ICE CREATURE CONTROL

SPIRIT AND MOTION FOR VIRTUAL LIFE-FORMS

NPC CONTROLLER FOR UNITY

USER MANUAL

Version 1.4.0 (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 is an abstract copy of nature and should be familiar to you. 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

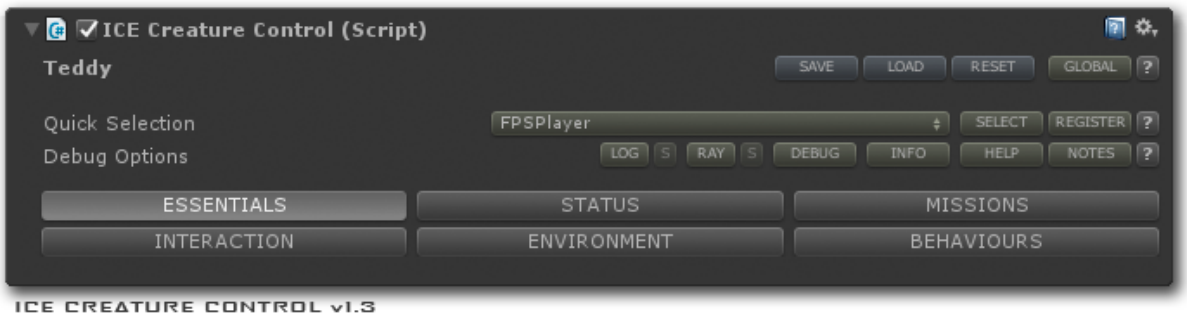
Targets and Behaviours are the moving spirit of your creature and you can find these settings in nearly each content module. ICECreatureControl provides your creature the ability to handle a large number of situations and reactions by specify Targets and Behaviours but please note, that everything your 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 FIRST STEPS

3.1 THINGS YOU NEED TO KNOW

As above-mentioned ICE comes with tons of features and possibilities and all the potential settings could overwhelm you while doing the first steps. But don't worry, the most settings are optional and will not be obligatory required to get attractive results also ICE provides you tidy and logical structured User Interfaces with integrated support features.

The User Interface of ICECreatureControl is subdivided in 7 sections – the Main Control Section and 6 Content Sections.



3.1.1 Main Control Section

The Main Control Section contains all relevant elements to handle the component content but also to control the debug options and to navigate within your global ICE World. You can find such a Main Control Section on the head of nearly all ICE components.

3.1.1.1 Quick Selection

The Quick Selection Popup is directly connected to the Creature Register Component and allows you to navigate quickly between all your registered creatures and objects and to the inspector view of the register as well. Use the Quick Selection Popup in conjunction with the GLOBAL display options, to get a quick access to your focused settings sections, without searching in several foldouts.

3.1.1.2 Debug Options

Here you can choose your individual debug 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.

3.1.2 Content Sections

3.1.2.1 Essentials

The Essential Section contains the basic settings to breathe life into your virtual character and covers also technical details, such as the desired motion control, gravity, obstacle avoidance, ground handling, culling options etc.

3.1.2.2 Status

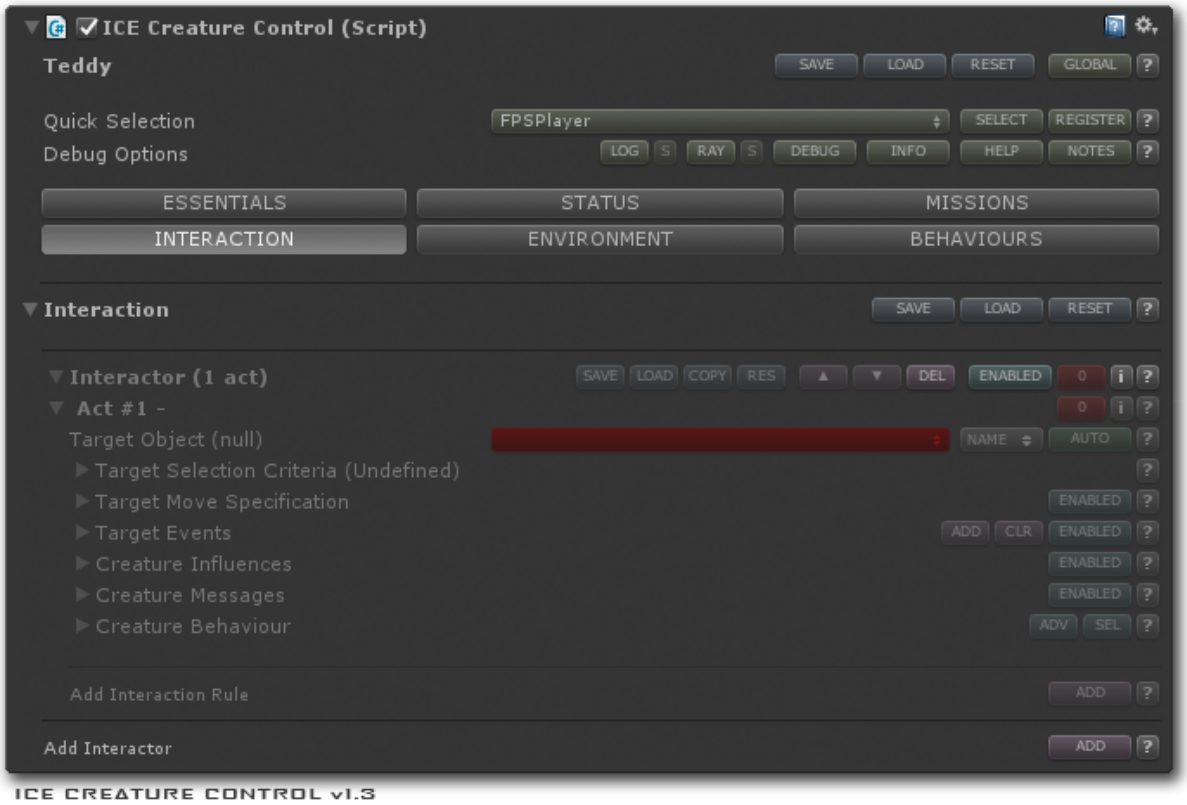
The Status Section contains individual characteristics and the vital signs of your creature.

3.1.2.3 Missions

The Mission Section contains all the default tasks your creature have to fulfil during its lifetime.

3.1.2.4 Interaction

The Interaction Sections contains all the desired rules your creature have to use to interact with other GameObjects. Such interactable Objects could be your Player, other Creatures or usable Tools like a Weapon or a Flashlight etc.

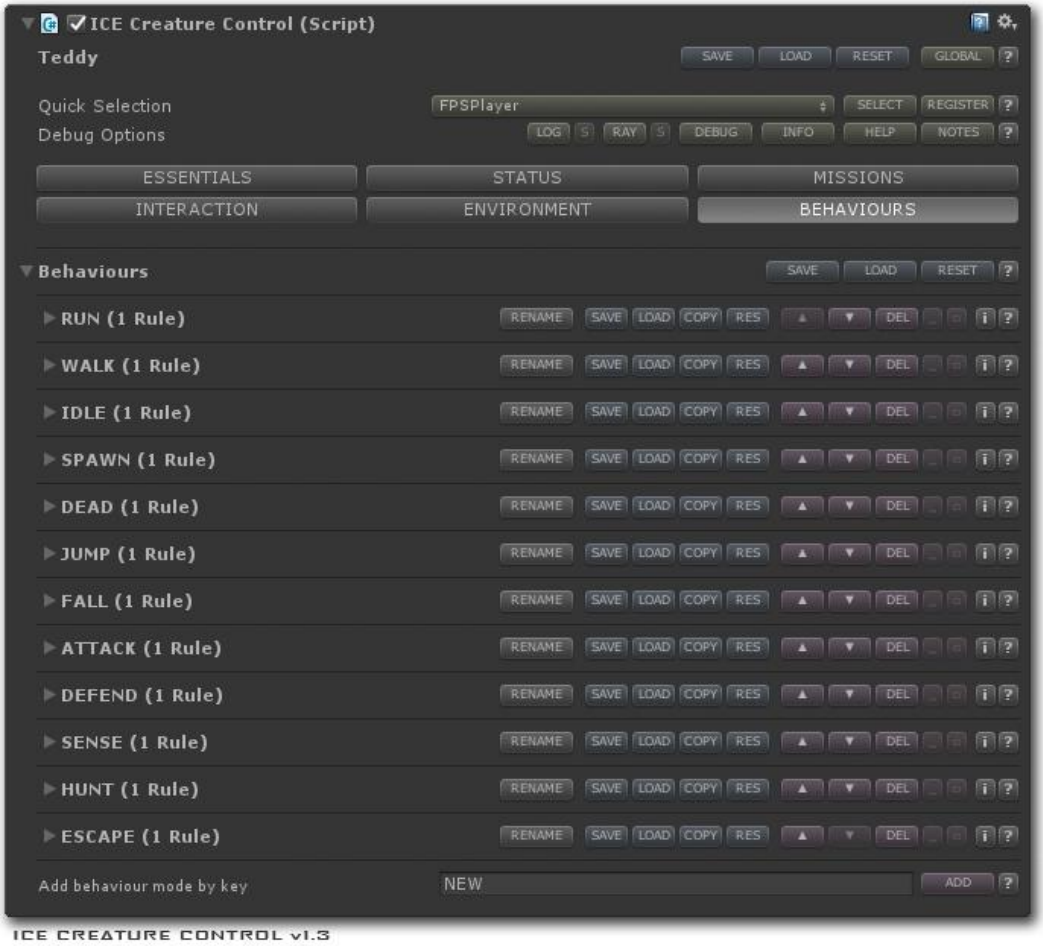


3.1.2.5 Environment

The Environment Section contains all the rules your creature have to use to interact with its surrounding environment.

3.1.2.6 Behaviour

The Behaviour Library contains all created behaviours your creature can use to react to specific objects and/or situations. It can have as many behaviours as you want.



3.1.3 Usability

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 and there are already some features to clean up the chaos.

3.1.3.1 The HELP Button (?)

First of all you'll find a HELP button behind each control, so you have not to work with an open manual to find information about a specific function. Nearly each feature will be explain in the inspector view by pressing the question mark.

3.1.3.2 The INFO Button (!)

In addition to the HELP button, some controls provides you an INFO button, which allows you and your team to enter own notes and comments about your settings.

3.1.3.3 The DEFAULT Button (D)

Many settings of ICE are optional and will not be required in each case, so if you adapt such settings but you are unhappy with the result or if you are not sure which of your settings is wrong and forced a negative effect, you should note, that changed values will be marked with an yellow DEFAULT button, so you can press such a DEFAULT button to adjust the control to its default values.

3.1.3.4 Save, Load, Reset Buttons

ICE provides you to save, load and reset specific sections or the complete settings, so you could play with different configurations to determinate the best setup and reuse your settings for other creatures again. But please note, that embedded objects (e.g. effect or sound prefabs) will not been saved.

3.1.3.5 Colour Codes

The controls of ICE are grouped by different colours according to their given functionality, so blue coloured controls represents file and list operations (e.g. save, load, add etc.), lime coloured controls represents optional features (e.g. Enabled or Advanced Buttons), yellow coloured controls represents debug and navigation features. Red coloured controls represents missing values of obligatory sections.

Some controls (e.g. Targets, Target Selection Criteria, and Behaviours) will be changed their colour automatically during the runtime, according to their current state. Basically RED means that the control is inactive, yellow means wasn't checked within the last frame and green means that the control is valid and active. This will be extremely helpful to observe the correct behaviour of a creature during the runtime.

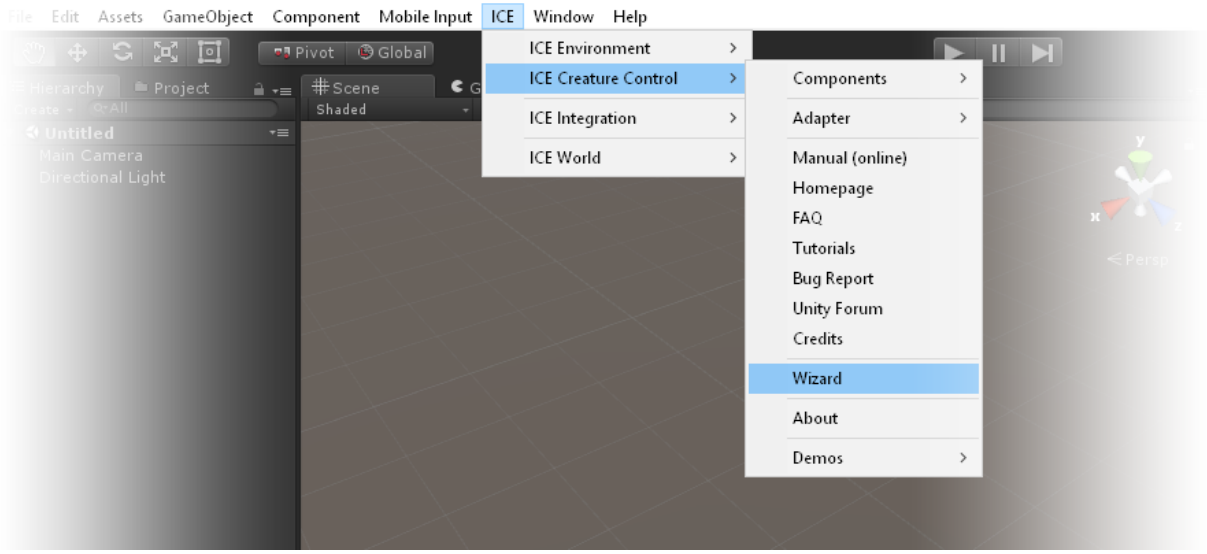
3.1.3.6 Familiar Control Elements

While working a while with ICE you'll see that you're working predominantly with some familiar elements, such as Targets and Behaviours, but also with Events or the Impulse Timer etc. and in fact the majority of control elements based on a collection of core objects, which will be used in the whole ICE World and provides you a familiar editor environment.

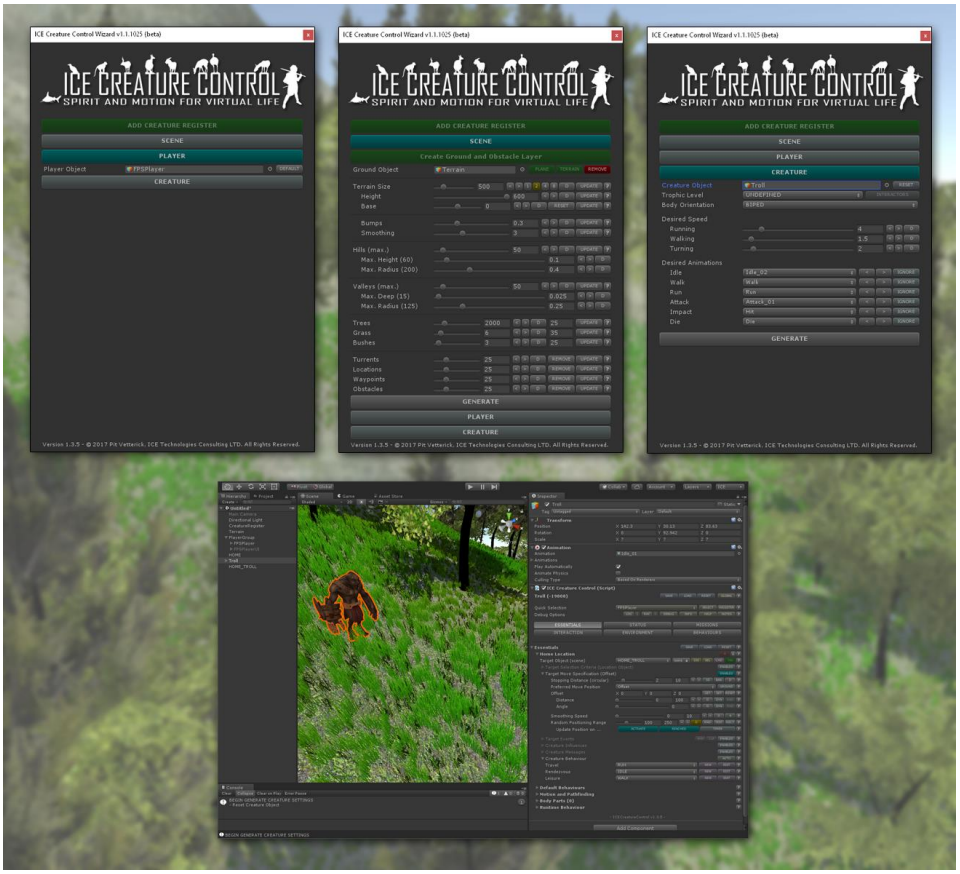
Btw. this simplifies the familiarization for creative user but for developer as well, because the whole code of ICE is strictly object orientated in the above mentioned way.

3.2 SETUP WIZARD

The easiest way to setup a creature and to get quick results will be using the ICECreatureControl Wizard. Simply open the ICE menu and press the Wizard menu item.



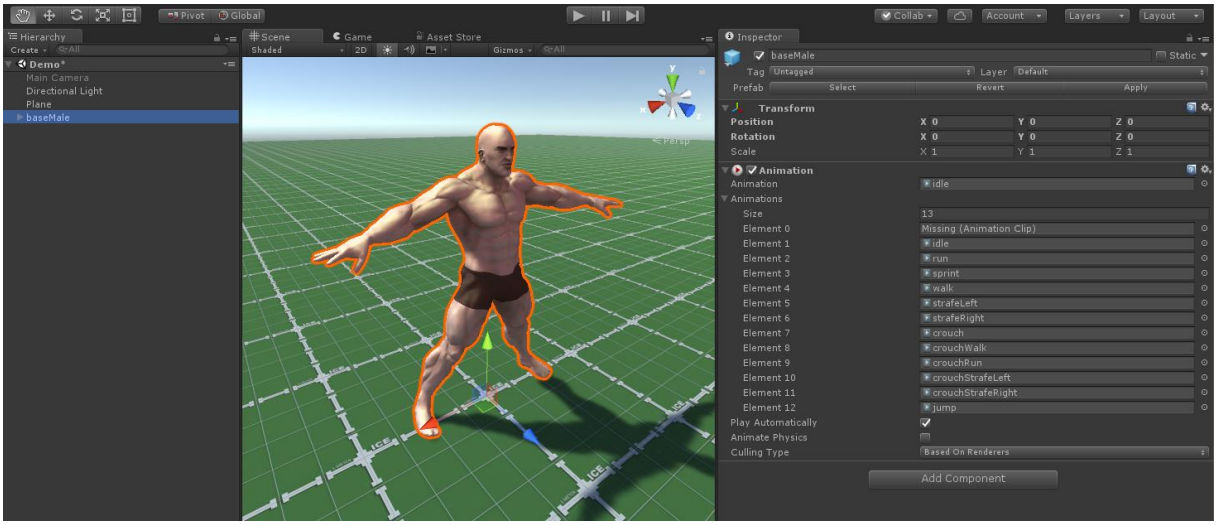
The Wizard allows you to create a complete test scene from scratch with some mouse clicks, so you can use the Wizard to configure a single creature but also to create the complete environment to test it.



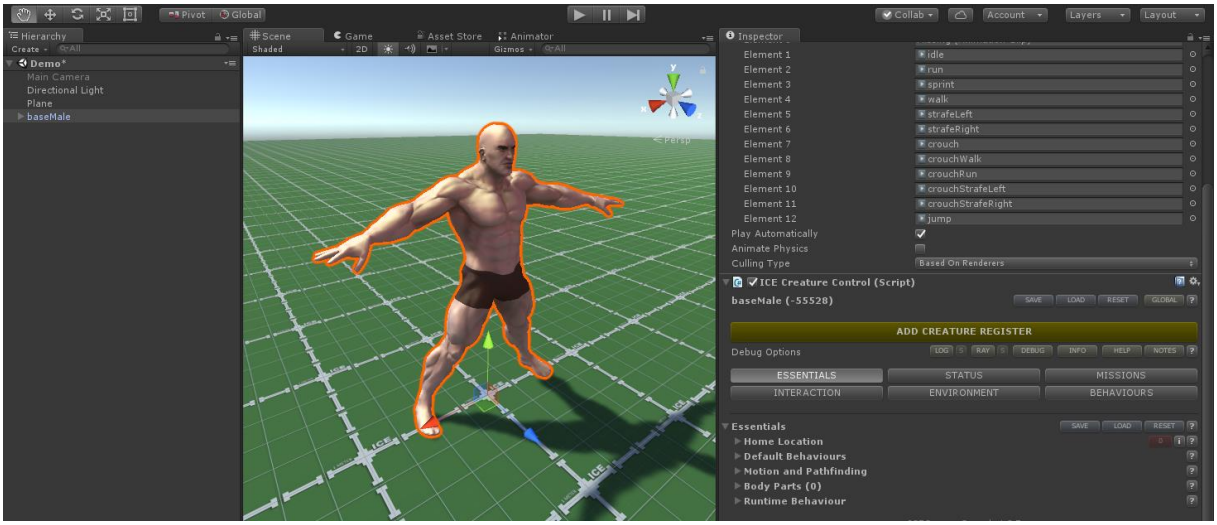
3.3 MANUAL SETUP

3.3.1 Fitness

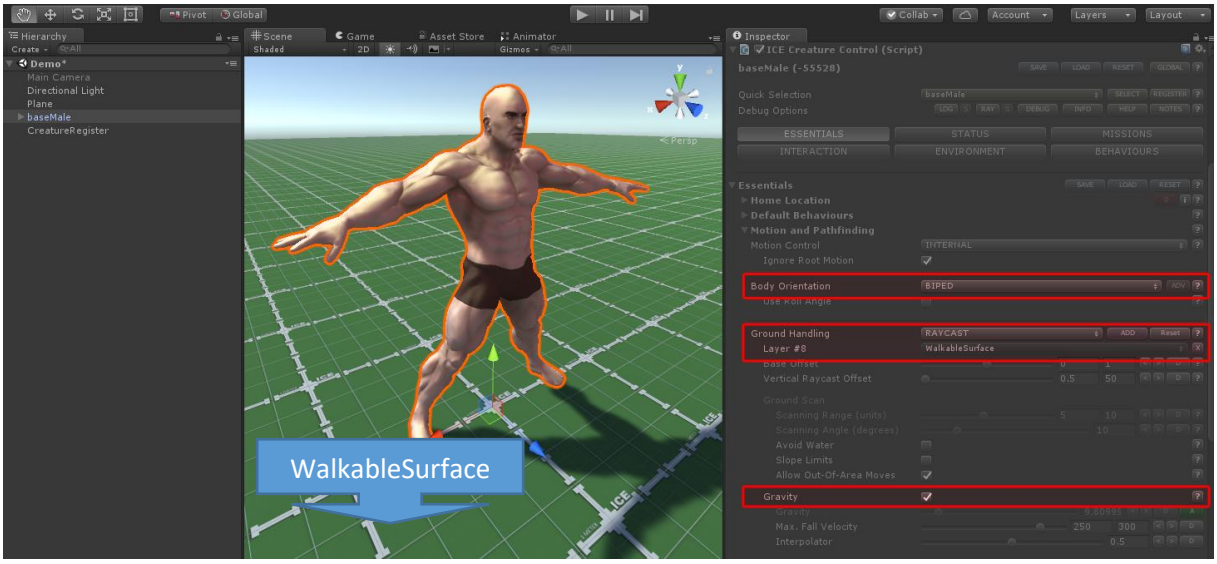
Choose your desired creature and place it somewhere in your scene. Make sure that the local axes of your creature are correct aligned (the forward direction of your creature should be positive z). Your creature should have at least two animations which are suitable for idle and move behaviour. ICECreatureControl 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.



