ADVANCED RANDOM

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

The **Advanced Random** object provides expressions for advanced pseudo-random number generation (PRNG), including two- and three-dimensional noise functions like Perlin noise (referred to as "classic noise" in the plugin). These are useful for procedural generation, such as providing an endless supply of interesting and unique level designs.

It also provides seeded random functions, which provide the same pseudo-random number sequence when given the same seed. This can also be used to override the system random function (covering the *random()* expression, and any randomness used in behaviors) with a seeded random, which can provide deterministic random number generation for the whole runtime. By default the seed is always itself random, meaning random number generation is different between different runs of the game.

Click here to open an example of the Advanced Random plugin. This uses the Drawing Canvas object to display randomly generated textures using Advanced Random.

Scripting

When using JavaScript or TypeScript coding, the features of this object can be accessed via the IAdvancedRandomObjectType script interface.

Gradients

The Advanced Random object also provides tools to calculate custom gradients. This is useful for procedural generation. The random numbers it generates are in the range 0-1, but gradients allow you to specify values or colors for certain ranges of that. For example making the range 0-0.5 blue, and the range 0.5-1 green, allows you to look up the color using a random number in the range 0-1 and get either a water or land color. Combined with expressions like 2D perlin noise, this provides a way to randomly generate a level with mixed regions of water and land.

Probability tables

Advanced Random can also create *probability tables*, which are a way of generating weighted random numbers. For example if you add three items with a weight of 1, and then a fourth item with a weight of 2, that item is twice as likely to be picked as any other item. The value of a probability table entry can be either a number or a string, but the weight must be a number. This is useful for doing things like random pickups with lots of common cheap items, but also some rare valuable items.

Permutation tables

Another feature of Advanced Random is creating *Permutation tables*. These are simply a sequence of numbers that are randomly ordered. This is useful for retrieving random, non-repeating numbers. For example you could use the numbers 0-51 to represent a deck of cards, and create a permutation table to represent a shuffled deck of cards. Then if you read through the permutation table in order with the *Permutation* expression, it will return a random sequence of cards, like dealing from a shuffled deck.

Advanced random properties

Seed

A string of characters used as the seed for random number generation. The same seed will always provide the same sequence of numbers. An empty string (the default) will use a random seed, ensuring the sequence of numbers is different every time.

Replace system random

If enabled, the system random function, which covers the *random* system expression as well as randomness in behaviors, is overridden to use the Advanced Random object's PRNG. Since the Advanced Random object can control the seed, this provides a way to seed the random number generation of the entire runtime.

Advanced random conditions

The Advanced Random object has no conditions.

Advanced random actions

Set octaves

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

Update seed

Set a new seed for random number generation, using a string.

> *Note if you pass an empty string, it still uses the empty string as the seed. If you want to go back to using a random seed, pass the RandomSeed expression as the seed to set.*

Add gradient stop

Adds a stop to the current gradient. Use after *Create gradient* to specify the 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. The stop value should be an expression of the form *rgbEx(r, g,*

b) or *rgba(r, g, b, a)* when the gradient uses color mode; otherwise it can be a simple number. The default gradient is a simple black to white gradient, using *rgbEx(0, 0, 0)* at position 0 and *rgbEx(100, 100, 100)* at position 1.

Create gradient

Create a new gradient. Multiple gradients can be managed by giving them different names. Creating a gradient also sets it as the current gradient, so this action can be immediately followed by *Add gradient stop* to specify the gradient. By default gradients work in color mode, which uses values based on the *rgbEx* or *rgba* expressions; however they can also be set to number mode which uses simple numbers.

Set gradient

Set the current gradient by its name. This allows switching between multiple gradients.

Create permutation table

Generate a randomly ordered sequence of numbers. *Length* 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.

Shuffle permutation table

Re-shuffle an existing permutation table.

Add probability entry

Add an entry to the current probability table. The value can be a string or a number. The weight affects how likely the item is to be picked, relative to other item's weights.

Create probability table

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

Remove probability entry

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.

Set probability table

Set the current probability table from which weighted random values are taken.

Create probability table from JSON

Create a new probability table from a JSON string. The input should be an array of `[weight: number, value: number|string]` tuples, e.g. `[[1, "Apple"], [2, "Banana"], [3, "Carrot"]]`.

Advanced random expressions

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.

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.

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.

RandomSeed

Generate a random seed string that can be used to set the seed, restoring an unpredictable number sequence.

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.

Seed

The currently set seed, as a string.

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.

Gradient(Position)

GradientByName(Name, Position)

Sample a gradient at the given position. This returns a color value for color mode gradients, otherwise a simple number. The *Gradient* variant refers to the current gradient, whereas the *GradientByName* variant refers to any gradient using a case-insensitive string of its name.

Permutation

Get a value at a zero-based index in the permutation table, from 0 (for the first item) up to but not including the length of the table.

Weighted

WeightedByName(Name)

Get a random value from a probability table. The relative likelihood of values is affected by their weight. The *Weighted* variant refers to the current probability table, whereas the *WeightedByName* variant refers to any probability table using a case-insensitive string of its name.

ProbabilityTableAsJSON

Get the current probability table as a JSON string. This can be read back using the **Create probability table from JSON** action.