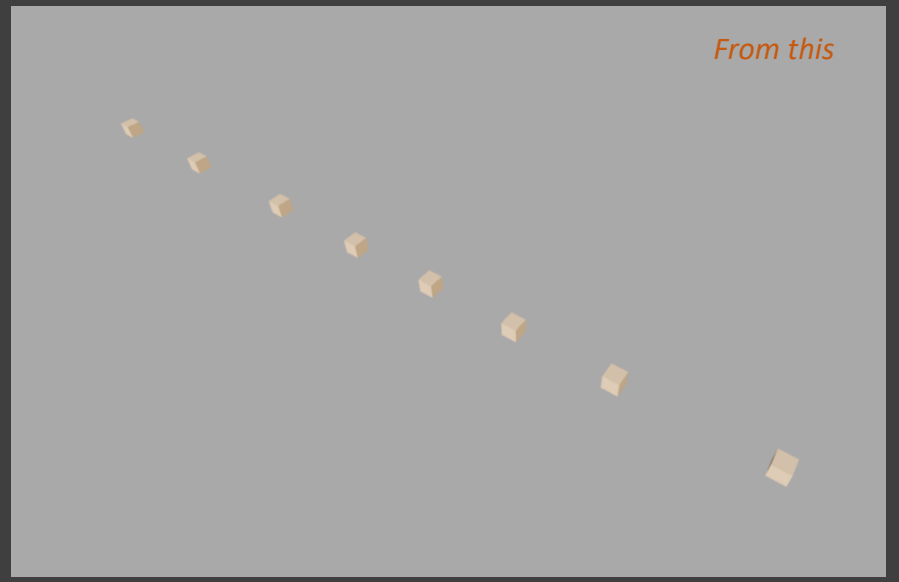
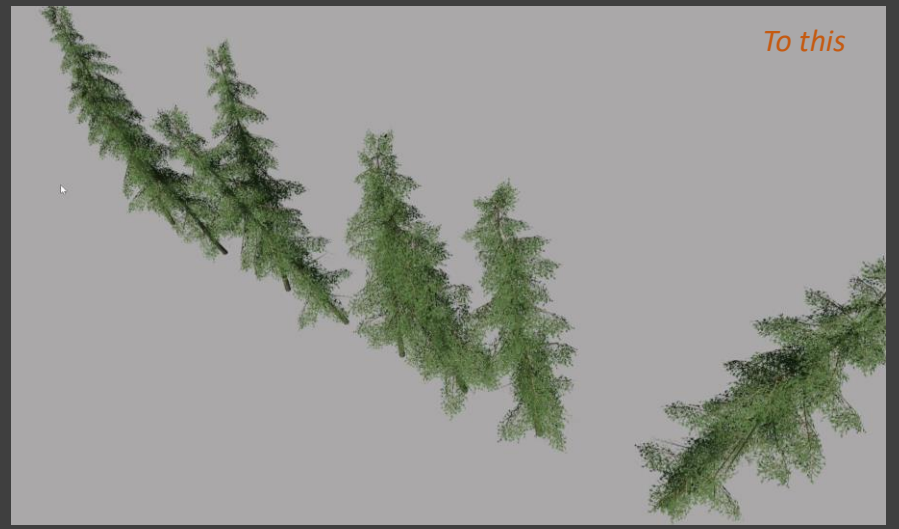
What is Object Distributor?

At the heart of it, *Object Distributor* is a script for Giants Editor that will replace any number of objects with any custom object, similar to an advanced search and replace. The script can be used to place trees, light poles or other objects on a map. This is especially useful in combination with a random/procedurally[1] generated collection of objects (destination markers).

The script gives control of the rotation and tilt of the replaced object and also support different kinds of randomization (e.g. rotation and angle of trees or replacing placing rocks of different sizes).

[1] *E.g. with tools like Houdini*



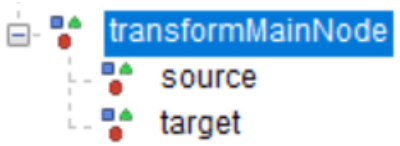


Created by w33zl (WZL Modding), idea by OlaHaldor

How does Object Distributor work?