Open the Essentials section, scroll down to the 'Motion and Pathfinding' settings and choose the desired 'Body Orientation' and the desired Ground Handling. While using RAYCAST you should define at least one ground layer e.g. WalkableSurface which represents all walkable object and surfaces. Also make sure, that 'Gravity' will be flagged.

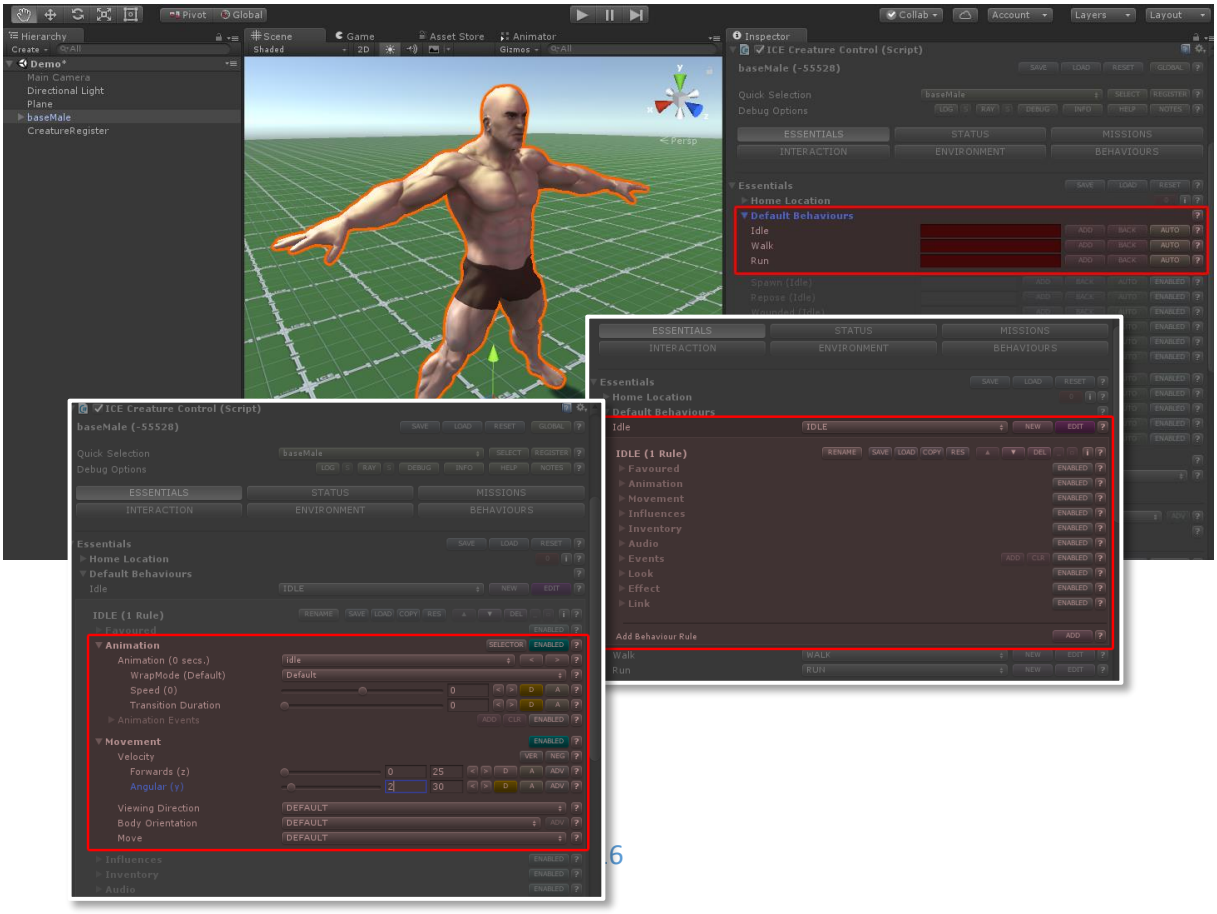


Note: To simplify this introduction, your creature will be nearly naked – no Collider, no Rigidbody, no NavMeshAgent - so you can see and understand how ICECreatureControl works. Btw. all the mentioned components are supported, but for the first steps the only additional component should be an Animation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

3.3.2 Behaviours

Your creature has successfully passed the physical fitness check and you can continue to define some basic behaviours your creature will need to for its first steps.

Scroll up to the 'Default Behaviours' section, press the AUTO buttons for IDLE, WALK and RUN and afterwards the EDIT button to open the Behaviour Editor of the selected behaviour.



The Behaviour Editor allows you to define several settings for Animation, Movements, Sound, and Effects etc. but for now we only need some suitable values for the Animation and the associated Movements, so select the desired animation. In addition you can also adapt the WrapMode, the animation speed and the desired Transition Duration. By default the Animation Speed should be 1 and the Transition Duration around 0.5 (should be the default values).

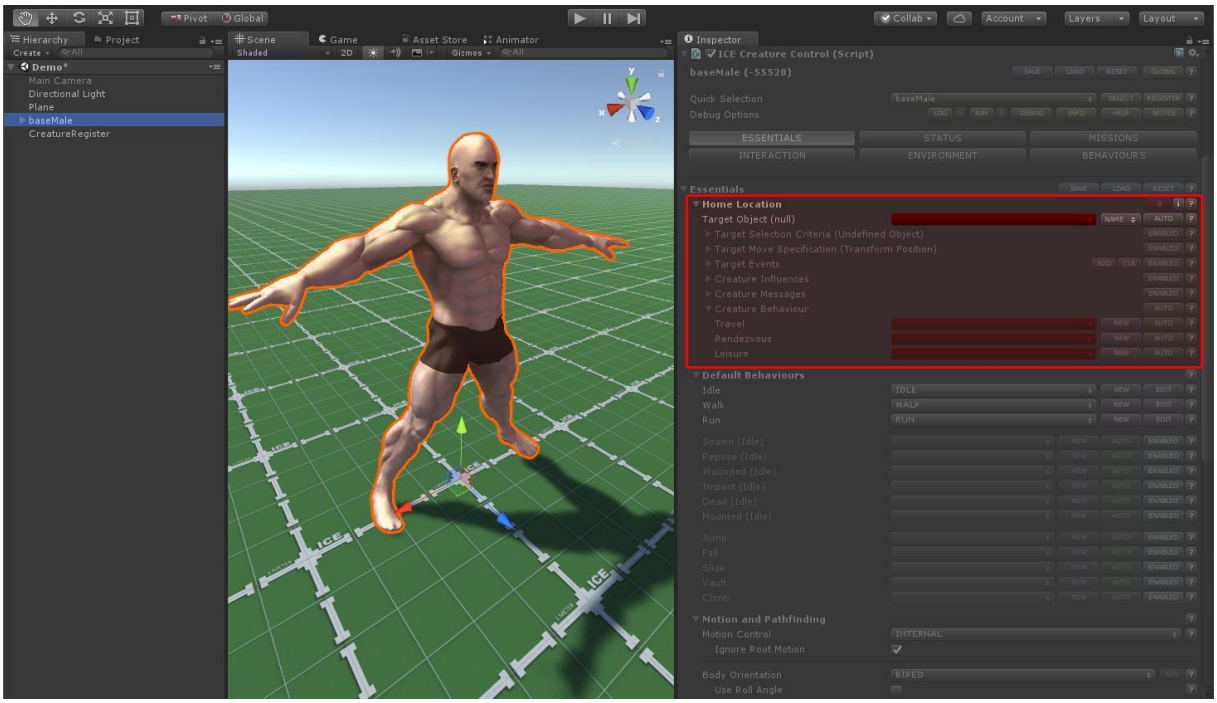
If the Animation settings are ready you have to adapt the desired Forward (z) and Angular (y) speed of your creature, both values should be suitable to your selected animation and will need to be adapted for the best result, but for starting you could use 3 for a 'walk' and 6 for a 'run' animation and for the angular you could use the half of the forward value. Please make sure, that the Popups of 'Viewing Direction', 'Body Orientation' and 'Move' displays 'DEFAULT' and please don't forget the Angular values, because without a specified turning speed your creature can't change its course and will running in one direction only.

Now you could close the Behaviour Editor by pressing the EDIT Button again and repeat the behaviour configuration steps for 'IDLE, 'WALK' and 'RUN'. (Btw. you can find all behaviours also listed in the Behaviour section)

3.3.3 Target

Your creature has successfully passed the fitness check and knows some basic behaviours, so there is only one step more to bring your creature alive.

Scroll up to the 'Home Location' settings and define the home target for your creature.



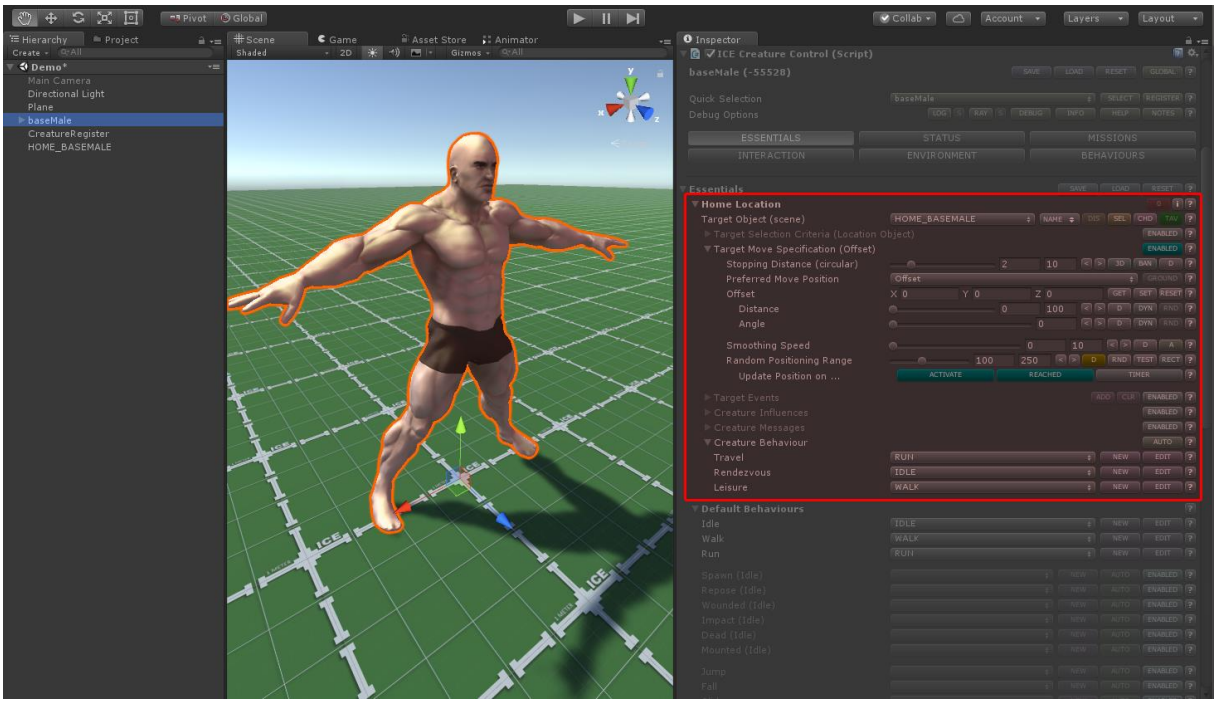
Press AUTO to create a new Target Object. If you have already registered GameObjects in your CreatureRegister you can also select a suitable Target by using the Popup but basically you can use each reachable GameObject in your scene (include the terrain or your player object).

Press now the AUTO button of the Creature Behaviour to use the already defined IDLE, WALK and RUN behaviours for this target.

As soon as your creature have suitable behaviours and a reachable target it will try to reach it, so if you press play now the creature should run to the target and if you move the target in the scene view, you will see that your creature will follow. But you don't need to move targets by hand, we can handle this with some clicks also with dynamic waypoints.

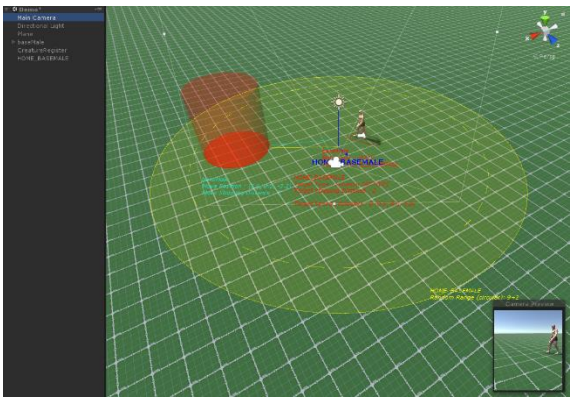
For this open the Target Move Specifications and select OFFSET. Adapt now the Random Positioning Range and activate 'Update Position on … ACTIVE and REACHED'.

The Random Positioning Range defines a circular area around the target in which your creature will randomized walking around. The flags ACTIVE, REACHED and TIMER specifies the update conditions.



To get a better understanding of your settings and their resulting effects, you should activate the DEBUG options to visualize the values. Targets, paths and movements can be displayed as gizmos and will be extremely helpful to understand and fix some issues and/or to adapt the settings in real-time.

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 reached the 'Stopping Distance', the TargetMovePosition will be relocated within the 'Random Range' and your creature will follow according to its 'LEISURE' behaviour.

4 ESSENTIALS

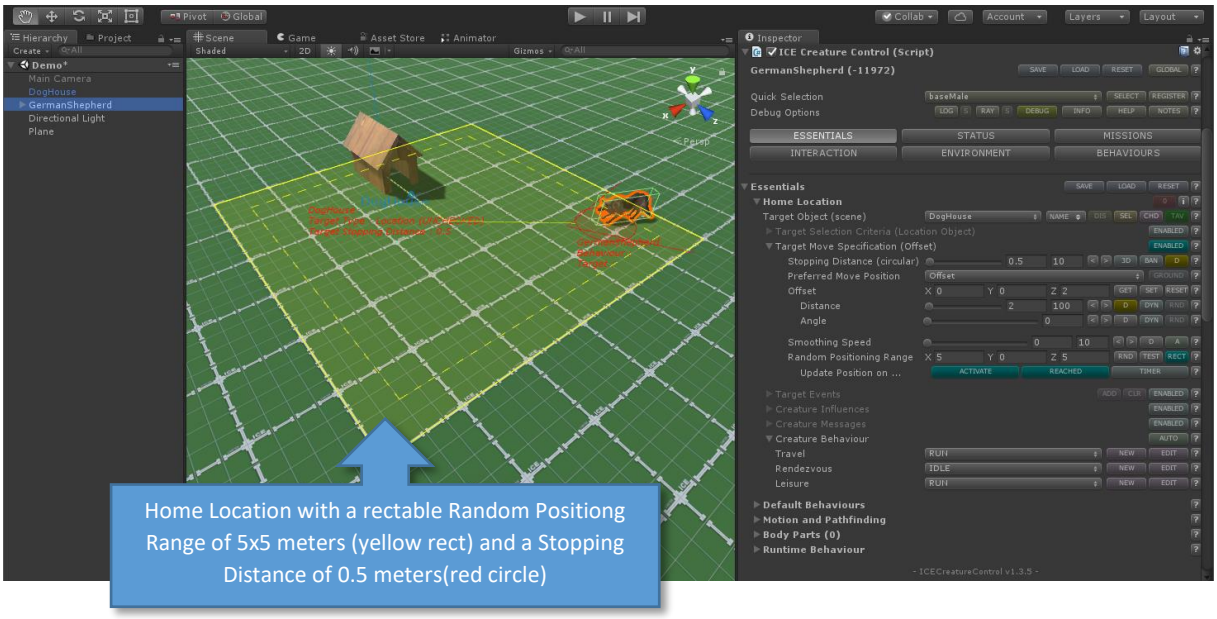
Essentials covers all fundamental settings your creature needs for its first steps, such as a home location, basic behaviours and the general motion and pathfinding settings as well.

4.1 HOME LOCATION

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.

4.1.1 Creature Behaviour

By default a typical Target have two behaviours only - Standard and Rendezvous. One move behaviour to reach the TargetMovePosition and one behaviour in cases this position was reached. The Home Target knows a further behaviour in cases a creature will be within the Random Positioning Range, so it can do some additional idle activities while it is close to its home.



As long as the Dog will be within the yellow area it will run its Leisure behaviour to reach the next dynamic waypoint, otherwise it will run its travel behaviour to run home.

Tip: By default your creature will always run its Rendezvous behaviour whenever it is reaching the stopping distance of a waypoint, if this Rendezvous behaviour have no movements your creature will stop. If you don't want this to force continuous movements (e.g. a guard who is patrolling an area) you can add a suitable move to the Rendezvous behaviour or you can assign the Leisure behaviour also as Rendezvous (if you don't need a special behaviour for the rendezvous)

4.2 Default Behaviours

The Default Behaviours represents the proposed minimum performance requirements your creature should be able to fulfil. Please note that you can define additional behaviours as required by using the Behaviour section of your creature.

4.2.1 Idle Behaviour

Static default behaviour while the creature has reached a Target Move Position and has no other tasks to do or if it needs to stop temporary its movement for some reasons.

4.2.2 Walk Behaviour

Default move behaviour while approaching to a Target Move Position.

4.2.3 Run Behaviour

Default move behaviour for reaching the given Target Move Position.

4.2.4 Spawn Behaviour

Optional action behaviour while the creature is spawning. If the SPAWN behaviour is disabled the creature will use its IDLE behaviour during the spawning process.

4.2.5 Dead Behaviour

Optional action behaviour while the creature is dead or dying. You can combine the DEAD behaviour with the Corpse options to run a dying animation before spawning the corpse object. If the DEAD behaviour is disabled the creature will spawn the Corpse only or run its IDLE behaviour in cases that the Corpse feature is disabled as well.

4.2.6 Jump Behaviour

Optional action behaviour, which will be used if your creature needs to jump.

4.2.7 Fall Behaviour

Optional action behaviour, which will be used if your creature is falling.

4.2.8 Slide Behaviour

Optional action behaviour, which will be used if your creature needs to pass under an obstacle.

4.2.9 Vault Behaviour

Optional action behaviour, which will be used if your creature needs to cross over an obstacle.

4.2.10 Climb Behaviour

Optional action behaviour, which will be used if your creature needs to climb.

4.3 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.

4.3.1 Motion Control Type *(NEW)*

4.3.1.1 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.

4.3.1.2 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.

4.3.1.3 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.

4.3.1.4 CHARACTERCONTROLLER (NEW)

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

4.3.1.5 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.

4.3.1.6 Ignore Root Motion

When using animations with root motions, no additional positional changes are required, but if such changes are desired, they can be forced with Ignore Root Motions. In such a case, both position changes would add up and possibly reflect themselves in an excessive speed and / or in unexpected movements.

4.3.2 Ground Handling

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

4.3.2.1 Base Offset

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

4.3.2.2 Vertical Raycast Offset

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.

4.3.2.3 Avoid Water (IMPROVED)

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

4.3.2.4 Use Slope Limits

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.

4.3.2.4.1 Max. Slope Angle

Maximum slope angle your creature can use.

4.3.2.4.2 Max. Walkable Slope Angle

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

4.3.2.5 Allow Out-Of-Area Moves

By default your creatures are able to leave intended game areas without to get lost in space while passing the border of a restricted terrain or mesh. If a creature can't detect a ground, it will continue with its current activities by keeping its last known operating level. This behaviour should bridge potential leaks while loading or generating its surrounding environment. If you deactivate Out-Of-Area moves, your creature will break its current activities by updating its move position to avoid border zones.

4.3.3 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.

4.3.3.1 Advanced Body Orientation (IMPROVED)

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

4.3.3.2 Use Leaning Turn

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.

4.3.4 Overlap Prevention *(IMPROVED)*

The Overlap Prevention allows your creature to detect and avoid intersections with other GameObjects without using additional Components such as a non-kinematic Rigidbody and Collider, but please consider, that the Overlap Prevention detect other objects by scanning their colliders, so make sure that each detectable object (incl. your detectable creatures) will need at least a collider. Apart from that, you are free to combine this feature also with a Rigidbody or a CharacterController, only while using your creature with a NavMeshAgent the Overlap Prevention will be disabled.

4.3.4.1 Overlap Prevention Type

The Overlap Prevention Type provides you three different types (CAPSULE, SPHERE, BOX) your creature can use to scan its surrounding environment. Each type comes with characteristic settings to define the optimal scanning range, so you can define the best type to cover the body of your creature. Tip: to avoid hard collisions or course corrections, you should be

4.3.5 Obstacle Avoidance *(IMPROVED)*

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.

4.3.5.1 Layer

4.3.5.2 Scanning Range

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.

4.3.5.2.1 Dynamic Scanning Range

If you are activate the DYN button the creature will determine the scanning range automatically by using its current speed and the given multiplier.

4.3.5.3 Scanning Angle

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.

4.3.5.4 Vertical Raycast Offset

The Raycast Offset defines the vertical offset of the scanning ray in relation to the transform position. A suitable orientation value will be the third of the body height, so your creature can detect also low obstacles which would block its legs or other lower body-parts. While using the Overcome feature you should adapt the offset to the half of the body height, because the lower and upper body areas will be covered by separate raycasts to evaluate the best method to avoid or rather to overcome an obstacle.

4.3.5.5 Allow Overcome Obstacles

While 'Allow Overcome Obstacles' is active the creature will try to overcome an obstacle by crossing it over or below, so your creature will try to crawl or slide under an obstacle or vault and climb over it before avoid it by going around.

4.3.5.5.1 Overcome Offset Difference

The Overcome Offset Difference specifies the level difference to the Vertical Raycast Offset and thereby the vertical position of the additional rays, which handles the lower and upper scan areas while the overcome feature is active. By default the difference will be the half of the defined Vertical Raycast Offset value, in which the value will be subtracted from the Vertical Raycast Offset to specify the origin of the lower ray and added to specify origin of the upper ray. Activate the RAY flag to display the rays in the editor.

4.3.5.5.2 Cross Below Starting Distance

The Starting Distance defines the desired distance to the contact point to trigger the required procedures to overcome the obstacle. While the SPEED flag is enabled the Starting Distance will be determined automatically by using the current speed and the given multiplier.

4.3.5.5.3 Cross Over

The Starting Distance defines the desired distance to the contact point to trigger the required procedures to overcome the obstacle. While the SPEED flag is enabled the Starting Distance will be determined automatically by using the current speed and the given multiplier.

4.3.6 Overlap Prevention *(IMPROVED)*

The Overlap Prevention can be used to obviate intersections with other objects but also to bypass obstacles in a smart way. Compared or rather additional to the Obstacle Avoidance, which will avoid collisions with potential obstacles, the Overlap Prevention will handle such collisions to avoid unrealistic mesh intersections. By default the Overlap Prevention will stop the creature if a collision with another object was detected but you can activate the AVOID option, so your creature will change its direction and will try to find the best way along the colliders' surface to avoid further collisions.

