Automatic light probe generator

By Jacob (Kopter) Andersson

Table of contents

Table of contents **1**
Philosophy **2**
Functionality **2**
Nav mesh based 2
Clustered merging process 2
Collision masking (Tag Check) 3
Collision checking (Merge through colliders) 3
Out of bounds cleanup (Vis Check) 3
Probe volume component **4**
Controls & settings 4
Operation 4
Synced reflection probe 4
Settings **5**
Setup **6**
First steps 6
[Step: 1] 6
[Step: 2] 7
[Step: 3] 8
[Step: 4] 9
Wrap-up **10**
Contact me or follow me 10
FAQ **11**

Philosophy

The creation of this tool was fueled by the raw hatred for Unity's light probe "support". Why would you ever *want* to manually place these little points out around your levels, in order to have some semblance of "proper" lighting? And why would you ever want to *keep editing* these as you're iterating on your layouts? The answer is; You wouldn't.

I love it when you can boil down a task to be more or less a literal singular mouse click. This is what I've attempted to apply when making this tool, just to give an extra big middle finger to the stupidly annoying problem at hand.

Hopefully this can connect with more of you devs out there that have felt a similar frustration with the lighting systems in Unity.

Functionality

Nav mesh based

The main crux of this tool is that it is dependant on a baked nav mesh. As long as this is supplied, the rest of the generation process should be all set for. There is a built in function for generating a separate nav mesh surface with its own settings. This enables you to specify tile density and other agent settings specific to the probe generation process. It's highly advised that you create a new agent for this process, to avoid overwriting any other nav mesh data you've already calculated. One thing to note with this function is that it's an "additive process" to your already existing nav mesh data. This doesn't matter too much due to the merging steps though.

Deriving the probe locations from the vertices in the nav mesh gives a rather nicely spaced out, and sparse amount of probes that are tied to the play area in the same way you would be moving through it. One big issue with this that I had to solve, was the erratic frequency of some nav mesh shapes mostly created around bent or "organic" geometry. Sometimes, probes just land too close to each other to be seen as efficient or necessary. To solve this, we merge.

Clustered merging process

Merging probes based on distance is more than easy to say the least. The problem was having thousands and thousands of probes comparing themselves to each other, even though they weren't even close in a relative sense. It was a total waste of processing, and we want speed. This is where clustering the level helps.

By using a big volume that we envelop our play-space in, we can first of all cull the probes that land outside of it. Cool, lots saved by just that alone. Then we fill this volume with a variable amount of cluster volumes that we then section off the probes into. By doing this, we can get the amount of calculations done, down to way lower figures.

If we have 50 000 probes to cycle through, we end up with “50 000 ^ 2 = **2 500 000 000**” calculations. If we have 16 clusters, and uniformly distribute these 50 000 probes into them, we end up with:

50 000 / 16 = 3125
3125 ^ 2 = 9 765 625
9 765 625 * 16 = **156 250 000** calculations

That’s 6,25% of the calculations needed to be performed by the main merge pass compared to the naive process. Of course, it’s not a “free” process to place the probes into the clusters, but it’s still a quicker process than without them. The tradeoff is, the more clusters you have, the more you have to go through when clustering, but it reduces the total amount of comparisons needed exponentially. This feature is great for larger levels.

Collision masking (Tag Check)

On top of this merge process, I’ve added in a simple collision masking pass, that makes probes inside of geometry be culled. This includes concave mesh colliders, by temporarily making them convex, and then switching it back in the final steps. This can come with some issues when trying to place probes around certain colliders, since they end up masking outside of their usual bounds.

You can of course disable this process, but I recommend keeping it turned on, since the nav mesh always bakes wherever it wants inside colliders…

There is another empty component included in the package to mask certain objects or prefabs from the process. Simply add the “LightProbePlacement_SkipObject” component on a prefab root object (to exclude entire prefab), or individual object to exclude it.

