1.1 DYN Button (Min, Max, Speed) (NEW)

Activate the DYN function and adapt the values to use a dynamic Offset Distance during the runtime.

4.1.4.1.1.2 RND Button (NEW)

Activate the RND function to randomize the dynamic values during the runtime

4.1.4.1.2 Target Offset Angle

Additional to adapt the offset position by enter the coordinates, you can use distance and angle to define the desired position. Angle defines the offset angle related to the transform position of the target. Zero (or 360) degrees defines a position in front of the target, 180 degrees consequently a position behind it etc.

4.1.4.1.2.1 DYN Button (Min, Max, Speed) (NEW)

Activate the DYN function and adapt the values to use a dynamic Offset Angle during the runtime.

4.1.4.1.2.2 RND Button (NEW)

Activate the RND function to randomize the dynamic values during the runtime

4.1.4.2 Target Random Positioning Range

Random Positioning Range specifies the radius of a circular area around the TargetOffsetPosition to provide a randomized positioning. The combination of TargetOffsetPosition and Random Range produced the TargetMovePosition as the final target position, which will used for all target related moves. While using a random position you can define the update conditions to reposition the TargetMovePosition during the runtime. Please note, that you can combine also two or all conditions as well.

4.1.4.2.1 Update On Activate

While using a random position you can define the update conditions. Update On Active will refresh the position whenever the target becomes active.

4.1.4.2.2 Update On Reached

While using a random position you can define the update conditions. Update On Reached will refresh the position whenever the creature has reached the given TargetMovePosition.

4.1.4.2.3 Update On Timer

While using a random position you can define the update conditions. Update On Timer will refresh the position according to the defined interval.

4.1.4.3 Target Stopping Distance

The Target Stopping Distance defines the minimum distance related to the TargetMovePosition to complete the current move. If your creature is within this distance, the TargetMovePosition was reached and the move is complete (that's for example the precondition to run a RENDEZVOUS behaviour).

4.1.4.3.1 Button 3D (Consider Level Differences)

By default the distance between your creature and the selected target will measured without differences in height, because it covers the most cases and tolerates also roughly target position settings. But in some situations (e.g. levels or buildings with walkable surfaces on several elevations etc.) you will need also the differences of y-axis and in such cases you can activate the 3D Button to consider the correct level differences

4.1.4.3.2 Button BAN (Restricted Zone) *(NEW)*

While the BAN function is active, your creature can't enter the inner circle of the Stopping Distance, this area will be restricted for your creature and its position will be automatically corrected to keep it out of this range.

4.1.4.4 Smoothing

The Smoothing Multiplier affects step-size and update speed of the TargetMovePosition. If Smoothing is adjusted to zero the TargetMovePosition will relocated directly during an update, if it is adjusted to one the TargetMovePosition will changed extremely slow and soft.

4.1.5 Target Influences *(NEW)*

The Target Inluences enables to adapt influences if the target is active (e.g. your prey creature have detected a predator target and therefore its stress level goes up)

4.1.6 Target Group Message *(NEW)*

The Target Group Message is a new feature which allows your creature to communicate with other creatures in its group.

4.1.6.1 Target Tutorials

4.1.6.1.1 TutorialStaticTarget

This tutorial provides you to test the interplay between the Target Move Specification values during the runtime. Modify the different values to see the result. Open also the inspector view of the Essential Settings to follow and understand the functionality of the Target Offset values, the Random Positioning Range and the Stopping Distance. Play also with the velocity values to understand the effect of the forward and angular velocity, curve radius and the given Stopping Distance to reach the given Target Move Position.

4.1.6.1.2 TutorialMovableTarget

This tutorial provides you to test the interplay between the Target Move Specification values during the runtime. Modify the different values to see the result. Open also the inspector view of the Essential Settings to follow and understand the functionality of the Target Offset values, the Random Positioning Range and the Stopping Distance. Play also with the velocity values to understand the effect of the forward and angular velocity, curve radius and the given Stopping Distance to reach the given Target Move Position.

Reassign the targets by flag the "Use … as Target" toggles to understand how you could force your creatures to follow another movable object (e.g. wandering herds etc.)

4.2 BEHAVIOURS

While a Target represents a goal, Behaviours defines the way to reach it. The Behaviour settings provides you to design and manage complex behaviour instructions and procedures, to reach your needs and goals and finally a realistic and natural behaviour of your creature.

4.2.1 Behaviour Modes

The behaviour of your creature is subdivided into single Behaviour Modes. Each of these modes contains the instructions for specific situations and can be assigned to target-related or condition-based events. Furthermore Behaviour Modes are not bounded to specific assignments and can generally be used for several targets and situations, in case they are suitable for them.

Each Behaviour Mode contains at least one Behaviour Rule, but to reach a more realistic behaviour you can add additional rules, which allows your creature to do things in different ways, break and resume running activities, run intermediate animation sequences, start effects or to play audio files as well.

4.2.1.1 Rename

Renames allows you to change the key of the selected Behaviour Mode. Please note, that renaming will remove all existing assignments.

4.2.1.2 Copy

Creates a copy of the selected Behaviour Mode

4.2.1.3 Remove

Removes the selected Behaviour Mode

4.2.1.4 Favoured

The ‘Favoured’ flag allows you to block other targets and behaviours until the defined conditions of the active mode are fulfilled. By using this feature you can force a specific behaviour independent of higher-prioritised targets, which will normally determines the active behaviour. You can select several conditions, in this case the selected ones will combined with OR, so that just one of them must be true. Please consider, that the active mode will in fact stay active until the conditions are fulfilled, so please make sure, that your creature can fulfil the conditions, otherwise you will provide a deadlock.

4.2.1.4.1 Priority

4.2.1.4.2 Minimum Period

4.2.1.4.3 Next Move Position

4.2.1.4.4 Target Move Position

4.2.1.4.5 Specific Target

4.2.1.4.6 Detour

4.2.2 Behaviour Rules

To provide a more realistic behaviour each mode can contains several different rules of instructions at the same time, which allows your creature to do things in different ways, break and resume running activities, run intermediate animation sequences, start effects or to play audio files as well.

4.2.2.1 Length

Here you can define the play length of a rule by setting the 'min' and 'max' range. If both values are identical, the rule will be playing exactly for the specified time-span, otherwise the length will be randomized based on the given values. Please note, that these settings are only available, if your selected 'Behaviour Mode' contains more than one rule. If you ignore the play length settings while you have more than one rule, the control tries to use the animation length but this could originate unlovely results and is inadvisable.

4.2.2.2 Animation

Here you can define the animation you want to use for the selected rule. Simply choose the desired type and adapt the required settings.

4.2.2.2.1 Animation Types

- NONE
  To use an animation is not obligatory required, so you can control also each kind of unanimated objects, such as dummies for testing and prototyping, simple bots and turrets or movable waypoints.

- ANIMATOR (MECANIM)
  By choosing the ANIMATOR ICECreatureControl will using the Animator Interface to control Unity's powerful Mecanim animation system. To facilitate setup and handling, ICECreatureControl provide three different options to working with Mecanim:

  - DIRECT – similar to the legacy animation handling
  - CONDITIONS – triggering by specified conditions (float, integer, Boolean and trigger)
  - ADVANCED – similar to CONDITIONS with additional settings for IK (ALPHA)

- ANIMATION (LEGACY)
  Working with legacy animations is the easiest and fastest way to get nice-looking results with some mouse clicks. Simply select the desired animation, set the correct WrapMode and go. Legacy animations are perfect for the tests and rapid prototyping, but please consider that Unity intends to phase out the Legacy animation system over time, so you should not use it for new and especially not for larger Projects.

- CLIP
  The direct use of animation clips is inadvisable and here only implemented for the sake of completeness and for some single cases it could be helpful to have it. Apart from this it works like the animation list. Simply select the desired animation, set the correct WrapMode and go.

4.2.2.3 Movement

Additional to the Default Move, which you can adapt in the Essential section, each Behaviour Rule provides enhanced Movement Options to customize the spatial movements of your creature according to the selected Animation, the desired behaviour or other needs and requirements.

In difference to the Default Move settings, the Behaviour Movements contains in addition to the known move specifications, further settings to define advanced movements, the viewing direction and the velocity, which is absolutely essential if a desired behaviour is to be provided spatial position changes. In such cases it's indispensable to adapt the velocity settings.

4.2.2.3.1 Velocity

4.2.2.3.1.1 Forward Velocity

Forward Velocity defines the speed of your creature in its z-direction. Please note, that the adjustment of the velocity is absolutely essential for all spatial movements. By activating the AUTO function, your creature will adjusts its velocity according to the given target. The NEG flag allows you to use negative velocity values. Make sure that the velocity values are suitable to the defined animation, otherwise your creature will do a moonwalk and please consider, that a zero value means no move.

4.2.2.3.1.2 Velocity Forward Variance

Use the Velocity Variance Multiplier to randomize the Forward Velocity Vector during the runtime, to force non-uniform movements of your creature (this will be helpful while using several instances of your creature)

4.2.2.3.1.3 Inertia (Mass Inertia)

The Inertia value will be used to simulate the mass inertia to avoid abrupt movements while the speed value changed.

4.2.2.3.1.4 Angular Velocity (y)

Angular Velocity defines the desired rotational speed of your creature around its y axis. This value affects the turning radius of your creature – the smaller the value, the larger the radius and vice versa. For a realistic behaviour, this value should be consider the given physical facts and therefore suitable to the specified speed and the naturally to the animation and the kind of creature as well.

4.2.2.3.2 Viewing Direction

Viewing Direction defines the direction your creature will look at while the behaviour is active. By default your creature will into the move direction, but in some cases it can be helpful to force a specific direction independent of the move direction.

4.2.2.3.3 Move

By default ICECreatureControl will use the Default Move for all standard situations, which describes a direct manoeuvre form the current transform position to the TargetMovePosition. This manoeuvre will be sufficient in the majority of cases, but it is less helpful if your creature have to veer away from a target, such as in an escape situation. In such cases you can overwrite the Default Move Settings by using one of the offered move options.

4.2.2.4 Influences

Influences defines the impact of a triggering event to your creature. These impacts could be positive for your creature, such as a recreation processes by reducing the damage while your creature is sleeping or eating, or negative through increasing the damage or stress values while your creature is fighting. Impacts will be directly affect the status values of your creature. While a triggering event is active, influences will refresh the status values during the framerate-independent update cycle of FixedUpdate (default 0.02 secs.), so please make sure, that your defined impact values are suitable to this short time-span or increase the interval value, otherwise your creature could die immediately.

4.2.2.4.1 Interval

Interval defines the time delay in seconds between two influence calls. By default this value is adjusted to zero, which means that an influence call will affect your creature in each framerate-independent update cycle of FixedUpdate (default 0.02 secs. .), so please make sure, that your defined impact values are suitable to this short time-span or increase the interval value, otherwise your creature could die immediately.

4.2.2.4.2 Damage

Damage specifies the impact to the damage status attribute and depending on the associated multiplier to the default indicators as well.

4.2.2.4.3 Stress

Stress specifies the impact to the stress status attribute and depending on the associated multiplier to the default indicators as well.

4.2.2.4.4 Debility

Debility specifies the impact to the debility status attribute and depending on the associated multiplier to the default indicators as well.

4.2.2.4.5 Hunger

Hunger specifies the impact to the hunger status attribute and depending on the associated multiplier to the default indicators as well.

4.2.2.4.6 Thirst

Thirst specifies the impact to the thirst status attribute and depending on the associated multiplier to the default indicators as well.

4.2.2.5 Inventory

4.2.2.5.1 Collect Active Item

While ‘Collect Active Item’ is flagged your creature will collect the GameObject of the active Target if this contains an own Inventory such as the ICECreatureItem type, otherwise this instruction will be ignored.

Example: Add the desired Item Object as Interactor to your creature, so that your creature can detect and reach the object. Add an additional Interactor Rule in case that your creature have reached the target and define a ‘Pick Item Up’ behaviour, set a suitable ‘pick’ animation and activate ‘Collect Active Item’. Now your creature will pick the item up as soon as the condition for the additional Interactor Rule are true. The collected object will be removed, added to the inventory of your creature and your creature will try to detect the next one according the given target selection criteria.