4.3.6.1 Overlap Prevention Type and Settings

To use the Overlap Prevention you have to select the desired type which will be suitable to cover the mesh of your creature in the best way. You could use the AUTO button to handle the required settings automatically by using the settings of an existing collider.

4.3.7 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.

4.3.8 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.

4.3.8.1 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.

4.3.8.2 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.

4.3.8.3 Test Interval

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

4.3.8.4 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.

4.3.8.5 Deadlock Action

Deadlock Action defines the desired procedure in cases of deadlocks.

4.4 RUNTIME BEHAVIOUR

4.4.1 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.

4.4.2 Don't Destroy On Load

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

5 STATUS

Status represents the mental and physical fitness of your creature, based on several settings, measurements and dynamic values which will affect the durability and the behaviour of your creature during the runtime as well. You can use this component section to adapt species-typical characteristics, such as the sense and reaction time, maximum age etc.

Dependent to your needs and requirements, ICECreatureControl provides you a Basic and an Advanced Status. The Basic Status contains the essential elements your creature will need for a basic life-cycle, any damage will directly affect the health and with it the durability and your creature will die as soon as its durability is exhausted. This procedure is similar to the 'hit point' concept but you can expand this functionality by activating the Advanced Status, which provides you an extensive Status System to picture the characteristics of a realistic life-form.

While using the Advanced Status the mental and physical fitness based primarily on three dynamic attributes (health, stamina and power) which in turn consists of further detail settings (e.g. age, armor, aggressivity etc.) and several sets of indicators, multipliers and random variances for tuning the final influence.

Please note that all these settings are optional.

5.1 INFO

5.2 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 Dynamic Vital Signs to use the extensive Status System.

5.2.1 Initial Durability

Initial Durability defines the fundamental capability of resistance of a creature in terms of physical integrity and its vital fitness. The durability will be affected by several influences (e.g. damage, age etc.) during the runtime and the creature will die as soon as its durability is exhausted. By default the Initial Durability is adjusted to 100 but you are free to define a suitable value according to your needs and requirements; the lower the value, the greater the impact of several influences and vice versa (e.g. increase this value to 1000 for your level boss or decrease it to 10 for homely antagonists). Please note that changing the durability value is ineffective while using influence values in percent, in such a case a damage of 50% for example will also reduce the durability value to 50%, independent of the defined initial durability value.

The Initial Durability based on a minimum and maximum value, which allows you to define a random range. If you prefer to define a fixed value, simply set minimum and maximum to the same value, also you can adapt the third field to modify the range of the slider.

5.2.2 Damage Transfer Multiplier

Typically a creature should be the highest entity within its transform hierarchy, but it could be that your creature needs to be a child of another object (e.g. horse and horseman) and in such cases both creatures would have their own independent status which is mostly suitable but you could also want to merge two or more creature to a single individual (e.g. an alien, monster or zombie with one or

more agile abscesses etc.) and in such cases, if the child creature should be a fixed part of the main body, you could enable the Damage Transfer Multiplier to forward received damages from the child to the main entity according to the specified multiplier value.

5.2.3 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.

5.2.4 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.

5.2.5 Recovery Phase

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.

5.2.6 Removing Delay

Removing Delay defines the delay time in seconds until a deceased creature will be added to the respawn queue. Within this time-span the creature will be visible in the scene and can be used for loot activities. While ‘Use Corpse’ is active the Removing Delay will also affect the spawning process of the corpse, therefore you should adjust this value to zero or you can adapt it to the dead animation length to use a defined dead animation before spawning the corpse.

5.2.6.1 Variance Multiplier

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

5.2.7 Fitness Multiplier

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

5.2.8 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.

5.2.9 Odour

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

5.2.9.1 Intensity

5.2.9.2 Max. Range

5.2.10 Gender

Gender defines the biological characteristics of your creature and in this context also the social and/or cultural role of the character.

5.2.11 Trophic Level

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.

5.2.11.1 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.

5.2.11.2 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.

5.2.11.3 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.

5.2.12 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.

5.2.12.1 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.

5.2.12.2 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.

5.2.13 Use Environment 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)

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

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

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

5.2.13.4 Comfort Temperature
Comfort Temperature defines the ideal temperature value for your creature.

5.2.13.5 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.

5.2.14 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.

5.2.14.1 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.

5.2.15 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.

5.2.16 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.

5.3 Dynamic Vital Signs

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 Dynamic Vital Signs the procedure to evaluate the Fitness is affected by several different initial values, indicators, multipliers and random variances.

5.3.1 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.

5.3.1.1 Fitness

5.3.1.2 Health

5.3.1.3 Stamina

5.3.1.4 Power

5.3.2 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.

5.3.2.1 Aggressivity

5.3.2.2 Anxiety

5.3.2.3 Experience

5.3.2.4 Nosiness

5.3.3 Dynamic Influences

5.3.4 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.

5.3.4.1 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.

5.3.4.2 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.

5.3.4.3 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.

5.3.4.4 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.

5.3.4.5 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.

5.4 CORPSE

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.

5.4.1 Corpse Removing Delay *(NEW)*

Corpse Removing Delay defines the delay time in seconds until the spawned corpse object will be removed from scene. You can adjust this value to zero to handle the removing process by using external scripts or to keep the spawned corpse durable into your scene.

5.5 SENSORIA *(NEW)*

Sensoria represents the perceptive capabilities of your creatures. If you activate this feature your creature can use its senses to explore its surrounding environment or rather to detect relevant targets. Please note, if Sensoria is disabled, the Sensoria features of the Target Selection Criteria will be disabled as well.

5.5.1 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.

5.5.1.1 Visual Range

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

5.5.1.2 Visual Sensor Position

The Visual Sensor Position defines the origin of the visual field or with other word the eye position. You can define a fixed offset position (relating to your creatures transform position) or you could activate the DYN button to use the local position of a subordinated transform (e.g. head or eyes). In this cases the Visual Sensor Position will be bounded to the specified transform and will follow all movements automatically.

5.5.1.3 Visual Horizontal Offset

While the Visual Sensor Position defines an exact point as origin of the visual field, the Visual Horizontal Offset defines the radius of an additional circular range around this point, to relocate the origin out of a critical range and to avoid conflicts with additional child objects or rather their colliders (e.g. helmet or other additional outfits), which could mask or interfere the visual field.

5.5.1.4 Sensoria Attributes

Sensoria attributes represents the general 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.

5.5.1.4.1 Visual

5.5.1.4.2 Auditory

5.5.1.4.3 Olfactory

5.5.1.4.4 Gustatory

5.5.1.4.5 Tactile

5.6 MEMORY

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.

5.7 INVENTORY *(IMPROVED)*

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.)

5.7.1 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.

5.7.1.1 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.

5.7.1.1.1 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.

5.7.1.1.2 Amount

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

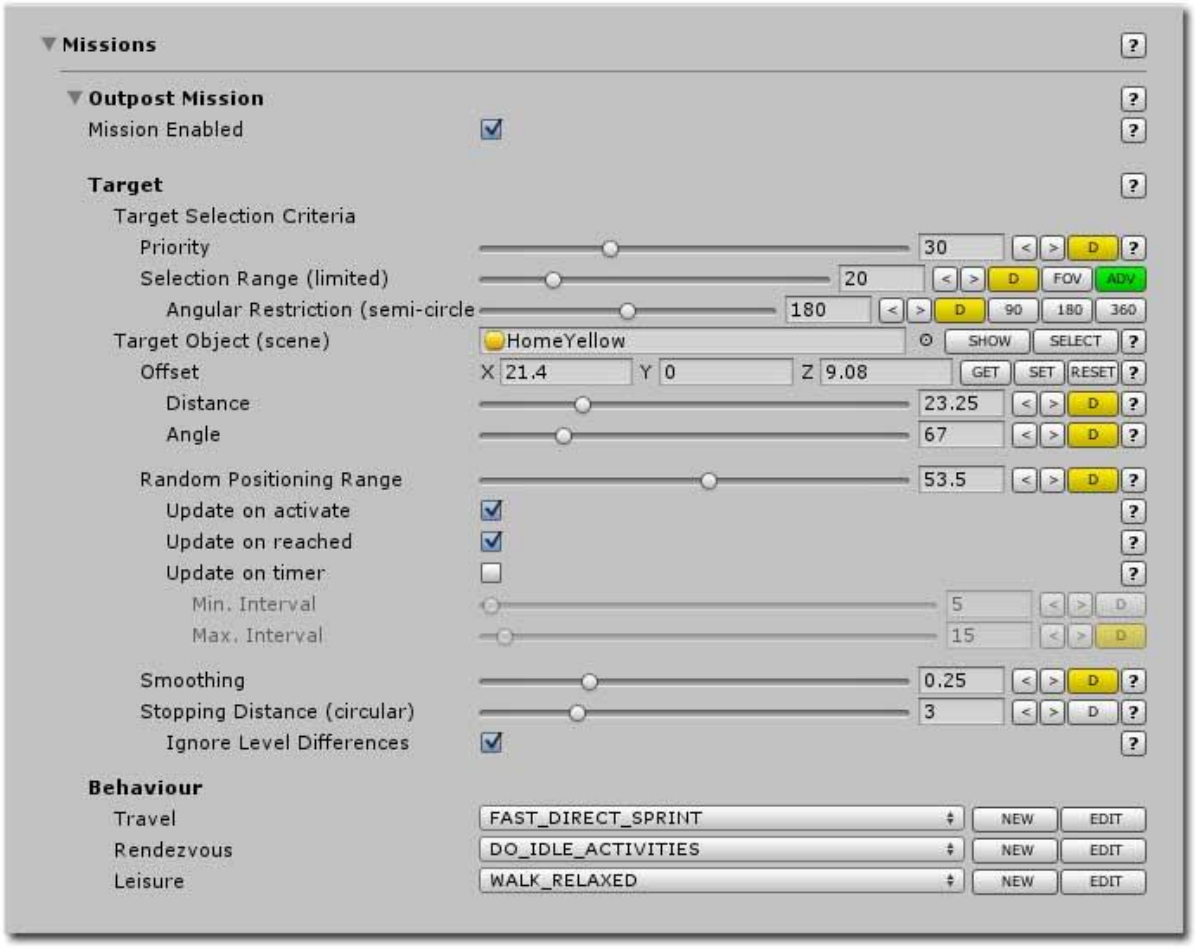
(See also: Behaviour Rules -> Inventory)

6 MISSIONS

6.1 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.



6.2 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.

6.3 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.

6.3.1 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.

6.3.2 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.

6.3.3 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.

6.3.4 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.

7 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.

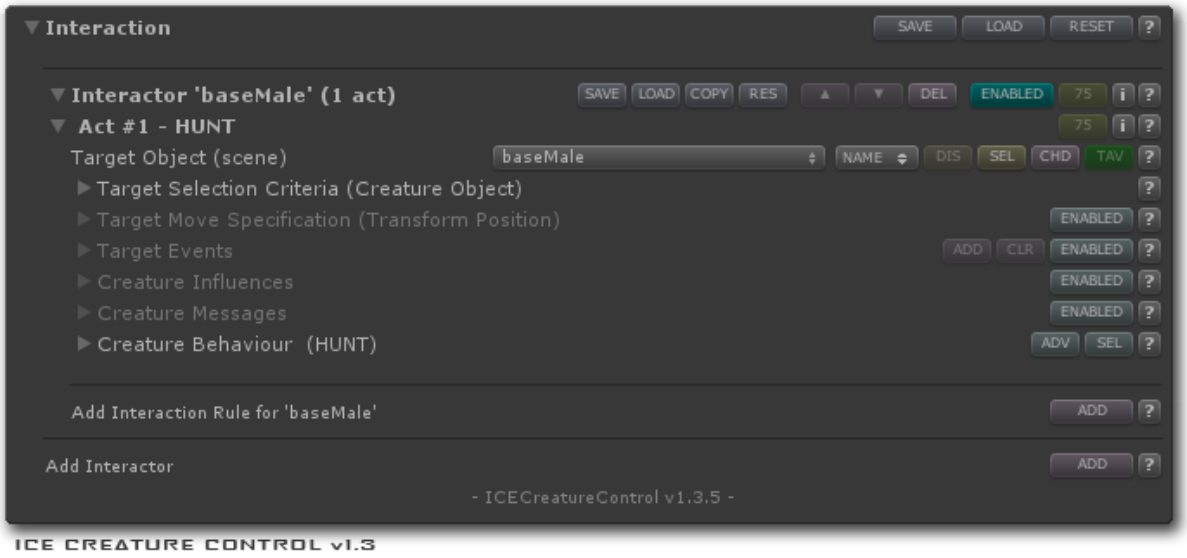
To using the interaction system you have to add one or more Interactors. An Interactor represents another GameObject as potential Target for your creature and contains a set of conditions and instructions to define the desired behaviour during a meeting. By default interactors are neutral, they could 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.

7.1 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.



7.2 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 activate or deactivate ENABLED also by your own scripts to manipulate the gameplay.

7.3 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.

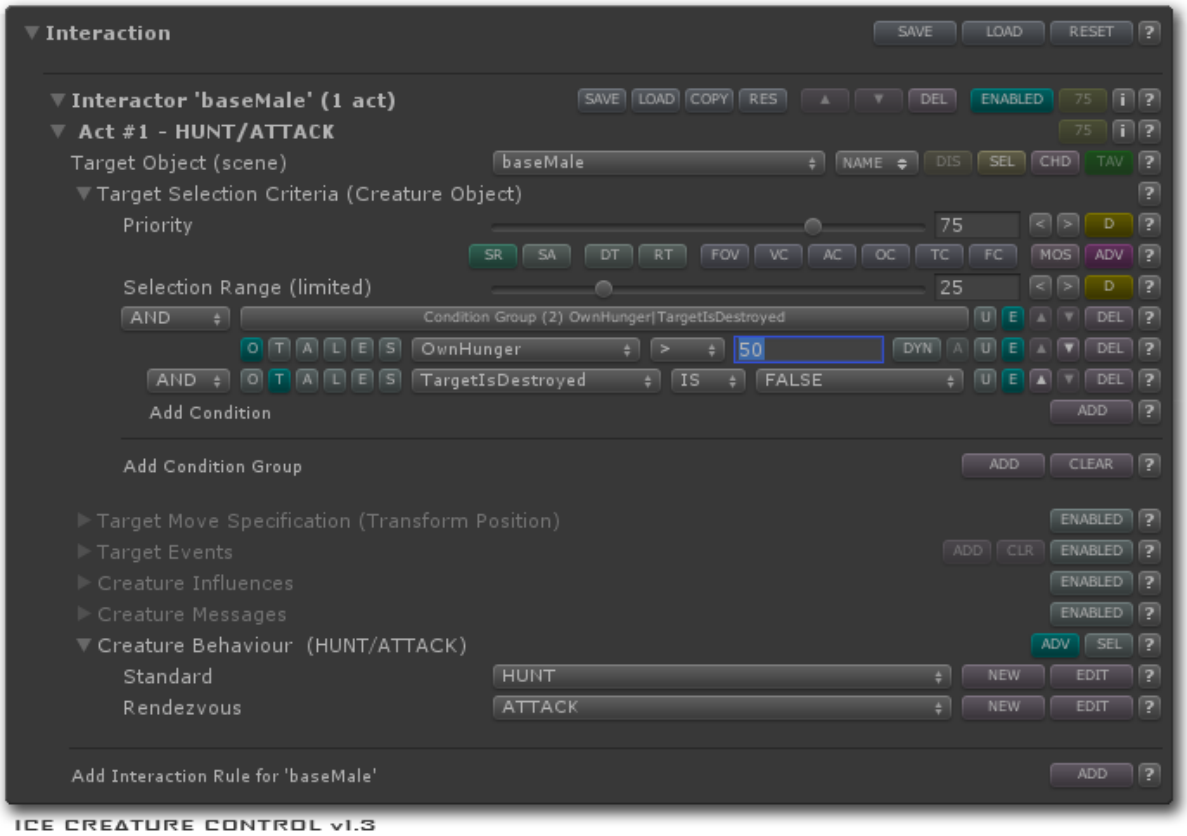

Interaction
SAVE LOAD RESET
Interactor 'baseMale' (3 acts)
SAVE LOAD COPY RES DEL ENABLED 76
Act #1 - SENSE
75
Target Object (scene)
baseMale
NAME DIS SEL CHD TAV
Target Selection Criteria (Creature Object)
Priority
75
SR SA DT RT FOV VC AC OC TC FC MOS ADV
Selection Range (limited)
25
AND O T A L E S OwnHunger > 50 DYN A U E DEL
Add Condition
ADD
Add Condition Group
ADD CLEAR
Target Move Specification (Transform Position)
ENABLED
Target Events
ADD CLR ENABLED
Creature Influences
ENABLED
Creature Messages
ENABLED
Creature Behaviour (SENSE)
ADV SEL
Act #2 - HUNT
COPY RES DEL ENABLED 76
Target Object (scene)
SHOW BASEMALE (48536)
SELECT
Target Selection Criteria (Creature Object)
Priority
76
SR SA DT RT FOV VC AC OC TC FC MOS ADV
Selection Range (limited)
20
AND O T A L E S OwnHunger > 50 DYN A U E DEL
Add Condition
ADD
Add Condition Group
ADD CLEAR
Target Move Specification (same as in Act One)
OVERRIDE ENABLED
Target Events
ADD CLR ENABLED
Creature Influences
OVERRIDE
Creature Messages
ENABLED
Creature Behaviour (HUNT)
ADV SEL
Act #3 - ATTACK
COPY RES DEL ENABLED 77
Target Object (scene)
SHOW BASEMALE (48536)
SELECT
Target Selection Criteria (Creature Object)
Priority
77
SR SA DT RT FOV VC AC OC TC FC MOS ADV
Selection Range (limited)
3
AND Condition Group (2) OwnHunger|TargetIsDestroyed U E DEL
O T A L E S OwnHunger > 50 DYN A U E DEL
AND O T A L E S TargetIsDestroyed IS FALSE U E DEL
Add Condition
ADD
Add Condition Group
ADD CLEAR
Target Move Specification (same as in Act One)
OVERRIDE ENABLED
Target Events
ADD CLR ENABLED
Creature Influences
OVERRIDE
Creature Messages
ENABLED
Creature Behaviour (ATTACK)
ADV SEL
Add Interaction Rule for 'baseMale'
ADD
ICE CREATURE CONTROL v1.3

The above-example shows an Interactor with 3 rules to realize a typical attack scenario. The creature will SENSE its prey within a distance of 25 meter but only if its hunger will be over 50, otherwise the creature will ignore the prey for now. Also the creature will ignore this prey if there is another target with a higher priority.

If the prey comes closer than 20 meter the creature will start to hunt it, but also only if its hunger will be over 50 and if there is no other target with a higher priority.

Finally the creature will attack the prey if it is within a range of 3 meter and it will stop its attack if the prey can leave the range (than creature will hunt it again) or if the prey will be dead.

You could do now a fourth rule in which the creature will eat the killed prey also the creature could loot the dead body of the prey. You can design also much more complex rules by using the Advanced Selection Criteria to define different scenarios and vice versa you could also simplify these tree rules by combining the essential acts in one. Here the creature will directly start hunting and will attack its prey if it is within the stopping distance.



8 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.

8.1 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.

8.1.1 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.

8.1.2 Surface Rule

8.1.2.1 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.

8.1.2.2 Interval

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

8.1.2.3 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.

8.1.2.4 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.

8.2 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.

8.2.1.1 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.

8.2.1.2 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

8.2.2 Tag

8.2.3 Layer

8.2.4 Body Part

9 ELEMENTS

9.1 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.

9.1.1 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.)

9.1.2 Target Selection Criteria *(IMPROVED)*

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.

9.1.2.1 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 valid targets with the same priority, the creature will select the nearest one and in case of standoff gaps the active target will be selected by chance.

9.1.2.2 Selection Options

In addition to the Selection Priority you can refine the desired selection criteria with additional options and complex conditions rules.

9.1.2.2.1 Selection Range (SR)

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.

9.1.2.2.2 Selection Angle (SA)

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.

9.1.2.2.3 Retaining Time (RT)

The Retaining Time defines the time-span in which an active target will stay selected even if the given selection scenario failed. The Retaining Time is helpful to avoid flickering target changes, caused by changeable and/or unsteady conditions, which squeezes the creature to swap quickly between two

or more targets. Aside from that, a small coasting also refines the realistic behaviour of the creature, because a real creature needs time to evaluate situations and to correct its course of motions as well.

9.1.2.2.4 Delay Time (DT)

The Delay Time defines the minimum time-span in which an inactive target have to wait before it can be activated if all other conditions will be fulfilled. The Delay Time is helpful to avoid unwanted change cycles, caused by changeable and/or unsteady conditions, which squeezes the creature to swap quickly between two or more targets.

9.1.2.2.5 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.

9.1.2.2.6 Visibility Check (VC)

While Visibility Check is active the target must be visible for the 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.

9.1.2.2.7 Audibility Check (AC)

While the Audibility Check is active the target must be hearable for the creature.

9.1.2.2.8 Odour Check (OC)

While the Odour Check is active the target must be smellable for the creature.

9.1.2.2.9 Tactile Check (TC)

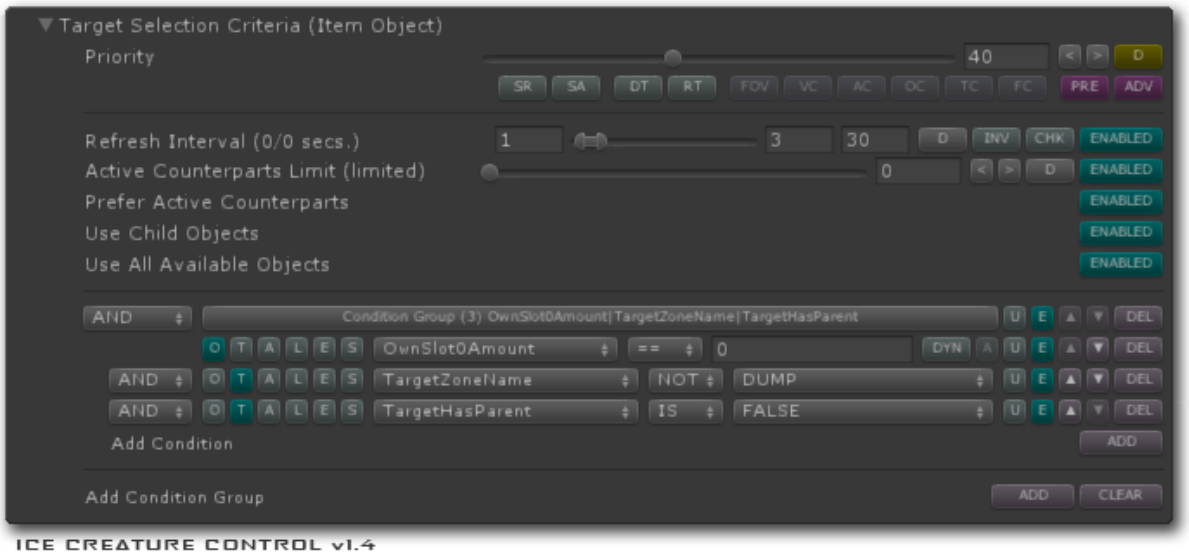
While the Tactile Check is active the target must be palpable for the creature.