Collision checking (Merge through colliders)

You can choose whether or not to avoid merging probes through level geometry using a simple collision checking pass in the main pass. Since it performs quite a few raycast, it’s not recommended to use this if you can avoid it!

Out of bounds cleanup (Vis Check)

A questionably useful post-op step added to clean up probes outside of the immediate play-space. What this operation does is, from each corner of the levels terrain (with a tweakable vertical offset applied)- it looks at each probe, and if they’re seen from any corner, they’re removed. This means an operation like this is only really useful if, first of all; you’ve got a terrain component in your level, but also that your play-space is enveloped by geometry or colliders.

I added this pass from working on a project that more or less purely has this type of level design, so it works wonders in there! But for other projects, I’m not sure you’d want this to be enabled by default. It has an option to check visibility through specific layers, so maybe you can find a use-case that works well for your project.

Probe volume component

Controls & settings

This component creates a volume “filled with probes” in a tweakable 3d grid pattern. The transformation of the volume is fully free-form and isn’t “axis locked”. You’re free to scale, rotate and translate it however you want. The density is not dependant on the volume scaling either, but keeps it consistent with world units. Note: setting the density to be higher than 1 probe/ 1m^3 is not recommended as it would be unnecessarily dense.

You also have the option to weigh the different volume’s “axis densities”. This means that you’re able to maintain a higher “horizontal density” for more frequent variations in lighting on the “XZ-plane”, but don’t need it for the vertical “Y-space” for example.

Operation

After you’re happy with the density settings and positioning, it waits with generating the probes until it’s time to generate the probes using the master.

Once the master starts its operation, it will look through the scene and find all of these volumes that has been placed out *(It doesn’t matter if they’re placed inside prefabs or other objects, just make sure the transforms on those parent objects are un-rotated and un-scaled!)*. If you enable “Apply merge pass”, the volume’s probes get all of the benefits of the main process with collision and visibility checking!

If you’d rather just have it as a separate light probe group though, you can click the “Bake Volume” button at the bottom to separate out the component into a Light Probe Group with the current settings applied.

Synced reflection probe

As an addition to this component, you have the option to create a reflection probe component that can automatically sync it’s bounds to the volume’s shape, since you’d most likely use this component in interior environments that would need reflection probes with a similar bound-shape.

Settings

The parameters exposed to you in the script's inspector layout should be very self-explanatory, but here's a quick rundown of each setting:

- **Merge distance**: The threshold distance probes merge under
- **Merge distance (airProbe)**: The threshold distance airProbes merge under (should be large to decrease unnecessary probe placements above play-space).
- **Floor offset**: The height offset from lower nav mesh vertices.
- **Height checking distance**: Distance to raycast upwards from each floor point, to place out airProbes or fill up interior/ vertically blocked spaces.
- **Collision check radius**: Radius for overlap checks. (should be rather small, or at least smaller than floor offset. Otherwise overlaps will happen with floor, and it will cull).
- **Collision check layers**: Layers to check probe collisions with.
- **Cluster count**: Amount of clusters to place probes in. (Smaller levels/ spaces should have lower amounts to speed up processing).
- **Light probe nav mesh**: Generate a nav mesh surface with custom settings for the light probe generation process. This can enable better threshold ranges from walls and densities across open terrains, but makes the process slower. Depending on your map size, this can considerably increase calculation times!
- **Force build**: Force the process to rebuild the custom nav mesh surface, rather than reusing a cached one.
- **Agent type**: Choose what agent type to use for nav mesh generation. It's highly recommended to create a new agent for this process to avoid other agent's data being intervened with.
- **Nav Tile Size**: The density of tiles in the nav mesh. This can increase/ decrease the amount of probes being placed out in open spaces.
- **Nav mesh bake on layers**: Choose what layers the custom nav mesh surfaces takes into account for calculation.

- **Merge through colliders**: Enable/ disable probes being able to merge through collision.
- **Tag Check**: Enable/ disable collision exclusion pass.
- **Vis Check**: Enable/ disable out of bounds checking.
- **Vis Check Height**: Vertical offset for Vis Check process. (Terrain Y + Offset)
- **Vis Check Layers**: Layers to block Vis Checking rays.

