AUTO PROBE

User Documentation v2.6
6/2/2021

CONTENTS

WELCOME

Thanks for checking out **AutoProbe**! Although there are several different assets on the Asset Store that attempt to place light probes for you, nothing can optimize the final probe set like **AutoProbe**. We're proud to offer this package at such a competitive price point, considering the number of features, quality of output, and heavy time investment in building and supporting the asset for the Unity game developer community.

Be aware that Unity versions 2018.2.0 through 2018.2.10 have a bug (in Unity) that crashes if you click Optimize. All other versions prior or after this, and all other features of AutoProbe function fine.

If you like the asset, *please* do take a moment to write a positive review or give it a ★★★★★ rating. Be sure to recommend it to your friends. Building assets for the Unity developer community takes a lot of time and effort, so be sure to support the ones that make your life easier. Otherwise we won't be able to keep doing it for you.

The URL to leave a review is right here:

https://assetstore.unity.com/packages/slug/105295

FIRST STEPS (BORING, BUT RECOMMENDED)

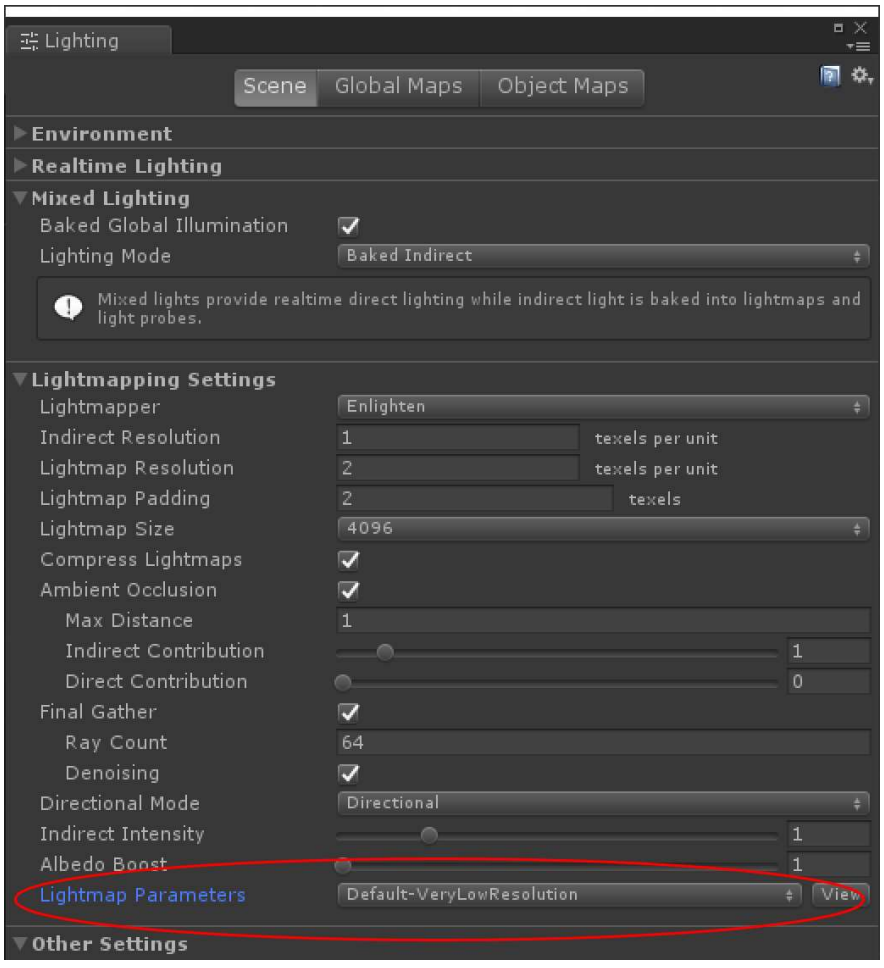
This document is the best place to learn about the features and uses about this asset. Please read it thoroughly, as we want to make sure you get the most out of your investment.

1. Create a ***brand new project***. This may seem unnecessary, and while we go through great lengths to make sure there are no collisions with other assets, it is impossible to provide a guarantee for every combination of settings. Overall, this will give you the best possible experience to first figure out how to use the asset without introducing other variables.