9.1.2.2.10 Flavour Check (FC)

While the Flavour Check is active the target must be tasty for the creature.

9.1.2.2.11 Pre-Selection (PRE) *(NEW)*

Basically the creature will preselect a GameObject for each defined target before running the final target selection process for all available targets. By using the 'Pre-Selection' option you can adapt several conditions to optimize the pre-selection process. Please note that changes of the preselection settings are normally only useful if several target objects of the same type are available simultaneously during the runtime.



9.1.2.2.11.1 Refresh Interval ***(NEW)***

The TargetRefreshInterval defines a time delay in seconds, in which the creature looks for matching target game objects. If the TargetRefreshInterval is disabled or the time span is adjusted to zero, the creature will search and replace appropriate target game objects according to the given perception time.

Please note: especially if a creature has to handle many targets of the same type, it is recommended to use the TargetRefreshInterval to increase the performance.

9.1.2.2.11.2 Active Counterparts Limit ***(NEW)***

By default, a creature always selects the next best target, which can cause that too many creatures are focused on the same object while ignoring surrounding target-options. By activating and adapting the 'Active Counterpart Limit' the creature takes into account how many other objects have currently selected the target and will only select it if the number of the active counterparts is below the specified limit. If the limit is adjusted to -1 this feature will be ignored. This feature works similar to the ActiveCounterpartsLimit expression of the advanced conditions.

9.1.2.2.11.3 Prefer Active Counterparts ***(NEW)***

By default, all available objects of the given type are taken into account in the target preselection, but in some cases, it may be helpful to favor specific objects which have already selected our creature as their own target, so our creature can react to objects, which have already a clear focus to our creature (e.g. an active attacker or something like this). By activating the 'Prefer Active Counterparts' option, the target selection procedure will favor all objects that have been selected our creature as their own active target.

9.1.2.2.11.4 Use Child Objects ***(NEW)***

By default, potential target objects within the creature's hierarchy will be ignored, but in some cases (e.g. for weapons, tools or other useable objectes) it might be useful to allow such subordinated child objects as a target, so you can enable this feature by activating the 'Allow Child Objects' option.

9.1.2.2.11.5 Use All Available Objects (NEW)

By default the creature will preselect the next best GameObject for each defined target and will run the final target selection with the advanced conditions for all available target groups just with such pre-selected objects.

This is suitable and performance friendly while using many different types of target objects and in the majority of cases the nearest one will be most likely the right one.

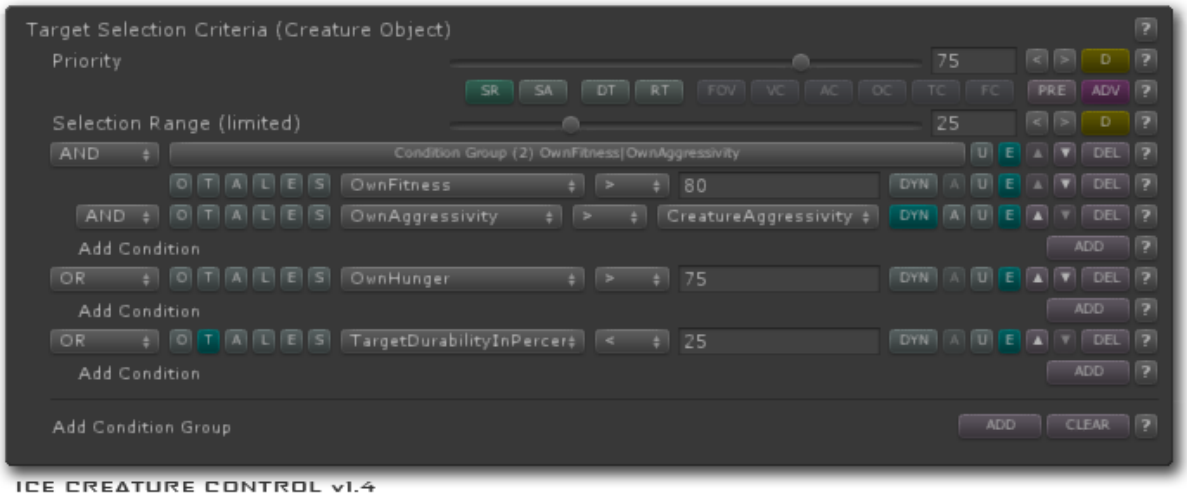
But in cases where multiple GameObjects of the same type are available, the Advanced Target Selection Criteria will check object-related conditions only for the above-mentioned preselected GameObject, which may lead to confusing and misleading behaviour since other potential target candidates are apparently ignored.

To avoid such cases you can activate 'Use All Available Objects' to consider all available GameObjects of the given type during the final target selection process.

Please note that this feature is not required whenever the specified TargetGameObject will be the only one of its type or if there are no object-related conditions defined in the Target Selection Criteria, such as distance, direction or other individual attributes.

9.1.2.2.12 Advanced Target Selection Criteria

In addition to the standard selection criteria, you can use the advanced criteria to refine the conditions. The Advanced Target Selection Criteria provides you to create combined constructs of conditional expressions by using a large set of runtime parameter.



You can find further information about the Advanced Target Selection Criteria in the annexe at the end of this manual.

9.1.3 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.

9.1.3.1 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.

9.1.3.1.1 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.

*9.1.3.1.1.1 DYN Button (Min, Max, Speed) **(NEW)***

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

*9.1.3.1.1.2 RND Button **(NEW)***

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

9.1.3.1.2 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.

*9.1.3.1.2.1 DYN Button (Min, Max, Speed) **(NEW)***

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

*9.1.3.1.2.2 RND Button **(NEW)***

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

9.1.3.2 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.

9.1.3.2.1 Update On Select

While using a random position you can define the update conditions. Update On Select will refresh the position whenever the target will be preselected.

9.1.3.2.2 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.

9.1.3.2.3 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.

9.1.3.2.4 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.

9.1.3.3 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).

9.1.3.3.1 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

9.1.3.3.2 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.

9.1.3.4 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.

9.1.4 Target Events *(NEW)*

Target Events allows your creature to send messages to the GameObject of an active target. You could use this feature for example to call a damage handler or to activate specific functions (e.g. open a door etc.). To use the Target Events enable this feature and ADD your desired events.

9.1.4.1 Method Name

By default you can use the popup list to select a desired method. The popup displays all available Behaviour Events of the target but you can also use the CUSTOM option to enter custom method names and parameter values (here please make sure to use the correct name and parameter type).

9.1.4.2 Timer

9.1.4.2.1 Start Time

9.1.5 Creature Influences

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)

9.1.6 Creature Messages *(IMPROVED)*

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

9.1.7 Creature Behaviour *(IMPROVED)*

Whenever your creature select a target it needs to know how to react to the target or rather how to handle the changed situation. A typical example would be to set a RUN behaviour so your creature will try to reach the Target Move Position of the target. That's absolutely suitable as long as the target is moving or if your creature is using a dynamic position around the target (see Random Positioning Range) or basically as long as your creature will hunting but never reach the given Target Move Position, because in cases the creature reaches the target a RUN behaviour could force your creature to spinning around the target. To avoid such a misbehaviour you can enable the advanced option (ADV button) to define a suitable Rendezvous Behaviour for cases your creature will reach the Target Move Position or rather the given Stopping Distance.

9.1.7.1 Advanced Option (ADV Button) (NEW)

The ADV Button enables or disables the advanced options, which allows your creature to use an additional Rendezvous Behaviour for cases the creature has reached the Stopping Distance of the given Target Move Position.

9.1.7.2 Selective Option (SEL Button) (BETA)

The SEL Button enables or disables the use of condition based selective behaviours. This feature is currently under construction and not available in the given version.

9.1.7.3 Standard Behaviour

The Standard Behaviour defines the default behaviour for the selected target. The creature will use this behaviour whenever this target is active.

9.1.7.4 Rendezvous Behaviour (NEW)

The Rendezvous Behaviour defines the behaviour which will be used in cases your creature has reached the Target Move Position or rather the given Stopping Distance. To use the Rendezvous Behaviour you need to enable the advanced option by using the ADV button.

9.1.7.5 Target Tutorials

9.1.7.5.1 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.

9.1.7.5.2 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.).

9.2 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.

9.2.1 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.

9.2.1.1 Rename

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

9.2.1.2 Copy

Creates a copy of the selected Behaviour Mode

9.2.1.3 Remove

Removes the selected Behaviour Mode

9.2.1.4 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.

9.2.1.4.1 Priority

9.2.1.4.2 Minimum Period

9.2.1.4.3 Next Move Position

9.2.1.4.4 Target Move Position

9.2.1.4.5 Specific Target

9.2.1.4.6 Detour

9.2.2 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.

9.2.2.1 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.

9.2.2.2 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.

9.2.2.2.1 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.

9.2.2.2.2 Animation Events *(NEW)*

AnimationEvent lets you call a script function similar to SendMessage as part of playing back an animation. Animation events support functions that take zero or one parameter. The parameter can be a float, an integer or a string. In cases you would like to use also events with an object reference or an AnimationEvent you have to define such events directly in the Animation Window.

Please note that Animation Events calls their methods on MonoBehaviours of the 'animated' GameObject only, if you want to call a method within one of its children you could use the Methods feature instead. Also please consider that AnimationEvents will be assigned directly to the associated animation and will try to call their defined methods on each GameObject which used these animation, so please be careful by defining such events.

9.2.2.2.2.1 Event

Use the Event section to define as much events as you want. For this simply enable the section and ADD a new event. Disabling the section will deactivate all listed events, which means that all defined events will be activated in the list and removed from the animation. To reset the complete Event section press the RES button, this removes all listed events completely.

9.2.2.2.2.1.1 Event Popup

The Event Popup provides you a preselected list with available methods. These methods represents registered PublicMethods which are directly related to the game play and suitable to steering movements and/or behaviour. (see also: ICE World, PublicMethods)

In addition to the listed methods, you can activate CUSTOM to enter arbitrary function names, in doing so you can define each available function you want.

By default a new created event will be inactive and not assigned to the animation, so you have to activate ACTIVE to assign an event to the selected animation, also deactivate the ACTION flag to remove an event from the animation. To remove obsolete events completely from list simply press the X button, this will removes both the listed event template and the assigned AnimationEvent.

9.2.2.3 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.

9.2.2.3.1 Velocity

9.2.2.3.1.1 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.

9.2.2.3.1.2 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)

9.2.2.3.1.3 Inertia (Mass Inertia)

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

9.2.2.3.1.4 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.

9.2.2.3.1.5 Vertical Velocity

If vertical movements enabled you can use the Vertical Velocity to define the desired speed along the y-axis. Your creature will use this velocity to reach its given altitude (Operating Level).

9.2.2.3.1.5.1 Operating Level

In addition to the Vertical Velocity you have to define also the desired operating level which represents the desired altitude your creature will try to reach. If you define different minimum and maximum values, the final altitude will be randomized if the rule becomes active. By default the altitude represents the absolute level above zero, by activating the 'GND' button (Ground) the creature will use the altitude above the given ground level and will follow the contour map.

9.2.2.3.2 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.

9.2.2.3.3 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.

9.2.2.4 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.

9.2.2.4.1 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.

9.2.2.4.2 Damage

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

9.2.2.4.3 Stress

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

9.2.2.4.4 Debility

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

9.2.2.4.5 Hunger

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

9.2.2.4.6 Thirst

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

9.2.2.5 Inventory

9.2.2.5.1 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.

9.2.2.5.2 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

9.2.2.5.2.1 Item

9.2.2.5.2.2 Interval

9.2.2.5.3 Equip

9.2.2.5.3.1 Item

9.2.2.5.3.2 Parent

9.2.2.6 Audio

9.2.2.7 Events (NEW)

Events allows you to call public script function within your creature's hierarchy, so you can use this feature to run specific behaviour procedures (e.g. FireOneShot or FlashlightON etc.) in addition to the internal behaviour functionality.

Events support functions that take zero or one parameter. The parameter can be a Float, an Integer, a Boolean or a String.

To use this feature simply add a new message and choose the desired method by using the popup or activate the CUSTOM flag to enter the desired method by hand. The method must be a member of a MonoBehaviour script, which is assigned to an enabled child object of your creature.

Tip: The used Event object based on the ICEWorld BehaviourEvent. If you are using ICEWorldBehaviour as base class for your own component, you could register your own events by overriding the OnUpdateBehaviourEvents lists, so your own events will be automatically display in the event popup.

AnimationEvent lets you call a script function similar to SendMessage as part of playing back an animation. Animation events support functions that take zero or one parameter. The parameter can be a float, an integer or a string. In cases you would like to use also events with an object reference or an AnimationEvent you have to define such events directly in the Animation Window.

9.2.2.8 Look

9.2.2.9 Effect

9.2.2.10 Link

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

9.3 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.

9.3.1 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.

9.3.1.1 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.

9.3.1.2 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.

9.3.1.3 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.

9.3.1.4 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.

9.3.1.5 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.

9.3.1.6 Tutorials Move

9.3.1.6.1 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.

9.3.2 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.

9.3.2.1 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.

9.3.2.2 Random Move

9.3.2.3 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.

9.3.2.4 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.

9.3.2.5 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.

9.4 TIMER

Many features needs to be called at the right time, continuously and/or clocked by using complex pulse repetition frequencies. To fulfil all these requirements ICE provides a powerful timer, which is embedded in many control elements such as Audio, Events, Effects, Influences etc. This Impulse Timer allows you to define a flexible time schedule, so you could use this timer to fire a single timed event, multiple continuous events within a specified interval or randomized impulse sequences as well.

9.4.1 Start Time

The Start Time defines the timespan in seconds from the initial activation of a feature to firing the first impulse. If Start Time is adjusted to zero, the impulse will be fired directly at the activation of a feature or rather while END button is flagged at the end of an active feature. In both cases there will

be a single impulse only. If you want more than one impulse you can activate the Interval Function by pressing the INT button.

9.4.2 Impulse Interval

The Impulse Interval defines the minimum and maximum timespan between two impulses. If minimum and maximum are different, the interval will be randomized within the specified range, otherwise the interval will be used as defined. You could use the RND button to generate randomized settings or the D button to adjust the values back to zero. Please note, if the Impulse Interval is adjusted to zero, the impulse will be fired in each frame.

9.4.3 Impulse Limit

The Impulse Limit defines the total number of impulses. If you define different minimum and maximum values the limit will be randomized, otherwise the limit will be used as defined. You could use the RND button to generate randomized values or the D button to adjust the values back to zero. Adjusting the limit to zero will disable the limit.

9.4.4 Sequence Limit

While the Impulse Limit defines the total number of impulses, the Sequence Limit specifies the number of sequenced impulses until the next break. Sequence Limit will be available only as long as the Impulse Interval is not adjusted to zero. If you define different minimum and maximum values the limit will be randomized, otherwise the limit will be used as defined. You could use the RND button to generate randomized values or the D button to adjust the values back to zero. Adjusting the limit to zero will disable the Sequence Limit. Please consider to adjust also the Break Length if you define the Sequence Limit, otherwise the Sequence Limit will not work as expected if the Break Length is adjusted to zero.

9.4.5 Break Length

While the Sequence Limit defines the number of sequenced impulses until the next break, the Break Length defines the timespan in seconds of the subsequent interruption. The Break Length option will be available only if Sequence Limit is not adjusted to zero. If you define different minimum and maximum values the timespan will be randomized, otherwise the Break Length will be used as defined. You could use the RND button to generate randomized values or the D button to adjust the values back to zero. Adjusting the Break Length to zero will disable this option.

9.4.6 Examples

9.4.6.1 Starting an effect 5 seconds after the activation of the associated behaviour

- Set the Start Time of the effect timer to 5

9.4.6.2 Starting an effect 5 seconds and repeat it every 3 seconds

- Set the Start Time of the effect timer to 5
- Activate the INT button to enable the interval function of the timer
- Set the Impulse Interval to 3 (this will force an impulse each third second)

9.4.6.3 Starting an effect at the end of the associated behaviour

- Activate the END button of the effect timer

9.4.6.4 Simulate the ticking of a clock

- Activate the INT button of the Audio section to enable the interval function of the timer
- Set the Impulse Interval to 1 (this will fire an impulse each second)

9.4.6.5 Simulate a Machinegun sound

- Activate the INT button of the Audio section to enable the interval function of the timer
- Set the Impulse Interval to 0.5f (this defines the fire rate of the gun)
- Set the Sequence Limit to 3-5 (this defines the burst rate)
- Set the Break Length to 1-2 (this defines the interruption between the burst sequences)

To limit the total number of fire shots, set the Impulse Limit to the desired value.

9.5 EVENTS

Events allows your creature to call external functions within its own component hierarchy or the component hierarchy of a target (Target Events only)

9.6 IMPACT

Impact contains the settings to define the consequences for other entities while colliding with an Item or a BodyPart object in your scene. In addition to the Damage and Force values you can define Sounds and an Effect which will be used during the impact process.

9.6.1 Damage Transfer

The Damage Transfer Type defines the mode how the impact damage will be sent to the involved GameObjects.

9.6.1.1 Direct

While the Direct Mode is selected the specified damage and force values will be sent by calling the AddDamage method of an ICEWorldEntity class directly.

9.6.1.2 Message

While the Message Mode is selected the damage value will be sent by using SendMessageUpwards with the specified method name and float value.

9.6.1.3 DirectOrMessage

While the DirectOrMessage Mode is selected, the impact handler will try to call the AddDamage method and will use SendMessageUpwards method only in cases the first procedure failed.

9.6.1.4 DirectAndMessage

While the DirectAndMessage Mode is selected, the impact handler will try to call the AddDamage method and in addition also the SendMessageUpwards method. Please note, that this will results in a double damage for each affected ICEWorldEntity object while using ‘ApplyDamage’ as method name.

9.6.2 Damage Method

Damage Method defines the desired method name which shall be used to send the damage data.

9.6.3 Damage Value

Damage Value defines the desired damage value.

9.6.4 Force Type

The Force Type specifies the kind of the impact forces.

9.6.4.1 None

9.6.4.2 Direction

9.6.4.3 Explosion

9.6.5 Energy

Energy specifies the desired minimum and maximum energy range of the force.

9.6.6 Explosion Radius

Explosion Radius defines the radius in which objects will be affected.

9.6.7 Sound

Sound defines the desired audio clips which shall be used during an impact.

9.6.8 Effect

Effect defines the desired effect which shall be used during an impact.

9.6.9 Behaviour

The Behaviour section contains optional settings to optimize and adapt the impact according to the given object or rather your individual needs and requirements.

9.6.9.1 Layer Mask

The Layer Mask allows you to restrict the impacts to one or more specified layer.

9.6.9.2 Destroy On Hit

If DestroyOnHit is enabled the entity will be destroyed subsequent to processing the impact procedure.

9.6.9.2.1 Delay

Delay defines the delay time to destroy the object.

10 REGISTER

ICECreatureRegister 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.

10.1 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.

10.1.1 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.

10.1.1.1 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.

10.1.1.2 Player

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

10.1.1.3 Creature

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

10.1.1.4 Items

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

10.1.1.5 Locations

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

10.1.1.6 Waypoints

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

10.1.1.7 Marker

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

10.1.1.8 Other

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

10.1.1.9 Reorganize Hierarchy

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

10.1.2 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.

10.1.2.1 Spawn Ground Check

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

10.1.2.2 Ground Layer

10.1.2.3 Spawn Obstacle Check

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

10.1.2.4 Obstacle Layer

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

10.1.3 Use Scene Management

While UseSceneManagement is flagged

10.1.3.1 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.

10.1.3.2 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.

10.1.4 Use Debug

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

10.1.4.1 Draw Selected Only

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

10.1.4.2 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.

10.1.4.3 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.

10.1.4.4 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.

10.2 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)

10.2.1 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.

10.2.2 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.

10.2.2.1 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.

10.2.2.2 Pool Management

10.2.2.2.1 Max. Spawn Objects

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

10.2.2.2.1.1 INITIAL button

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

10.2.2.2.1.2 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.

10.2.2.2.2 Spawn Interval

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

10.2.2.2.3 Random Size

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

10.2.2.2.3.1 Size Variance

Use 'Size Variance' to define the minimum and maximum limits.

10.2.2.2.4 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.

10.2.2.2.5 Use Hierarchy Group

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

10.2.2.2.5.1 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.

10.2.2.2.6 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 object types the position of their given reference objects, but you are free to define as much points as you want; if multiple points available the selection will be randomized.

10.2.2.2.6.1 SpawnPoint

SpawnPoint defines the centre of the spawn area, which could be adapt by modifying the Random Range values. By default your creatures will using its HOME location as spawn point, but basically a SpawnPoint could also be each static or movable GameObject (e.g. a static building or your player as well).

10.2.2.2.6.1.1 Random Range

RandomRange could be used to define a minimum and maximum radius around the given spawn point to modifify the final spawn area. If both RandomRange values (minimum/maximum) are adjusted to zero the spawn position will be the transform position of the given spawn point.

10.2.2.2.6.1.2 Level Difference

LevelDifference defines the maximum altitude difference between the centre position and all potential spawn points within the given Random Range. Centre Position (y) + Level Difference will be the origin of the ray to detect the correct ground level, so LevelDifference will be only visible if the GroundCheckType is adjusted to RAYCAST.

10.2.2.2.7 Add Spawn Point

10.2.3 ICECreatureControl (CC)

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

10.2.4 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.

10.2.5 ICECreatureLocation (CL)

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

10.2.6 ICECreatureWaypoint (CW)

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

10.2.7 ICECreatureMarker (CM)

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

10.2.8 ICECreatureItem (CI)

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

11 COMPONENTS

In addition to the ICECreatureControl and the ICECreatureRegister you will find further components which you could use to prepare species-appropriate and familiar habitats for your creatures.

11.1 ITEMS

Items are ICECreatureEntities which represents moveable and collectable objects your creatures can collecting, owning and using during the runtime.

Each Item owns an Inventory which could contain other items, also each item provides impact features and could be used as melee weapon.

11.1.1 ICECreatureItem

ICECreatureItem represents an unspecified Item. You should add the ICECreatureItem script to each moveable object which should be collectable and/or useable for your creatures, if there is no higher item script available which fulfils the object related functionality.

11.1.1.1 Features

- Status
- Inventory
- Impact

11.1.2 ICECreatureTool

ICECreatureTool represents driven tools, such as a drilling machine, a vacuum cleaner or a jack-hammer etc. In addition to the standard item attributes, ICECreatureTool comes with switchable behaviours for Off, Standby and Operate which can be activated by pressing a key (in cases the player is using the tool) or by calling the associated public methods.

11.1.2.1 Features

- Status
- Inventory
- Impact
- Tool

11.1.3 ICECreatureFlashlight

ICECreatureFlashlight represents a Flashlight your creature can use during the runtime.

11.1.3.1 Features

- Status
- Inventory
- Impact
- Flashlight

11.1.4 ICECreatureTorch

ICECreatureTorch represents a Torch your creature can use during the runtime.

11.1.4.1 Features

- Status

- Inventory
- Impact
- Fire

11.1.5 ICECreatureMeleeWeapon

ICECreatureMeleeWeapons represents all types of melee weapons.

Please note that each item could be used as melee weapon.

11.1.5.1 Features

- Status
- Inventory
- Impact

11.1.6 ICECreatureRangedWeapon

ICECreatureRangedWeapon represents all types of ranged weapons, such as pistols, guns, assault-riffles, cannons etc.

