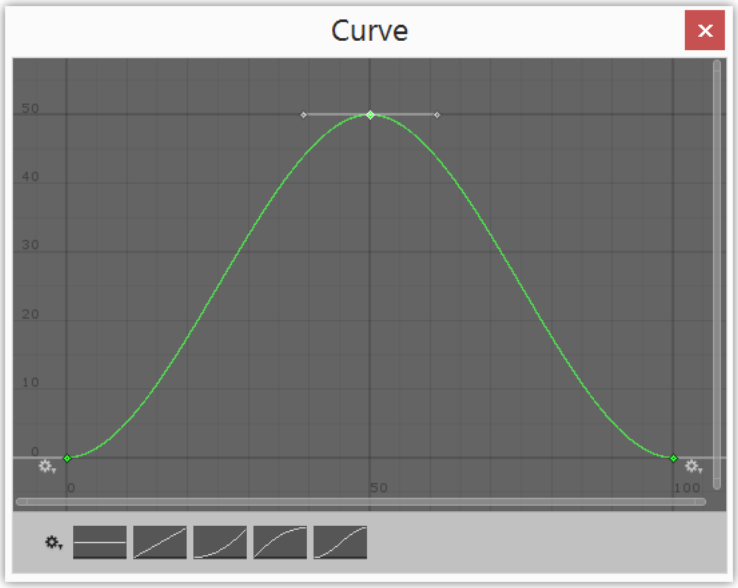
Generating Random Numbers According to Distribution

1. Add the RandomDistribution script to a GameObject.

2. Click on the curve in the inspector to adjust the distribution to your needs. The higher the curve is at a given number, the more likely that number will be.

3. Call the following methods on the script to get random numbers:
   - RandomFloat() returns a float value
   - RandomInt() returns an int value



Public Functions

You can all the following functions on a RandomDistribution script:

float **RandomFloat**() Returns a random float value using weighted chances from the distribution curve.
int **RandomInt**() Returns a random integer value using weighted chances from the distribution curve.
AnimationCurve **GetDistributionCurve**() Returns the distribution curve that is used for random number generation.
void **SetDistributionCurve**(AnimationCurve) Sets the distribution curve to be used for random number generation.

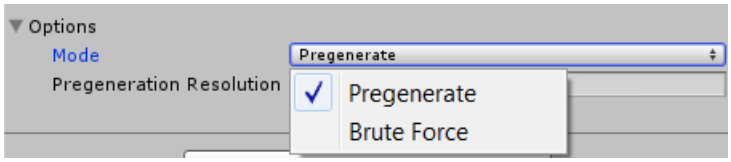
Options / Technical Details

The “options” section lets you choose between two different algorithms to use, which are explained in the following section. This setting is probably only interesting if performance really matters for your

project (the difference is roughly whether generating 1,000,000 random numbers takes 300 or 100 milliseconds).

The main difference here is that the “Pre-Generate” algorithm is about three times as fast as “Brute Force”, but also has a larger memory footprint and possibly less precision what approximating the curve.

Pre-Generate: This option generates an array of numbers on Awake(), the returned random numbers are then picked from that list. The weighting is implemented by adding more common numbers multiple times to the list.



The chosen resolution specifies how many numbers are generated and thus how precisely the curve is approximated. Higher resolutions will take more memory space and longer to generate.

Brute Force: When a random number is requested, this option picks random points within a rectangle that wraps the curve until a point is under the curve. The x-coordinate of that point is then returned as the random number.

This algorithm has the advantage of randomizing the numbers at full float precision rather than a specified resolution, however, it is slightly slower than picking numbers from a pre-generated list. With this algorithm, performance also depends on the shape of the curve.