You can use this feature to harvest a field, to collect food, tools or items but also to rob and loot other creatures or your player as well.

4.2.2.5.2 Distribute

If 'Distribute Item' is flagged your creature will distribute the selected item while the respective behaviour rule is active and the inventory amount of the item is larger zero.

Example: Select the desired inventory item and define the interval in which your creature should distribute the item. Define also a suitable animation to visualize the distribution process

4.2.2.5.2.1 Item

4.2.2.5.2.2 Interval

4.2.2.5.3 Equip

4.2.2.5.3.1 Item

4.2.2.5.3.2 Parent

4.2.2.6 Audio

4.2.2.7 Effect

4.2.2.8 Link

Link provides the forwarding to a specific Rule or another Behaviour Mode as well.

4.3 MOVEMENTS

The spatial movements of your creature are basically just position changes from one point to another or rather from the current transform position of your creature to the given TargetMovePosition. The raw results are consequently straight-line paths between these two points, which are usually insufficient for realistic movements and so the control provides several options to optimize movements.

4.3.1 Default Move

The Default Move settings will be used for all standard situations and describes the manoeuvre form the current transform position to the TargetMovePosition.

4.3.1.1 Move Segment Length

The final destination point is basically the given TargetMovePosition and as long as there are no obstacles or other influences, your creature will follow a straight-line path to reach this position. If Move Segment Length is not adjusted to zero, the linear path will subdivided in segments of the defined length and the outcome of this is a sub-ordinate MovePosition which can be used to modulate the path.

4.3.1.2 Segment Variance Multiplier

Use the Segment Variance Multiplier to randomize the Segment Length during the runtime. The length will be updated when the stopping distance at the end of a segment was reached.

4.3.1.3 Lateral Variance Multiplier

Use the Lateral Variance Multiplier to force a randomized sideward drift. The random value will be refreshed when the stopping distance at the end of a segment was reached.

4.3.1.4 Stopping Distance

The Move Stopping Distance determined the minimum distance related to the actual MovePosition to complete the current move. If your creature is within this distance, the MovePosition was reached and the move is complete.

4.3.1.5 Ignore Level Differences

While Ignore Level Differences is flagged, the distance between your creature and the given MovePosition will measured without differences in height. By default, this option is ON because it covers the most cases and tolerates also roughly target position settings, but in some cases (e.g. levels or buildings with walkable surfaces on several elevations etc.) you will need also the differences of y-axis.

4.3.1.6 Tutorials Move

4.3.1.6.1 Tutorial Move Examples

This tutorial provides you to modify and test several move settings during the runtime. Select the given example settings and see the result. Modify the values to follow and understand the functionality. Play also with the velocity values to understand the effect of the forward and angular velocity, curve radius and the given Stopping Distance to reach the given Move Position. Try to adapt the values to reach a suitable walk for a potential situation (e.g. wounded animal, drunken sailor etc.)

Note: Please consider that each behaviour rule could have its own custom move settings, which allows you to combine different moves for your desired scenarios.

4.3.2 Additional Behaviour Movement

By default ICECreatureControl will use the Default Move for all standard situations, which describes a direct manoeuvre form the current transform position to the TargetMovePosition. This manoeuvre will be sufficient in the majority of cases, but it is less helpful if your creature have to veer away from a target, such as in an escape situation. In such cases you can overwrite the Default Move Settings by using one of the offered behaviour move options.

4.3.2.1 Custom Move

Custom Move allows you to overwrite the Default Move settings. In all other respects, this option is identical with the Default Move, which defines a direct manoeuvre form the current transform position of your creature to the TargetMovePosition.

4.3.2.2 Random Move

4.3.2.3 Orbit Move

Orbit Move defines an orbital move around the TargetMovePosition. You can adjust the initial radius, a positive or negative shift value, so that your creature will following a spirally path and the associated minimum and maximum distances, which specifies the end of the move. Please consider, that an orbital move with a zero shift value will not have a logical end, so you should make sure that your creature will be not circling around the target infinitely. You could do this, for example, by setting a limited play length of the rule.

4.3.2.4 Avoid Move

By using the Avoid Move, your creature will try to avoid the target by moving to the side, left or right being based on the initial sighting line. Please consider, that you can affect the Avoid behaviour by adjust the angular restriction settings of the Target Selection Criteria and/or the Field Of View of your creature.

4.3.2.5 Escape Move

By using the Escape Move, your creature will move away from the target in the opposite direction of the initial sighting line. You can randomize this escape direction by adapt the RandomEscapeAngle. The EscapeDistance defines the desired move distance, which will added to the given SelectionRange of the target. Please consider, that you can affect the Escape behaviour by adjust the angular restriction settings of the Target Selection Criteria and/or the Field Of View of your creature.

5 FIRST STEPS

5.1 SETUP WIZARD

5.2 MANUAL SETUP

1. Fitness

   Choose your desired creature and place it somewhere in your scene. Make sure that the local axes of your creature are correct aligned (the forward direction of your creature should be positive z). Your creature should have at least two animations which are suitable for idle and move behaviour. ICECreatureControl provides both Unity animation systems, MECANIM and LEGACY as well. If you are using MECANIM make sure, that the Animator Controller is ready, otherwise simply create a new Animator Controller and add your desired Animations (btw. it's not necessary to create transitions, ICECreatureControl will do it on-the-fly). Add the ICECreatureControl Component to your creature and open the inspector view of the component. If you have no active Creature Register in your scene click the related Button to add or activate it. Select the START display option to display the relevant settings only and open the Essentials foldout. Scroll down to the 'Motion and Pathfinding' settings and choose the desired 'Ground Orientation' and the desired Ground Check Type (btw. by using RAYCAST you should also define the ground layer). Make sure, that 'Handle Gravity' is checked and the gravity value is around 9.8.

   Note: To simplify the introduction, your creature should be nearly naked – no Collider, no Rigidbody, no NavMeshAgent - so you can see and understand how ICECreatureControl works. Btw. all the named components are supported, but for the first steps the only additional component should be an Animation or Animator component filled with great animations ☺

2. Target

   Your creature has successfully passed the physical fitness check and you can continue with the Home settings. Go to the 'Home' settings and define the home target for your creature. Set the selection criteria to zero, so that the home will have the lowest relevance. Choose the desired Target Object, which could be each reachable GameObject in your scene (if you have currently no other objects in your scene, you could also use the terrain or the ground object as target). If this is done, you could adapt the offset (you could use the offset if the original target position is suboptimal and/or not reachable for your creature). Set the desired radius of the 'Random Range'. If this value is 0, your creature will move to the TargetOffsetPosition and will be waiting there during the idle time (that's okay for a guard) but if you want that your creature should do some activities during the idle time, choose a larger 'Random Range' and select the desired update trigger, so that your creature will get constantly new positions and it behaviour looks more natural. Now define the 'StopDistance', that's the minimum distance to the TargetMovePosition to complete the current move. Values between 2-3 working well in the most cases. The smaller the 'StopDistance' the more precise the move but if the 'StopDistance' is too small, it could be, that your creature can't complete the move, because of its own speed and/or turn limits (btw. in such a case your creature will move circular around the TargetMovePosition, if you see it just increase the 'StopDistance' to fix it).

3. Behaviour

   Finally you have to define the desired behaviours of your creature. Go to the behaviour section within the Essentials settings. You'll see the behaviour options for 'Leisure', 'Travel', 'Dead' and 'Respawn'. Press now the AUTO Button of the 'Leisure' behaviour to create an empty behaviour mode, which will automatically labelled and assigned as 'LEISURE'. Please note, that the name of a behaviour is its unique key, which allows you to reuse a behaviour for several targets and situations. Press now the Edit Button to configure the LEISURE behaviour. Choose the required animation type and select the desired Animation and a suitable WrapMode. Speed should be 1 and the Transition Duration around 0.5 (should be the default values)

   If your selected animation represents a move (such as walk, run etc.) you have to activate the Movement options to adapt the Velocity values for Forward (z) and Angular (y). The velocity values should be suitable to your selected animation and will needed to be adapted for the best result, but for starting you could use 3 for a 'walk' and 6 for a 'run' animation and for the angular you could use the half of the forward value. Please make sure, that the Popups of 'Velocity', 'Viewing Direction' and 'Move' displays 'DEFAULT'.

   Now you could close the behaviour editor box by pressing the Edit Button again and repeat the behaviour configuration steps for 'Travel', 'Dead' and 'Respawn'. (Btw. you can find all behaviours also listed in the Behaviour Foldout)

Congratulation, it's done! Please save your settings and press play to see the result. If everything is correct, your creature should 'TRAVEL' from his origin position to the current TargetMovePosition of its home location. As soon as it inside the 'StopDistance', the TargetMovePosition will relocated within the 'Random Range' and your creature will follow according to its 'LEISURE' behaviour.

6 ESSENTIALS

Essentials covers all fundamental settings your creature needs for its first steps, such as a home location, basic behaviours and the general motion and pathfinding settings as well.

6.1 HOME AND BEHAVIOURS

Here you have to define the home location of your creature. This place will be its starting point and also the area where it will go to rest and to respawn after dying. Whenever your creature is not busy or for any reason not ready for action (e.g. wounded, too weak etc.) it will return to this place.

The Home Behaviours represents the proposed minimum performance requirements your creature should fulfil. Leisure is an idle behaviour if your creature is at home and have no other tasks. Travel contains the move behaviour if it's on the way.

6.2 MOTION AND PATHFINDING

The Motion and Pathfinding options contains the basic settings for physics, motion and pathfinding, which are relevant for the correct technical behaviour, such as grounded movements, surface detection etc.

6.2.1 Motion Control Type *(NEW)*

6.2.1.1 INTERNAL (IMPROVED)

ICECreatureControl works fine without additional Components and can handle a lot of situations autonomously, that's particularly important if you need a large crowd of performance friendly supporting actors, but it will definitely not covering all potential situations and for such cases ICECreatureControl supports several Unity Components to enhance the functionality.

6.2.1.2 NAVMESHAGENT (IMPROVED)

The internal pathfinding technics are sufficient for open environments, such as natural terrains or large-scaled platforms, but less helpful for closed facilities, areas covered by buildings and/or constructions or walkable surfaces with numerous obstacles. For such environments you could activate the NavMeshAgent, so that your creature will using Unity's Navigation Mesh. Activating 'Use NavMeshAgent' will automatically add the required NavMeshAgent Component to your creature and handle the complete steering. The only things you have to do is to check and adjust the 'Agent Size', the desired 'Obstacle Avoidance' and 'Path Finding' settings. Needless to say, that the NavMeshAgent Component requires a valid Navigation Mesh.

6.2.1.3 RIGIDBODY (NEW)

Basically, each moveable object in your scene should have a Rigidbody and especially if it has also a collider to detect collisions. Without Rigidbody Unity's physics engine assumes that an object is static and static (immovable) objects should consequently not have collisions with other static (immovable) objects and therefore Unity will not testing collision between such supposed static objects, which means, that at least one of the colliding objects must have a Rigidbody additional to a collider, otherwise collisions will not detected. But no rule without exception and there are absolutely some cases your creature really doesn't need a Rigidbody or even a Collider as well.

Apart from that is a Rigidbody more than a supporting element to detect collisions and you can use the physical attributes of the Rigidbody to affect the behaviour of your creature but please note, that the steering by forces is not implemented in the current version (coming soon) and using the physics with custom settings could yield funny results.

For a quick setup you can use the Preset Buttons. FULL (not implemented in the current version) will prepare Rigidbody and ICECreatureControl to control your creature in a physically realistic way. SEMI deactivates the gravity, enables the kinematic flag and allows position changes. OFF deactivates the gravity, checks the kinematic flag and restricts position changes. In all cases rotations around all axes should be blocked.

6.2.1.4 CHARACTERCONTROLLER (NEW)

In addition to the other listed controller types you can also use Unity's CharacterController component to steering your creature.