11.1.6.1 Features

- Status
- Inventory
- Impact
- Primary Weapon
- Secondary Weapon
- Flashlight
- Laser

11.1.7 ICECreatureTurret

ICECreatureTurret represents a full automatic ranged weapon which can detect and destroy targets without external steering commands. Your player or your creatures can place such weapon systems to protect specific locations and areas.

11.1.7.1 Features

- Status
- Inventory
- Impact
- Primary Weapon
- Secondary Weapon
- Flashlight
- Laser
- Turret

11.1.8 ICECreatureProjectile

ICECreatureProjectile represents each kind of projectile which will be used as ammunition for ranged weapons. You will need projectiles while using ranged weapons with the Ammunition Type ‘Projectile’ or ‘Ballistic Projectile’. In both cases the weapon handles the shooting procedure only while the projectile handles the final impact damage.

11.1.8.1 Features

- Status
- Inventory
- Impact
- Projectile

11.1.9 ICECreatureExplosive

ICECreatureExplosive represents each kind of pyrotechnic materials.

11.1.9.1 Features

- Status
- Inventory
- Impact

11.1.10 ICECreatureMine

ICECreatureMine represents a land mine.

11.1.10.1 Features

- Status
- Inventory
- Impact

11.2 OBJECTS

Objects represents static scene elements, such as buildings or immovable building parts (e.g. doors and windows). Objects can be used by a creature but not collected or owned

11.3 LOCATIONS

Locations represents spatial areas which your creatures can use to navigate and to orient themselves

11.3.1 ICECreatureOrganism

ICECreatureOrganism represents any kind of life-forms, such as animals and humans but also cybernetic organism and plants as well.

11.3.1.1 ICECreatureControl

11.3.1.2 ICECreaturePlayer

11.3.1.3 ICECreaturePlant

11.3.2 ICECreatureObject

ICECreatureObject represents any kind of unanimated objects, such as usable items and environment elements.

11.3.2.1 ICECreatureAnimatedObject

11.3.2.2 ICECreatureItem

ICECreatureItem represents any kind of moveable and collectable objects

11.3.2.2.1 ICECreatureWeapon

11.3.2.2.1.1 ICECreatureRangedWeapon

11.3.2.2.1.2 ICECreatureMeleeWeapon

11.3.2.2.1.3 ICECreatureBullet

11.3.2.2.2 ICECreatureFlashlight

11.3.3 ICECreatureLocation

11.3.3.1 ICECreatureWaypoint

11.3.3.2 ICECreatureMarker

11.4 BODY

11.5 DEBUG

The Debug Folder contains additional debug scripts which will be automatically used by the different components to draw their gizmos or to display further information.

11.5.1 ICECreatureControlDebug

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.

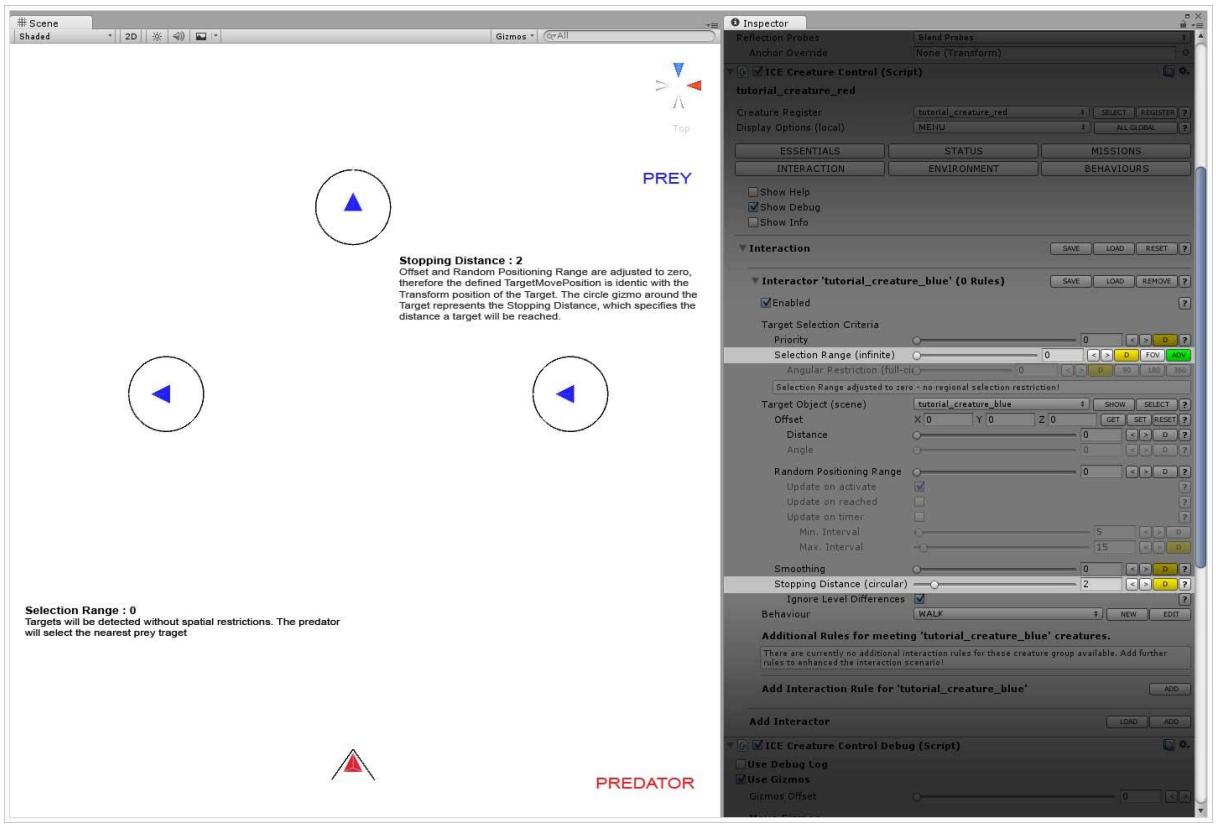
11.5.2 Path and Destination Pointer

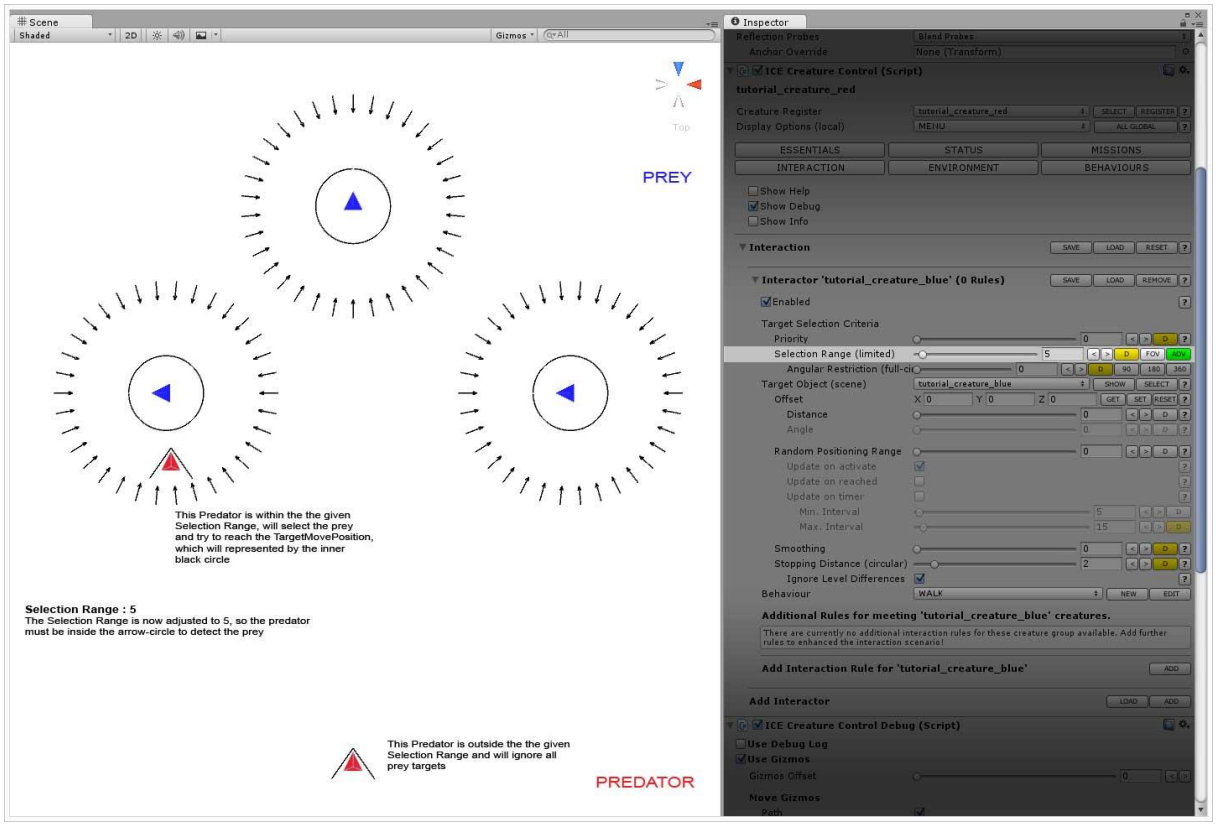
The Path and Destination Pointer are runtime features to display the current path and the final move position of your creature. The Pointer are primitive objects, which you can customize, so that you can easily tell them apart.

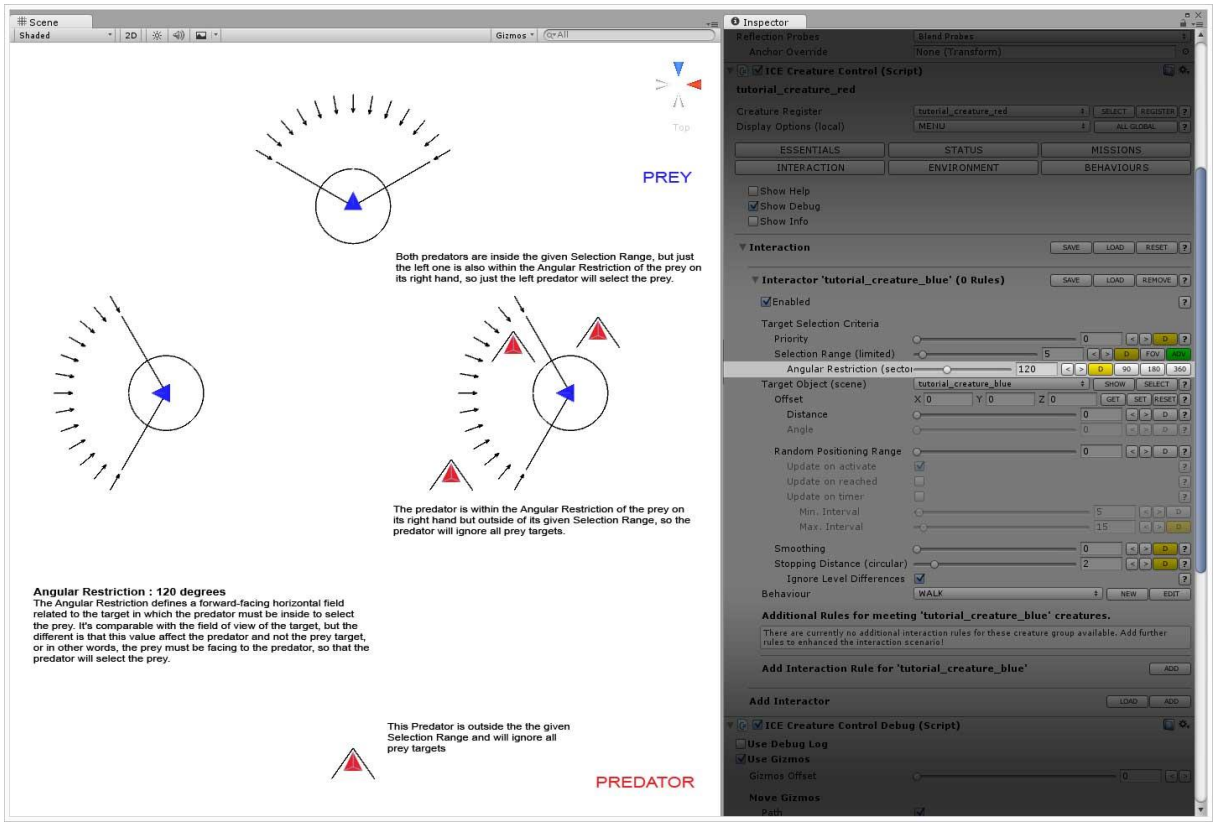
11.5.3 Debug Log

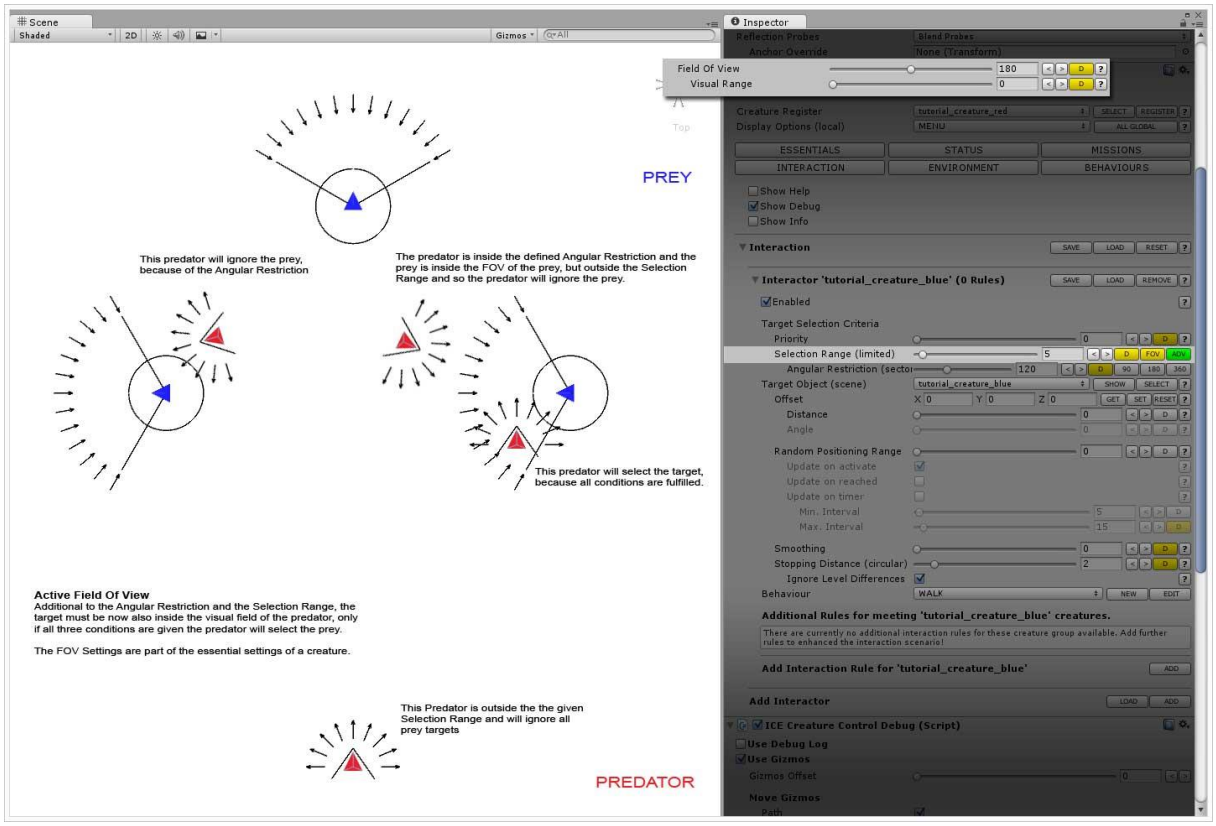
The debug information informs you about the status of your creature, so you can monitoring the active targets, behaviours and behaviour rules.

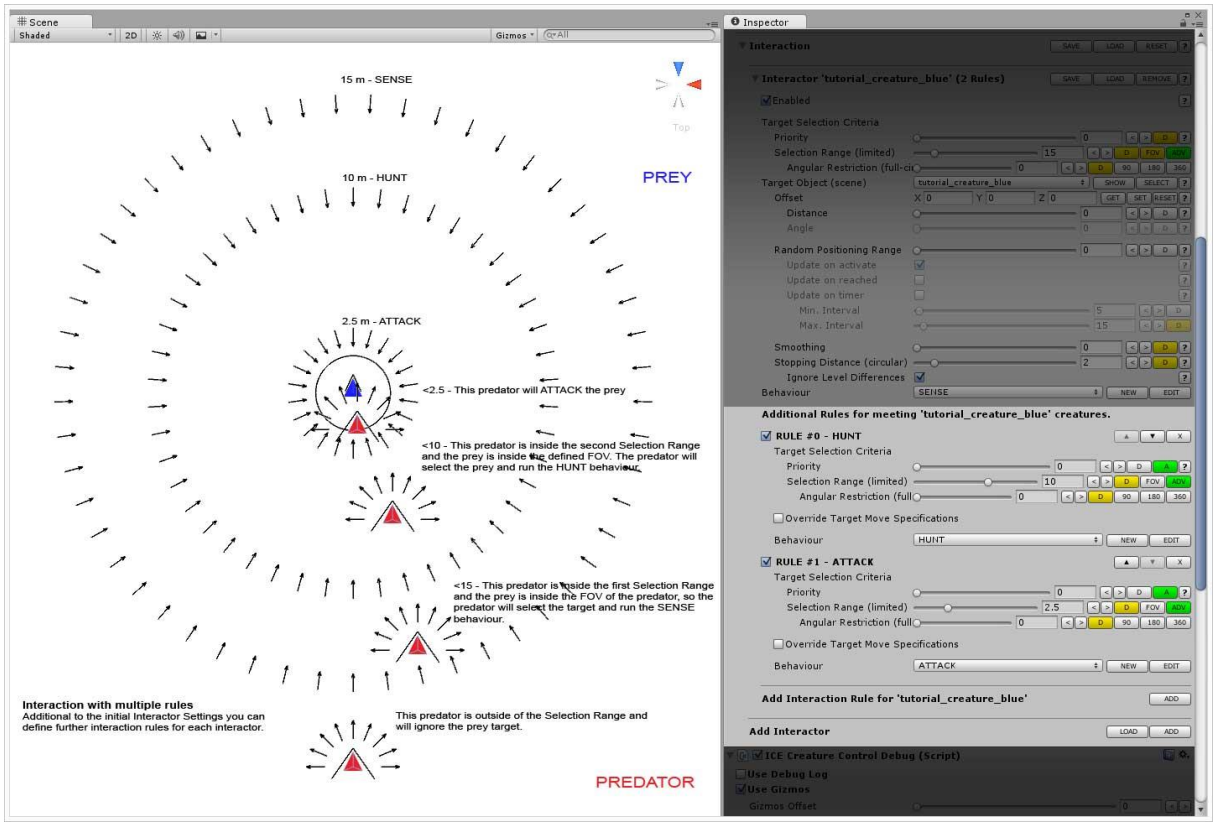
11.5.4 Gizmos




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





11.5.5 ICECreatureRegisterDebug

Small script without editable options to draw the register gizmos.

11.6 EXTENSIONS

11.6.1 ATTRIBUTES (beta)

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.

11.6.1.1 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.

11.6.1.2 ICECreatureInfluencesAttribute

11.6.1.3 ICECreatureSelectionAttribute

11.6.1.4 ICECreatureOdourAttribute

11.6.2 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.

11.6.2.1 ICECreatureInventoryExtension (beta)

11.6.3 MISSIONS (coming soon)

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

11.6.3.1 ICECreatureMissionTemplate (coming soon)

Template script to develop custom missions.

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

12 SOLUTIONS

12.1 DAMAGE HANDLING

12.1.1 Basics

The basic Damage Handling is part of the ICEWorldEntity, which is the base class of all ICE components, so each ICE object can be damaged and destroyed. To damage an entity simply call its ApplyDamage method with the desired damage value …

```
public virtual void ApplyDamage( float _damage )
```

… or if you want more control you can use the AddDamage method, which provides you more parameter to steering the damage behaviour.

```
public virtual void AddDamage(
      float _damage,
      Vector3 _damage_direction,
      Vector3 _damage_position,
      Transform _attacker,
      float _force = 0 )
```

By using the above-mentioned methods you can damage each entity, just make sure that Status and Durability of the entity are enabled. The Durability represents the 'Health Points' of an object. By default the Initial Durability is adjusted to 100 but you can adapt this value as desired. If you are define different min and max values, the initial durability will be adjusted automatically by chance according to the defined values.

If an entity object is attached to another entity you can activate and use the Damage Transfer Multiplier. If this feature is active incoming damage calls will be forwarded to the parent entity by using the specified multiplier value to increase or decrease the received damage value. You can use this for example to equip your character with a bullet proof vest to absorb impacts and decrease incoming damages or you can define specific body parts of your creature by using the

ICECreatureBodyPart component to adapt the received damage according to the hit point. However, you'll find such a status for each entity and if you don't want to use it simply deactivate it and your entity can't be destroyed.

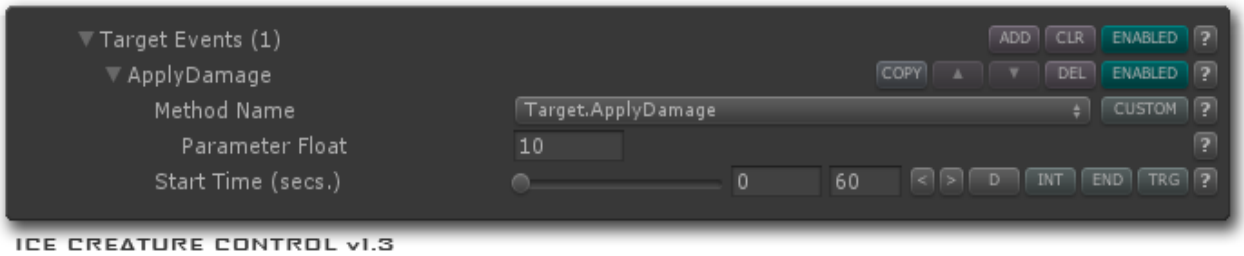
12.1.2 How a Creature can damage its targets

By default your ICECreatureControl creatures will have a peaceful nature and will be unprotected and defenceless, but there are several ways to change this.

12.1.3 Target Events

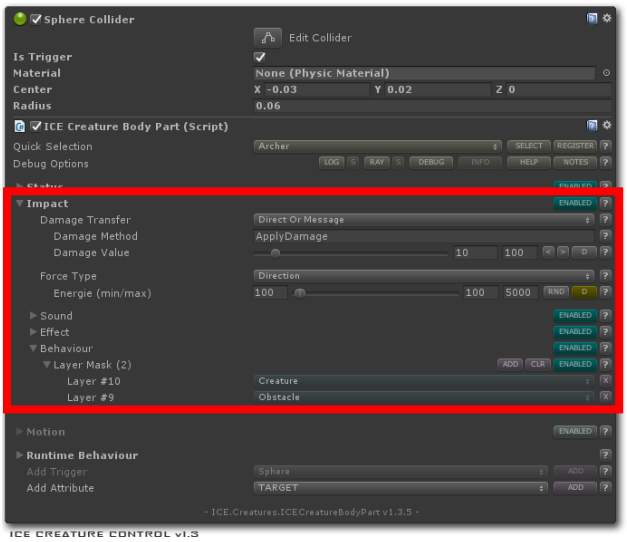
The easiest way to make your creature damaging other objects are Target Events. You'll find such an event section within the Target Settings of your creatures, just enable the feature, add a new event and select the desired method name and value. In addition to that you can also specify the interval settings to repeat the specified event as desired. If you want to sync the event with an animation you can activate the 'TRG' flag to trigger the event by an AnimationEvent. Simply set 'TriggerTargetEvent' with the desired event index parameter (btw. you can handle this in the Animation Section of the target related behaviour).