2. Install **AutoProbe** from the Asset Store.

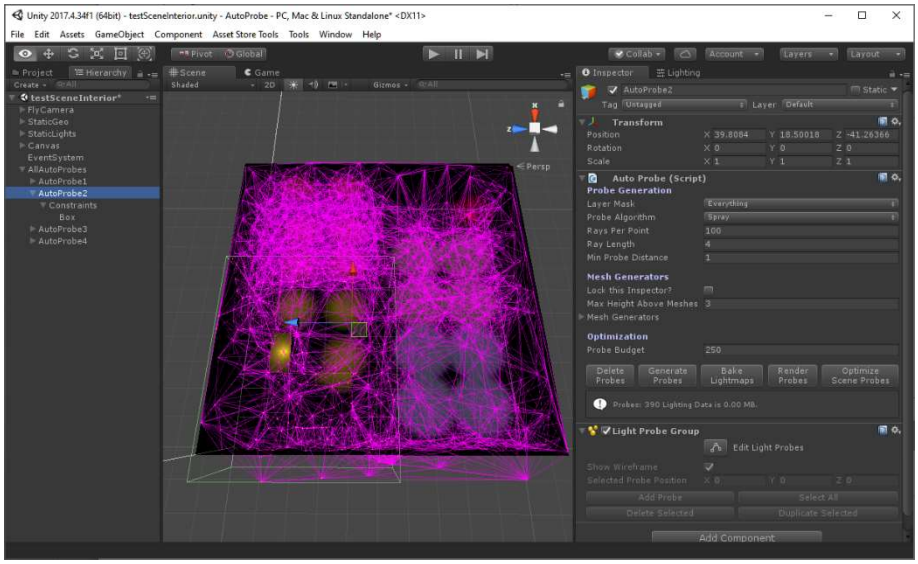
3. Go to the testScenes folder and open the scene called *testSceneInterior*

4. Configure the lighting to be quick to test. Go to Window→Lighting →Settings→LightmapParameters and change the selection to Default-VeryLowResolution. This lets us do a quick lighting pass just to see everything working.

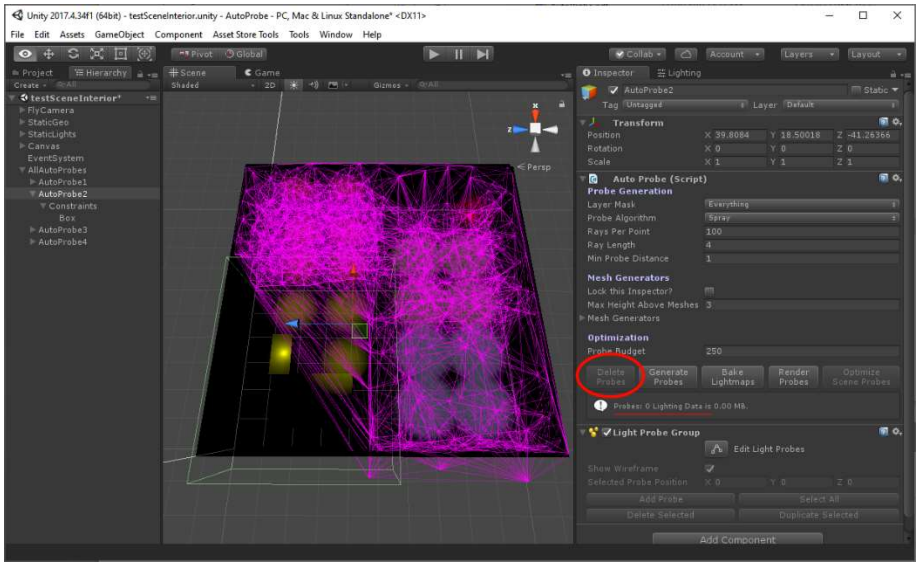


5. Open the "AllAutoProbes" object in the hierarchy view and click on it. There are four objects with *AutoProbe* scripts on them underneath it. **AutoProbe** allows for multi-selection editing, for easy rebuilding of probe sets.