- **Show progress bar**: Show more detailed information from generation process.
- **Obj: Light probe group**: The scene object reference to place the processed light probes into. It's recommended to just use the "ALPG_Master" prefab supplied in the package.
- **Obj: Probe zone**: The scene object reference the probe zone volume. All probes outside of this volume will be culled. It's recommended to just use the "ALPG_Master" prefab's zone that is supplied in the package.

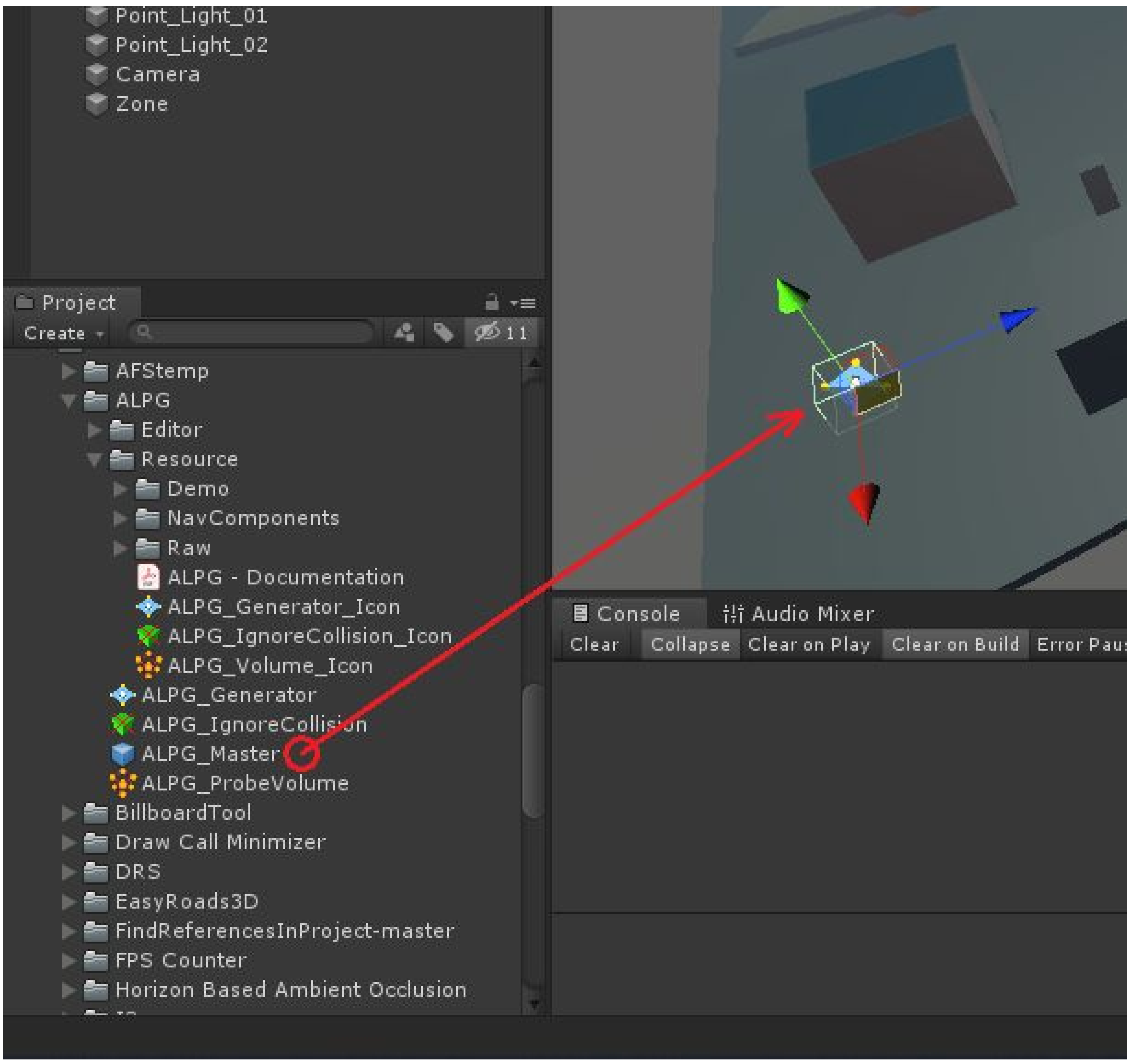
Setup

First steps

[Step: 1]