6.2.1.5 CUSTOM (NEW)

While using CUSTOM motion control, ICECreatureControl will calculate the move positions but the final movement of your creature will be handled by an external component, such as A*Pathfinding. Please add the desired pathfinding or character controller component and the associated adapter to your creature. If there is no suitable adapter available please contact the developer for further information.

6.2.2 Ground Check

Use the Ground Check option to specify the desired method to handle ground related tests and movements.

6.2.2.1 Base Offset

Base Offset defines the relative vertical displacement of the owning GameObject.

6.2.2.2 Vertical Raycast Offset (NEW)

The vertical Raycast Offset defines the height of the raycast origin related to the given creature position. By default this value is 0.5 but dependent to the terrain it could be helpful to increase this value to reach better results.

6.2.2.3 Avoid Water (NEW)

While Avoid Water is flagged your creature will avoid surfaces with the water layer.

6.2.2.4 Use Slope Limits (NEW)

While Use Slope Limits is flagged you can define the maximum slope limit your creature can use and in addition to that you can also adapt the maximum walkable slope angle, so that your creature will try to find a walkable way.

6.2.2.4.1 Max. Slope Angle *(NEW)*

Maximum slope angle your creature can use.

6.2.2.4.2 Max. Walkable Slope Angle *(NEW)*

Maximum walkable slope angle your creature will use to find a walkable way

6.2.2.5 Body Orientation (IMPROVED)

Here you can specify the vertical direction of your creature relative to the ground, which is important for movements on slanted surfaces or hilly grounds.

6.2.2.6 Advanced Body Orientation (NEW)

The Plus flag activates a more extensive method to handle the Ground Orientation for Crawler and Quadruped creature types.

6.2.2.7 Use Leaning Turn (NEW)

The Leaning Turn option allows your creature to lean into the turn. The used lean (roll) angle is related to the current speed but the angle can adjusted and limited by changing the multiplier and the maximum value. Please note, that this feature currently works well with Legacy Animations but shows no results by using the Mecanim Animation System. That's because Mecanim handles the Root Transform Rotations according to the given animation curves and ignores external changes. I'll fixed this in the course of the further integration of the Mecanim Animation System.

6.2.3 Obstacle Avoidance *(NEW)*

While Obstacle Avoidance is active your creature can detect and avoid obstacles automatically without additional pathfinding tools. Set the Obstacle Avoidance Popup to BASIC to activate this feature and add all available and/or desired obstacle layers. In addition you can adapt the Scanning Range and Angle options to improve the result and/or to optimize the performance.

6.2.3.1 Layer (NEW)

6.2.3.2 Scanning Range (NEW)

Scanning Range defines the maximum radius your creature will scan. The default values should be suitable for the majority of cases but finally the result will be dependent on the given scenario and situation e.g. if there are a lot of obstacles closely spaced you should decrease the range, but if the range is too small it could be that your creature will detect an obstacle to late and can't avoid in time.

6.2.3.3 Scanning Angle (NEW)

The Scanning Angle defines the steps in degrees within a full-circle and with this the density of the scan, a lower value will improve the result but also increase the required load, a higher value could be too imprecise.

6.2.4 Handle Gravity

Here you can activate/deactivate the internal gravity. If you are using the internal gravity you don't need additional components to handle it.

6.2.5 Handle Deadlocks

A deadlock is a situation in which your creature can't reach its desired move position. This could have several reasons, such as the typical case that its route is blocked by obstacles etc., but it could also be that its forward velocity is too high or the angular speed too low, so that your creature have - for the given situation - not the required manoeuvrability to reach the Stopping Distance to complete this move. In such cases you can observe two typical behaviours – if the path is simply blocked your creature will still walking or running on the same spot, but if the manoeuvrability is not suitable to the given stopping distance, your creature will moving in a circle or a loop. The Deadlock Handling allows your creature to detect such mistakes and you can define how your creature should react.

6.2.5.1 Test 1 - Move Distance

Move Distance defines the minimum distance which your create have to covered within the specified time. Please note that distance and interval should be suitable to the lowest forward speed of your creature.

6.2.5.2 Test 2 - Loop Range

Loop Range works analogue to the Move Distance Test but in larger dimensions. The Loop Range should be larger than the given Stopping Distance and the interval suitable to the lowest walking speed of your creature.

6.2.5.3 Test Interval

Test Interval defines the time in which your creature have to cover the Move Distance.

6.2.5.4 Max. Critical Positions

Max. Critical Positions defines the upper tolerance limit to trigger a deadlock. If this value is adjusted to zero, each critical event will directly trigger a deadlock, otherwise the limit must be reached.

6.2.5.5 Deadlock Action

Deadlock Action defines the desired procedure in cases of deadlocks.

6.2.6 Field Of View (FOV)

The Field Of View represents the maximum horizontal angle your creature can sense the surrounding environment. By default this value is adjusted to zero, which suspends the FOV restrictions and allows sensing in 360 degrees, alternative you could set the value also directly to 360 degrees, this will have the same effect except that the FOV still active and you can see the FOV gizmos. Please note, that the FOV settings will not automatically use to sense (select) a target. Please note that you have to activate the FOV flag of the Target Selection Criteria to use this feature.

6.2.6.1 Visual Range

Visual Range defines the maximum sighting distance of your creature. By adjusting this value to zero, the sighting distance will be infinite.

6.3 RUNTIME BEHAVIOUR

6.3.1 Coroutine

ICECreatureControl is using coroutines to handle all sense and preparing procedures separated from Unity's frame update. You can deactivate the coroutines for debugging or adjusting your creature settings.

6.3.2 Don't Destroy On Load

Makes that your creature will not be destroyed automatically when loading a new scene.

7 STATUS

Status represents the physical fitness of the creature, which is a dynamic attribute consists of several different settings and measurements to affect the behaviour during the runtime. You can use this component section to adapt species-typical characteristics, such as the sense and reaction time, maximum age etc. Dependent to the desired complexity and/or the given requirements, ICECreatureControl provides you a Basic and an Advanced Status. Please note that all these settings are optional.

7.1 BASICS

The Basics Status contains the essential elements your creature will need for a basic life-cycle, which allows your creature to sense and react, to receive damage and to recreate, to die and also to respawn. You can activate the Advanced Status Settings to use the extensive Status System.

7.1.1 Perception Time

Perception Time defines the necessary time to sense a situation, which in particular means, the interval in which your creature will sense the surrounding environment to detect potential interactor objects.

7.1.2 Reaction Time

Reaction Time defines the necessary time to identify a situation, which in particular means, the interval in which your creature will decide the kind of reaction and start the action through selection and activation of the most relevant target and the related behaviour.

7.1.3 Recovery Phase *(NEW)*

Recovery Phase defines a warm-up period in seconds after the spawning process in which the creature will be completely defenceless while running its respawn behaviour without any target. You can adjust this value to provide your player to detect a new spawned creature at the right time, otherwise it could be that a spawned creature appear from nowhere and will start its attack without any heads-up.

7.1.4 Respawn Delay

Respawn Delay defines the delay time in seconds until a deceased creature will added to the respawn queue. Within this time-span the creature will be visible in the scene and can be used for loot activities.

7.1.4.1 Variance Multiplier

The Variance Multiplier defines the threshold variance value, which will be used to randomize the associated interval during the runtime.

7.1.5 Use Corpse *(NEW)*

If Use Corpse is flagged you can assign a GameObject which will be used if your creature dies (e.g. a Ragdoll Object of your creature which will be used instead of the original model). The corpse object have to be a prefab which will be instantiate automatically if your creature dies.

7.1.6 Fitness Multiplier

The Fitness Multiplier defines the influence ratio of the fitness value on the associated interval.

7.1.7 Recreation Limit

The Recreation Limit defines the fitness threshold value at which your creature will return automatically to its home location to recreate its fitness. If this value is adjusted to zero, the Recreation Limit will ignored, otherwise the home target will be handled with the highest priority in cases the value goes below to the limit.

7.2 ADVANCED

Dependent to the desired complexity and the given requirements, ICECreatureControl provides you additional to the Basic Settings an enhanced Status System. While using the Basic Status, the fitness of your creature will be always identical with the health value and finally just the antipode of damage – increasing the damage will directly reduce the health and consequently the Fitness as well. By using the Advanced Status the procedure to evaluate the Fitness is affected by several different initial values, indicators, multipliers and random variances.

7.2.1 Trophic Level *(NEW)*

The Trophic Levels represents the type of food your creature prefer. This value will be used to affect automatically and randomized procedures (e.g. wizards) and runtime behaviours of your creature.

7.2.1.1 Herbivores

Herbivores are animals which only eat plant material. This means leaves, flowers, fruits or even wood. Sheep, horses, rabbits and snails are well known examples of herbivores which eat grass and leaves. A parrot, however, which eats fruits and nuts can also be called as herbivore.

7.2.1.2 Omnivores

Omnivores eat both plants and meat. Chickens are omnivores. They eat seeds, but they can also eat worms. Human beings are also omnivores, although some people choose not to eat meat. These people are called vegetarians.

7.2.1.3 Carnivores

Carnivores eat meat. A carnivore is a predator because it has to find and catch its prey. Some carnivores, such as wolves, hunt in a group called a pack. They move silently and slowly to form a circle around their prey before they attack.

7.2.2 Use Aging

The ‘Use Aging’ flag activates the aging process, which will limited the life-cycle of your creature and will have additional influence to the Fitness as well. Please note, that a limited life-cycle consequently means that your creature will die at the end of the cycle.

7.2.2.1 Current Age

Current Age represents the age of your creature at runtime. You can adjust this value to define an initial age or you can modify the value also during the runtime. Please note, that the effective time data are in seconds, the use of minutes is for the editor mask only.

7.2.2.2 Maximum Age

Maximum Age defines the maximum length of the life-cycle and consequently the time of death as well. Please note, that the effective time data are in seconds, the use of minutes is for the editor mask only.

7.2.3 Use Temperature

The 'Use Temperature' flag activates the thermal sensation of your creature, which will have additional influence to the fitness and finally to the behaviour as well. While 'Use Temperature' is active, your creature will receive and evaluate temperature values based on the environment data of the creature register, which you can easiest combine with your own scripts, third party products or external data sources. You can find the ICECreatureUniStormAdapter attached to your ICECreatureControl package (please note, that this adapter requires a valid licence of UniStorm)

7.2.3.1 Temperature Scale

Temperature Scale defines the desired measuring unit FAHRENHEIT or CELSIUS in degrees

7.2.3.2 Minimum Temperature

Minimum Temperature defines the lowest temperature value your creature can survive.

7.2.3.3 Maximum Temperature

Maximum Temperature defines the highest temperature value your creature can survive.

7.2.3.4 Comfort Temperature

Comfort Temperature defines the ideal temperature value for your creature.

7.2.3.5 Temperature

Temperature represents the current environment temperature, which your creature receives via the environment data of the creature register. This editor field is for testing only, the value will overwrite during the runtime.

7.2.4 Use Armour

The 'Use Armour' flag activates the Armour of your creature. Armour works as a buffer by absorbing incoming damage values. As long as the armour value is larger zero the damage value will remain unaffected.

7.2.4.1 Armour

Armour defines the initial armour value in percent and represents the armour during the runtime as well. You can adapt this value to customize the initial armour.

7.2.5 Use Shelter

While using 'Use Shelter' your creature will be protected against environment influences (e.g. rain, storm, low temperatures etc.) or attacks while enter a safe area which you can define by a tagged trigger. If your creature enter such a trigger the IsSheltered flag will be true and will reset to false again in cases it leaves such areas.

7.2.6 Use Indoor

While using 'Use Indoor' your creature will be protected against environment influences (e.g. rain, storm, low temperatures etc.) or attacks while enter a safe area which you can define by a tagged trigger. If your creature enter such a trigger the IsIndoor flag will be true and will reset to false again in cases it leaves such areas.

7.2.7 Odour *(NEW)*

Odour represents volatilized chemical compounds that other creature could perceive by their sense of olfaction.