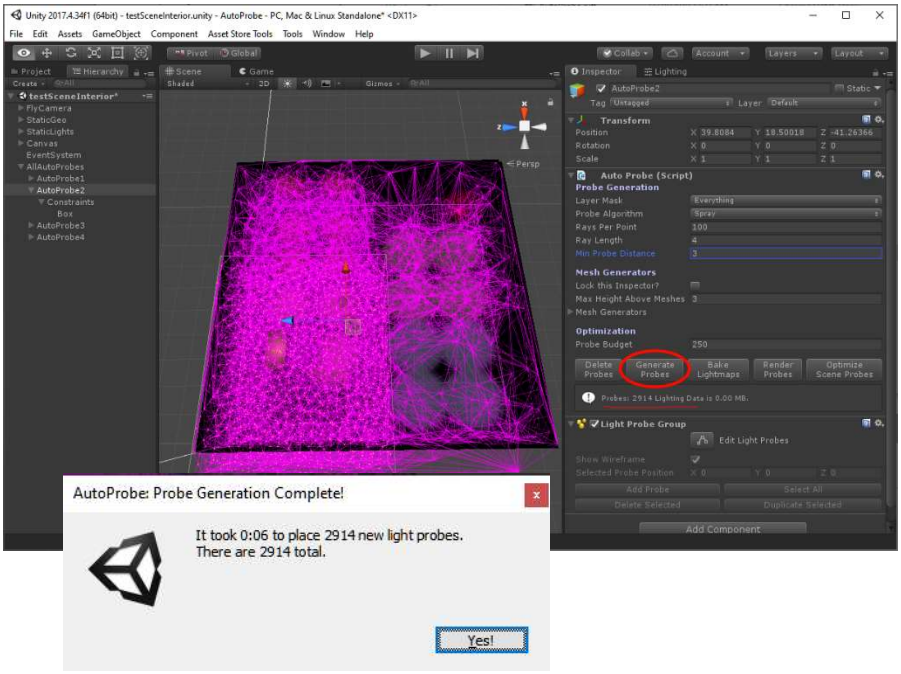
6. Select "AutoProbe2", which has both an *AutoProbe* and a *LightProbeGroup* component. This shows all the light probes in the scene, not just the one you're highlighting, and **if you select more than one, they disappear**. That's just how *LightProbeGroups* work ¯_(ツ)_/¯



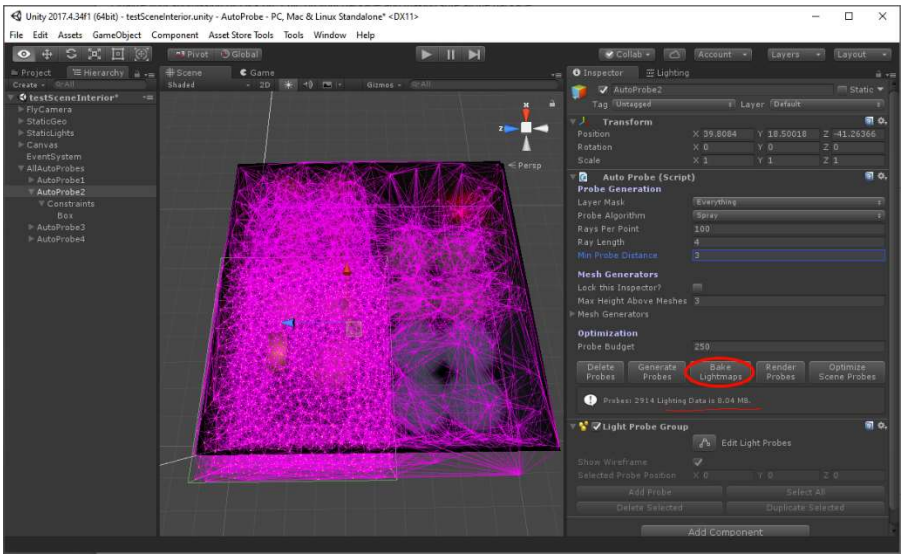
7. In the Inspector, click *Delete Probes*. This will remove all the light probes in the first "room". Your scene should look like this.



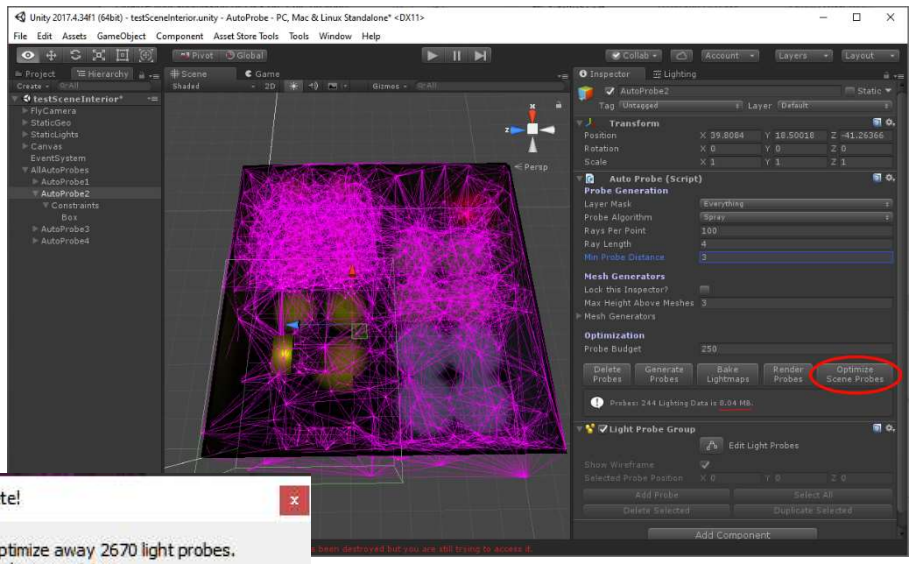
8. Next, click the *Generate Probes* button, which will fill the room with a dense blanket of light probes inside the bounding disabled collider under the Constraint object, stopping at the geometry it finds in the scene. You should get a popup that says probes were placed.