By default you'll get a menu with suggested methods and if your target contains an entity component you'll find here the above-mentioned ApplyDamage method, but you can also activate the custom flag to enter your own damage method (e.g. 'Damage' while using UFPS). If this is done your creature will call the specified method whenever the associated target become active.

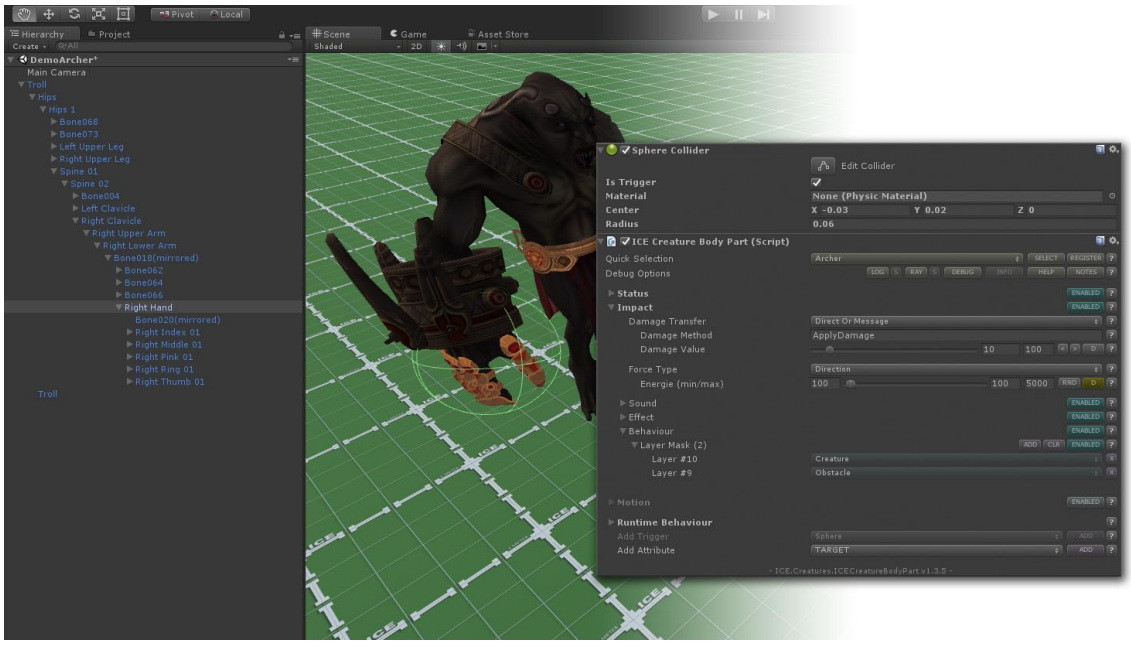


12.1.4 Body Part Impacts

In addition to damaging a target by using the Target Events, which serves the purpose but could be too imprecise for your desired scenario, you can use the Impact feature of your creatures Body Parts. As already mentioned you can define body parts of your creature by using the ICECreatureBodyPart component. Such BodyParts can receive and transfer damage but they can also strike back and deal powerful punches and kicks.



All you need is to select the desired body part within your creature's hierarchy and add the ICECreatureBodyPart component to it. Enable the Impact feature and define the desired damage values. In addition to that you can define also an impact sound and/or an impact effect, also you could define a layer mask to restrict the impacts to specific objects, so your creature will not destroy your complete world like a bull in a china shop. Finally you have to define the dimension of the body part by using a suitable resized collider/trigger. Here please consider that your creature or at least its target requires at least one Rigidbody, otherwise the body part can't detect impacts.

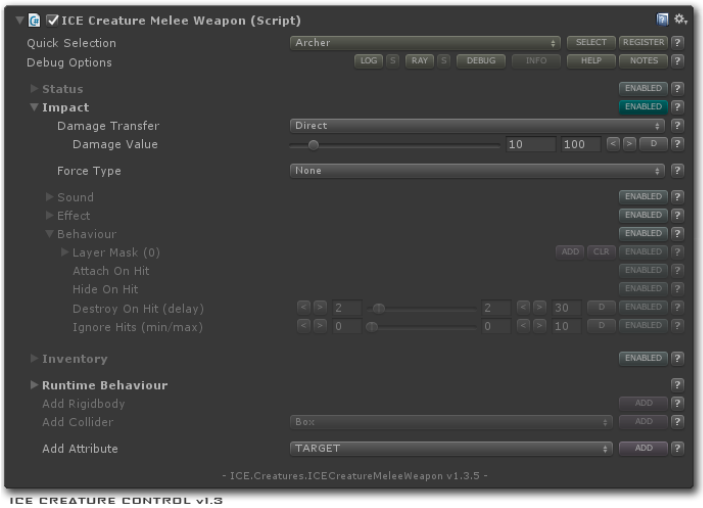


If this is done there is nothing more to do, your creature can use now the prepared body part as weapons and can damage and destroy all and everything, just define the desired attack behaviours and enjoy the brawl.

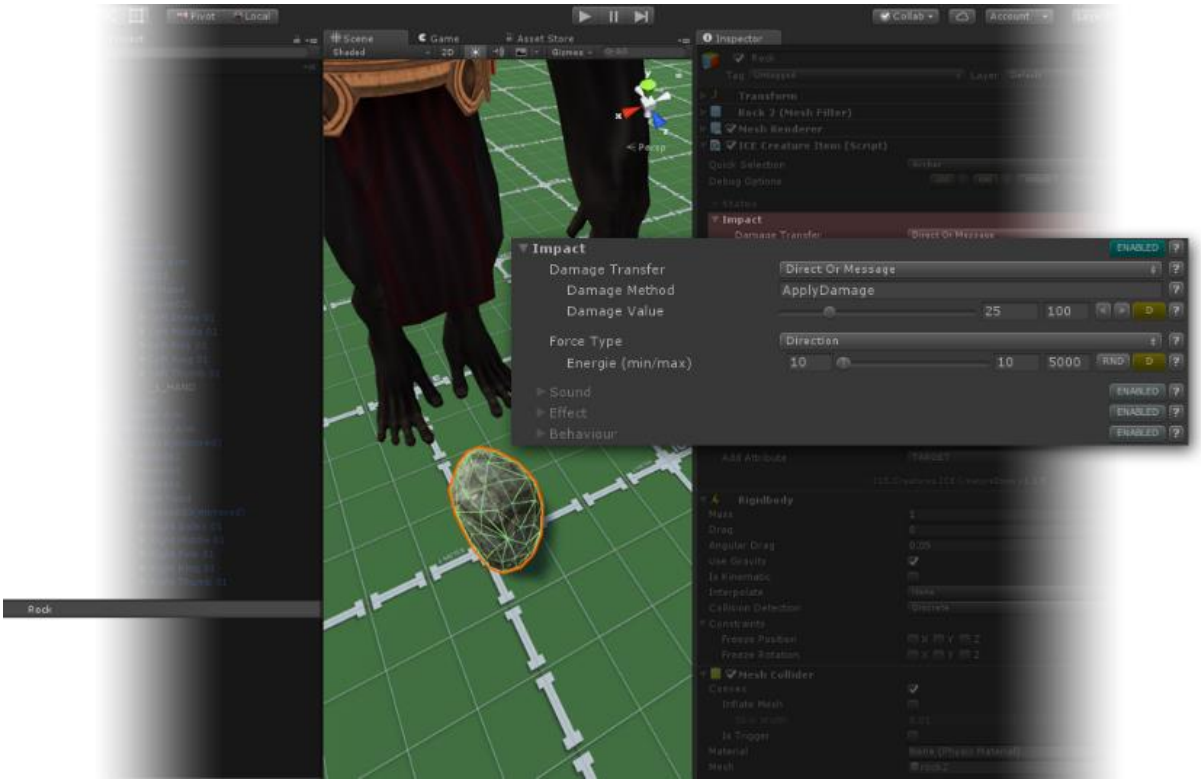
12.1.5 Melee Weapon Impacts

By using Body Parts with the Impact feature your creature is ready and well versed in the art of self-defence, but in addition to that your creature can also use other objects as weapons.

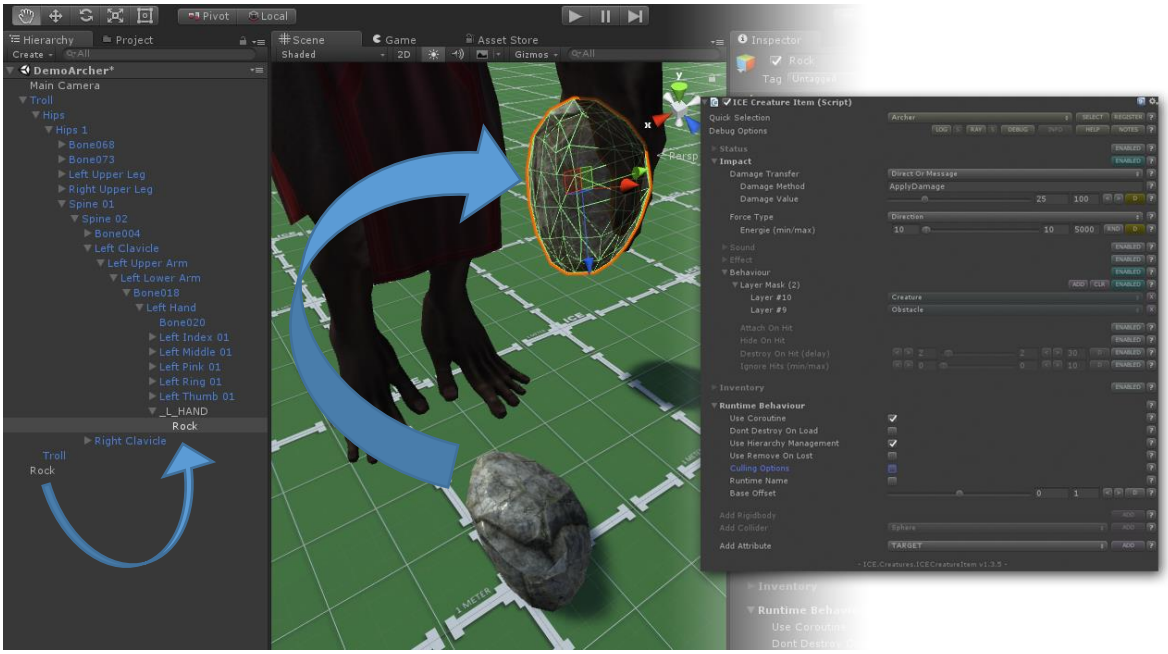
For this purpose you can add the ICECreatureMeleeWeapon component to all your typical melee weapons (e.g. swords, knifes, hatchets, clubs etc.), so your creature can find and use such weapons.



But in principle, similar to the real world, each moveable object could be divert from its intended use to inflict harm or damage on someone or something, therefore each Item or rather each object with a derived ICECreatureItem component contains an Impact section and could be used as weapon.



In this example you can see the Impact Settings of a rock item. Attach now the rock object to the desired hierarchy element (here the left hand) and your creature will be equipped and ready to use its new item.

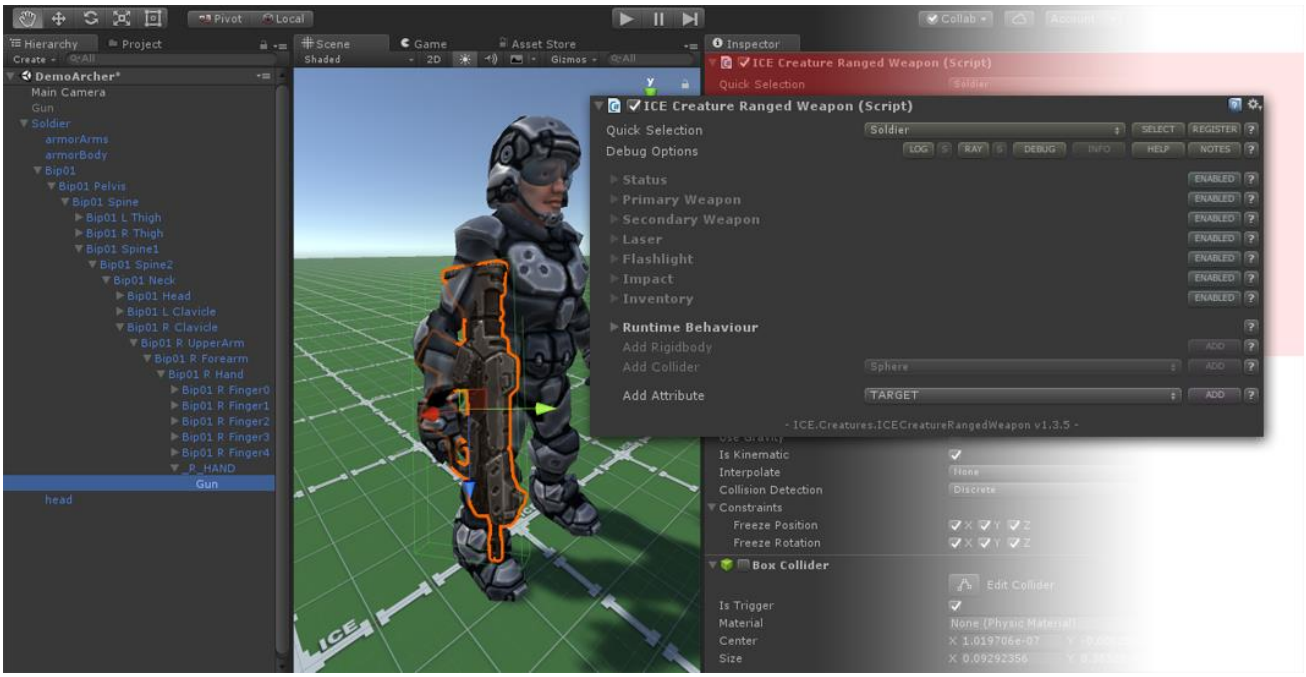


Btw. see also the inventory options of your creature (Creature Status Settings). If you assign such an attached item also to the inventory of your creature you will get several possibilities to control the equipment during the runtime (e.g. hiding an item, changing its mount point or detach items while your creature is dying etc.).

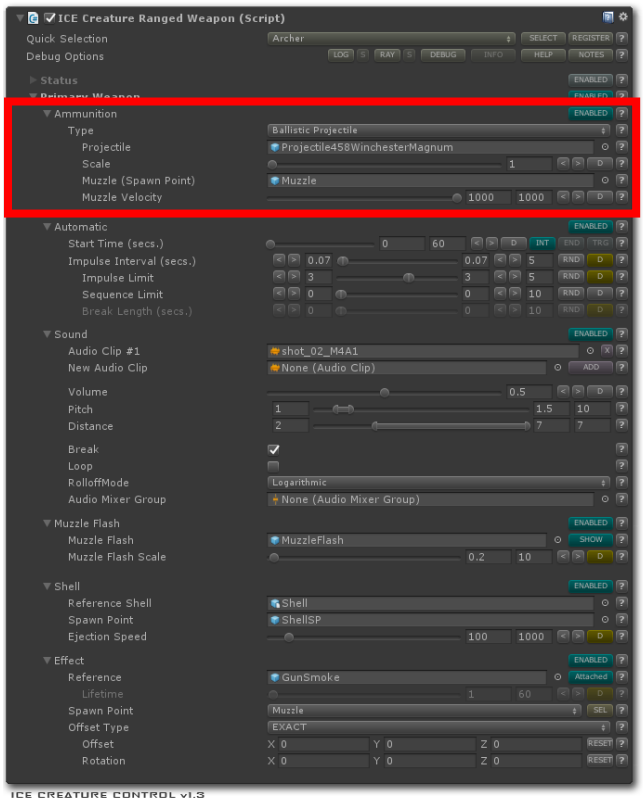
12.1.6 Ranged Weapons

In addition to the above-mentioned methods your creature are also able to use guns, pistols, bows or any other kind of ranged weapons to inflict harm or damage on its enemies.

For this purpose you can add the ICECreatureRangedWeapon component to all your typical ranged weapons, so your creature can find and use such weapons.



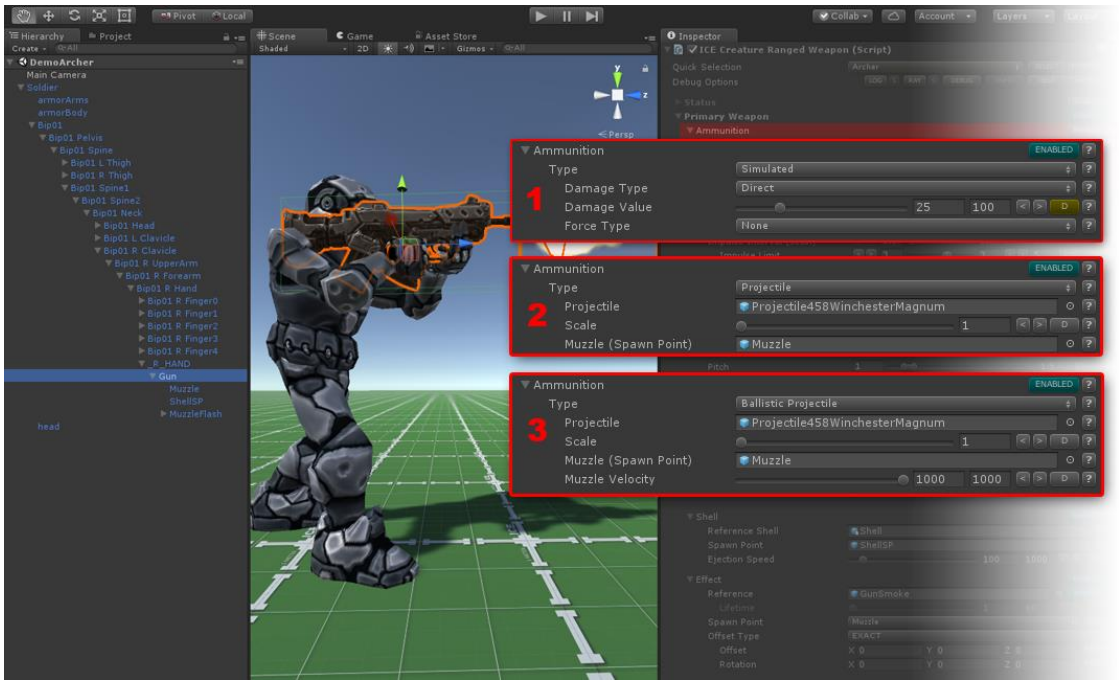
Additional to the above-mentioned Impact option, which allows to use an item as melee weapon, the ICECreatureRangedWeapon component provides two ranged weapon systems, in which the primary system represents the main weapon while the secondary could be optionally used for an integrated grenade launcher or a similar weapon attachment. Both systems are identical and will be used in the same way. Furthermore there are settings for a Flashlight and a Laser system, which could be optionally used dependent to the desired weapon configuration.



Regarding to the damage handling, all required settings can be found in the weapon sections. The weapon sections provides several settings to define the desired Ammunition, Fire Rate (Automatic), Launching Sound and Effect, Muzzle Flash and the Shell Ejection. However, all damage related settings can be done directly in the Ammunition part.

The Ammunition section of the ICECreatureRangedWeapon component offers three Ammunition Types, which represents different methods to handle the shooting procedure.

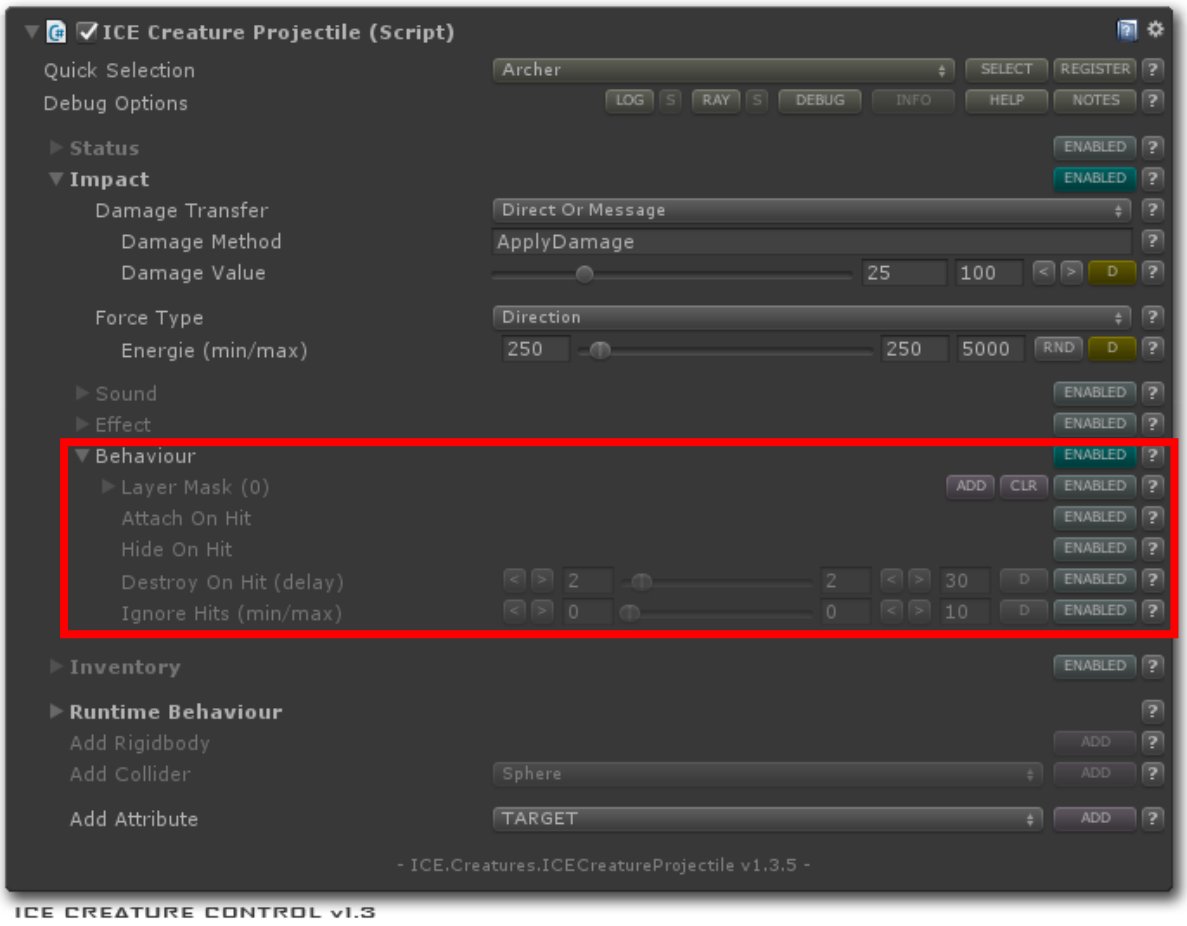
1. ‘Simulated’ will simulate a shot by damaging an aimed target directly.
2. ‘Projectile’ will place the Projectile directly at the hit-point of the targeted destination.
3. ‘Ballistic Projectile’ will launch the Projectile as a visible object on a ballistic trajectory.