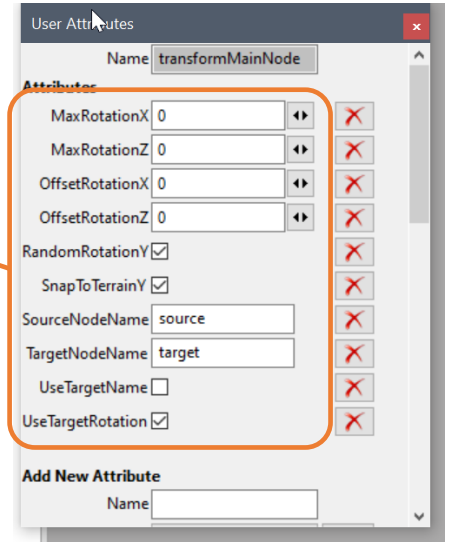
The script requires three nodes:

1. The **root/configuration** node (a transform group)
2. A ***"source"*** node (can be renamed)
3. A ***"target"*** node (can be renamed)

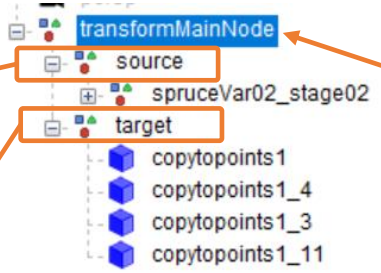


When the script is executed every child node in ***target*** will be replaced with a cloned child object from the ***source*** node.

1) The main/root node contains all configurations



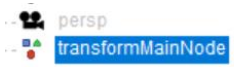
2) The ***source*** node contains one or more objects that will be cloned and replacing objects in the target node

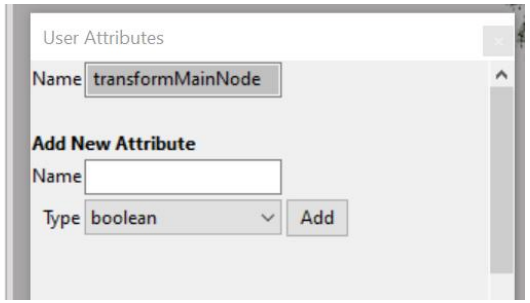


3) The ***target*** node contains objects that will be replaced with objects from the ***source*** node. These target objects is "location makers" (temporary objects) that indicates where the cloned objects will be placed on the map (and optionally also in which direction).

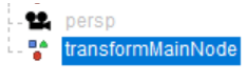
Get started (prepare your nodes)

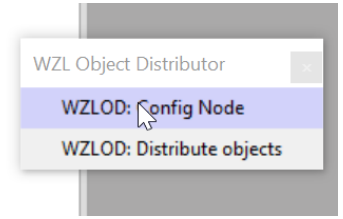
1. Create empty transform group (name doesn't matter)





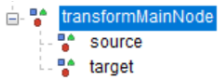
2. Execute script "Config Node" from Object Distributor



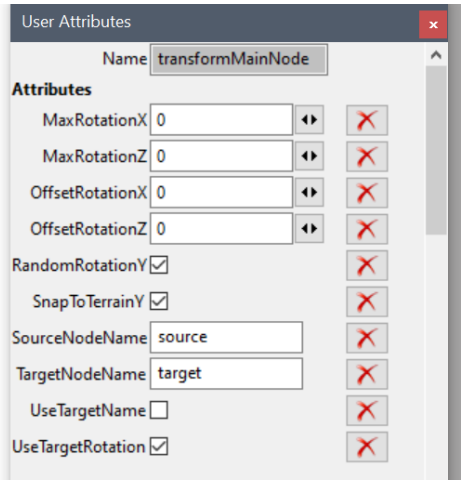


This is a mandatory step to ensure everything is setup properly before running the actual distribution script. You can configure manually as well, but this script ensure everything is properly in place. You can also run this multiple times as it is "non destructive", i.e. it will only add missing pieces, not changing something that is already there.