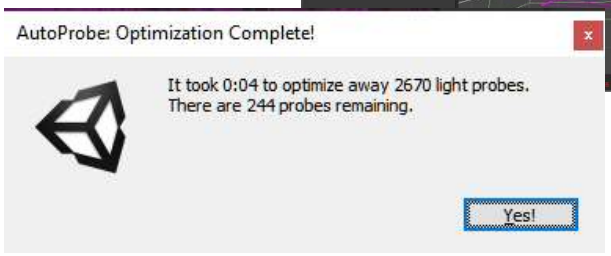


9. Next, click the *Bake Lights* button, which should quickly generate lightmaps for the room, then sample the light probes based on the static lightmaps in the scene. Notice that the helpful display in the Inspector shows **Lighting Data is 8.0MB** for this scene full of light probes. This is the light probe data for the **whole scene**, not just this light probe group. This would eat up your memory when you hit play, so clearly we want to get rid of the unnecessary probes.



10. Now that there is lighting information, AutoProbe can detect the least useful probes and delete them down to the budget you specify. So, click *Optimize* and watch as it makes several passes carefully checking for mathematical redundancy.





11. Notice, *the size of the Lighting Data has not changed!* The light probe set in the scene has been reduced in size, but until you re-bake the light probe data, the runtime doesn't know about it yet. They are two separate things. Light probes tell Unity where to sample, but the information stored in LightingData.asset is actually the runtime data it uses. So, click *Bake Lights* one more time, and it will quickly re-sample just the light probes and store out a new version of LightingData.asset that is smaller, resulting in **5MB memory savings** for this one area!

12. Hit Play and move around. There's a rotating cube attached to the front of the camera so you can look at the lighting from all angles. Different sections of this interior scene have different lighting configurations, to show off what can be baked into probes and how it looks at runtime. There are further instructions given on screen for movement controls and how each light group is configured.

13. You can also get a feeling for how the probes are working by creating a simple white 3D sphere and moving it around the scene. When you select a 3D object, the Scene View will helpfully render the tetrahedron it is picking to interpolate for light probes. Dragging it around will give you a sense of what the lighting will look like. It's also a quick way to spot-check for problems.

There are more details that you may miss if you don't read the rest of the documentation, so please take a moment to read through the description of the workflows provided by **AutoBuilder** in more detail. Although it is relatively straightforward, setup from scratch requires further understanding.

QUICKSTART

Here are the steps you need to go through to set up your own **AutoProbe** object in your project. Take a moment to read over the instructions once so you're familiar with the concepts before trying it. It's easy to configure, but there are some tricks you might miss if you just dive in.

1. Import **AutoProbe** into your project. You can remove the testScenes folder if you like.

2. Open one of your scenes that is setup for lightmaps. Light probes sample data from lightmaps or lights that are set up correctly for baking. You must have done at least one of the following:
    a. Tagged GameObjects as Static.
    b. Checked Mesh Renderers as Lightmap Static.
    c. Set the Light Mode = Mixed or Baked on one or more lights.

3. If you already have a light probe set, select that object. Otherwise, create an empty object and move it to the area of your scene where you want light probes. (Suggestion: If you are going to have several different light probe sets, collecting them under a single parent for organization purposes makes life easier, since you can quickly multi-select **AutoProbe** objects and regenerate probe sets.)

4. Add Component→AutoProbe.

    Notice there is a child object called *Constraints*, and grandchildren called *Box*, *Sphere*, and *Capsule*. You can duplicate or remove them as needed, just make sure the collider is **DISABLED**. If the colliders are enabled, objects in the scene will bump into them, which you certainly don't want. **AutoProbe** uses these as invisible boundaries when exploring spaces, as most geometry is not water-tight.

