

Object Array Blueprint

User Guide

[Object Array Blueprint - UE Marketplace \(unrealengine.com\)](https://unrealengine.com/marketplace/object-array-blueprint)

Please leave a review!

Support email: ScottreidUE4@gmail.com

Introduction	4
Example Level	4
Tutorials	5
Creating a Simple Array	5
Object Array with Modular Assets	7
Array Setup	12
Actor Type	12
Static Mesh	12
Blueprint	12
Static Mesh Instance	12
Static Mesh Actor/ Blueprint Actor	12
Object Array Type	12
End Point Set	12
Manually Set	13
Spaced Evenly	13
End Point Direct Distance	14
Spline Aligned	14
End Point	15
Manual X,Y,Z Count	15
Use Object Pattern?/ Object Pattern	15
Master Seed/ Random Master Seed	15
Refresh	16
Max object Quantity	16
Spawn Number	16
Unique Object Only	16
Enable Runtime	16
Object Offsets	16
XYZ Offsets	16
Use Offset Pattern?/ Offset Pattern	17
Offset Range Min/ Max	17
Object Rotation	17
Roll, Pitch, Yaw Rotation	17
Reset Rotation to Zero	17
Roll, Pitch, Yaw Random	17

Rotation Range?/ Rotation Range Min/ Max	17
Keep Object Flat?/ Inv Keep Object Flat	17
Align Vertically	17
Set Rotation List Toggle/ Set Rotation List	18
Use Rotation Seed?/ Rotation Seed/ Random Rotation Seed	18
Spline Rotation Align/ Spline Rotation Align Flat	18
Use Rotation Pattern?/ Rotation Pattern	18
Object Scale	18
Object Scale	18
Reset Scale to One	18
Scale Range?/ Scale Range Min/ Max	18
Uniform Scale	19
Set Scale List Toggle/ Set Scale List	19
Use Scale Seed?/ Scale Seed/ Random Scale Seed	19
Use Scale Pattern	19
Patterns	19
Object Pattern	19
Offset Pattern	19
Rotation Pattern	20
Scale Pattern	20
Snap to Ground	20
Snap to Ground	20
Snap Align to Surface	20
Snap to Ground Stack	20
Snap to Ground Vertical Adjust	21
Align Angle Sample Distance	21
Actor to Ignore	21
Physics Simulation	21
Simulate Physics?	21
Save Physics Transform/ Simulation Save Delay	21
Load Physics Transform	22
Apply Physics Transform	22
Physics Fall Transform	22
Do Not Spawn	22
Do Not Spawn List	22
Randomise Do Not Spawn List	22
Use Do Not Spawn Volume	23
Materials	23
Material	23
Keep Constant Material Scale	23
Manual Material Scale & UV	24
Material Scale	24

Material Scale X & Y	24
UV Shift X & Y	24
Receive Decals?	24
Spline	24
Use Spline Circle Calculate	24
Radius	24
Circle Type	24
End Tangent Curve	25
End Adjust	25
Use Reduce Spline Qty By?/ Reduce Spline Qty By	25
Troubleshooting	26
Warnings on Compile	26

Introduction

This will be an all levels of experience breakdown on how to use the Object Array Blueprint for Unreal Engine. It supports versions from Unreal Engine 4.10 up to and including UE5.

The Object Array Blueprint allows you to array out objects in the XYZ coordinates. It has many options and features that allow many variations, which creates a lot of potential with quickly placing objects in your environment.

Simply drop the blueprint into the level and add the objects you want to spawn in then grab the end point and drag it out. You then have the option to use the modifiers to create something that can randomise the spacing, rotation, scale and more!

The next few chapters will break down how to create specific common types of arrays to help you get ideas on how this can be used.

The following chapters will then break down each of the specific modifier sections of the blueprint so you can get the most out of the object array.

A few explainer videos can be found below.

Quick Overview: youtu.be/w9JYqx_uVZg

In depth look at all the features: youtu.be/pRG5aVvRn7c

Runtime Update: youtu.be/5LDkUKQv7-k

Please note that I have included a brief explanation on each variable using the tooltips if you hover your mouse over.

If you have any issues or need further explanation then please contact me via email at scottreidue4@gmail.com.

Example Level

After adding the Object Array Blueprint to your project via the Epic Games Launcher you will have access to an example level. The level will break down each modifier along with an information bubble further explaining each item.

...\Content\Object_Array\ExampleContent\Maps\Object_Array_Example

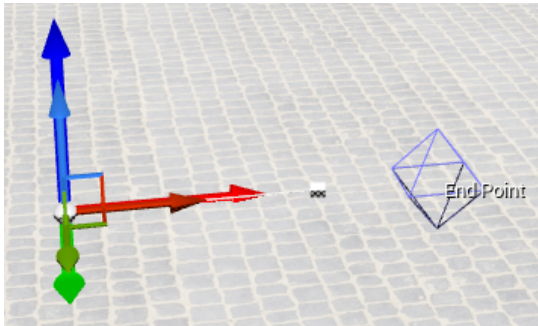
Note: First time opening this level will require it to prepare shaders, which can take some time.

Tutorials

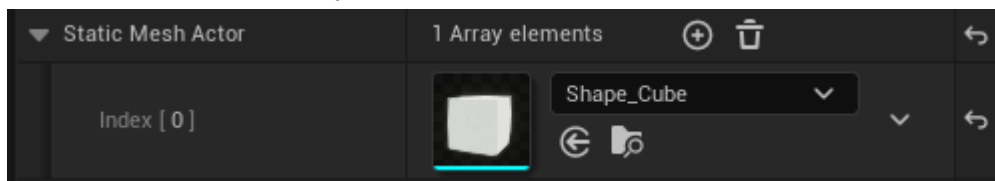
Creating a Simple Array

This will explain how to create a simple array, with some basic modifiers.

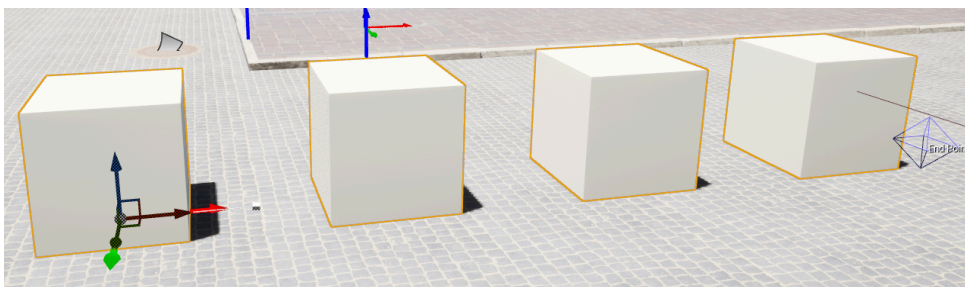
1. Drag in the BP_Object_Array from the content browser into your scene.
2. This will show as a blank object array consisting of three arrows indicating the axis directions and a wireframe diamond labelled 'EndPoint'.