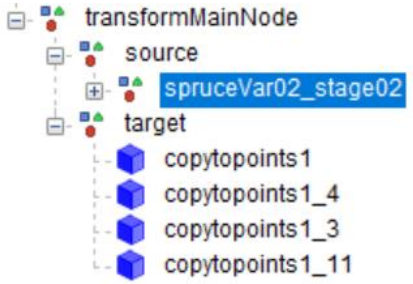
3. Change any settings on the root node (if needed)



After running the "Config Node" script you should now end up with these attributes on the root node as well as having two new child nodes called *"source"* and *"target"*



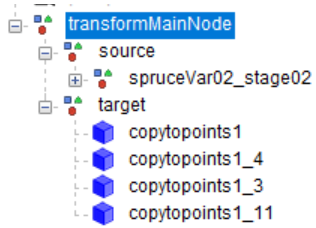
4. Add source and target child nodes

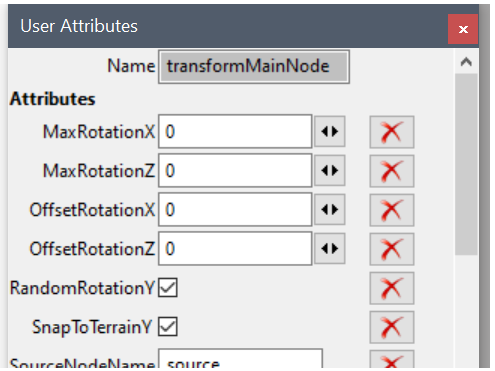


Add at least one source child object and one target child object (otherwise the script will fail)

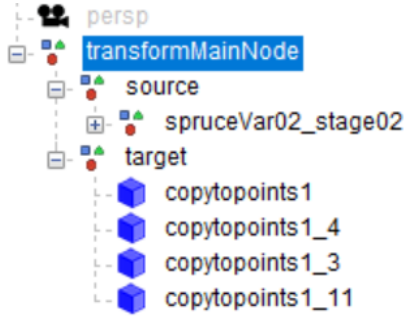
Use the *Distribute Objects* script

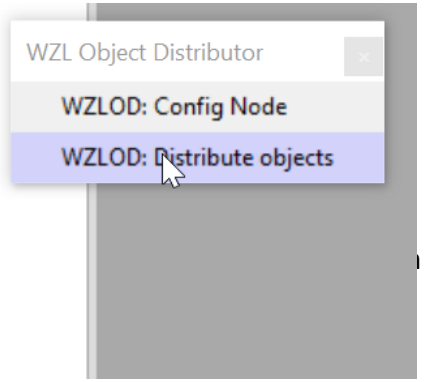
1. Select the root node (the transform group you created earlier and executed the ”Config Node”script on)