7.2.7.1 Intensity

7.2.7.2 Max. Range

7.2.8 Influence Indicators

Influence Indicators represents all status attributes which can be affected by direct influences, based on internal activities or external forces as well. You can modify this Indicators to customize initial values or to test the status settings. By default all this indicators are adjusted to zero.

7.2.8.1 Damage

The Damage attribute represents the effective damage level of your creature in percent. Depending on the associated multiplier the value will affect default indicators.

7.2.8.2 Stress

The Stress attribute represents the effective stress level of your creature in percent. Depending on the associated multiplier the value will affect default indicators.

7.2.8.3 Debility

The Debility attribute represents the effective debility level of your creature in percent. Depending on the associated multiplier the value will affect default indicators.

7.2.8.4 Hunger

The Hunger attribute represents the effective hunger level of your creature in percent. Depending on the associated multiplier the value will affect default indicators.

7.2.8.5 Thirst

The Thirst attribute represents the effective hunger level of your creature in percent. Depending on the associated multiplier the value will affect default indicators.

7.2.9 Sensory Indicators *(NEW)*

Sensory Indicators represents the perceptive capabilities of your creatures. By default all values are adjusted to 100 % but you can adapt the values manual to restrict specific senses. Also you can adjust the multiplier to restrict senses dynamically during the runtime.

7.2.9.1 Visual

7.2.9.2 Auditory

7.2.9.3 Olfactory

7.2.9.4 Gustatory

7.2.9.5 Tactile

7.2.10 Vital Indicators

Vital Indicators represents calculated status attributes which will be indirect affected by the influence indicators and the associated multipliers. By default the reference values are adjusted to 100, but

you can modify this values as desired to adapt the indicators to your existing environment. The calculated result will be expressed in percent.

7.2.10.1 Fitness

7.2.10.2 Health

7.2.10.3 Stamina

7.2.10.4 Power

7.2.11 Character Indicators *(NEW)*

Character Indicators represents calculated status attributes which will be indirect affected by the influence indicators and the associated multipliers. By default the reference values are adjusted to 100, but you can modify this values as desired to adapt the indicators to your existing environment. The calculated result will be expressed in percent.

7.2.11.1 Aggressivity

7.2.11.2 Anxiety

7.2.11.3 Experience

7.2.11.4 Nosiness

7.2.12 Dynamic Influences

7.3 MEMORY *(NEW)*

The Memory represents different kind of memorizations and empirical values your creature can use to remember e.g. specific situations, other creatures or locations but also your player. This feature is currently under construction and will be full available in one of the next versions.

7.4 INVENTORY *(NEW)*

The Inventory represents a list of items your creature (but also other ICE objects) can have with it. You can define an empty list, your creature can fill during the runtime or you can also define default items, your creature can lose while looting by another creature or your player but also while distribute items (e.g. sow seed or deliver a newspaper or pizza etc.)

7.4.1 Slots

Slots represents repositories your creatures but also other ICE objects as well can use to store items. To adapt the desired slots just set the maximum number and increase or decrease the slider or press the RES Button to remove all slots.

7.4.1.1 Slot

A slot represents a repository for an item type. By default a slot and especially the deposited items are virtual constructs without existing GameObjects, in this case the given amount is just a theoretical value which represents imaginary items. But you can also specify a parent object by using the Slot Popup which represents the creatures' hierarchy, so that your creature can use the defined parent as a real handle (e.g. hand, holster, chest holder etc.) and as long as the item amount is larger than zero the defined item will be represented as an instantiated object, assigned to the given handle and visible and detectable for other creatures. This allows you to equip or re-equip your creature during the runtime by using the inventory settings of the behaviour rules.

7.4.1.1.1 Item

Item represents the type of a stored object. Basically an item can be each GameObject or Prefab but in each case it must be listed as reference in the Creature Register and requires the ICECreatureItem script, if both is given the item will be listed in the Item Popup. Furthermore you can activate the EXCL Button, to mark the item as exclusive and reserved for the given slot. By default EXCL is deactivated and the slot is open for each kind of items. If you assign now an item to such an open slot the amount will automatically increase to 1 and the slot will be open again if the amount will reset to zero. By activate the exclusive flag the slot stays in each case reserved for the assigned item type independent of the amount. If the DOD Button is flagged all available items will be dropped in cases the inventory owner dies or its object will be destroyed.

7.4.1.1.2 Amount

Amount represents the current number of items. You can adjust the maximum number to restrict the maximum capacity.

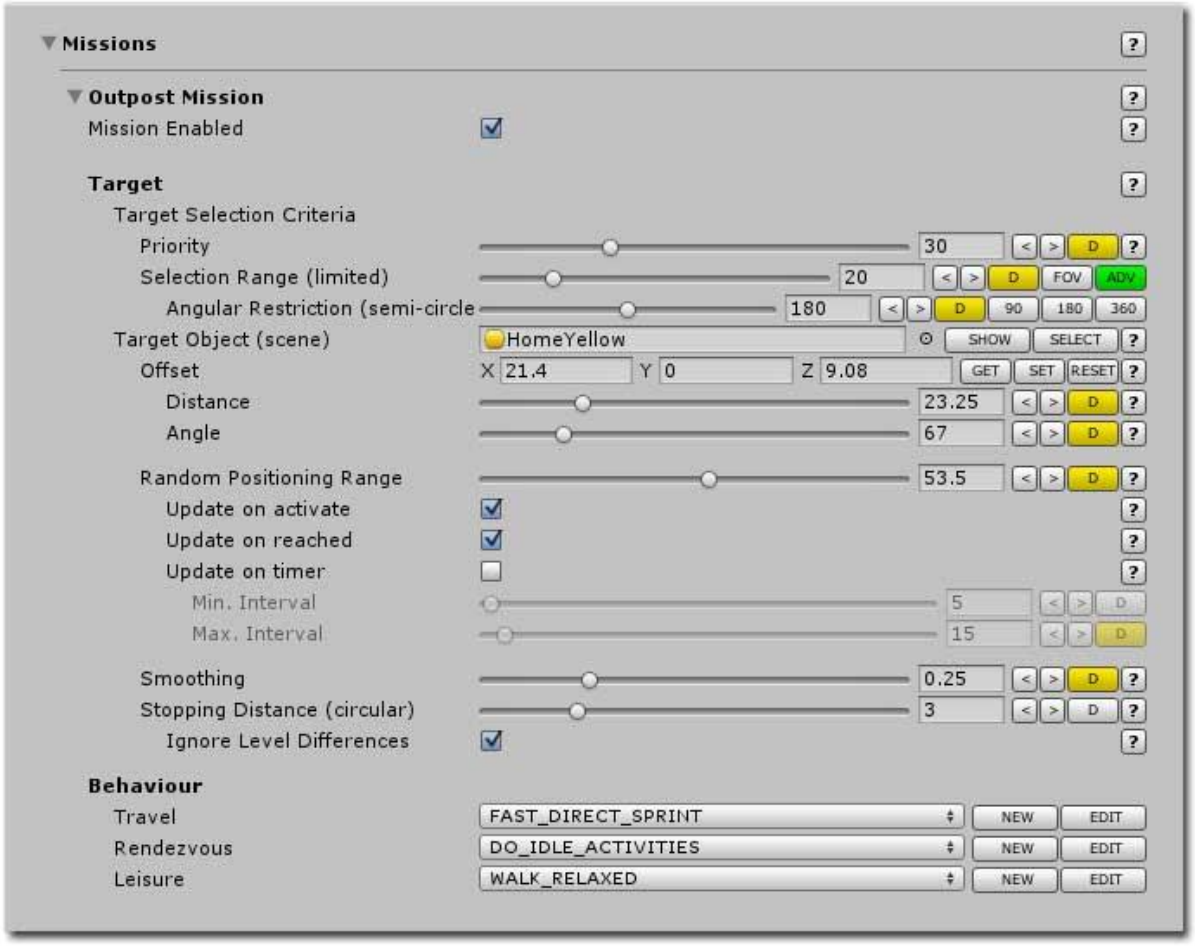
(See also: Behaviour Rules -> Inventory)

8 MISSIONS

8.1 OUTPOST MISSION

The Outpost Mission is absolutely boring for any high motivated creature and as expected the job-description is really short: go home and wait for action! But you could enlarge the Random Positioning Range to give your creature a larger scope, add additional rules for LEISURE and RENDEZVOUS and your creature will spend its idle time with some leisure activities. Furthermore you could use the Pool Management of the Creature Register to generate some clones, so that your creature isn't alone. On this way you could use the Outpost Mission to populate a village, to setup a camp with soldiers, some animals for a farm or a pack of wolves somewhere in a forest etc.

The Outpost object could be any reachable object in the scene and btw. movable objects as well. Adapt the distances so that your creature feel comfortable, have sufficient space for his idle activities and don't blunder into a conflict with the object size.



8.2 ESCORT MISSION

The Escort Mission offers your creature more entertainment, but the job-description is also simple: Your creature have to search and follow the leader wherever he is and goes! You could use this mission to specify a faithful and brave companion to your player or to any another NPC as well. You could also combine this mission with other Targets, such as the Patrol Mission, to use your creature as a guide which can show your player secret places.

The Leader object could be any reachable object in the scene. Adapt the distances so that your creature have enough space for his activities and don't blunder into a conflict with the leader moves.

8.3 PATROL MISSION

The Patrol Mission represents a typical Waypoint Scenario and is - up to now - the most varied standard task for your creature, so the job-description is a little bit more comprehensive: Find out and TRAVEL to the nearest waypoint. If you are reach the Max. Range (Random Positioning Range + Stopping Distance) and it's a transit-point, ignore LEISURE and RENDEZVOUS, find out the next waypoint accordind to the given path-type and start to PATROL. If it's not a transit-point, follow the LEISURE rules until reaching the RENDEZVOUS position (TargetMovePosition + Stopping Distance) and execute the RENDEZVOUS instructions over the given period of time (Duration Of Stay). Afterwards find out the next waypoint accordind to the given path-type and start to PATROL. Repeat these instructions for each waypoint.

To prepare a Patrol Mission you can add single waypoints, which could be any reachable objects in your scene, or you can add a complete waypoint group, which is a parent object with its children. By using this way, the children will be used as waypoints, while the parent will be ignored.

8.3.1 Patrol Enabled

The Enabled flag allows you to activate or deactivate the complete Mission, without losing the data. As long as a Mission is disabled, the creature will ignore them during the runtime. You could use this feature also by your own scripts to manipulate the gameplay.

8.3.2 Waypoint Order Type

The Order Type defines the desired sequence in which your creature have to visit the single waypoints. Please consider, that your creature will always starts with the nearest waypoint, so if you want that it will start with a special one you should place it in the near. By default the cycle sequence is ordered in ascending order, activate the DESC button to change it to descending.

8.3.3 Waypoint

A Patrol Mission can basically have any number of waypoints. Each waypoint represents a separate target and will also be listed with all target features in the inspector. You can move each waypoint item within the list up or down to change the order or you can delete completely as well. Furthermore, you can activate and deactivate each single waypoint as desired, in such a case, your creature will skip deactivated waypoints to visit the next ones.

8.3.4 Use Custom Behaviour

The 'Use Custom Behaviour' flag allows you to overwrite the default patrol behaviour rules for the selected waypoint. Activate the 'Custom Behaviour' flag to define your additional behaviour rules. Please note, that these rules will be used for the selected waypoint only.

9 INTERACTIONS

Additional to the standard situations defined in the home and mission settings, you can teach your creature to interact with several other objects in your scene, such as the Player Character, other NPCs and Bots, doors and hatches, melee weapons etc. The Interaction Settings provides you to design complex interaction scenarios with each object in your scene.

To using the interaction system you have to add one or more Interactors. An Interactor represents another GameObject as potential Target for your creature and contains a set of conditions and instructions to define the desired behaviour during a meeting. By default interactors are neutral, they could be best friends or deadly enemies as well and basically interactors can be everything your creature has to interact with it, such as a football your creature has to play with it or a door, which it has to destroy.