To get started; simply drag in the supplied <ALPG_Master> prefab into your scene. Zero out it's transform to be located at the origin to avoid offsets in the volume.

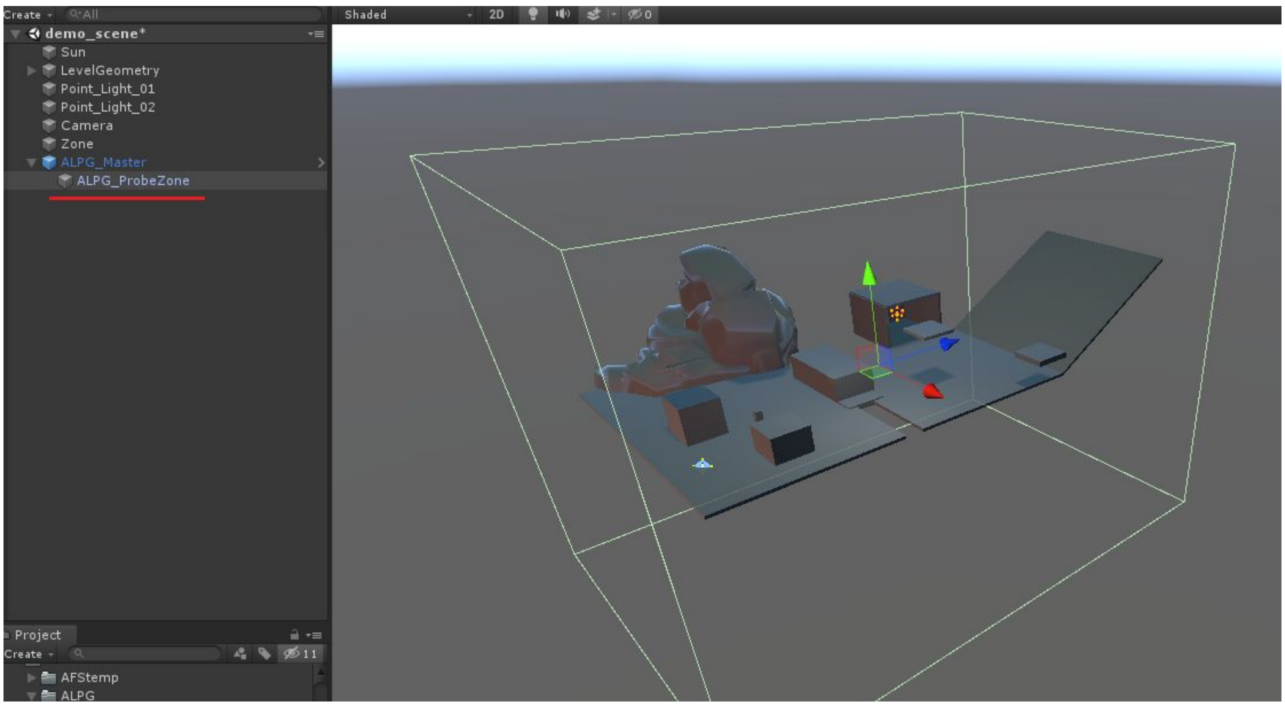
Note:
It is **not** recommended to re-create a new "master prefab" or add the <ALPG_Generator> component on a new GameObject! There's no need for this, and there would be room for error to be able to miss important setup steps!



[Step: 2]

Proceed with scaling, rotating and translating the <ALPG_ProbeZone> object, located within the prefab to cover whatever area you want to calculate the probes.