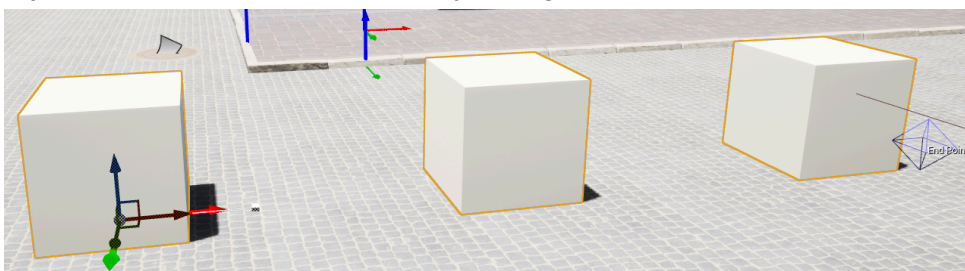
3. Select the Object Array and go to the details panel.
4. Under 'Array Setup' section click + on 'Static Mesh Actor', then drop down the list and select a static mesh asset. Try the 'Shape_Cube' static mesh provided in the sample content or one of your own.



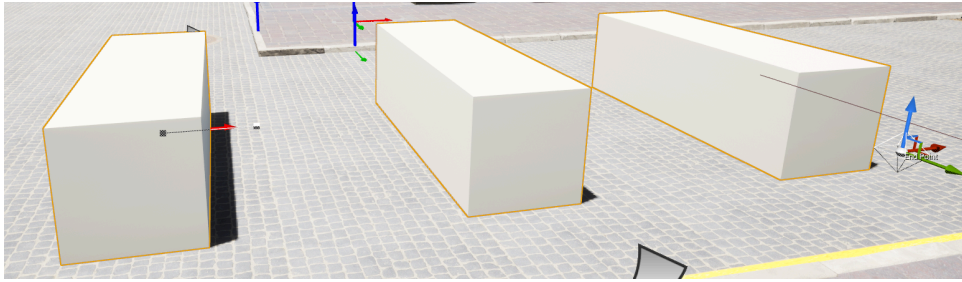
5. In the scene it should now display the object. Select the Endpoint diamond and drag it out using the red arrow (X axis) and it should replicate the object. Drag it out so 4 items are shown.



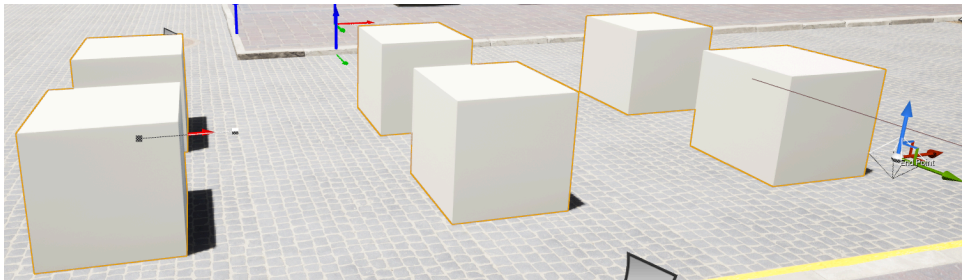
6. In the detail panel, go to the 'Object Offsets' section and change the default 'X Offset' to 300. This should now increase the spacing to 300 but reduce the total number of objects to three as there is now only enough space for three.



7. Select the 'EndPoint' in the scene and drag it using the green arrow (Y axis) so three objects show.



8. Go back to the 'Object Offsets' section and change the 'Y offset' to 200. This will again reduce the number of items as it is now using a larger offset for the space given.

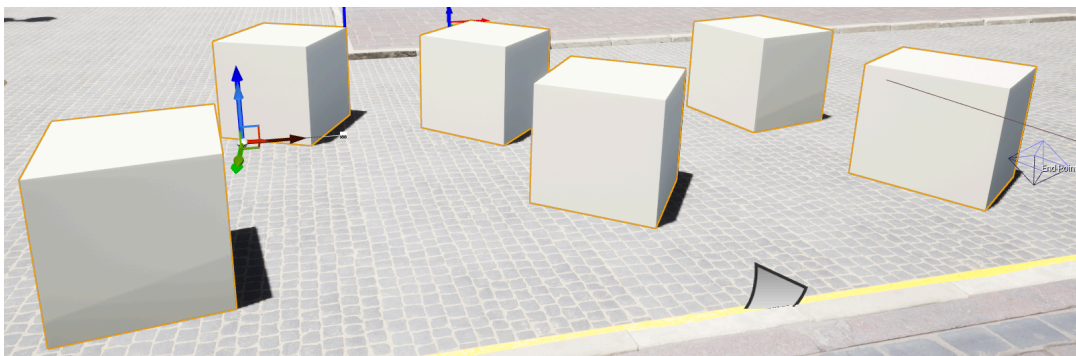


9. Let's add some random offsets, scale and rotations to our objects.
10. Under the 'Object Offsets' section change the 'Offset Range' Min and Max to -50,-50,0 min and 50,50,0 max.

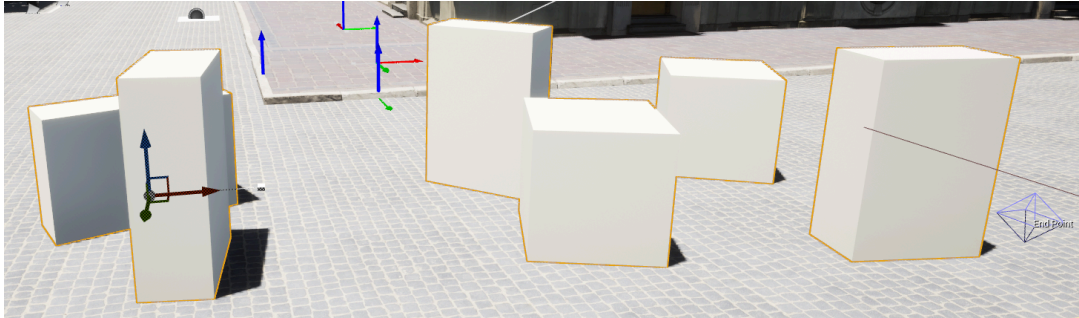
► Offset Range Min	-50.0	-50.0	0.0
► Offset Range Max	50.0	50.0	0.0

Take note how the cubes are moving each time you enter a value. It randomises within this range to move the objects based on the default XYZ offsets. In this example it has upto a ± 50 unit position change.

