ADVANCED RANDOM SCRIPT INTERFACE

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/scripting-reference/plugin-interfaces/advanced-random

---

The `IAdvancedRandomObjectType` interface derives from IObjectClass to add APIs specific to the Advanced Random plugin.

Note this class derives from the object class interface, not the instance interface. Typically it is used through `runtime.objects.AdvancedRandom`.

Advanced Random APIs

---

seed

Set or get a string with the current seed for the pseudo-random number generator. The same seed will produce the same sequence of pseudo-random numbers.

---

octaves

Set or get the number of octaves used for coherent noise generation, from 1-16. The default is 1. This affects the Billow, Classic and Ridged noise functions only. Using additional octaves adds layers of increasing detail to the noise functions, but is also slower to process.

---

billow2d(x, y)

billow3d(x, y, z)

Generate a random number using billow noise in the range 0-1, using either 2D or 3D co-ordinates.

---

cellular2d(x, y)

cellular3d(x, y, z)

Generate a random number using cellular noise in the range 0-1, using either 2D or 3D co-ordinates.[/dd classic2d(x, y)
classic3d(x, y, z)
Generate a random number using classic (perlin) noise in the range 0-1, using either 2D or 3D co-ordinates.

---

ridged2d(x, y)

ridged3d(x, y, z)

Generate a random number using ridged noise in the range 0-1, using either 2D or 3D co-ordinates.

---

voronoi2d(x, y)

voronoi3d(x, y, z)

Generate a random number using Voronoi noise in the range 0-1, using either 2D or 3D co-ordinates.

---

random()

Generate a random number in the range [0, 1) using the current seed. This allows generating random numbers with a predictable sequence even when *Replace system random* is not used.

---

createGradient(name, mode)

Create a new gradient with a given string for its name. The `mode` must be one of `"float"` or `"rgb"`.

---

setCurrentGradient(name)

Set the current gradient that is the default to sample from.

---

addGradientStop(position, value)

Adds a stop to the current gradient. The stop position can be any number, but is generally kept within the 0-1 range so it can be used with the random expressions.

---

sampleGradient(name, position)

Sample a gradient at the given position. The `name` can be omitted (pass `null`) to use the current gradient; otherwise it specifies a case-insensitive string of the gradient to sample.

---

createProbabilityTable(name)

Create a new probability table, using a string of its name to identify it.

---

createProbabilityTableFromJSON(name, jsonStr)

Create a new probability table with a name, using a string of JSON data from a prior call to `getProbabilityTableAsJSON()` for its entries.

---

getProbabilityTableAsJSON()

Return a string of JSON data representing the current probability table.

setCurrentProbabilityTable(name)

Set the current probability table by a string of its name.

addProbabilityTableEntry(weight, value)

Add an entry to the current probability table with the given weight and value.

removeProbabilityTableEntry(weight, value)

Remove an existing entry from the current probability table. If a weight of 0 is specified, the first entry with the given value is removed regardless of its weight. Otherwise an entry is only removed if it matches both the value and the weight.

sampleProbabilityTable(name)

Get a random value from a probability table. The relative likelihood of values is affected by their weight. The `name` can be omitted (pass `null`) to use the current probability table; otherwise it specifies a case-insensitive string of the probability table to sample.

createPermutationTable(length, offset)

Generate a randomly ordered sequence of numbers. The `length` parameter is how many numbers to generate, and `offset` is the first number in the sequence. For example a `length` of 3 with an `offset` of 1 will generate the numbers 1, 2 and 3, and then randomly shuffle them.

shufflePermutationTable()

Re-shuffle an existing permutation table.

getPermutation(index)

Get a value at a zero-based index in the permutation table.