12.1.7 Projectile Impacts

In contrast to the 'Simulated' method, which is able to damage a target directly, the projectile based methods (2+3) require an additional object to inflict damage on a target.

For this purpose you can add the ICECreatureProjectile component to all your bullets, arrows, cannonballs or similar projectiles.



The ICECreatureProjectile represents an item with specific abilities but finally the damage handling based of the already-mentioned Impact method and is therefore identical to the damage handling of all other item types. Important here will be the Impact Behaviour settings, which defines the behaviour of your projectile during an impact.

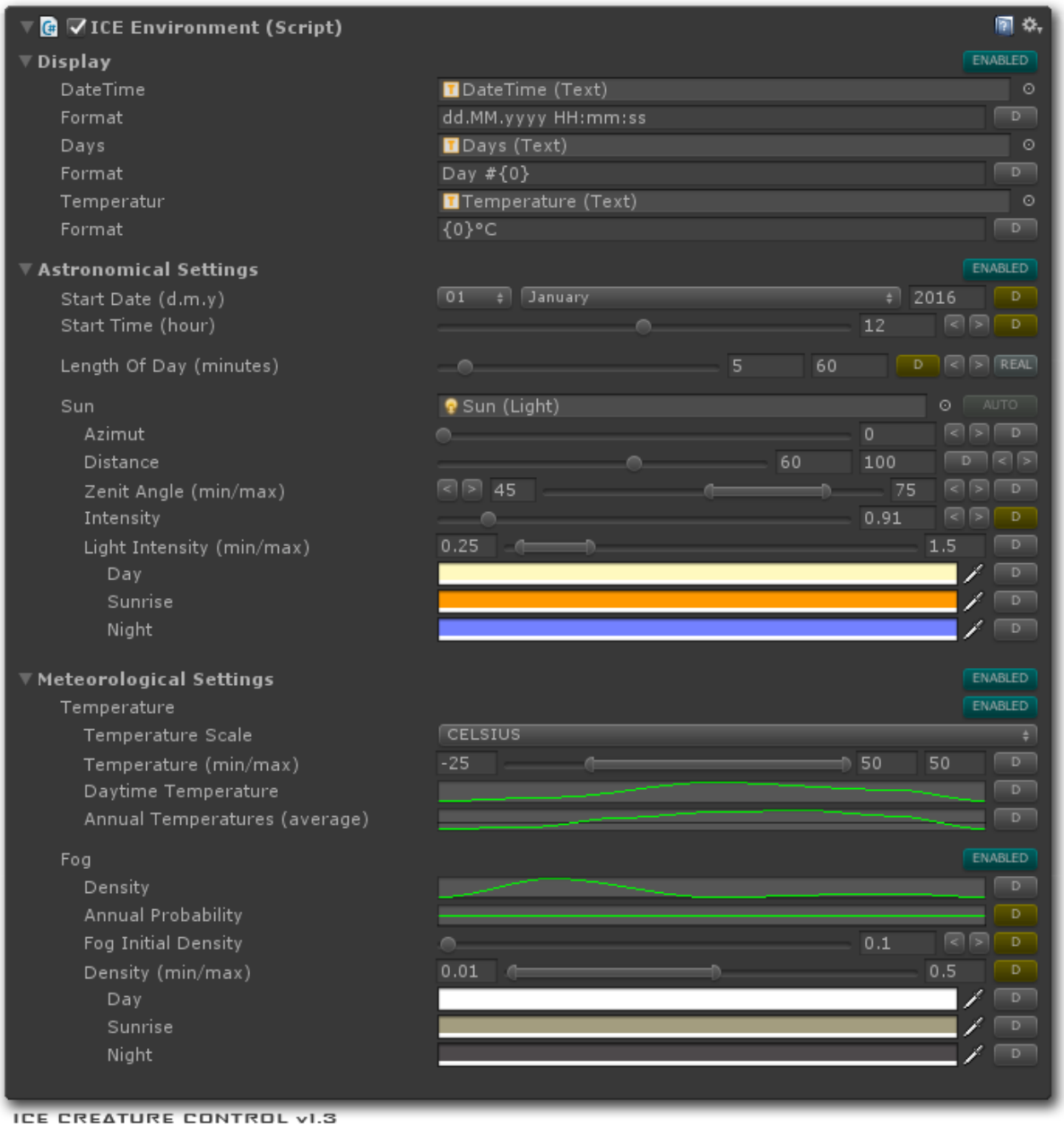
12.2 Flying and Diving Creatures

Basically each creature can fly, swim or dive, for this you only need to define the desired vertical movements in the Movement section of a behaviour. Activate the 'VER' button to enable vertical movements, define the desired operation level and the vertical speed – that's all! Now your creature will try to reach the given operating level. If you want more dynamic, you could add additional behaviour rules with different operating level, so your creature will change the altitude by changing the behaviour rules.

13 ICE ENVIRONMENT

ICEEnvironment is an optional component to handle natural events, such as the Day&Night cycle, environment temperature etc. ICEEnvironment based on the ICEWorldEnvironment, which provides your creatures to perceive and react to different environment conditions, so your creature could take cover from a thunderstorm or will sleeping during the night etc.

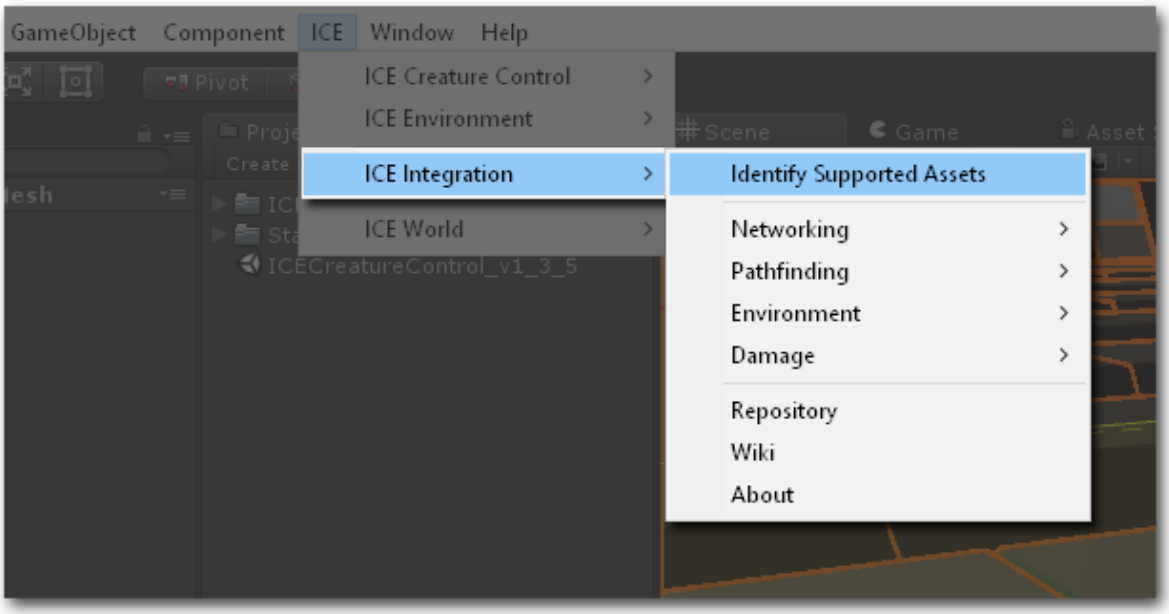
Please note that this component is under construction.



14 ICE INTEGRATION

To support the smooth integration of 3rd Party Products ICE Integration comes with a couple of optional scripts which handles the integration automatically without custom code.

To combine ICE with supported third party products, just open the ICE Integration Menu and press 'Identify Supported Assets'. ICE will scan now your project folder for supported files and will add a custom define for each detected and useable asset.



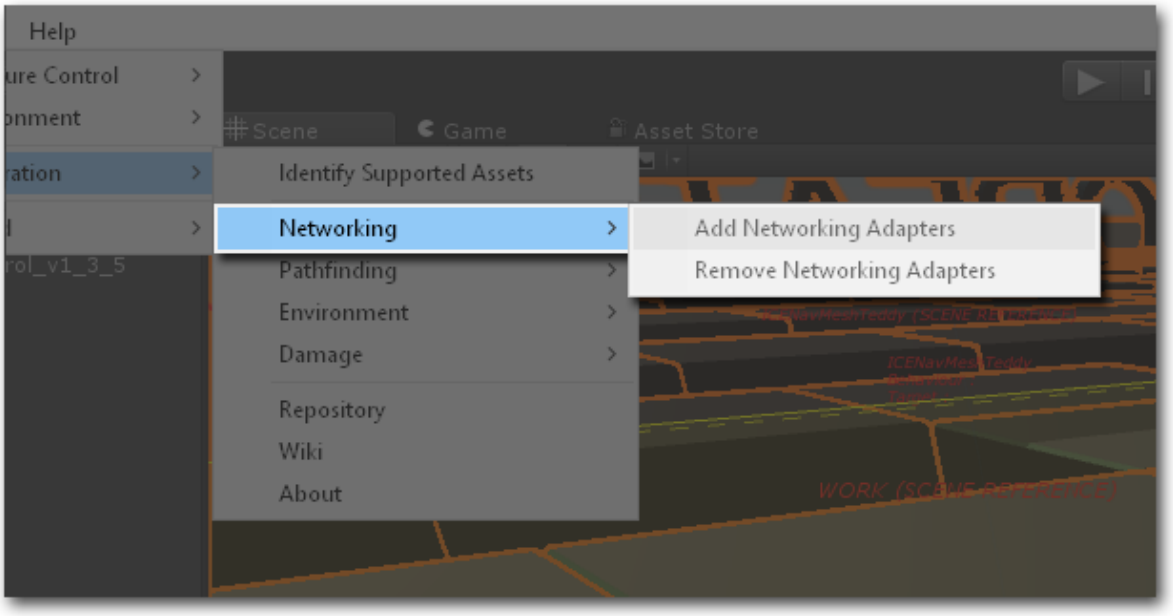
Alternatively, this step can also be performed manually. To do this, you just have to enter the required Custom Define Symbols in the Player Settings.

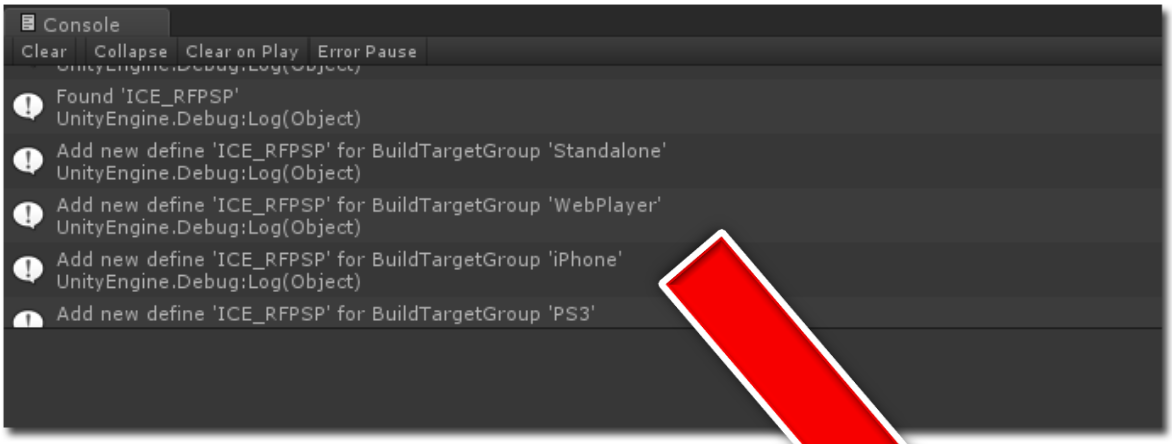
14.1.1 Custom Define Symbols

- Opsive's Ultimate FPS - ICE_UFPS
- Opsive's Third Person Controller - ICE_OPSIVE_TPC
- Ultimate Survival - ICE_ULTIMATE_SURVIVAL
- Invector's Third Person Controller - ICE_INVECTOR_TPC
- Azuline Studio's Realistic FPS Prefab - ICE_RFPSP
- Hardworker Studio's UnitZ - ICE_UNITZ
- Calvin Weibel's Easy Weapons - ICE_EASY_WEAPONS
- Noobtuts uMMORPG - ICE_UMMORPG
- Photon Unity Network - ICE_PUN
- Black Horizon Studio's UniStorm - ICE_UNISTORM
- Aron Granberg's A* Pathfinding Project - ICE_ASTAR
- Apex Game Tools' Apex Path - ICE_APEX

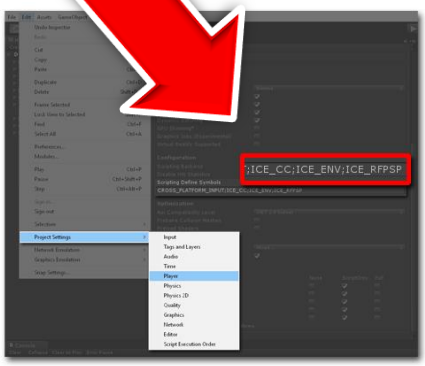
To integrate now specific assets, just select the desired menu group and press 'Add Adapters'. ICE will scan now your hierarchy to find relevant Components by their type and will add the corresponding adapter to its GameObject. You can reverse this step at all times by pressing 'Remove

Adapters'.





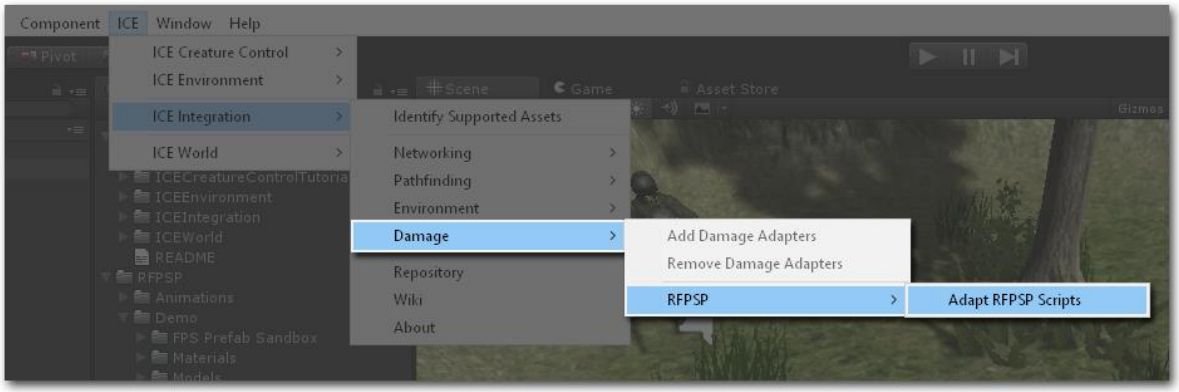
All steps and automatic procedures will be logged in the console, so you can follow and retrace all changes. Also you can open your Player Settings to see which Assets was detected. All ICE based defines will starting with 'ICE'. Please feel free to remove all needless defines as desired, this will avoid overhead by deactivating the unneeded code sections.



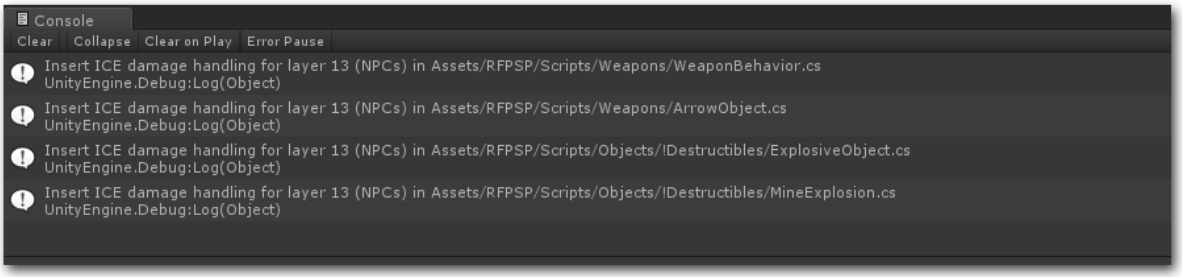
Basically the both above-mentioned steps will handle the complete integration process, so there is nothing else to do besides of the typical component settings. The most adapters do not need any additional configurations and will be ready to work as soon as they are added to their GameObject, but there are some asset packages we have to pay a bit more attention, because in addition to the

adapters such packages requires also to adapt parts of their code. Basically I always try to avoid such modifications but in some cases there are no other ways to get access to the required class member, because of unfavourable code structures with missing access modifier or similar annoyances.

However, ICE can handle such required code modification for you. In cases an adapter requires code changes you will find one or more additional menu items within the Integration menu structure, just press the corresponding 'Adapt Script' item and ICE will doing the rest.



ICE will inform you also here about all changes in the console and if you want to check the modified code section after the adaptations you can searching for 'MODIFIED BY ICE' within your project scripts. ICE will comment each entry with a '// BEGIN …' and '// END …' and the above-mentioned 'MODIFIED BY ICE'.



Btw. please consider that you will get errors or malfunctions while using an adapter who need such code adaptions without the required changes. Also make sure that you are working with the latest version of the asset and that the code is untouched. If ICE can't find a required code sections you will get a warning and have to implement the changes by yourself.

Please note! Automated code adaptations are irreversible, so I would like to recommend you to back up your project before running the adaptations.

Well, if all this is done the integration process should be complete and your ICE product should be able to working with the 3rd party asset.

14.2 Third Party Support

Currently following assets are already supported by ICE or requested and will coming soon.

14.2.1 Damage

14.2.1.1 Opsive's Ultimate FPS (UFPS)

14.2.1.2 Opsive's Third Person Controller (TPC)

14.2.1.3 Ultimate Survival (US)

14.2.1.4 Invector's Third Person Controller (VTPC)

14.2.1.5 Azuline Studio's Realistic FPS Prefab (RFPSP)

14.2.1.6 Hardworker Studio's UnitZ (UNITZ)

14.2.1.7 Gaming is Love's ORK Framework - RPG Engine (coming soon)

14.2.1.8 Calvin Weibel's Easy Weapons

14.2.1.9 Noobtuts uMMORPG

14.2.2 Networking

14.2.2.1 Photon Unity Network (PUN)

14.2.3 Environment

14.2.3.1 Black Horizon Studio's UniStorm

14.2.4 Pathfinding

14.2.4.1 Aron Granberg's A Pathfinding Project*

14.2.4.2 Apex Game Tools' Apex Path (coming soon)

14.2.5 Animation

14.2.5.1 RootMotion's Final IK

14.2.5.2 Rune Skovbo Johanson's Locomotion System

14.2.6 Visual Scripting

14.2.6.1 Hutong Games LLC PlayMaker

14.2.6.2 Evasion Games' GameFlow

14.2.7 User Interface

14.2.7.1 Mad Pixel Machine's Energy Bar Toolkit

14.2.7.2 Devdog's Inventory Pro (coming soon)

Finally I would like to mention that ICE Integration and all the included scripts are completely optional and solely complementary solutions to facilitates potential integration tasks, they are NOT required to run ICECreatureControl and also NOT part of the core product, so please consider that I create and offer all these scripts for free to make the integration of 3rd Party Products easier to you. I try to realize all requested adapters as soon as possible but please understand that there is no warranty or claim for benefits for such adapters.

14.3 Additional Supported Assets

14.3.1 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 legacy 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.

14.3.1.1.1 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*.

14.3.1.1.2 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

15 ICE WORLD

ICE World represents an open framework for Unity 3D, which contains several base classes to simplify the integration of 3rd Party Packages and own projects.

Please feel free to use this framework as groundwork for your own Projects and consider that you are welcome to take part in the creation of this framework as well.

15.1 ESSENTIAL BASE CLASSES

15.1.1 ICEWorldBehaviour

ICEWorldBehaviour represents the lowermost base class for all ICE components (e.g. ICEWorldSingleton or ICEWorldEntity)

ICEWorldBehaviour is derived from MonoBehaviour and declared as abstract.

15.1.1.1 PublicMethods

PublicMethods represents a dynamic collection of public methods which can be used during the runtime by external scripts to steering the behaviour of an ICE component (e.g. ApplyDamage). To make public methods available they have to be registered by using the RegisterPublicMethod method within OnRegisterPublicMethods event.

To simplify the integration of such public methods ICEWorldBehaviour provides a list of all available methods which can be displayed in the editor by using the MethodPopup (see ICE.World.EditorUtilities.WorldObjectEditor.DrawMethodsObject)

15.1.2 ICEWorldSingleton

ICEWorldSingleton represents the lowermost base class for all singleton components within the ICE World (e.g. ICECreatureRegister or ICEEnvironment).

ICEWorldSingleton is derived from ICEWorldBehaviour and declared as abstract.

Use ICEWorldSingleton or one of its derived world classes as base class for your own management components, so your code will be automatically downwards compatible with the rest of the ICE world.

15.1.3 ICEWorldEntity

Within the ICE World an entity is something that exists as itself, as a subject or as an object, actually or potentially, concretely or abstractly, physically or not. On this note an ICE World Entity represents an interactive GameObject within your scenes which could affect the gameplay in someways.

ICEWorldEntity is the lowermost base class for all interactive components within the ICE World (e.g. ICECreatureControl or ICEPlayer, but also items, vehicles, construction elements etc.).

ICEWorldEntity is derived from ICEWorldBehaviour and declared as abstract.

Use ICEWorldEntity or one of its child classes as base class for your own interaction components, such as your own FPSController or your custom AI script, so your code will be automatically downwards compatible with the rest of the ICE world.

Each Entity contains a status and can be destroyed

15.1.4 ICEWorldEntityPart

15.1.5 ICEWorldCamera

ICEWorldCamera represents the lowermost base class for all camera components within the ICE World (e.g. ICEPlayerCamera)

ICEWorldCamera is derived from ICEWorldBehaviour and declared as abstract.

15.2 Specific Base Classes

15.2.1 ICEWorldEnvironment

ICEWorldEnvironment contains several environment parameter, such as Date and Time, Temperature and Weather Conditions etc.

Use ICEWorldEnvironment as base class for your own Day&Night Cycle and/or Weather System, so it will be automatically compatible with the rest of the ICE world.

16 SPECIAL THANKS

16.1 3DMAESEN (BÜMSTRÜM)

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

16.2 MISTER NECTURUS

https://www.assetstore.unity3d.com/en/#!/publisher/290

16.3 DAVID STENFORS

https://www.assetstore.unity3d.com/en/#!/publisher/13489

Demo Scenes: Campfire (https://www.assetstore.unity3d.com/en/#!/content/45038)

16.4 SOU CHEN KI

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

16.5 LESHIY3D

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

16.6 QUENTIN HUDSPETH

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

16.7 FLYINGTEAPOT

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

16.8 JON FINLAY

Many thanks for his great artwork and images

16.9 CHRIS SPOONER BLOG SPOONGRAPHICS

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

17 ANNEXE

17.1 GLOSSARY

17.1.1 ActiveTarget

A creature can have any number of targets, but only one active target at the same time (see also Target).