11. Let's now add some random rotations. Under the 'Object Rotation' section tick the three boxes for Roll (X), Pitch (Y) and Yaw (Z) Random. You should notice your objects go a little crazy and now look like some modern art. You would typically only use one of these at a time depending on what you are doing.
12. We will undo some of these random rotations by unticking Roll (X) and Pitch (Y) but leaving Yaw (Z) ticked. This will still look weird until resetting the Roll and Pitch rotations back to zero. This can be done simply by clicking on 'Reset Rotations to Zero' (or manually change the Rotations back to zero).
13. This should create some randomly placed and rotated objects.



14. Let's now play with scale. Under the 'Object Scale' section, enable the 'Set Scale list Toggle'. Click on the + next to 'Set Scale List' and increase to 3 array elements.
15. You should notice that some of the objects have now disappeared, this is because the new indexes in the array have a zero scale so the objects won't display. Leave 'index 0' at scale 1,1,1, change 'index 1' to 0.5,1.5,1 and change 'index 2' to 1.0,0.5,1.5.
16. The array will now apply these scales randomly to the objects.

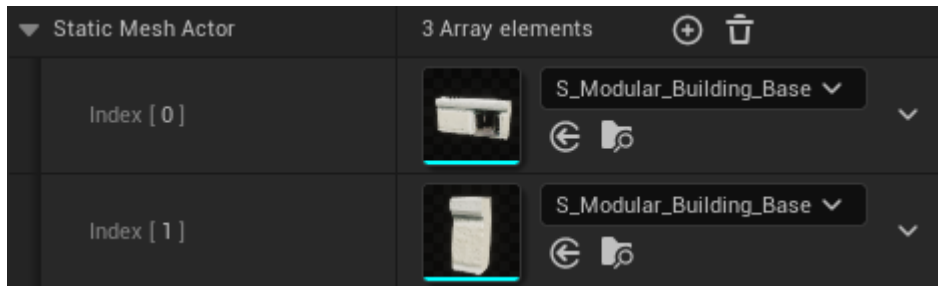


17. Your array will look different to the one shown above and this is due to randomisations of the modifiers used in this tutorial. If you are unhappy with how it looks, all of these random features are controlled by seeds (a master seed and a seed for each rotation and scale). Under the 'Array Setup' section click on 'Random Master Seed', this will randomly select another seed, which will change the randomisations on the offsets, rotations and scales.
18. This same process could be used to place a scattering of rocks, some light posts down a street, a picket fence and many other things.
19. Feel free to play with the other features to see how they work. Rest your mouse on each feature in the details panel and it will give you a bit more information on how it works or refer below to the specific feature explanations.

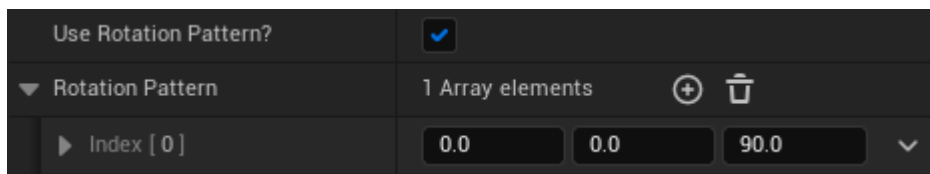
Object Array with Modular Assets

Using modular assets is a simple task when using the various patterns (object, offset, rotation and scale) in the object array. This allows different sized and rotated modular assets to be grouped together in the same array and easily stretched out.

1. First let's get some modular assets from Quixel Bridge. Epic Games has generously provided all assets on Bridge free for all users of Unreal Engine.
2. Load Bridge and click on Collections => Urban => Renaissance Revival Modular Building.
3. Download and load into Unreal Engine both 'Modular Building Ground Floor Kit' and 'Modular Building Base Kit'. I will be working with the low quality level but select whatever suits your needs.
4. Drag in a new BP_Object_Array and add the long and short base kit static meshes to the 'Static Mesh Actor'. Make sure you don't add the inside or external corner pieces by mistake.



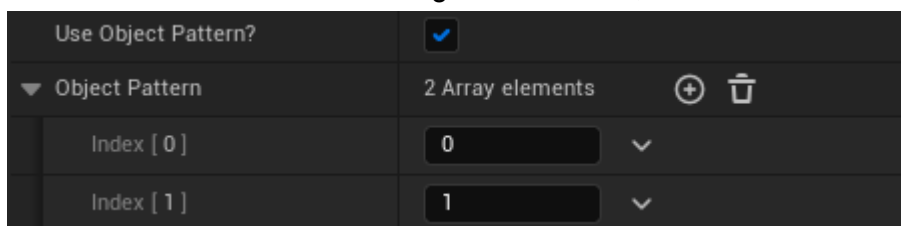
5. We will be using Patterns. Patterns will only work along the Y axis (green arrow). So keep this in mind if you find you are getting unexpected results.
6. Select the 'EndPoint' in the scene and stretch it out in the Y axis (green arrow) direction.
7. You might find that some modular assets might need rotating. To do this go to the 'Object Rotation' section and tick 'Use Rotation Pattern?' and change the first index to 0,0,90.



8. You should have an array of assets similar to below overlapping itself.



9. Next we will create a pattern for what order the static meshes spawn in. Go to the 'Array Setup' section and tick 'Use Object Pattern?' and add two elements to 'Object Pattern'. Object Pattern uses the index numbers from the 'Static Mesh Actor'. So in our case we want to have the long then the short, so 0 then 1.



This will make it spawn long, short, long, short, etc

10. We now need to specify the offset pattern so it knows what width each object is. Under the 'Object Offsets' section tick 'Use Offset Pattern?' and add two elements to 'Offset Pattern'.
11. If you rest your mouse on the static mesh in the content browser it will show the approx size **Approx Size: 250 × 52 × 55**. In our case the long mesh is 250 and the short is 50. So in the 'Offset Pattern' enter 250 in 'index 0' and 50 in 'index 1'.
12. To show things are working correctly and not overlapping you can increase each offset slightly to show a gap. As you can see below it is long then short repeating.