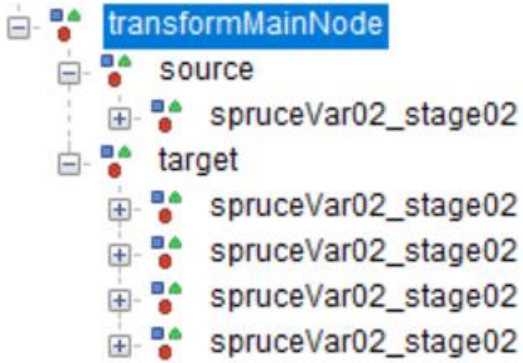


2. Execute script “Distribute objects” from Object Distributor



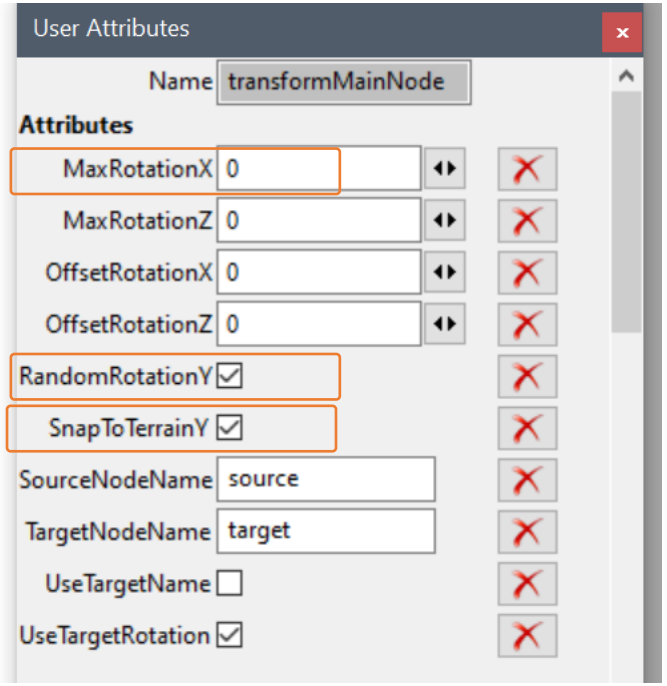


3. Target child nodes is now replaced with clones of the source child nodes



The log will display useful information, e.g. what settings are active

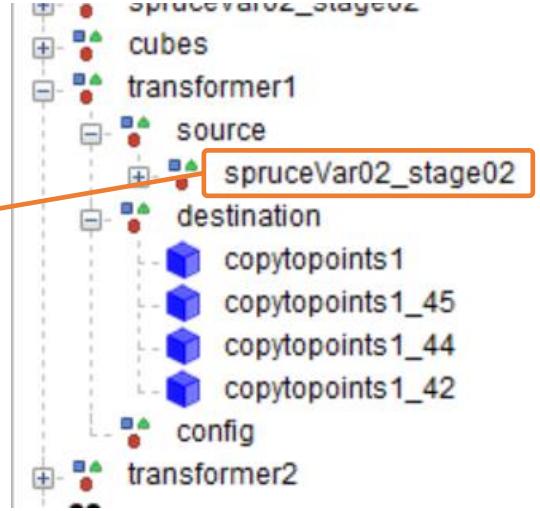
```
==[ WZL Object Distributor by w33zl - Distribute objects ]================================
Source node 'source' (2886) contains only one object, all replaced object will be the same
Found 8 target objects to be replaced
X max rotation set to 22.000000, objects will be tilted randomly on the X axis
Random Y rotation is enabled, objects will be randomly rotated 360 degrees on the Y axis
Snap to terrain is enabled, the height of objects will be adapted to the ground (if possible)
Replacing objects, please wait (it might take some time)...
```



Source object can be either randomly selected or static (“cherry picked” single object)

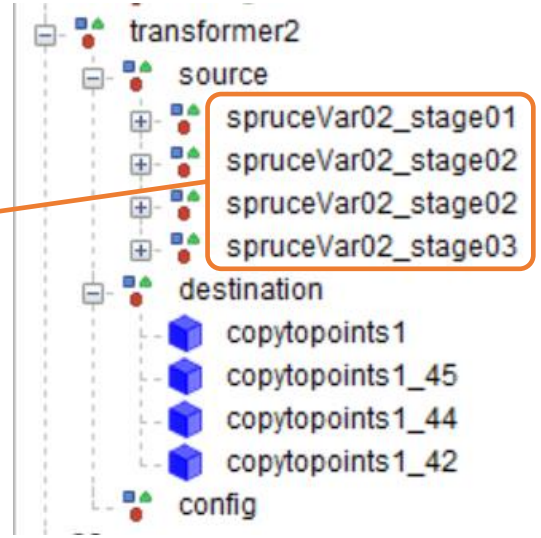
One/single source object = static source object
(i.e. every target will be exactly the same)

Every replaced target node will be “spruceVar02_stage02”, no variation.



Multiple source objects = random selection of source object
(i.e. there will be some variation)

The replaced target node can be any of the four source nodes, and with multiple copies of “stage02” those are more likely to appear



Settings and configuration

Controls the tilt/lean of the objects. This will allow a random deviation up to the supplied value, calculated from the neutral point (zero degrees), in both directions (positive and negative value). I.e. the actual maximum deviation in any give direction would effectively be half of the supplied value, e.g. a *MaxRotationX* value of “90” would result in a tilt between -45° and +45° on the X axis. A value of 0 means “disabled” (no randomization).