Note:
Keeping this volume as small as possible is ideal to speed up calculation times and keep down memory footprint! Keep in mind that the clusters are currently only divided on the X and Z axis, as Unity is a Y-up editor, and we're only doing basic divided checks to speed up calculation times!

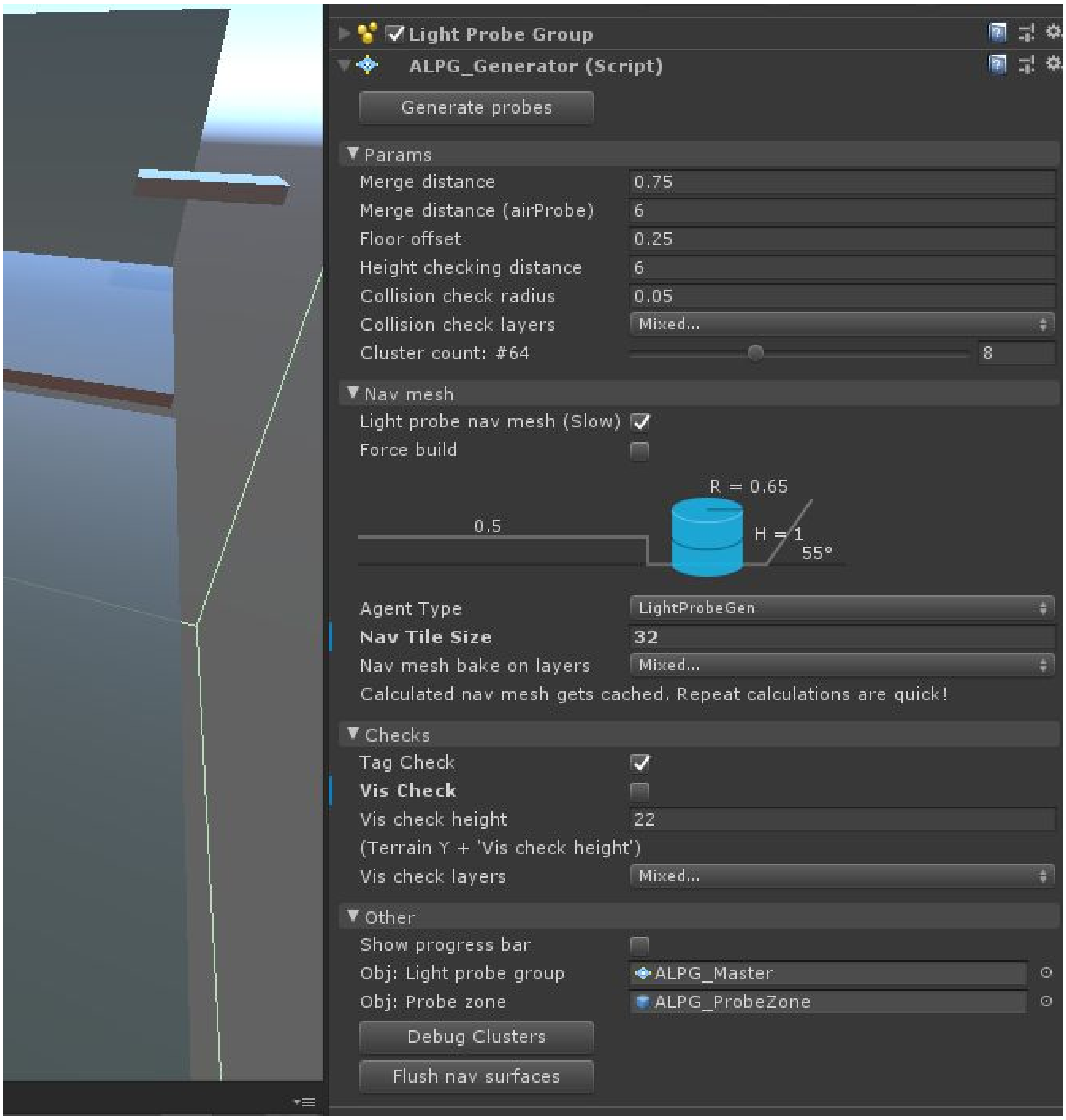


[Step: 3]

Check your settings to make sure you've got them tweaked to your needs. Also supply a nav mesh actor if you're planning on using the "Light probe nav mesh" function!