13. Move the 'EndPoint' in the scene so we have 5 long and 4 short. Like the above screenshot (but without the gaps).
14. We now want to modify this array to include a door in the middle. Add a third element to 'Static mesh Actor' and add the door from the Ground floor kit.
15. Add 9 elements to the 'Object Pattern' and update as per below. Notice it is the same repeating 0, 1 but now with index 2 (the door) in the middle replacing one of the long meshes.

▼ Object Pattern		9 Array elements	⊕	🗑
Index [0]	0	▼		
Index [1]	1	▼		
Index [2]	0	▼		
Index [3]	1	▼		
Index [4]	2	▼		
Index [5]	1	▼		
Index [6]	0	▼		
Index [7]	1	▼		
Index [8]	0	▼		

16. It should look the same as below. With 2 long and 2 short on each side of the door. If the door mesh was a different length than the long base mesh then we would have needed to update the 'Offset Pattern' similar to how we have done the 'Object Pattern' in step 15. Luckily for us it was the same so we could have the same simple pattern.



17. We now want to create a copy of this object array for the ground floor windows that will sit on the base meshes. Select the array and hold the ALT key and drag on the Z axis (blue arrow). This will create a copy. Alternately right click the object then edit then duplicate. Move the copy to the top of the base mesh.
18. Swap out the new Object Array copy with the ground floor kit window and pillar. Make sure you swap the long base with the window and the short base with the pillar so you then don't have to change any of the offset distances.
19. Remove the mesh reference from index 2, which was the door, making it blank. This will create a gap where the door is as it is now trying to spawn an object from index 2, which no longer exists. For an alternate way of removing specific objects, refer to the ['Do Not Spawn'](#) section.
20. Adjust the height of the Object Array in the scene so it aligns to the door mesh. This modular set embeds into the base quite a bit. You will find other modular sets sit perfectly on top of the mesh row below.



21. The modular sets from Quixel Bridge have multiple floors, so repeat the same process as per above for the other assets. Once a building is built I recommend

grouping the arrays together (Ctrl+G) so they can be duplicated and moved without leaving bits behind.

22. This is only scratching the surface on what can be done for modular assets using the Object Array Blueprint. See below for some example shots of different modular sets. This scene uses the object array for basically everything. From the curb to the footpath to the buildings and light posts.



Array Setup

Actor Type

Actor Type determines what type of objects spawn in the array.

Static Mesh

The typical array. Nothing fancy, just static meshes. Uses 'Static Mesh Actor' below.

Blueprint

Uses blueprints for the objects. Can be useful by combining objects together, like a point light and a static mesh of a light. Uses "Blueprint Actor" below.

Static Mesh Instance

The same as 'Static Mesh' above but it is using instances for each duplication. This can improve performance by reducing the amount of draw calls. Uses 'Static Mesh Actor' below.

Static Mesh Actor/ Blueprint Actor

This is where you load the static meshes or blueprint actors in to be used in the Object Array. It depends on what is set in 'Actor Type' above on what actor is used in this section.

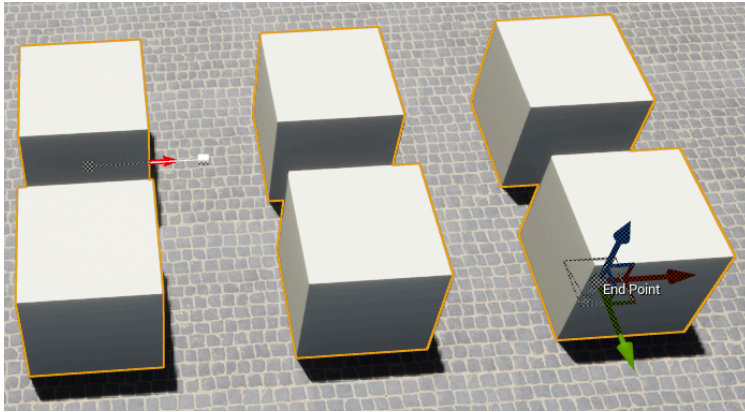
Object Array Type

The Object Array Type is how you manipulate the Object Array to create the objects.

End Point Set

This uses the relative distance of the 'EndPoint' from the Object Array origin and the XYZ offsets to determine how many objects spawn in each axis.

For example if the 'EndPoint' is at 400,150,0 and the offsets are X = 200, Y = 150 then it will spawn in 3 objects on the X axis and 2 on the Y axis. For the X axis it always spawns an object at the origin then two more fit within the 400 with an offset of 200. Same for Y, it spawns 1 (uses the same object as X) at the origin then 1 more fits within 150 at an offset of 150.

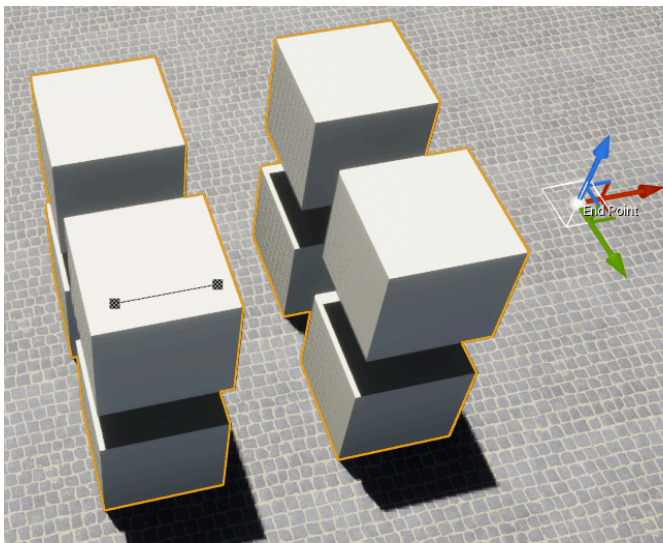


An example use case could be a set distanced object you want repeating. For instance a picket fence mesh that you want to repeat out to form a complete fence.

Manually Set

This uses the variable 'Manual X,Y,Z Count' and ignores the 'EndPoint'. Set it to whatever quantities you need and it will spawn that quantity (There is a safeguard called 'Max object Quantity', which prevents too many objects being spawned in at once and making your system unresponsive, adjust as necessary).