    New in v2.6: Also notice there is a child object called *StartingPoints*, and a grandchild that is just an empty transform. You should move this to a spot in open air (not inside any colliders). AutoProbe uses this to know where to start generating light probes. You may give it several starting points by duplicating and positioning it. Or, if you remove *StartingPoints* entirely, AutoProbe will return to previous behavior for spawning initial points. This is a more intuitive way to start for most people, and easier to validate the starting position.

5. Move and Scale the transforms or adjust the collider sizes directly until they completely enclose the area where dynamic objects will be able to go.

6. Click the buttons on AutoProbe in this sequence:
    a. Delete Probes
    b. Generate Probes
    c. Bake Lightmaps
    d. Render Probes
    e. Optimize Scene Probes
    f. Render Probes
    (this second bake of the probes rewrites the reduced lighting set for the runtime to use)

Multi-selecting AutoProbe objects works. Feel free to break up levels into different areas that you can iterate on separately. They all get merged together at runtime anyway.

Now you have a single working fully optimized probe set ready for dynamic objects. There are many more features, so read further to get the most out of your investment!

WORKING WITH BAKERY

Please follow the instructions on the Quickstart page, plus understand the following important details:

1. AutoProbe automatically detects that Bakery is installed. If present, the buttons in the AutoProbe inspector simply call Bakery's Lightmapping and Light Probe rendering functions rather than the built-in Unity ones.
2. Be aware that generating light probe data is quite a bit slower in Bakery than with Unity's lightmapper, and depending on the desired result, may require some careful tuning of light settings. Consequently, you might want to start with looser probe sets, to save baking time.
3. The test scene included with AutoProbe has been configured with Distance Shadowmask enabled, Occlusion Probes enabled. With this configuration, where dynamic shadows are desired, a Bakery light component is added with "Indirect and Shadowmask" and a Unity light component added as Realtime only with shadows enabled. There are upper limits for how many shadowmask lights may apply to one area, and this is demonstrated in the Yellow lights. This is a Unity limitation, not Bakery or AutoProbe. Overall, this configuration is as close to the Mixed lightmapper mode as I could get, although there may be other configurations. Feel free to experiment and let me know if there's a better way. I'm not an expert with how Bakery works.
4. If you are used to using Unity's light probe baking, the main difference seems to be that baking Unity "Mixed" mode lights will store direct lighting information into lightmaps, and generate light probe data from this lightmap data alone. Bakery's "Direct" mode is the only one that stores direct lighting information into lightmaps, however when baking light probes suspended between the light and the surface, they are also sampling the light source rather than just the surface, which essentially fills the air with light, in much the way fully "Baked" mode lights work in Unity.

My assumption is that mixed lighting mode is the most useful, where the majority of light complexity is best precalculated and direct lighting + shadows are necessary for dynamic objects only.

USING AUTOPROBE

AutoProbe has been improving over the years for usability. Review the following features and see how they can apply to your project. You will find Tooltips on every element in the Inspector and some deeper explanation of each feature here in the documentation. Contact us if something is not clear; we can help.

GENERATING PROBES

There are two phases of the probe generation algorithm. Initial seeding is how AutoProbe figures out one or more valid starting points, followed by a recursive ray casting algorithm. Understand that no probes are ever generated outside the Constraint colliders, and Mesh Generators are treated similarly to Constraints, where probes cannot be above the Max Height Above Meshes, which gives you extreme control over placement, if you need that.

SEEDING PROBE SETS

Initial seeding of the probe algorithm is simple. Either there are already probes, or we need to generate some. Using existing probes as seeds for the raycasting phase means you can iteratively add or remove probes and AutoProbe will continue wherever you left off.