After adding a new interactor you will see primarily the familiar target settings as they will be used in the home and mission settings, but instead of the object field the interactor settings provides a popup to select the target game object. That's because, interactors are normally OOIs (objects of interest), which could be also interesting for other objects of interest, such as the Player Character or other NPCs and therefore such objects have to be registered in the creature register to provide a

quick access to relevant data during the runtime. So you have to use the popup to add the desired interactor.

But the pivotal difference to the home and missions settings is, that you can define an arbitrary number of additional interactor rules, which allows you to overwrite the initial target related selection and position settings for each rule. By using this feature you could define a nearly endless number of conditions and behaviours for each imaginable situation, but in the majority of cases 3-5 additional rules will be absolutely sufficient to fulfil the desired requirements.

9.1 Interactor

An Interactor represents another GameObject as potential Target for your creature and contains a set of conditions and instructions to define the desired behaviour during a meeting. By default interactors are neutral, they could be best friends or deadly enemies as well and basically interactors can be everything your creature has to interact with it, such as a football your creature has to play with it or a door, which it has to destroy.

9.2 Interactor Enabled

The Interactor Enabled flag allows you to activate or deactivate the Interactor, without losing the data. As long as an Interactor is disabled, the creature will ignore it during the runtime. You could use this feature also by your own scripts to manipulate the gameplay.

9.3 Interactor Target

Interactors are mostly objects of interest, which will be normally interesting for other objects of interest as well, such as the Player Character or other NPCs and therefore such objects have to be registered in the creature register to provide a quick access to relevant data during the runtime. For this reason, you have to use the popup to select your desired interactor. If your desired interactor isn't listed currently, switch to your creature register to add the interactor. Please note, that your interactor object doesn't need additional scripts to be listed in the register, unless your interactor is a player character or a NPCs, which is not controlled by ICECreatureControl, in such a case you should add the ICECreatureResident Script to your interactor, which will handle the registration and deregistration procedures during the runtime.

10 ENVIRONMENT

Complementary to the HOME, MISSIONS and INTERACTION features, which are all dealing with the interaction between your creature and other GameObjects, the Environment section handles the interaction with the surrounding environment. The current Environment System provides your creature two different abilities to sense its surrounding space – SURFACES and COLLISIONS detection.

10.1 SURFACES

The Surface Rules defines the reaction to the specified textures. You could use this feature for example to handle footstep sounds and/or footprint effects, but you could also start explosion effects to simulate a minefield, or dust effects for a dessert, or you could define textures as fertile soil, where your creature can appease its hunger or thirst etc. Please make sure, that all used Textures will have a unique name to avoid misfeatures.

10.1.1 Scan Interval (1.1)

The Scan Interval value defines the desired time period in seconds to check the current ground texture. It's not required and recommended to do this scan in each frame (adjusted to zero), by default the creature will do this scan each second, which should be suitable for the most cases. You should increase this value if you are using large herds or crowds.

10.1.2 Surface Rule

10.1.2.1 Name

Name defines just the display name of the rule and have further impact. You can rename it to get a more comprehensible and context related term.

10.1.2.2 Interval

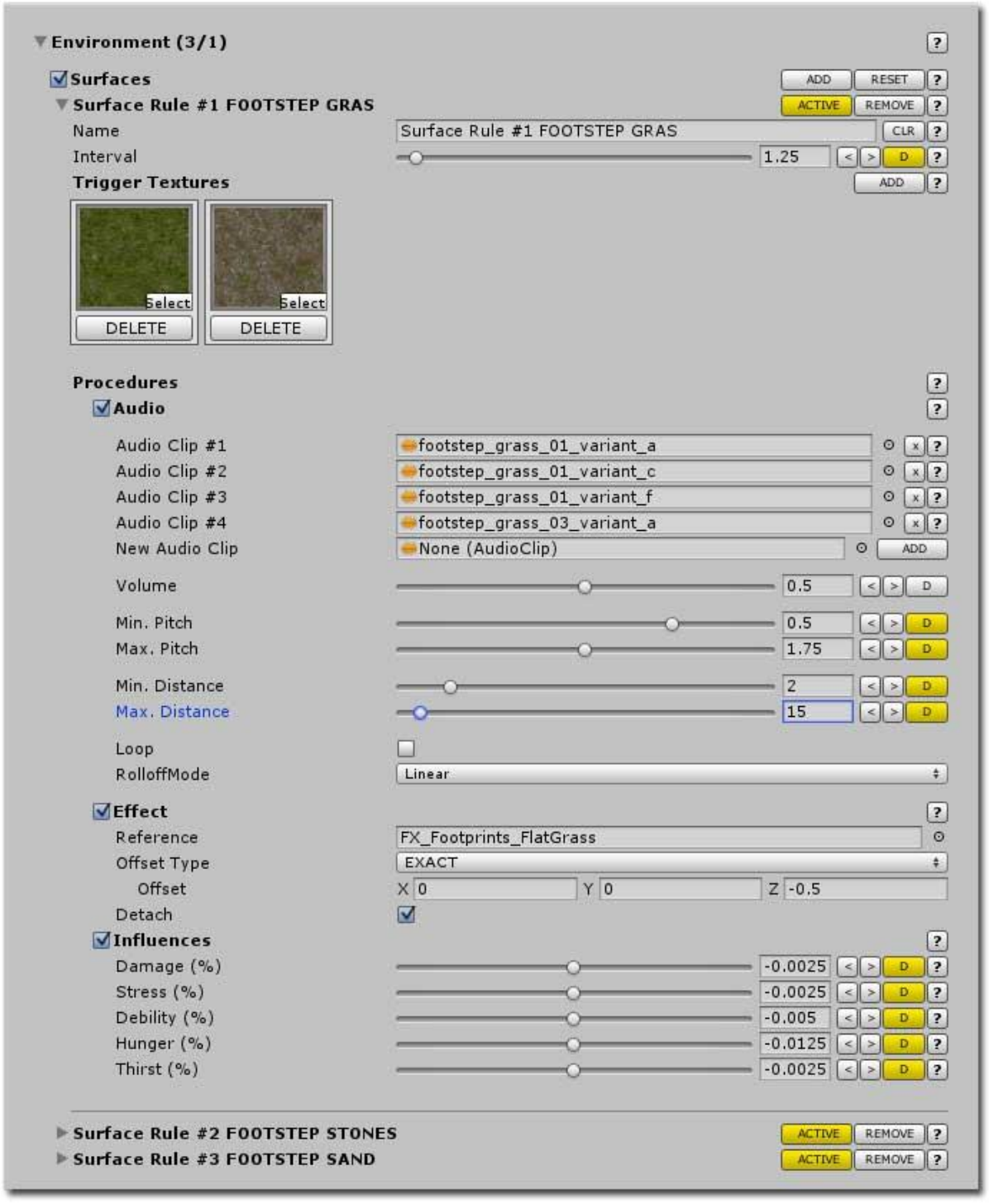
The Interval value defines the desired repeating time period in seconds.

10.1.2.3 Trigger Textures

The Trigger Textures specifies the conditions to activate the assigned procedures. As soon as your creature comes in contact with one of the defined textures, the specified procedures will start. Use the Interval settings to adjust the desired repeating interval. Please make sure, that all used Textures will have a unique name to avoid misfeatures.

10.1.2.4 Procedures

Each Surface Rule can initiate several procedures, in cases the given trigger conditions are fulfilled. You can adapt the Procedure setting to define the desired behaviour. You could use the procedure settings for example to define footstep sounds and/or footprint effects, but you could also start explosion effects to simulate a minefield, or dust effects for a dessert, or you could define textures as fertile soil, where your creature can appease its hunger or thirst etc.



10.2 COLLISIONS

The Collision Rules defines the reaction to detected collisions. You could use this feature for example to adjust the damage if your creature was hit by a bullet, or comes in contact with a melee weapon or a spike wall.

10.2.1.1 Name

Name defines just the display name of the rule and have further impact. You can rename it to get a more comprehensible and context related term.

10.2.1.2 Type

Type specifies the condition type, which will be using to filter the incoming collisions. Currently you can filter incoming collision objects by TAG, LAYER or TAG&LAYER Tag

10.2.2 Tag

10.2.3 Layer

10.2.4 Body Part

11 REGISTER – ICECREATUREREGISTER

The Creature Register is an additional component, which will installed automatically as soon as you place your first ICE CC creature on your scene. This component serves as a central population register and pool manager for all your creature related objects, to provide an easy management and performance friendly interactions. During the runtime ICE objects will join and leave the register automatically, but you can register also your player character or any other kind of GameObject which should active interact with your virtual wildlife. Simply add one of the target Script to the specific GameObject and press update - that's all.

11.1 OPTIONS

Options contains several optional features which could be helpful to you to organize your project and to reach the desired goals without custom scripts, but in any case you are free to implement also your own solutions to handle these functions.

11.1.1 Use Hierarchy Management

By using the Hierarchy Management the ICECreatureRegister makes sure that your scene stay clean and tidy during the runtime. If UseHierarchyManagement is flagged all spawned Objects will be sorted according to the given structure. You are free to modify the given structure as desired to adapt it to your project.

11.1.1.1 Root

By default the root node will be the CreatureRegister element but you can define also your own object or deactivate this node to arrange all groups to the top level of your scene hierarchy.

11.1.1.2 Player

The Players node contains all GameObjects who using the ICECreaturePlayer script for their registration (see also ICECreaturePlayer).

11.1.1.3 Creature

The Creatures node contains all GameObjects who using the ICECreatureControl script for their registration (see also ICECreatureControl).

11.1.1.4 Items

The Items node contains all GameObjects who using the ICECreatureItem script for their registration (see also ICECreatureItem).

11.1.1.5 Locations

The Locations node contains all GameObjects who using the ICECreatureLocation script for their registration (see also ICECreatureLocation).

11.1.1.6 Waypoints

The Waypoints node contains all GameObjects who using the ICECreatureWaypoint script for their registration (see also ICECreatureWaypoint).

11.1.1.7 Marker

The Markers node contains all GameObjects who using the ICECreatureMarker script for their registration (see also ICECreatureMarker).

11.1.1.8 Other

The Other node contains all other GameObjects without specific ICE scripts.

11.1.1.9 Reorganize Hierarchy

Press Reorganize Hierarchy to clean up your scene by sorting all listed Reference Objects according to the given structure.

11.1.2 Use Pool Management

By using the Pool Management the Register can handle the population of your creatures and all your other related objects such as locations, waypoints and items etc. as well. While UsePoolManagement is flagged you can activate the POOL functions for each reference object to define the desired spawn and respawn settings. The Pool Management is an optional feature, you are free to handle it also by your own scripts or third party products.

11.1.2.1 Spawn Ground Check

Use Spawn Ground Check to define how the ground level will be detected during a spawning process.

11.1.2.2 Ground Layer

11.1.2.3 Spawn Obstacle Check

Use Spawn Ground Check to define how the ground level will be detected during a spawning process.

11.1.2.4 Obstacle Layer

Use Spawn Obstacle Check to define the obstacle layers which should be avoided during a spawning process.

11.1.3 Use Scene Management

While UseSceneManagement is flagged

11.1.3.1 Don't Destroy On Load

While DontDestroyOnLoad is flagged the CreatureRegister will not be destroyed automatically when loading a new scene.

When loading a new level all objects in the scene are destroyed, then the objects in the new level are loaded. In order to preserve an object during level loading call DontDestroyOnLoad on it. If the object is a component or game object then its entire transform hierarchy will not be destroyed either.

11.1.3.2 Random Seed

Random Seed defines the seed for the random number generator.

The random number generator is not truly random but produces numbers in a preset sequence (the values in the sequence "jump" around the range in such a way that they appear random for most purposes).

The point in the sequence where a particular run of pseudo-random values begins is selected using an integer called the *seed* value. The seed is normally set from some arbitrary value like the system clock before the random number functions are used. This prevents the same run of values from occurring each time a game is played and thus avoids predictable gameplay. However, it is

sometimes useful to produce the same run of pseudo-random values on demand by setting the seed yourself.