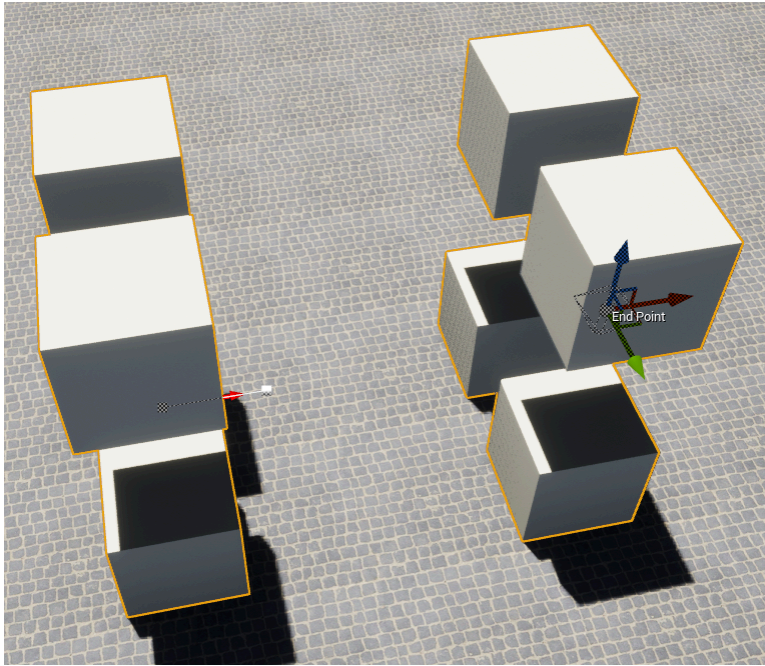
Below shows a 2,2,2 'Manual X,Y,Z count', notice the 'EndPoint' is ignored.



An example use case could be if you know exactly how many objects you want and at what spacing, for instance a row of chairs in a theatre.

Spaced Evenly

This is similar to 'End Point Set' except that it locks an object to the 'EndPoint' location and then spaces the object evenly back to the origin. It uses the XYZ offsets to determine how many objects spawn and when the 'EndPoint' goes over a whole number divisible to the distance from the origin it will spawn in another object in that row or column.

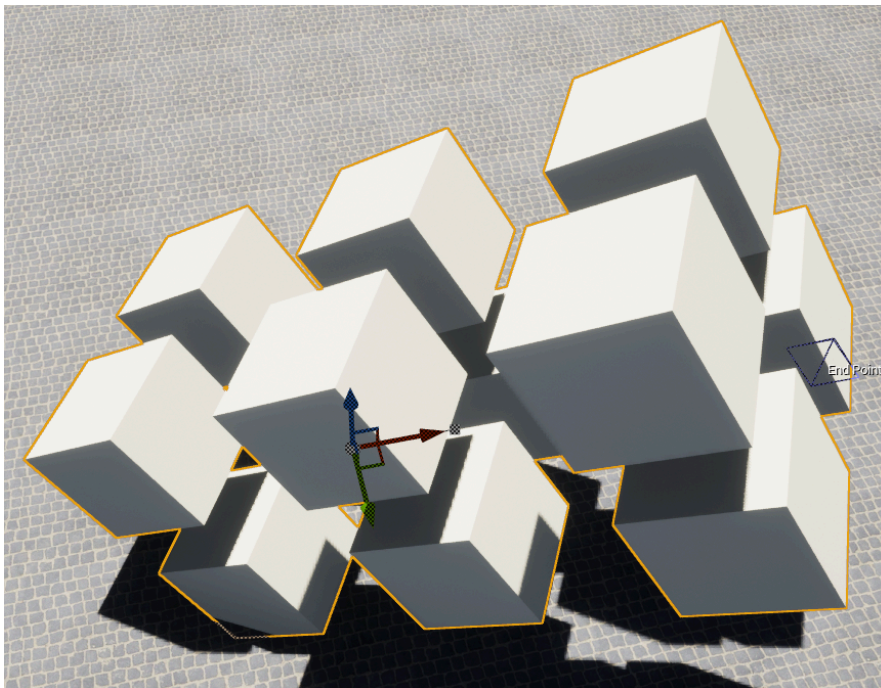


An example use case could be if you want the array to start and finish in specific locations. For instance some street lights that you want to go from corner to corner with even spacing between.

End Point Direct Distance

This is a bit of a hybrid between 'End Point Set' and 'Manually Set' with a twist (in rotation). The X axis always points towards the 'EndPoint' so the quantity of objects along the X axis is based on the direct distance back to the origin using the X offset. The quantity of objects for Y and Z is from the 'Manual X,Y,Z Count' variable.

Below shows the array pointing to the 'EndPoint' with 2 sets for both Y and Z on the 'Manual X,Y,Z Count' variable.



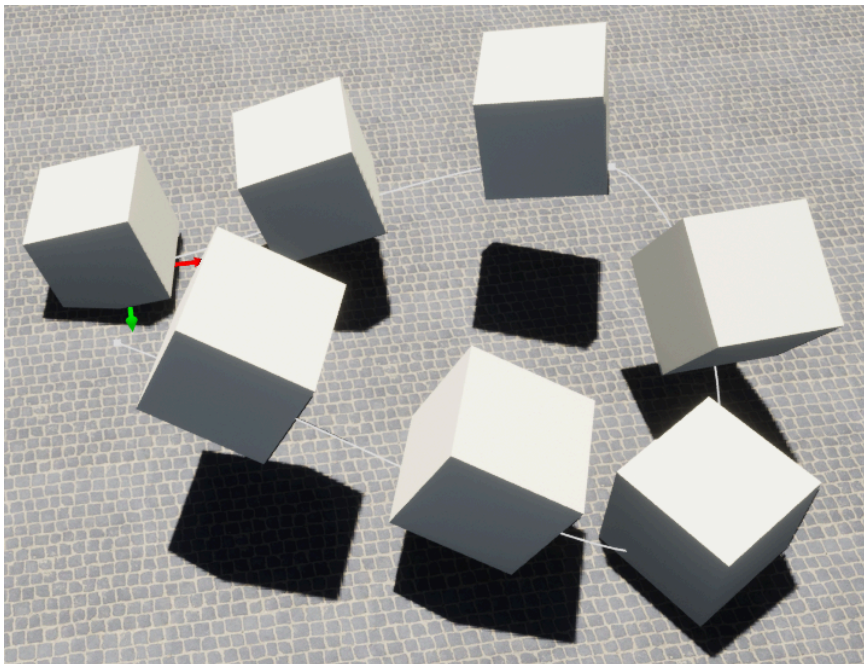
An example use case could be to align to another sloped asset. For instance used in conjunction with 'Keep Object Flat?' and 'Inv keep Object Flat' you can make some steps up a hill.