You can mostly leave the settings to whatever default values the prefab comes set up with. The "Vis Check" function should most likely be ticked off, unless you're sure about what this feature does, and that you want it active!

Make sure the object references at the bottom are NOT empty!



Light Probe Group ✓
ALPG_Generator (Script)
Generate probes

Params

Merge distance	0.75
Merge distance (airProbe)	6
Floor offset	0.25
Height checking distance	6
Collision check radius	0.05
Collision check layers	Mixed...
Cluster count: #64	8

Nav mesh

Light probe nav mesh (Slow)	✓
Force build	☐

Agent Type	LightProbeGen
Nav Tile Size	32
Nav mesh bake on layers	Mixed...

Calculated nav mesh gets cached. Repeat calculations are quick!

Checks

Tag Check	✓
Vis Check	☐
Vis check height	22

(Terrain Y + 'Vis check height')

Vis check layers	Mixed...

Other

Show progress bar	☐
Obj: Light probe group	ALPG_Master
Obj: Probe zone	ALPG_ProbeZone

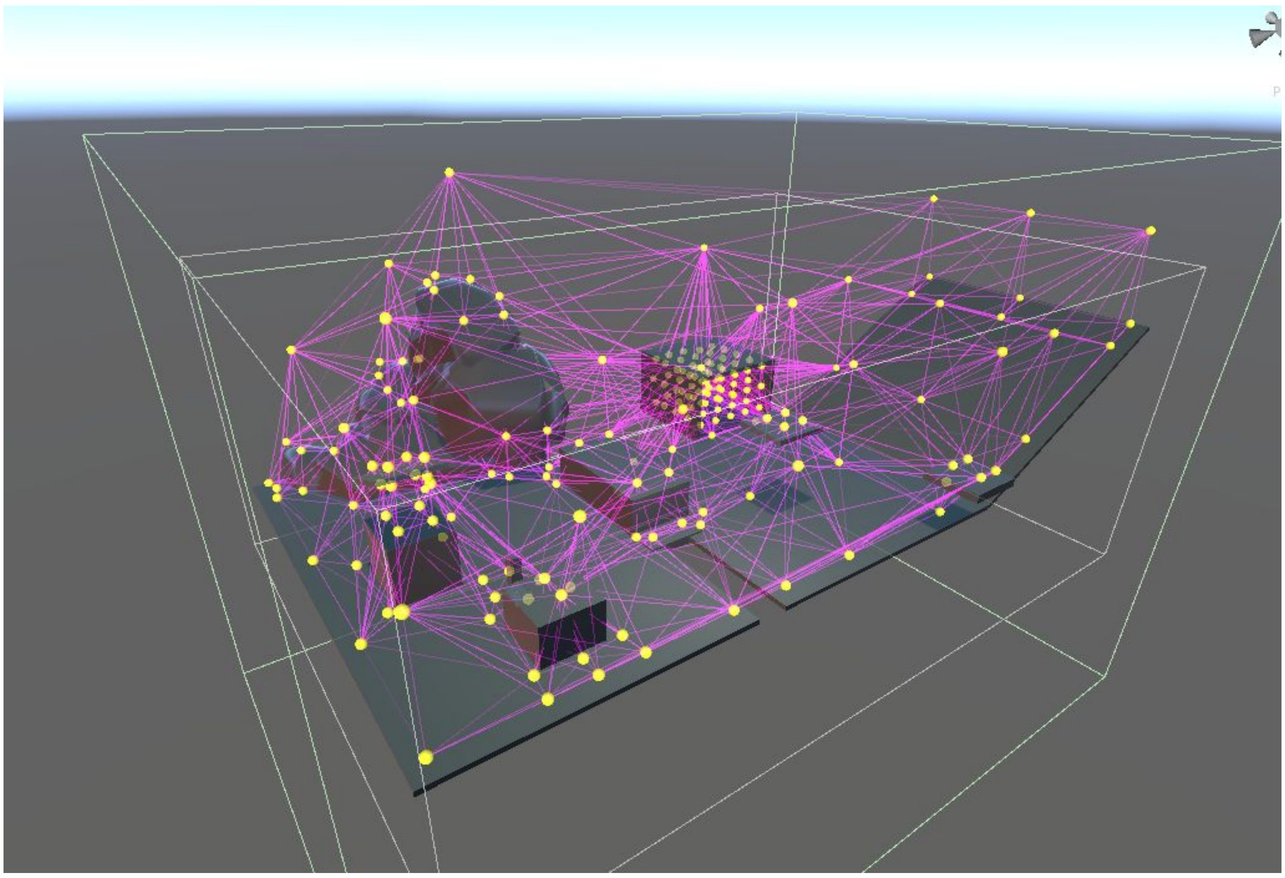
Debug Clusters
Flush nav surfaces

[Step: 4]

That's it! Hit the "Generate probes" button, and let the magic happen!

Note:
Keep in mind that if you're using the "Light probe nav mesh" function, and change something like the object's transform after baking, a force rebuild would be required to properly update the cached nav mesh component!

Click the "Flush nav surfaces" to remove any generated nav mesh components within the object. You will be seeing them when previewing nav mesh data under the nav mesh inspector tab, which might get annoying if you're long done with the probe generation!



Wrap-up

Even though the results might be “too coarse” or “imprecise”, you can supplement with manually created external light probe groups to feed in more specific placements. If anything it can give you a nice base placement to work with.

Again, the tool isn’t meant to replace any and every interaction you have with the light probe placement in Unity. It saves you time for quick iterations and still leaves you with quite decent results.

Hope it helps your development in any case!

Contact me or follow me

Email: jopterineractive@gmail.com
Twitter: https://twitter.com/K0pter

FAQ

[Nav mesh components conflict]