You might set your own seed to make sure that the same "random" pattern is produced each time the game is played.

11.1.4 Use Debug

Activate the Debug feature to show the Reference, Clones and SpawnPoint Gizmos.

11.1.4.1 Draw Selected Only

While 'Draw Selected Only' is flagged the Gizmos will be only drawn when their GameObjects are selected.

11.1.4.2 References

Use the Reference settings to adapt the colour of the REFERENCE gizmos, also you can activate/deactivate TEXT to display or hide the description and use ENABLED to activate or deactivate the REFERENCE gizmos.

11.1.4.3 Clones

Use the Clones settings to adapt the colour of the CLONE gizmos, also you can activate/deactivate TEXT to display or hide the description and use ENABLED to activate or deactivate the CLONE gizmos.

11.1.4.4 SpawnPoints

Use the SpawnPoints settings to adapt the colour of the SPAWNPOINT gizmos, also you can activate/deactivate TEXT to display or hide the description and use ENABLED to activate or deactivate the SPAWNPOINT gizmos.

11.2 Reference Objects

Reference Objects represents a list with all different types of GameObjects your creatures should interact with it during the runtime (e.g. your Player, other creatures and NPCs, locations and waypoints, loot items etc.). In Editor Mode this list provides a Popup with all object names while using the target access by name, also you can use the internal pool management of the register to adapt the spawning and population management. During the runtime the list contains all spawned objects, provides a quick and performance friendly access to the superior groups and also to each single element, handles the population management and the communication between groups and objects. Therefore you should add at least one reference object of each GameObject which you want to use as potential target or interaction object. Basically you can add each desired GameObject (scene objects or prefabs) as a reference, which will be listed also as potential target in the selection popup while using the target access by name but please consider that all objects have to handle their registration and deregistration during the runtime alone, otherwise it could be that a target will not detect correctly and your creatures will ignore it. ICE provides several target scripts (e.g. Player, Location, Waypoints, Marker and Items etc.) to handle these registration procedures and you should classify all unknown objects by assigning one of these scripts to your object according to its function (Tip: you can use the 'C' buttons to add the desired script)

11.2.1 Add Reference Object

Use ‘Add Reference Object’ to add a new reference object to the register. Each Reference Object represents a group of objects with the same characteristics, such as a specific species or item classes etc.

11.2.2 Reference Object Group

A ‘Reference Object Group’ represents a group of objects with the same characteristics, such as a specific species, an item class or a location etc. which should be used as potential target during the runtime. Basically a Reference Object Group based on a single reference object which will be used as original for all runtime initialized clones. Such a reference object could be each GameObject but the ICE framework provides several types of harmonised objects to increase the functionality and to optimize the interplay but also to simplify the usability.

11.2.2.1 Reference Object

The ‘Reference Object’ represents the prototype and will be used as original for all runtime initialized clones. The ‘Reference Object’ should be a Prefab, to make sure that it will not be destroyed during the runtime but you are free to use also scene objects, but please consider that this could trigger a couple of problems if it got lost during the runtime.

11.2.2.2 Pool Management

11.2.2.2.1 Max. Spawn Objects

‘Max. Spawn Objects’ defines the maximum number of objects which should be spawn during the runtime.

11.2.2.2.1.1 INITIAL button

Activate ‘INITIAL’ to spawn all objects on start, otherwise they will spawn according to the given spawn interval.

11.2.2.2.1.2 Initial Spawn Priority

If INITIAL is flagged you can adjust the ‘Spawn Priority’ to specify in which order the given reference groups should spawn their clones. This could be important to make sure that locations and waypoints are already exists before spawning your creatures.

11.2.2.2.2 Spawn Interval

Adapt the ‘Spawn Interval’ values to define the minimum and maximum range in seconds in which the objects should be spawn.

11.2.2.2.3 Random Size

To force a more natural scenario you can activate ‘Random Size’ to randomize the size of your initialized objects.

11.2.2.2.3.1 Size Variance

Use ‘Size Variance’ to define the minimum and maximum limits.

11.2.2.2.4 Soft Respawn

Activate ‘Soft Respawn’ to reuse already initialized objects during the runtime. If ‘Soft Respawn’ is flagged already initialized but unused objects will be reset and reused without destroying and creating new instances.

11.2.2.2.5 Use Hierarchy Group

Activate 'Use Hierarchy Group' to use an own group node for all spawned instances of the reference object.

11.2.2.2.5.1 Custom Hierarchy Group

By default the Hierarchy Group Node will be a child of the given reference node, but you can also define a custom GameObject as group node, in this case the instances will only initialized if the defined node is active and available inside your scene and they will be also hidden or removed if the custom group node will be deleted or deactivated. You can use this feature for your zone management to steering the visibility of initialized items, just by activate or deactivate or load or unload the custom group node.

11.2.2.2.6 Spawn Points

Spawn Points represents potential locations your creatures and objects will be spawned during the runtime. By default your creatures will use its HOME location as spawn point and all the other types the position of their reference objects, but you are free to define as much points as you want; if multiple points available the selection will be randomized.

11.2.2.2.6.1 SpawnPoint

11.2.2.2.6.1.1 Random Range

11.2.2.2.7 Add Spawn Point

11.2.3 ICECreatureControl (CC)

Creature Object represents an ICECreatureControl controlled character. Press the "CC" button to add the required ICECreatureControl script to your creature.

11.2.4 ICECreaturePlayer (CP)

Player Object represents an active ICECreaturePlayer character. ICECreaturePlayer is a small script to handle the registration process of your player during the runtime. Just add this script to your player character and all your ICECreatureControl characters can percept your player and interact with him (see also chapter 9 Interactions).

Press the "CP" button to add an ICECreaturePlayer script to your player. Please note: if you are using an obsolete version of a player adapter script such as ICECreatureUFPSPlayer, ICECreatureRFPSPAdapter or ICECreatureUnitZAdapter it could be necessary to add the ICECreaturePlayer script too.

11.2.5 ICECreatureLocation (CL)

Location Object represents a location target. Press the "CL" button to add the required ICECreatureLocation script to your unknown object.

11.2.6 ICECreatureWaypoint (CW)

Waypoint Object represents a waypoint target. Press the "CW" button to add the required ICECreatureWaypoint script to your unknown object.

11.2.7 ICECreatureMarker (CM)

Marker Object represents a marker target. Press the "CM" button to add the required ICECreatureMarker script to your unknown object.

11.2.8 ICECreatureItem (CI)

Item Object represents a item target. Press the “CI” button to add the required ICECreatureItem script to your unknown object.

12 TARGETS

12.1 ICECreatureControl

12.2 ICECreaturePlayer

12.3 ICECreatureItem

12.4 ICECreatureLocation

12.5 ICECreatureWaypoint

12.6 ICECreatureMarker

13 UTILITIES

13.1 ICECREATURECONTROLDEBUG SCRIPT

The debug options are part of the General Settings and provide several tools to monitoring the movement and behaviour of your creature, so it's easier to you to detect and avoid misfeature and nonconformities, such as potential deadlocks, collisions etc.

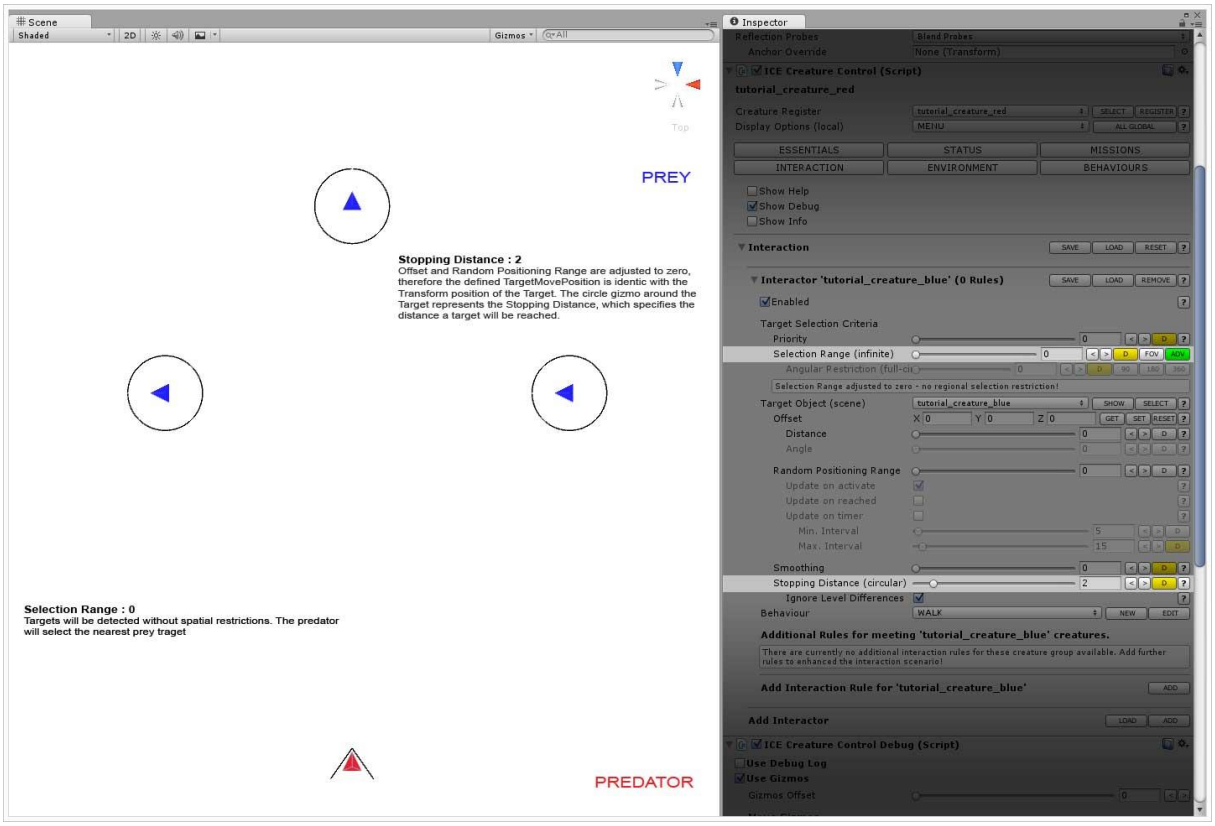
13.1.1 Path and Destination Pointer

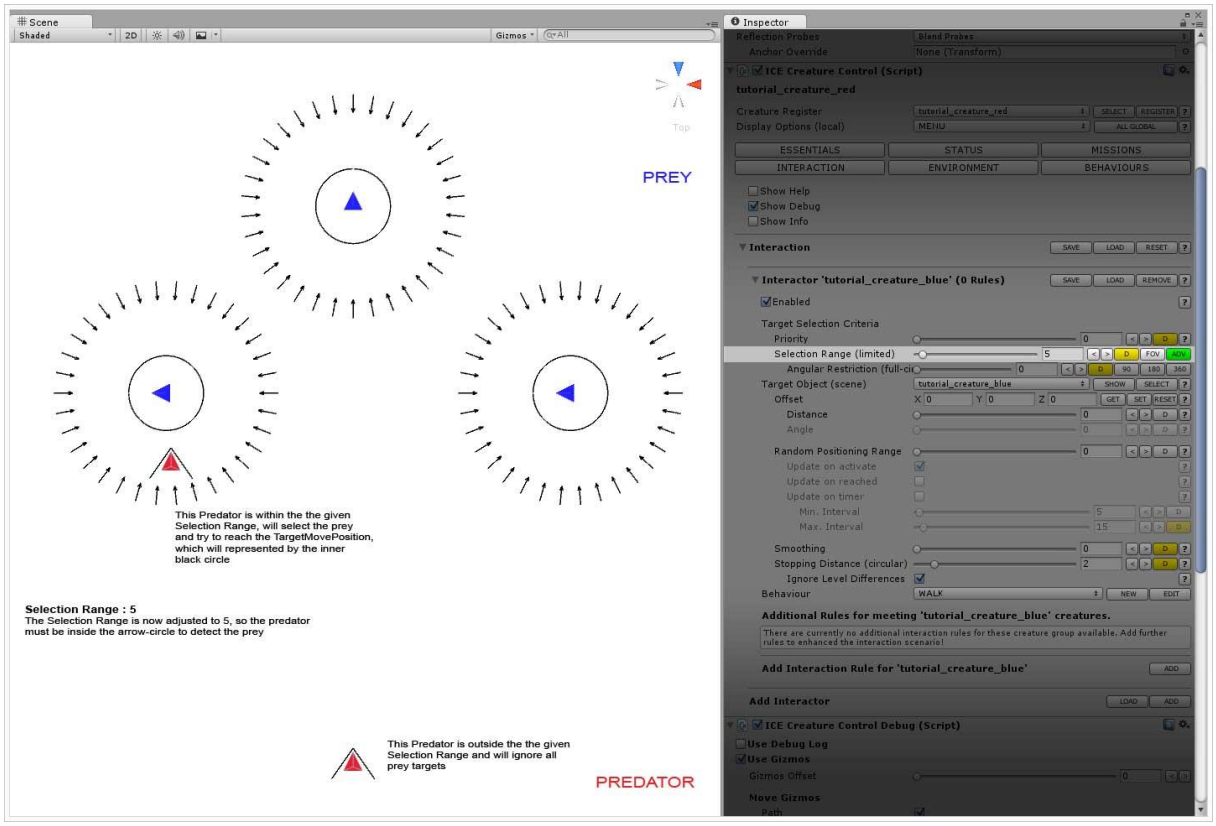
The Path and Destination Pointer are runtime features to display the current path and the final move position of your creature. The Pointer are primitive objects, which you can customize, so that you can easily tell them apart.