Spline Aligned

The objects align to a spline. There is a 100 unit long spline that can be moved and manipulated. By default the objects stay square to the array, but can enable 'Spline Rotation Align' to make the object align to the splines tangent.

Note: As of version UE 5.4, spline arrays need to be refreshed using the refresh button if the spline has been modified. The engine has now implemented that you can select individual entities and manipulate them.

Below shows a wavy spline with 'Spline Rotation Align' enabled.



An example use case could be to array out a collection of objects around a curved object. For instance some bricks to act as garden edging along a curved path.

End Point

Exposed to the scene and used to create arrays for 'End Pont Set', 'Spaced Evenly' and 'End Point Direct Distance'. Refer to [Object Array Type](#) above for more details.

Manual X,Y,Z Count

Used to create arrays for 'Manually Set', 'End Point Direct Distance' and 'Spline Aligned'. Refer [Object Array Type](#) above for more details.

Use Object Pattern?/ Object Pattern

Allows you to set a pattern for what order the objects spawn in your array. Uses the ID from the list of actors. Patterns along the Y axis only.

Refer to the [Patterns](#) Section for more details.

Refer to the tutorial [Object Array with Modular Assets](#) for a good use case.

Master Seed/ Random Master Seed

This is the seed that all the randomisation is based on. This allows you to replicate results using the same seed.

You can input a seed directly into the 'Master Seed' or use 'Random Master Seed' to randomise the seed.

The Rotation seed and Scale seed can override the master seed so you can have a specific seed for each. Refer [Use Rotation Seed?/ Rotation Seed/ Random Rotation Seed](#) and [Use Scale Seed?/ Scale Seed/ Random Scale Seed](#) for more information.

Refresh

You may find that the array needs to refresh under certain situations. This button does just that!

For instance using the 'Snap to Ground' feature, if the object that it has snapped down to has moved, then the array needs to be refreshed to adjust itself to the moved location.

Note: The array also refreshes itself when moved or any variable is changed.

Max Object Quantity

This is a safety feature to help prevent you accidentally spawning in too many objects and locking up your computer. If you accidentally add a decimal place to the offset or an extra zero or two to the Manual count then this has you covered to make sure your computer doesn't freeze.

If you have a simple object and need more than the default 200 then feel free to adjust it.

If you do reach this max quantity then a little message will appear up the top left corner saying that the max object quantity has been reached.

Spawn Number

For information only, it shows the base zero quantity of how many objects are in the array.

E.g if it shows 7 then there are 8 objects.

Unique Object Only

Enabling this will only ever spawn in 1 of each actor type. It will randomly place them within the array. Useful if you have many different objects in 'Static Mesh Actors' or 'Blueprint Actors' but only want to display 1 of each.

Enable Runtime

This allows the Object Array to be modifiable via blueprint during gameplay.

Refer to the following video showing how to update the Object Array via blueprint

youtu.be/5LDkUKQv7-k

If updating via blueprint then you may want to use 'Enable Runtime Update Repeat', which will automatically update the array at what interval you have in 'Runtime Update Repeat Delay'.

Note: The Object Array Blueprint wasn't designed for runtime updates, it was designed for constructing a level, so it isn't optimised to be updated frequently. If you have a large number of objects you are updating during gameplay then it could have a large performance hit.

Object Offsets

XYZ Offsets

Used to offset the objects in the array in the XYZ axis directions.

Use Offset Pattern?/ Offset Pattern

Allows you to set a pattern for what offsets objects spawn in your array. Patterns along the Y axis only.

Refer to the [Patterns](#) Section for more details.

Refer to the tutorial [Object Array with Modular Assets](#) for a good use case.

Offset Range Min/ Max

Minimum and Maximum random spawn offset from the default Object Array offset

Object Rotation

Roll, Pitch, Yaw Rotation

You can set an override of each axis rotation. It will also display the last spawned rotation if using any random modifiers.

Reset Rotation to Zero

Resets the 'Roll, Pitch and Yaw Rotations' to zero. If there are modifiers still active then it won't change that value.

Roll, Pitch, Yaw Random

Check boxes for each axis to randomise the rotation. It uses the 'Master Seed' or the 'Rotation Seed' if set.

Rotation Range?/ Rotation Range Min/ Max

Allows a min and max range that the randomised rotation can use. Based on 360°. It uses the 'Master Seed' or the 'Rotation Seed' if set.

Keep Object Flat?/ Inv Keep Object Flat

Aligns the object to the XY plane of the Object Array. Best used in conjunction with [‘End Point Direct Distance’](#). This rotation is relative to the Object Array and not world.

You can use ‘Inv Keep Object Flat’ to invert the result depending on the array direction.

Align Vertically

Aligns all Z axis objects directly above the base object. Best used in conjunction with [‘End Point Direct Distance’](#). This rotation is relative to the Object Array and not world.

Set Rotation List Toggle/ Set Rotation List

You can create a list of rotations that the Object Array will randomly select for each object. It uses the ‘Master Seed’ or the ‘Rotation Seed’ if set.

Use Rotation Seed?/ Rotation Seed/ Random Rotation Seed

Enabling ‘Use Rotation Seed?’ will use the ‘Rotation Seed’ and will override the ‘Master Seed’ for all rotation randomisations.

This allows you to replicate results using the same seed. You can input a seed directly into the ‘Rotation Seed’ or use ‘Random Rotation Seed’ to randomise the seed.

Refer to the [Master Seed/ Random Master Seed](#) for detail on the Master Seed.

Spline Rotation Align/ Spline Rotation Align Flat

‘Spline Rotation Align’ is used in conjunction with [Spline Aligned](#) to align the objects to the splines tangent in all axes.

‘Spline Rotation Align Flat’ is used in conjunction with ‘Spline Rotation Align’ to flatten the object so only the Z axis is rotated to align to the spline. You may need to reset the rotations to zero to get it to display correctly.

Use Rotation Pattern?/ Rotation Pattern

Allows you to set a pattern for what rotation objects spawn in your array. Patterns along the Y axis only.

Refer to the [Patterns](#) Section for more details.