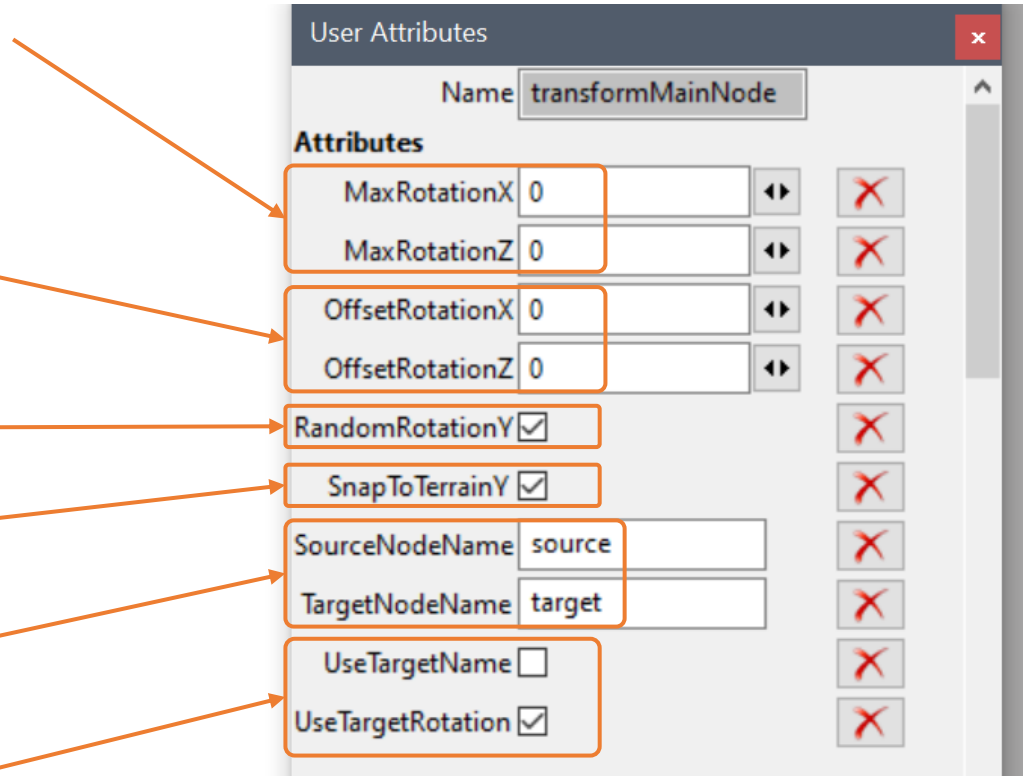
This is a *fixed* rotation *offset* which could be used to make *every* object to lean a bit in a given direction (i.e. not random). Can be used in combination with the MaxRotation*{X/Z}* to offset the randomized value (see next image).

Should the objects rotate on their own (Y) axis (i.e. facing different directions)

Should objects be snapped to the ground (regardless off Y value of the target object) or should the Y value of the target be respected

The parent *“source”* and *“target”* nodes can be renamed if desired, just make sure these two settings is the same as the actual node names

Use the name/rotation values from the target object (otherwise these values will be picked from the source object)



How the *MaxRotation* {X/Z} + OffsetRotation{X/Z} works in combination

MaxRotationX=45
OffsetRotationX=22

MaxRotationX=45
OffsetRotationX=0

How the *UseTargetRotation* setting affects the orientation of the final (replaced) objects

UseTargetRotation=false
(objects will always face upwards, unless tilted/rotated via other settings)

UseTargetRotation=true
(objects will face in whatever direction the original object was)

Warning: Not recommended together with any random rotation/tilt, causes unexpected results

Using *MaxRotationX* and *MaxRotationZ* together

It is possible, but can give quite strong effect, use with caution.

Using *MaxRotationX* in combination with *RandomRotationY*

Similar to using MaxRotationX and MaxRotationZ in combination, but with less dramatic effect (with the added benefit of rotation on it's own/Y axis, i.e. objects will be facing different directions)