Q: My project already contains the nav mesh components, and I'm getting a conflict! How do I fix these errors?

A: Delete either folder containing these components! Your project can only have one version set of these files without throwing errors!

[Long generation times]
Q: The generation process is really slow! How can I speed this up?

A: A mix of cluster amounts and probe nav mesh tile size (smaller = longer generation times), will be the more considerable factors in mending slow generation times. If your level is on the smaller end, using fewer clusters make more sense- since there is a cost for placing each probe within each cluster.

Other than that, make sure to avoid using the "Show progress bar" setting, as it will slow down the process to redraw the interface! The process can end up taking around 10x as long.

[No probes on scene with a terrain]
Q: When I generate probes on my level with a terrain, I only get just a few probes here and there, or none at all. Most of the probes show up inside buildings or other interiors.

A: The "Vis Check" checkbox will enable a cleanup process that culls the probes "outside of the play space". This means that all probes visible to the edges of your map (with a vertical offset) will be culled unless something blocks this "line of sight" (you can make a separate layer for this tracing if you want). I'd recommend you leave this setting turned off unless you have a rather particular level structure or shape.

[No probes in buildings]
Q: When I attempt to generate probes for interiors, I'm left with empty rooms.

A: The "Tag Check" checkbox will alter the "inside collider" cleanup process. You don't want to have probes inside objects, but in order to detect this type of "collider overlap", mesh colliders need to be convex. In order to not just ignore mesh colliders in this step, the process collects all mesh colliders in your scene, temporarily switches them over to be convex, and then after the process is done- it switches them back again.

This can result in buildings with mesh colliders for encapsulating walls to become big boxy masks that removes all probes within them. To avoid this, simply put the <ALPG_IgnoreCollision> component on these meshes/ prefabs to exclude them as masks for the process!