Refer to the tutorial [Object Array with Modular Assets](#) for a good use case.

Object Scale

Object Scale

You can set an override of each axis scale. It will also display the last spawned scale if using any random modifiers.

Note: Do not use zero in any field as this will result in the object not showing.

Reset Scale to One

A quick way to reset back to the default scale of 1.

Scale Range?/ Scale Range Min/ Max

Allows a min and max range that the randomised scale can use. It uses the 'Master Seed' or the 'Scale Seed' if set.

Note: Do not use zero in any field as this will result in the object not showing.

Uniform Scale

Used with 'Scale Ranges', it allows for the scale to remain uniform and only uses the X scale of the Min/ Max Scale Range.

Set Scale List Toggle/ Set Scale List

You can create a list of scales that the Object Array will randomly select for each object. It uses the 'Master Seed' or the 'Scale Seed' if set.

Note: Do not use zero in any field as this will result in the object not showing.

Use Scale Seed?/ Scale Seed/ Random Scale Seed

Enabling 'Use Scale Seed?' will use the 'Scale Seed' and will override the 'Master Seed' for all scale randomisations.

This allows you to replicate results using the same seed. You can input a seed directly into the 'Scale Seed' or use 'Random Scale Seed' to randomise the seed.

Refer to the [Master Seed/ Random Master Seed](#) for detail on the Master Seed.

Use Scale Pattern

Allows you to set a pattern for what scale objects spawn in your array. Patterns along the Y axis only.

Refer to the [Patterns](#) Section for more details.

Patterns

This section will go into more detail of the pattern system explaining each pattern type. There are 4 types of patterns, 'Object Pattern', 'Offset Pattern', 'Rotation Pattern' and 'Scale Pattern'. They can be found under each of their sections on the details panel for the Object Array Blueprint.

Patterns follow the sequence in the pattern then repeats the sequence if the array is long enough.

Refer to the [Object Array with Modular Assets](#) tutorial for a good way to use the pattern system.

Object Pattern

Allows you to set a pattern for what order the objects spawn in your array. Use the ID from the list of actors in either 'Static Mesh Actor' or 'Blueprint Actor'. Patterns along the Y axis only.

Offset Pattern

Allows you to set a pattern for what offsets objects spawn in your array. Patterns along the Y axis only.

For example, used in conjunction with 'Object Pattern', you could have two different width meshes that could alternate back and forth. You could then use the 'Offset Pattern' to alternate back and forth the two width values.

Rotation Pattern

Allows you to set a pattern for what rotation objects spawn in your array. Patterns along the Y axis only.

For example, it can be used to create a sweeping curve with set rotations for each next iteration. You can also use it to lock in a rotation value for an object in case you duplicate the array it won't reset to default values.

Scale Pattern

Allows you to set a pattern for what scale objects spawn in your array. Patterns along the Y axis only.

For example, you can have each object get larger or smaller as the array progresses.

Snap to Ground

Snap to Ground

The toggle to turn 'Snap to Ground' on and off.

'Snap to Ground' works by casting the objects in the array vertically down in the Z axis until it hits something with collision. It uses the 'Snap to Ground Distance' to determine how far it looks vertically for the collision from the object array's origin point.

It uses the 'Z Offset' to know where to look for the next collision. This feature means you can have rows of shelves that the objects automatically snap to the shelf.

Note: 'Snap to Ground' only casts down on the world's Z axis, not the object arrays relative Z axis.

Note (again): If the array has rotations already set in the X, Y or Z fields then it won't align correctly. These values will need to be reset to zero manually or by clicking on 'Reset Rotations to Zero'

Snap Align to Surface

The toggle to turn 'Snap Align to Surface' on and off.

'Snap Align to Surface' works by taking sample traces of the surface in each direction to determine the angle of the surface directly under the object. It uses this angle to align the objects to the surface. It is used in conjunction with 'Align Angle Sample Distance'.

Snap to Ground Stack

The toggle to turn 'Snap to Ground Stack' on and off.

By default the 'Snap to Ground' feature ignores the other objects in the array. Turning this on will allow it to stack the object in the Z axis on top of the ones below it.

A good example is to create a stack of boxes that will use its own collisions to determine where to place the next.

Snap to Ground Vertical Adjust

This allows you to adjust the height the object has snapped to the surface. E.g. you can lift or sink the object relative to the surface.

A good use case can be for sinking in some pavers or trees into the landscape a little to bed them into the world better.

Align Angle Sample Distance

Used in conjunction with 'Snap Align to Surface'. This is the distance between the sample traces that determines the angle of the surface under the object. You will need to change this to suit your object size. It should be set similar to the object size you are spawning in. A larger size will likely miss the smaller deformations.

Actor to Ignore

You can select objects in your scene that the 'Snap to Ground' feature will ignore and not snap to. Add elements to the array and pick the objects from your scene using the drop down menu or the eye dropper.

Physics Simulation

Simulate Physics?

This allows you to simulate physics with the objects in the array. On play they will be affected by gravity.

Note: Will not work with 'complex as simple' collisions set on the static mesh. Will only work with Static meshes.

Save Physics Transform/ Simulation Save Delay

This system can save the locations of a physics simulation. This allows you to prerun the physics simulation and save on running any simulations during gameplay.

Enable 'Save Physics Transform' and set 'Simulation Save Delay' to a delay that will ensure you capture the objects once they have fully rested on the ground.

Run Simulate so the player character won't be in the way and wait until you see the '<Object Array Name> Locations Saved!' in the top left corner. Stop Simulate and move to [Load Physics Transform](#).

If the delay isn't long enough, adjust and run Simulate again.

Note: Will not work with complex collisions set on the static mesh. Will only work with Static meshes.

Load Physics Transform

Once you have run '[Save Simulate Transform](#)' click on 'Load Physics Transform' and it should load the saved locations from the physics simulation and auto check 'Apply Physics Transform'.

If unhappy, uncheck 'Apply Physics Transform' and rerun '[Save Physics Transform](#)'.

Once happy, uncheck 'Simulate Physics?' and 'Save Physics Transform' to ensure the locations don't get overridden.

Apply Physics Transform

This is automatically checked when 'Load Physics Transform' is selected. If unchecked it will revert back to the default array. At this point you can change the Object array to a static Mesh instance or blueprint array as Physics simulation is no longer being used. Just be mindful that if you use different objects to match the size or there might be clashes or gaps.