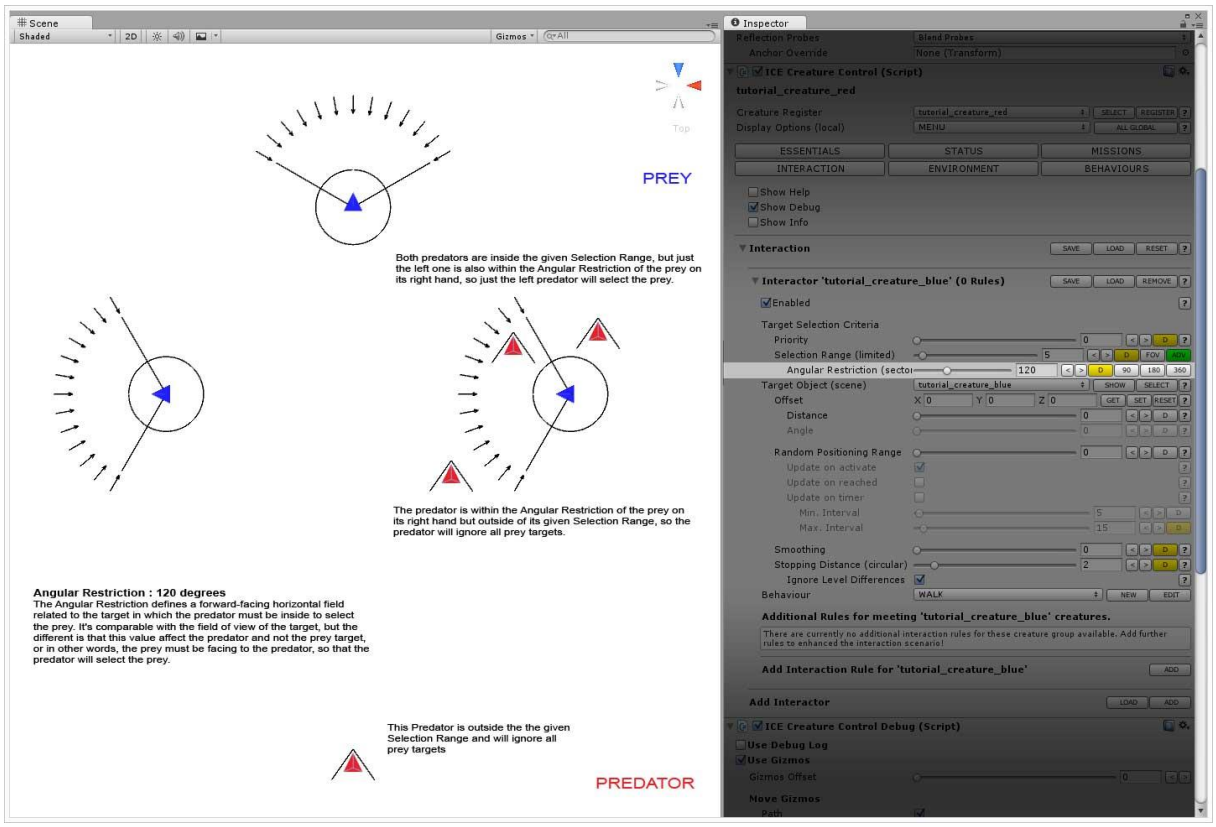
13.1.2 Debug Log

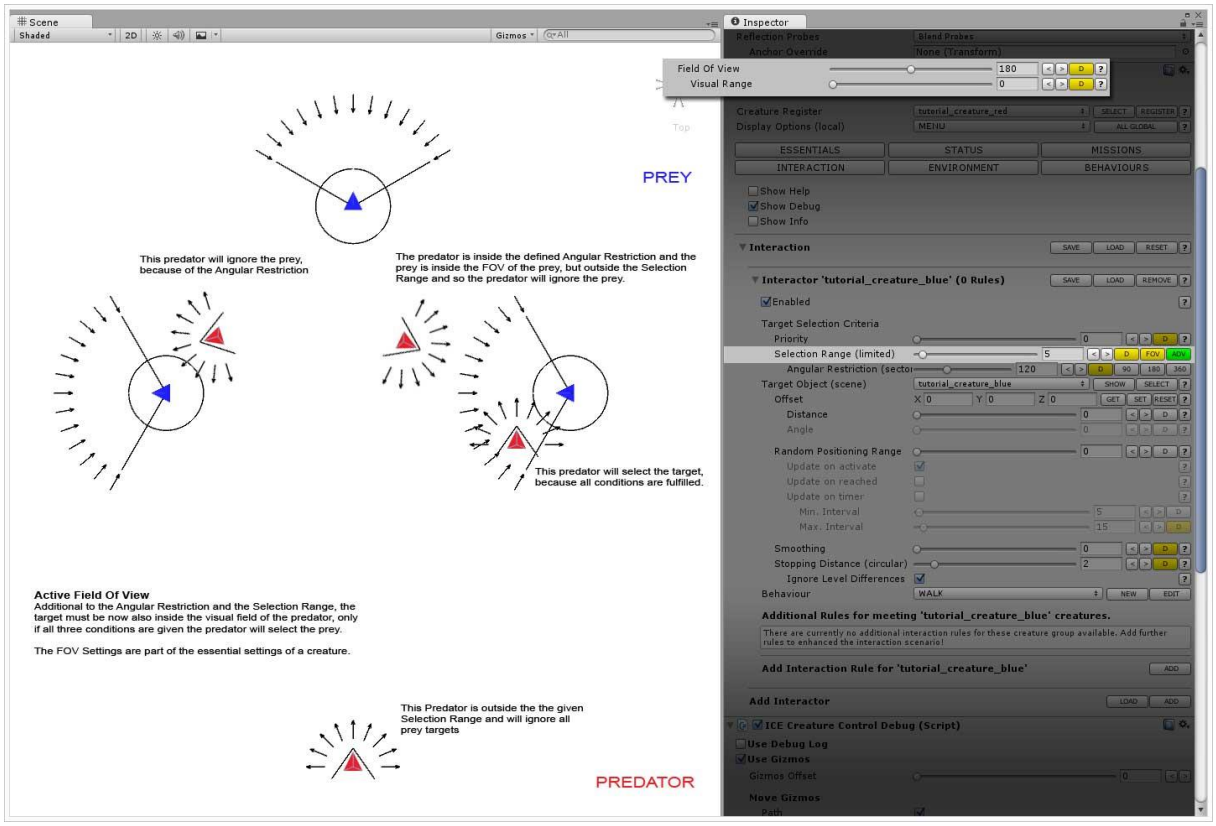
The debug information informs you about the status of your creature, so you can monitoring the active targets, behaviours and behaviour rules.

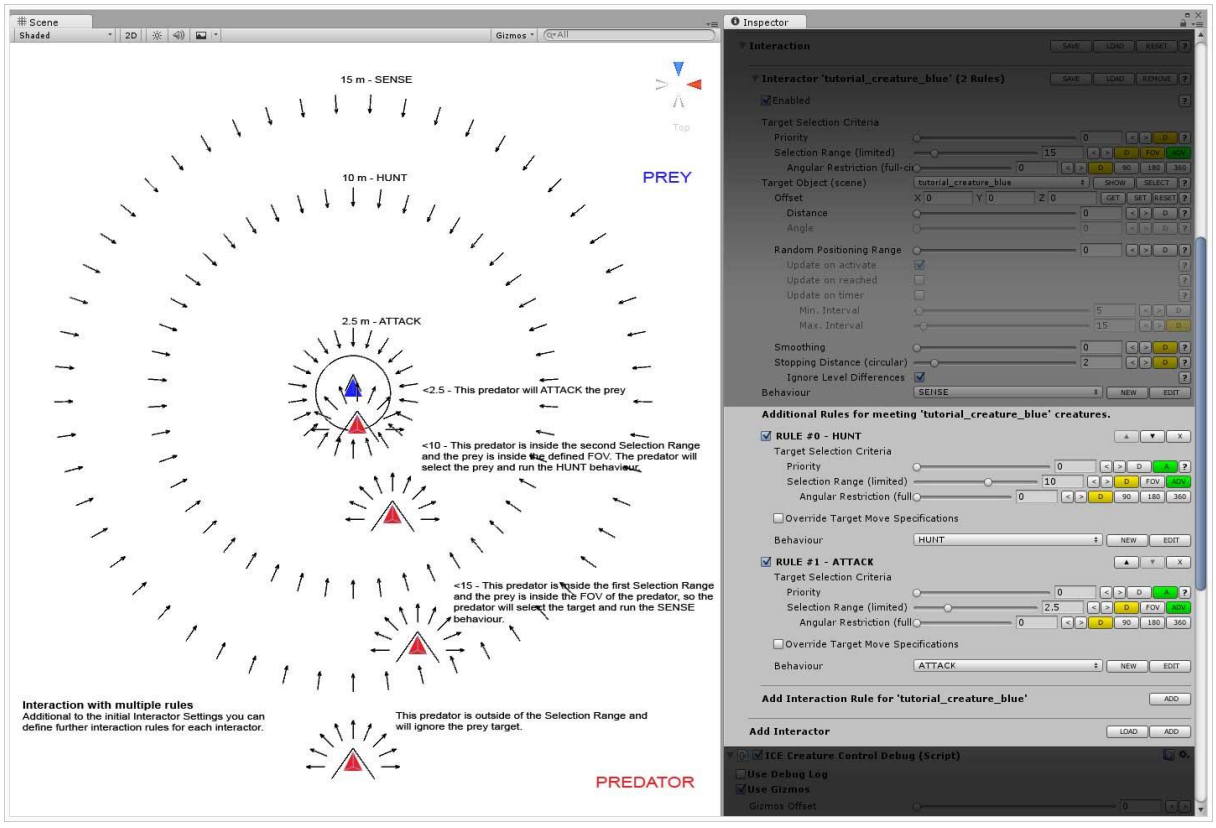
13.1.3 Gizmos




PREY
This Predator is within the the given Selection Range, will select the prey and try to reach the TargetMovePosition, which will represented by the inner black circle
Selection Range : 5
The Selection Range is now adjusted to 5, so the predator must be inside the arrow-circle to detect the prey
This Predator is outside the the given Selection Range and will ignore all prey targets
PREDATOR


PREY
Both predators are inside the given Selection Range, but just the left one is also within the Angular Restriction of the prey on its right hand, so just the left predator will select the prey.
The predator is within the Angular Restriction of the prey on its right hand but outside of its given Selection Range, so the predator will ignore all prey targets.
Angular Restriction : 120 degrees
The Angular Restriction defines a forward-facing horizontal field related to the target in which the predator must be inside to select the prey. It's comparable with the field of view of the target, but the different is that this value affect the predator and not the prey target, or in other words, the prey must be facing to the predator, so that the predator will select the prey.
This Predator is outside the the given Selection Range and will ignore all prey targets
PREDATOR





13.2 ICECREATUREREGISTERDEBUG

Small script without editable options to draw the register gizmos.