17.1.2 LastActiveTarget

The LastActiveTarget represents the previous active target (see also ActiveTarget).

17.1.3 Target

A Target represents a group of a particular type of GameObject that a creature should interact with during the runtime. How a creature has to react to specific targets can be defined in the settings of a target.

17.1.4 TargetGameObject

While a target represents the group of a particular type of GameObject, the TargetGameObject represents a specific GameObject of that group (see also Target).

17.1.5 TargetMovePosition

The TargetMovePosition represents the final target position related to the TargetGameObject. Dependent to the specified TargetMoveSpecifications this position can be the exact Transform Position of the TargetGameObject but also a randomized position around it. The TargetMovePosition is the position the creature wants to reach.

17.2 Advanced Target Selection Criteria Expression Values

17.2.1 Own Creature Related Values

All expression values starting with OWN refer to the current creature and not to a possible target object. You can use this values with static comparative values or with runtime values of the selected or active target.

17.2.1.1 OwnGameObject

OwnGameObject represents the GameObject of the given creature. You can use this value to compare it with other GameObjects.

17.2.1.2 OwnBehaviour

OwnBehaviour represents the current Behaviour of the given creature. You can compare this value with other available behaviours to ensure that the creature is running a specific behaviour before selecting the given target.

17.2.1.3 OwnReceivedCommand

OwnReceivedCommand represents the last received command. You can compare this value with a specified text to react to a certain command e.g. ATTACK etc.

17.2.1.4 OwnAge

OwnAge represents the current age of the creature. You can compare this value with a defined value to ensure that a creature will select a target only if the creature will be old enough.

17.2.1.5 OwnGenderType

OwnGenderType represents the gender of the creature. You can compare this value for example with the TargetGenderType to ensure that the creature will select only targets of a specific gender.

17.2.1.6 OwnTrophicLevel

OwnTrophisLevel represents the trophic level of a creature. You can compare this value for example to ensure that your carnivore creature will only attack herbivores.

17.2.1.7 OwnOdour

OwnOdour represents the current odour of the creature. You can compare this value for example with defined limits, so the creature will not select a specific target while its own odour is to strong.

17.2.1.8 OwnOdourIntensity

OwnOdourIntensity represents the current odour intensity of the creature. You can compare this value for example with defined limits, so the creature will not select a specific target while its own odour is to strong.

17.2.1.9 OwnOdourRange

OwnOdourRange represents the current odour range of the creature. You can compare this value for example with defined limits, so the creature will not select a specific target while its own odour is to strong.

17.2.1.10 OwnEnvTemperatureDeviation

17.2.1.11 OwnFitness

OwnFitness represents the current fitness value of the creature. You could compare this value for example with the fitness value of a target creature, so ensure that your creature will only attack the other one if its own fitness is higher than the fitness of the other.

17.2.1.12 OwnHealth

17.2.1.13 OwnStamina

17.2.1.14 OwnPower

17.2.1.15 OwnDamage

17.2.1.16 OwnStress

17.2.1.17 OwnDebility

17.2.1.18 OwnHunger

17.2.1.19 OwnThirst

17.2.1.20 OwnAggressivity

17.2.1.21 OwnExperience

17.2.1.22 OwnAnxiety

17.2.1.23 OwnNosiness

17.2.1.24 OwnZoneName

17.2.1.25 OwnVisualSense

17.2.1.26 OwnAuditorySense

17.2.1.27 OwnOlfactorySense

17.2.1.28 OwnGustatorySense

17.2.1.29 OwnTactileSense

17.2.1.30 OwnSlot0Amount

OwnSlot0Amount represents the current amount of items in slot 0 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.31 OwnSlot1Amount

OwnSlot1Amount represents the current amount of items in slot 1 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.32 OwnSlot2Amount

OwnSlot2Amount represents the current amount of items in slot 2 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.33 OwnSlot3Amount
OwnSlot3Amount represents the current amount of items in slot 3 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.34 OwnSlot4Amount
OwnSlot4Amount represents the current amount of items in slot 4 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.35 OwnSlot5Amount
OwnSlot5Amount represents the current amount of items in slot 5 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.36 OwnSlot6Amount
OwnSlot6Amount represents the current amount of items in slot 6 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.37 OwnSlot7Amount
OwnSlot7Amount represents the current amount of items in slot 7 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.38 OwnSlot8Amount
OwnSlot8Amount represents the current amount of items in slot 8 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.39 OwnSlot9Amount
OwnSlot9Amount represents the current amount of items in slot 9 of the own inventory. You can compare this value with a static number but also with several runtime values.

17.2.1.40 OwnSlot0MaxAmount
OwnSlot0MaxAmount represents the maximum amount of items in inventory slot 0.

17.2.1.41 OwnSlot1MaxAmount
OwnSlot1MaxAmount represents the maximum amount of items in inventory slot 1.

17.2.1.42 OwnSlot2MaxAmount
OwnSlot2MaxAmount represents the maximum amount of items in inventory slot 2.

17.2.1.43 OwnSlot3MaxAmount
OwnSlot3MaxAmount represents the maximum amount of items in inventory slot 3.

17.2.1.44 OwnSlot4MaxAmount
OwnSlot4MaxAmount represents the maximum amount of items in inventory slot 4.

17.2.1.45 OwnSlot5MaxAmount
OwnSlot5MaxAmount represents the maximum amount of items in inventory slot 5.

17.2.1.46 OwnSlot6MaxAmount
OwnSlot6MaxAmount represents the maximum amount of items in inventory slot 6.

17.2.1.47 OwnSlot7MaxAmount
OwnSlot7MaxAmount represents the maximum amount of items in inventory slot 7.

17.2.1.48 OwnSlot8MaxAmount
OwnSlot8MaxAmount represents the maximum amount of items in inventory slot 8.

17.2.1.49 OwnSlot9MaxAmount
OwnSlot9MaxAmount represents the maximum amount of items in inventory slot 9.

17.2.1.50 OwnerIsWithinHomeArea

17.2.1.51 OwnerPosition

17.2.1.52 OwnerIsDead

17.2.1.53 OwnerIsInjured

17.2.1.54 OwnerIsSheltered

17.2.1.55 OwnerIsIndoor

17.2.1.56 OwnerIsGrounded

17.2.1.57 OwnerAltitude

17.2.1.58 OwnerIsSelectedByTarget

17.2.1.59 OwnHomeDistance

17.2.2 Active Target Related Values
All expression values starting with ACTIVE refer to the active target. You can use this values with static comparative values or with runtime values of the selected or active target.

17.2.2.1 ActiveTargetGameObject

17.2.2.2 ActiveTargetName

17.2.2.3 ActiveTargetEntityType

17.2.2.4 ActiveTargetTime

17.2.2.5 ActiveTargetTimeTotal

17.2.2.6 ActiveTargetAge

17.2.2.7 ActiveTargetHasParent

17.2.2.8 ActiveTargetParentName

17.2.2.9 ActiveTargetIsDestroyed

17.2.2.10 ActiveTargetHasOwnerActiveSelected

17.2.2.11 ActiveTargetActiveCounterpartsLimit

17.2.2.12 ActiveTargetDurability

17.2.2.13 ActiveTargetDurabilityInPercent

17.2.2.14 ActiveTargetIsInFieldOfView

17.2.2.15 ActiveTargetIsVisible

17.2.2.16 ActiveTargetIsAudible

17.2.2.17 ActiveTargetIsSmellable

17.2.2.18 ActiveTargetVisibilityByDistance

17.2.2.19 ActiveTargetAudibilityByDistance

17.2.2.20 ActiveTargetSmellabilityByDistance

17.2.2.21 ActiveTargetSlot0Amount
ActiveTargetSlot0Amount represents the current amount of items in inventory slot 0 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.22 ActiveTargetSlot1Amount
ActiveTargetSlot1Amount represents the current amount of items in inventory slot 1 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.23 ActiveTargetSlot2Amount
ActiveTargetSlot2Amount represents the current amount of items in inventory slot 2 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.24 ActiveTargetSlot3Amount
ActiveTargetSlot3Amount represents the current amount of items in inventory slot 3 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.25 ActiveTargetSlot4Amount
ActiveTargetSlot4Amount represents the current amount of items in inventory slot 4 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.26 ActiveTargetSlot5Amount
ActiveTargetSlot5Amount represents the current amount of items in inventory slot 5 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.27 ActiveTargetSlot6Amount
ActiveTargetSlot6Amount represents the current amount of items in inventory slot 6 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.28 ActiveTargetSlot7Amount
ActiveTargetSlot7Amount represents the current amount of items in inventory slot 7 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.29 ActiveTargetSlot8Amount
ActiveTargetSlot8Amount represents the current amount of items in inventory slot 8 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.30 ActiveTargetSlot9Amount
ActiveTargetSlot9Amount represents the current amount of items in inventory slot 9 of the active target. You can compare this value with a static number but also with several runtime values.

17.2.2.31 ActiveTargetSlot0MaxAmount
ActiveTargetSlot0MaxAmount represents the maximum amount of items in inventory slot 0 of the active target.

17.2.2.32 ActiveTargetSlot1MaxAmount
ActiveTargetSlot1MaxAmount represents the maximum amount of items in inventory slot 1 of the active target.

17.2.2.33 ActiveTargetSlot2MaxAmount
ActiveTargetSlot2MaxAmount represents the maximum amount of items in inventory slot 2 of the active target.

17.2.2.34 ActiveTargetSlot3MaxAmount
ActiveTargetSlot3MaxAmount represents the maximum amount of items in inventory slot 3 of the active target.

17.2.2.35 ActiveTargetSlot4MaxAmount
ActiveTargetSlot4MaxAmount represents the maximum amount of items in inventory slot 4 of the active target.

17.2.2.36 ActiveTargetSlot5MaxAmount
ActiveTargetSlot5MaxAmount represents the maximum amount of items in inventory slot 5 of the active target.

17.2.2.37 ActiveTargetSlot6MaxAmount

ActiveTargetSlot6MaxAmount represents the maximum amount of items in inventory slot 6 of the active target.

17.2.2.38 ActiveTargetSlot7MaxAmount

ActiveTargetSlot7MaxAmount represents the maximum amount of items in inventory slot 7 of the active target.

17.2.2.39 ActiveTargetSlot8MaxAmount

ActiveTargetSlot8MaxAmount represents the maximum amount of items in inventory slot 8 of the active target.

17.2.2.40 ActiveTargetSlot9MaxAmount

ActiveTargetSlot9MaxAmount represents the maximum amount of items in inventory slot 9 of the active target.

17.2.2.41 ActiveTargetOdour

17.2.2.42 ActiveTargetOdourIntensity

17.2.2.43 ActiveTargetOdourIntensityNet

17.2.2.44 ActiveTargetOdourIntensityByDistance

17.2.2.45 ActiveTargetOdourRange

17.2.3 Last Target Related Values

17.2.3.1 LastTargetGameObject

17.2.3.2 LastTargetName

17.2.3.3 LastTargetEntityType

17.2.3.4 LastTargetTime

17.2.3.5 LastTargetTimeTotal

17.2.3.6 LastTargetAge

17.2.3.7 LastTargetHasParent

17.2.3.8 LastTargetParentName

17.2.3.9 LastTargetIsDestroyed

17.2.3.10 LastTargetHasOwnerActiveSelected

17.2.3.11 LastTargetActiveCounterpartsLimit

17.2.3.12 LastTargetDurability

17.2.3.13 LastTargetDurabilityInPercent

17.2.3.14 LastTargetIsInFieldOfView

17.2.3.15 LastTargetIsVisible

17.2.3.16 LastTargetIsAudible

17.2.3.17 LastTargetIsSmellable

17.2.3.18 LastTargetVisibilityByDistance

17.2.3.19 LastTargetAudibilityByDistance

17.2.3.20 LastTargetSmellabilityByDistance

17.2.3.21 LastTargetSlot0Amount

LastTargetSlot0Amount represents the current amount of items in inventory slot 0 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.22 LastTargetSlot1Amount
LastTargetSlot1Amount represents the current amount of items in inventory slot 1 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.23 LastTargetSlot2Amount
LastTargetSlot2Amount represents the current amount of items in inventory slot 2 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.24 LastTargetSlot3Amount
LastTargetSlot3Amount represents the current amount of items in inventory slot 3 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.25 LastTargetSlot4Amount
LastTargetSlot4Amount represents the current amount of items in inventory slot 4 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.26 LastTargetSlot5Amount
LastTargetSlot5Amount represents the current amount of items in inventory slot 5 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.27 LastTargetSlot6Amount
LastTargetSlot6Amount represents the current amount of items in inventory slot 6 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.28 LastTargetSlot7Amount
LastTargetSlot7Amount represents the current amount of items in inventory slot 7 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.29 LastTargetSlot8Amount
LastTargetSlot8Amount represents the current amount of items in inventory slot 8 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.30 LastTargetSlot9Amount
LastTargetSlot9Amount represents the current amount of items in inventory slot 9 of the last active target. You can compare this value with a static number but also with several runtime values.

17.2.3.31 LastTargetSlot0MaxAmount
LastTargetSlot0MaxAmount represents the maximum amount of items in inventory slot 0 of the last active target.

17.2.3.32 LastTargetSlot1MaxAmount
LastTargetSlot1MaxAmount represents the maximum amount of items in inventory slot 1 of the last active target.

17.2.3.33 LastTargetSlot2MaxAmount
LastTargetSlot2MaxAmount represents the maximum amount of items in inventory slot 2 of the last active target.

17.2.3.34 LastTargetSlot3MaxAmount
LastTargetSlot3MaxAmount represents the maximum amount of items in inventory slot 3 of the last active target.

17.2.3.35 LastTargetSlot4MaxAmount
LastTargetSlot4MaxAmount represents the maximum amount of items in inventory slot 4 of the last active target.

17.2.3.36 LastTargetSlot5MaxAmount
LastTargetSlot5MaxAmount represents the maximum amount of items in inventory slot 5 of the last active target.

17.2.3.37 LastTargetSlot6MaxAmount
LastTargetSlot6MaxAmount represents the maximum amount of items in inventory slot 6 of the last active target.

17.2.3.38 LastTargetSlot7MaxAmount
LastTargetSlot7MaxAmount represents the maximum amount of items in inventory slot 7 of the last active target.

17.2.3.39 LastTargetSlot8MaxAmount
LastTargetSlot8MaxAmount represents the maximum amount of items in inventory slot 8 of the last active target.

17.2.3.40 LastTargetSlot9MaxAmount
LastTargetSlot9MaxAmount represents the maximum amount of items in inventory slot 9 of the last active target.

17.2.3.41 LastTargetOdour

17.2.3.42 LastTargetOdourIntensity

17.2.3.43 LastTargetOdourIntensityNet

17.2.3.44 LastTargetOdourIntensityByDistance

17.2.3.45 LastTargetOdourRange

17.2.4 Target Related Values

17.2.4.1 TargetGameObject

17.2.4.2 TargetName

17.2.4.3 TargetEntityType

17.2.4.4 TargetTime

17.2.4.5 TargetTimeTotal

17.2.4.6 TargetAge

17.2.4.7 TargetHasParent

17.2.4.8 TargetParentName

17.2.4.9 TargetZoneName

17.2.4.10 TargetIsActive

17.2.4.11 TargetIsLastTarget

17.2.4.12 TargetHasOwnerActiveSelected

17.2.4.13 TargetIsDestroyed

17.2.4.14 TargetDurability

17.2.4.15 TargetDurabilityInPercent

17.2.4.16 TargetActiveCounterpartsLimit

17.2.4.17 TargetIsInFieldOfView

17.2.4.18 TargetIsVisible

17.2.4.19 TargetIsAudible

17.2.4.20 TargetIsSmellable

17.2.4.21 TargetVisibilityByDistance

17.2.4.22 TargetAudibilityByDistance

17.2.4.23 TargetSmellabilityByDistance

17.2.4.24 TargetDistance

17.2.4.25 TargetOffsetPositionDistance

17.2.4.26 TargetMovePositionDistance

17.2.4.27 TargetLastKnownPositionDistance

17.2.4.28 TargetOdour

17.2.4.29 TargetOdourIntensity

17.2.4.30 TargetOdourIntensityNet

17.2.4.31 TargetOdourIntensityByDistance

17.2.4.32 TargetOdourRange

17.2.4.33 TargetSlot0Amount
TargetSlot0Amount represents the current amount of items in slot 0 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.34 TargetSlot1Amount
TargetSlot1Amount represents the current amount of items in slot 1 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.35 TargetSlot2Amount
TargetSlot2Amount represents the current amount of items in slot 2 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.36 TargetSlot3Amount
TargetSlot3Amount represents the current amount of items in slot 3 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.37 TargetSlot4Amount
TargetSlot4Amount represents the current amount of items in slot 4 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.38 TargetSlot5Amount
TargetSlot5Amount represents the current amount of items in slot 5 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.39 TargetSlot6Amount
TargetSlot6Amount represents the current amount of items in slot 6 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.40 TargetSlot7Amount
TargetSlot7Amount represents the current amount of items in slot 7 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.41 TargetSlot8Amount
TargetSlot8Amount represents the current amount of items in slot 8 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.42 TargetSlot9Amount
TargetSlot9Amount represents the current amount of items in slot 9 of the targets inventory. You can compare this value with a static number but also with several runtime values.

17.2.4.43 TargetSlot0MaxAmount
TargetSlot0MaxAmount represents the maximum amount of items in inventory slot 0 of the last active target.

17.2.4.44 TargetSlot1MaxAmount
TargetSlot1MaxAmount represents the maximum amount of items in inventory slot 1 of the last active target.

17.2.4.45 TargetSlot2MaxAmount
TargetSlot2MaxAmount represents the maximum amount of items in inventory slot 2 of the last active target.

17.2.4.46 TargetSlot3MaxAmount
TargetSlot3MaxAmount represents the maximum amount of items in inventory slot 3 of the last active target.

17.2.4.47 TargetSlot4MaxAmount
TargetSlot4MaxAmount represents the maximum amount of items in inventory slot 4 of the last active target.

17.2.4.48 TargetSlot5MaxAmount
TargetSlot5MaxAmount represents the maximum amount of items in inventory slot 5 of the last active target.

17.2.4.49 TargetSlot6MaxAmount
TargetSlot6MaxAmount represents the maximum amount of items in inventory slot 6 of the last active target.

17.2.4.50 TargetSlot7MaxAmount
TargetSlot7MaxAmount represents the maximum amount of items in inventory slot 7 of the last active target.

17.2.4.51 TargetSlot8MaxAmount
TargetSlot8MaxAmount represents the maximum amount of items in inventory slot 8 of the last active target.

17.2.4.52 TargetSlot9MaxAmount
TargetSlot9MaxAmount represents the maximum amount of items in inventory slot 9 of the last active target.

17.2.5 Creature Related Values (only available if the target is a creature)

17.2.5.1 CreatureActiveTargetGameObject

17.2.5.2 CreatureActiveTargetTime

17.2.5.3 CreatureActiveTargetTimeTotal

17.2.5.4 CreatureBehaviour

17.2.5.5 CreatureCommand

17.2.5.6 CreatureAltitude

17.2.5.7 CreatureEnvTemperatureDeviation

17.2.5.8 CreatureFitness

CreatureFitness represents the current fitness value of the target creature. You could compare this value for example with the fitness value of your own creature, to ensure that your creature will only attack the other one if its own fitness is higher than the fitness of the other.

17.2.5.9 CreatureHealth

17.2.5.10 CreatureStamina

17.2.5.11 CreaturePower

17.2.5.12 CreatureDamage

17.2.5.13 CreatureStress

17.2.5.14 CreatureDebility

17.2.5.15 CreatureHunger

17.2.5.16 CreatureThirst

17.2.5.17 CreatureAggressivity

17.2.5.18 CreatureExperience

17.2.5.19 CreatureAnxiety

17.2.5.20 CreatureNosiness

17.2.5.21 CreatureVisualSense

17.2.5.22 CreatureAuditorySense

17.2.5.23 CreatureOlfactorySense

17.2.5.24 CreatureGustatorySense

17.2.5.25 CreatureTactileSense

17.2.5.26 CreaturePosition

17.2.5.27 CreatureIsDead

17.2.5.28 CreatureIsInjured

17.2.5.29 CreatureIsInjuredOrDead

17.2.5.30 CreatureIsSheltered

17.2.5.31 CreatureIsIndoor

17.2.5.32 CreatureIsGrounded

17.2.5.33 CreatureGenderType

17.2.5.34 CreatureTrophicLevel

17.2.6 Environment Related Values

17.2.6.1 EnvironmentTimeHour

EnvironmentTimeHour represents the current hour of the day. You can use this value to ensure that your creature will select a defined target only at the specified time. You can also combine EnvironmentTimeHour with a second one to define a time span with > and < etc.

17.2.6.2 EnvironmentTimeMinute

EnvironmentTimeMinute represents the current minute of the hour. You can use this value to ensure that your creature will select a defined target only at the specified time. You can also combine EnvironmentTimeMinute with a second one to define a time span with > and < etc.

17.2.6.3 EnvironmentTimeSecond

EnvironmentTimeSecond represents the current second of the minute. You can use this value to ensure that your creature will select a defined target only at the specified time. You can also combine EnvironmentTimeSecond with a second one to define a time span with > and < etc.

17.2.6.4 EnvironmentDateYear

EnvironmentTimeYear represents the current year of the game. You can use this value to ensure that your creature will select a defined target only at a certain year. You can also combine EnvironmentTimeYear with a second one to define a time span with > and < etc.

17.2.6.5 EnvironmentDateMonth

EnvironmentTimeMonth represents the current month of the year. You can use this value to ensure that your creature will select a defined target only at a certain month. You can also combine EnvironmentTimeMonth with a second one to define a time span with > and < etc.

17.2.6.6 EnvironmentDateDay

EnvironmentTimeDay represents the current day of the month. You can use this value to ensure that your creature will select a defined target only at a certain day. You can also combine EnvironmentTimeDay with a second one to define a time span with > and < etc.

17.2.6.7 EnvironmentTemperature

EnvironmentTemperature represents the current environment temperature. You can use this value for example to ensure that a creature only select a specific target if the temperature is within a suitable range, for this you can combine two EnvironmentTemperature values to define a temperature range.

17.2.6.8 EnvironmentWeather

EnvironmentWeather represents the current weather. You can use this value to ensure that a creature have to select a target according to the general weather conditions.

17.2.7 System Related Values

17.2.7.1 SystemInputKey

SystemInputKey represents the value of the specified key. You can use this value for example to send specific commands to a creature, so a creature will select a target only while the specified key is pressed.

17.2.7.2 SystemInputAxis

SystemInputAxis represents the value of the specified axis. You can use this value for example to send specific commands to a creature, so a creature will select a target only if the conditions for specified axis are given.

17.2.7.3 SystemUIToggle

SystemUIToggle represents the value of specified UI toggle object. You can use this value for example to send specific commands to a creature, so a creature will select a target only if the conditions for specified toggle are given.

17.2.7.4 SystemUIButton

SystemUIButton represents the value of specified UI button object. You can use this value for example to send specific commands to a creature, so a creature will select a target only if the specified button was pressed.