Note: With this enabled it will lock the Object Array to any changes, so will need to be unchecked if you need to change any defaults.

Physics Fall Transform

This lists all the array elements locations once 'Save Physics Transform' and 'Load Physics Transform' have been run.

This list can be modified to adjust any separate element.

Do Not Spawn

Allows a couple of ways to remove certain objects from an array. Can be used for creating gaps in the objects or making something more random and natural looking by removing some of the objects.

Do Not Spawn List

Create an array of the object IDs that get culled from spawning. This is not base zero so the IDs start from one. E.G. if you have a straight array 5 objects long and you add two elements in the array with values 1 and 5, then it will cull the first and last object.

Note that when implementing more than one axis into the array it can be difficult to determine what order the object has spawned in and hence what object you want culled.

Refer to the do not spawn volume for an easier way of culling objects.

Randomise Do Not Spawn List

Used to randomise the 'Do Not Spawn List' with up to the maximum number of objects and not repeating the same number twice.

A good use for this is if you had a scattering of objects across a surface, culling some of the objects randomly will make the scene more natural looking.

Note: It will automatically cull the list size if it is larger than the maximum number of objects the array allows.

Use Do Not Spawn Volume

Next to the BP_Object_Array in the content browser is the BP_Object_Array_DoNotSpawnVolume, which can be dragged into the scene. If an object overlaps with the volume with this active it will be culled. If you move the volume only, the array will need to be refreshed by clicking on 'Refresh' or by adjusting it in any way.

Note: Do not spawn volumes do not work with instance static mesh arrays.

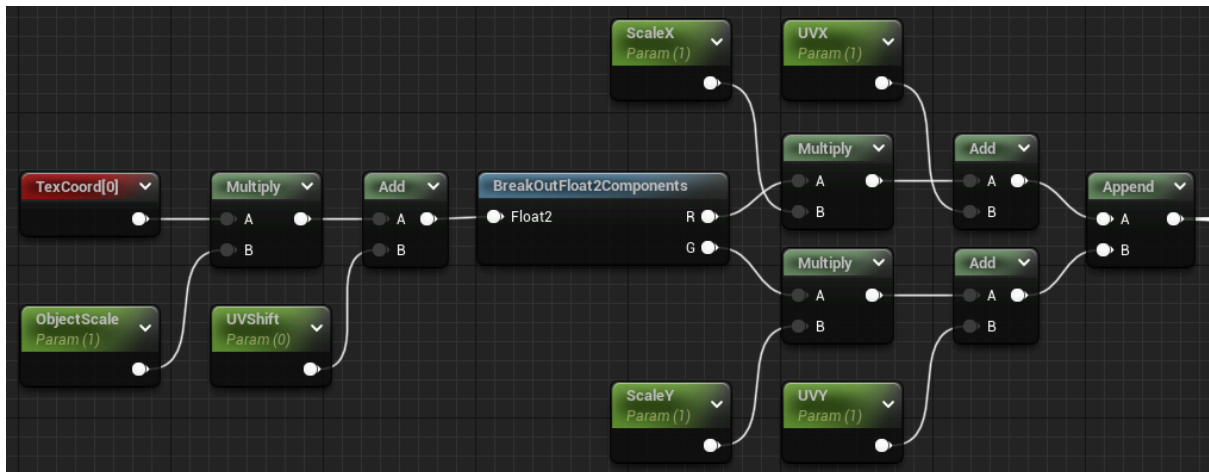
Materials

Material

You can set a material that will be applied to the objects in the array. If you add more than one material then it will randomly select the material for each object.

Note: This only works with static mesh arrays. It doesn't work for blueprint arrays or static mesh instance arrays.

To get the full function out of the following material controls you need to add the below nodes to any UV input within your material. Typically the UV input is in the back of texture samples, but it may also feed through more advanced mathematics.



There are 6 'Scalar Parameters' each labelled as follows. 'ObjectScale', 'UVShift', 'ScaleX', 'ScaleY', 'UVX' and 'UVY'. If you find that one isn't working then likely the parameter name here has been spelt wrong.

Keep Constant Material Scale

Turning this on will keep the scale of the material the original scale and consistent even if the scale of the objects vary within the Object Array. You need 'Manual Material Scale' turned off for this to function.

Manual Material Scale & UV

Turning this on allows 'Material Scale', 'Material Scale X', 'Material Scale Y', 'UV Shift X' and 'UV Shift Y' to function. Turning this on also disables 'Keep Constant Material Scale'. This has been added to version UE5 and above only.

Material Scale

Adjust the variable to scale the material uniformly. You need 'Manual Material Scale' checked for this to function.

Material Scale X & Y

Adjust the variables to scale each X & Y independently. You need 'Manual Material Scale' checked for this to function.

UV Shift X & Y

Allows the texture to be moved in the X & Y direction. Perfect for aligning the texture to another object.

Receive Decals?

This allows the objects within the array to receive decals or not.

Spline

The following functions only work with the 'Object Array Type' set to 'Spline Aligned'.

Use Spline Circle Calculate

Turning this on will change the spline to a circle. It uses the 'X Offset' for the object spacing. Refer to the below spline controls to customise this circle.

Note: The spline will be displayed but only as read-only, it can't be modified. If you have already moved the default spline, then it will remain in its set position even if 'Use Spline Circle Calculate' is turned on, however it will still create a circle but you won't see the circle spline but instead your modified default spline.

Radius

This is the radius of the 'Spline Circle Calculate'.

Circle Type

You can change the type of circle to the following. ' $\frac{1}{8}$ Circle', ' $\frac{1}{4}$ Circle', ' $\frac{1}{2}$ Circle' and a 'Full Circle'

End Tangent Curve

This variable adjusts each end of the spline's tangent point. Its default is 0.8, which should suit most applications but it can be adjusted if needed.

End Adjust

This variable moves the first and last point of the spline apart. This can be useful to slightly adjust the spline if needed.

Use Reduce Spline Qty By?/ Reduce Spline Qty By

Turning this on and using 'Reduce Spline Qty By' removes objects at the end of the array by the number set. This can be useful if you need to remove overlapping objects at the start and end of the spline or if you don't want a complete circle.

Troubleshooting

Warnings on Compile

There will be some warnings on compile due to dividing by zero. This happens in features that are not active. When these features aren't active they change some variables to zero in some calculations that use division, resulting in the error. The compiler will see these calculations in the turned off features and show a warning even though it will never see gameplay.

This can be safely ignored.