14 ATTRIBUTES

Attributes are optional components to update and modify given settings of your creatures during the runtime. You can add attribute scripts to your target objects to override the pre-defined settings of a creature who select such a target during the runtime. This allows you to define different settings for the same kind of target. You can also use multiple attributes of the same type in such a case the final selection will be randomized.

Example: Your predator creature have to hunt a prey creature, so you have to define the prey object as interactor and define the conditions your predator will use to detect the prey – this settings will be used for all prey objects of the given type. In addition to that you can add now optional attributes to your prey creature to overwrite the given interactor settings of your predator. By doing this you can force a variable behaviour of your predator (e.g. in one case your predator will attack the prey cause its hungry, in the second case just while the prey is visible and too close, in the third case the predator will ignore the prey completely but preferred the player)

This feature is currently under construction and will be improved within the next versions.

14.1 ICECREATURETARGETATTRIBUTE

ICECreatureTargetAttribute contains Target Settings which will be overwrite the specified target values of a creature as soon as the target object will be selected.

14.2 ICECREATUREINFLUENCESATTRIBUTE

14.3 ICECREATURESELECTIONATTRIBUTE

14.4 ICECREATUREODOURATTRIBUTE

15 EXTENSIONS (BETA)

Extensions are optional components to upgrade given features or to increase the given capacities of your creatures during the runtime. This feature is currently under construction and will be improved and full available within the next versions.

15.1 ICECREATUREINVENTORYEXTENSION (BETA)

16 MISSIONS (COMING SOON)

Additional to the implemented standard missions you will find here further missions which you can assign to your creature.

16.1 ICECREATUREMISSIONTEMPLATE (COMING SOON)

Template script to develop custom missions.

Please note: This script is just a template and requires programming skills in c#.

17 VERSION CHANGES

17.1 VERSION 1.0 RC3 – INITIAL RELEASE

17.2 VERSION 1.1

new – Odour Influences
new - Inventory System
new - ICECreatureAstarAdapter – A* Pathfinding Project support
new - ICECreatureMissionTemplate (to create own missions)
new - Further Demo Scene and Tutorials
new - Support of sea life and avian species
new - Setup Wizard (to set-up several standard scenarios such as herds, wp mission etc.)
new - ICECreatureRegister (Pool Management - zones/scenes)
new - ICECreatureRegister (Pool Management - range and FOV culling)
new - Support Menu
improved – Environment Surface – Scan Interval
improved - Code optimizing and comments
improved - Target Selection Criteria (status values based perception)
improved - Status (Feed Type - HERBIVORE, OMNIVORES, CARNIVORE)
improved - ICECreatureRegister (Pool Management - spawn at start)
improved - ICECreatureUFPSAdapter (Player Damage)
improved - Body Orientation and the slope handling
improved - Rework Motion Control
new - Obstacle Avoidance
improved - Target Selection Criteria (Extended Visibility Check for Target Perception)
improved - Interactor Copy-Function
bugfix - Error message if invalid target

18 ICE FRAMEWORK

18.1 ICECREATURECONTROL (1.1.8)

18.2 ICEENVIRONMENTCONTROL (BETA)

18.3 ICECONSTRUCTIONCONTROL (COMING SOON)

19 Frequently Ask Questions

Symptom: Sometime your creature pop into the air or spin off into the sky

Problem: By using Rigidbody and Collider to detect collisions, your creature will be handled as a closed, impenetrable unit, which requires sufficient space on the ground. If this free space is not given, because your creature comes in contact with other colliders the influencing forces will push your creature in a physically possible position. Depending on the physical forces and settings, such as mass, gravity or the physical material settings as well, your creature will lift off.

You can see this phenomenon predominant if have a large number of creatures in a cramped scope or a spatial constriction.

Solution: Reduce the size of the collider, to minimize the required space. If this is not directly possible, because of using a mesh collider, you could use the convex collider from the mesh as a trigger to detect impacts and an additional, adaptable collider to handle the collisions.

Please note: Basically ICECreatureControl doesn't need a Collider or Rigidbody to handle the movement, so if visual intersections with other objects are acceptable for the gameplay (e.g. your creature is just a supporting actor and will not be in the focus of the player or you have dozens of creatures in a crowd and nobody could see details etc.) you could also remove the Rigidbody to solve this problem and if your creature also don't need any kind of interactions you could remove the collider as well.

20 THIRD PARTY SUPPORT

20.1 ULTIMATEFPS

UFPS is a fantastic framework to create excellent First Person Shooter within a short time, but each Shooter also needs condign and inveterate disputants. ICECreatureControl can handle this job without writing one line of code. Just add the ICECreatureUFPSPlayer script to your UFPS Player and the ICECreatureUFPSAdapter script to your NPC creature and instantly both of them can interact as a team or as fierce antagonists.

20.1.1 ICECreatureUFPSPlayer

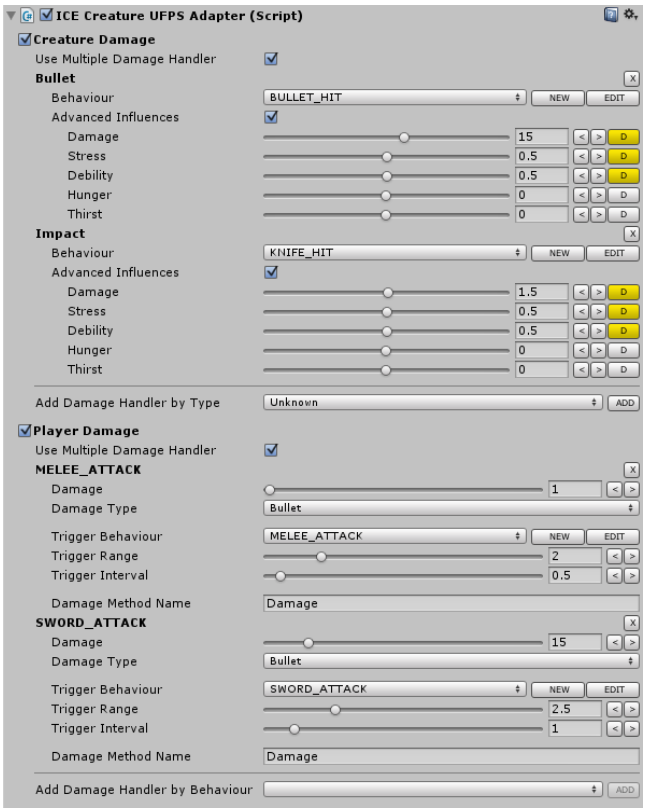
The ICECreatureUFPSPlayer will register your UFPSPlayer in the ICECreatureRegister so he will be selectable as an Interactor in the Interaction Section of your ICECreatureControl and you can define the desired behaviours, furthermore the script will handle also all impacts your UFPS Player receives from your ICECreatureControl controlled creature.

20.1.2 ICECreatureUFPSAdapter

The ICECreatureUFPSAdapter will handle all damages your ICECreatureControl NPC receives from your UFPS Player.

Note: To using the optional ICECreatureUFPSPlayer and ICECreatureUFPSAdapter adapter scripts you required a licensed version of the UFPS Package from VisionPunk.

https://www.assetstore.unity3d.com/en/#!/content/2943



20.2 UniStorm Weather System

Your ICECreatureControl NPC can react to environmental influences, such as temperature, weather or time. By default you can set all these parameter in the ICECreatureRegister component, but the register is only the distributor and does not define or manage own environmental values.

20.2.1 ICECreatureUniStormAdapter

With the ICECreatureUniStormAdapter you can receive the environment status directly from the UniStorm Weather System, so your creature can sleeping in the night, abort a fight because of extreme weather conditions or freezing to death while low temperatures.

Just add the ICECreatureUniStormAdapter to your Creature Register and specify the desired behaviours.

Note: To using the optional ICECreatureUniStormAdapter you required a licensed version of the UniStorm Weather System by Black Horizon Studios.

https://www.assetstore.unity3d.com/en/#!/content/2714

20.3 Locomotion System

Maybe the free Unity Asset 'Locomotion System' by Rune Skovbo Johanson is a little bit aged but it is still perfect to enhance the quality of movements, especially if you have a creature with only some few animations.

The Locomotion System and ICECreatureControl working absolutely fine together, in this combination the Locomotion System handles all ground animations while ICECreatureControl is working as character motor and defines the movements by following the given rules and execute all his other tasks.

To using the Locomotion System combined with ICECreatureControl just add the 'Leg Animator' and the 'Leg Controller' script of the Locomotion System in addition to ICECreatureControl. Configure the locomotion scripts by settings the required entries for your creature, such as the root and leg bones and the animations which are to be controlled by the locomotion system. If this is done, open the Behaviour Rules of ICECreatureControl and deactivate all animations which are controlled now by the Locomotion System – Note: don't delete the complete rule, just set the Animation Type to NONE, all other values and especially the movement settings are continue required.

Now you can press play, to check the movements. Of course you have to do the fine adjustment but you should see now, how your creature adapt his steps to the ground.

Note: To using the 'Locomotion System' with Unity 5 you have to adapt the code a little bit but the changes are easy.

20.3.1.1.1 LocomotionEditorClass.cs line 45

The 'Root Bone' Object should be a scene object, but the *ObjectField* parameter *allowSceneObjects* is flaged as *false*, so simply change it to *true*.

20.3.1.1.2 LegAnimator.cs line 252 and line 286

There are two code segments where the LegAnimator creates *new AnimationClips*, these new clips must be flagged as *legacy* otherwise the Animation Component can't handle these clips. You can fixed it easy by doing something like this:

AnimationClip _clip = new AnimationClip();

_clip.legacy = true;

GetComponent<Animation>().AddClip(_clip, "LocomotionSystem");

Note: The Locomotion System works currently only with legacy animations, therefore please consider that Unity intends to phase out the Legacy animation system over time and it is NOT advisable to use it for new and especially not for larger Projects.

Tip: The Locomotion System requires at least two keys for the Position and Rotation for each assigned Animation Curve otherwise the curve will not be valid and you can't initialize your settings but not all animations fulfil these requirements because not each node of the original animation needs a move and dependent to the used animation program needless keys was not stored by saving the animation. But it's easy to solve it. Open the desired animation in the Animation View (Ctrl-6), select the required node and check the keys for Position and Rotation, if they are not exists simply add the missing parts. Btw. Imported FBX animations are listed as read-only, so just duplicate the animation by using Edit => Duplicate and modify the copy as above-mentioned.

https://www.assetstore.unity3d.com/en/#!/content/7135

21 SPECIAL THANKS

21.1 BÜMSTRÜM

https://www.assetstore.unity3d.com/en/#!/publisher/909

21.2 SOU CHEN KI

https://www.assetstore.unity3d.com/en/#!/publisher/1796

21.3 LESHIY3D

https://www.assetstore.unity3d.com/en/#!/publisher/12156

21.4 QUENTIN HUDSPETH

http://qhudspeth.deviantart.com/art/Fighter-Silhouette-164760918

21.5 FLYINGTEAPOT

https://www.assetstore.unity3d.com/en/#!/publisher/3317

21.6 JON FINLAY

Many thanks for his great artwork and images

21.7 CHRIS SPOONER BLOG SPOONGRAPHICS

http://www.vectorjunky.com/free-vector/Animals/Free-Vector-Pack--Safari-and-Zoo-Animals.html