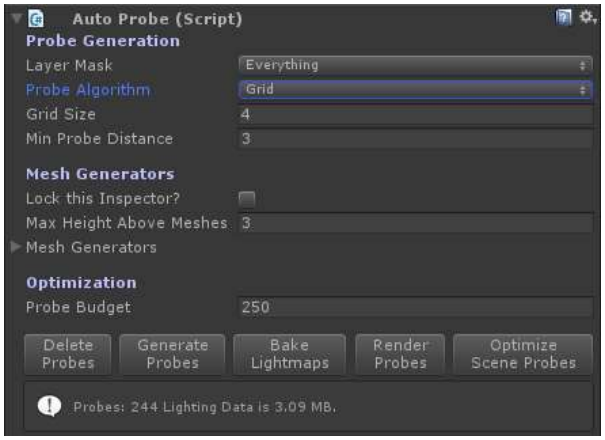
In the event no probes are present, AutoProbe looks for a child object called *StartingPoints* and uses any child transform positions as initial places to grow the light probe set from. If that's missing, AutoProbe will use Mesh Generators to generate probe positions at the floor and ceiling heights specified. If no Mesh Generators are present, the initial light probes are created at the centers of the Constraint colliders, which are hopefully valid and not inside any other collider. The *StartingPoints* method is easiest to control.

All raycasts obey a **Layer Mask** that allows you to restrict what geometry is considered static in your levels. Either tune this to ignore dynamic geometry, or disable all the dynamic objects while generating probes, otherwise you will have odd lighting gaps when objects move.

There are two ray casting algorithms for generating light probes, *Grid* and *Spray*. Grid is fastest and gives very uniform sampling of space. Spray is more organic and is better for capturing complex lighting environments, but may require more probes to capture the whole environment.

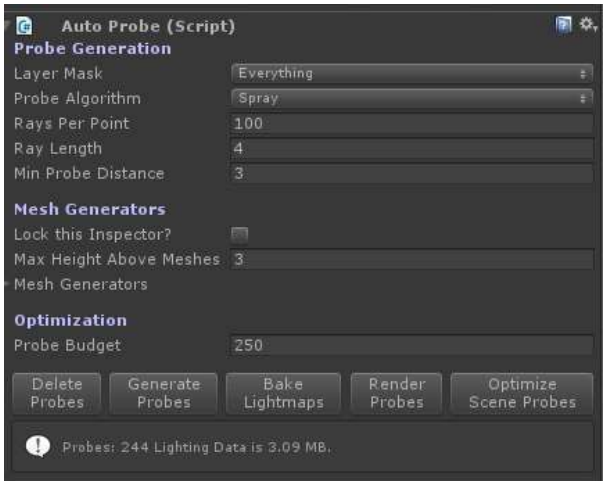
GRID METHOD

In this mode, you can set the spacing for the grid with *Grid Size*. The way this works is by casting rays from each point in all 6 cardinal directions as far as *Grid Size* specifies. If the raycast hits some geometry, it generates a light probe there, but only if it is at least *Min Probe Distance* from any other visible point. This effectively samples the walls and floors of most levels, placing light probes inside the lit areas rather than behind them. The grid is intentionally jittered slightly to prevent mathematical errors in Nvidia's tetrahedralizer. Don't be surprised when you look down the grid and it does not look perfectly straight. This is a feature not a bug.



SPRAY METHOD

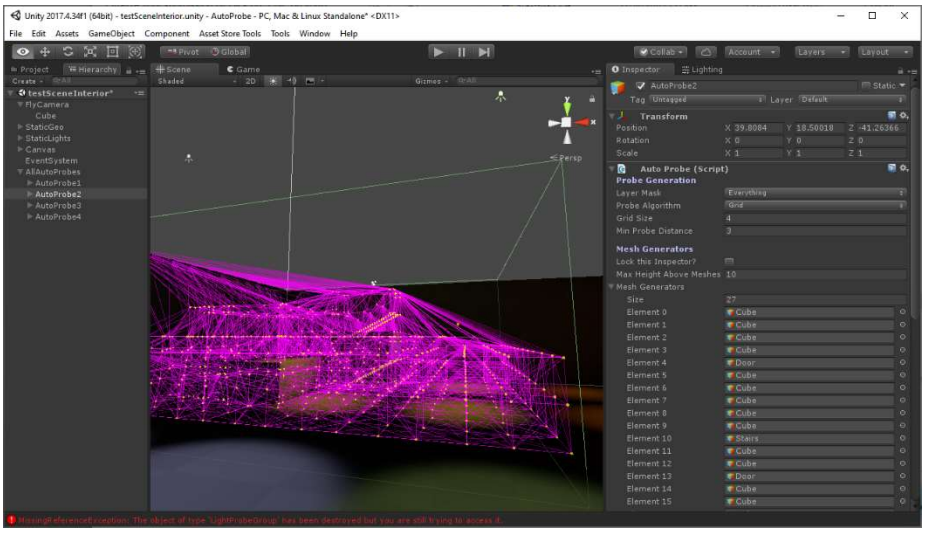
The Spray method is more organic. In this mode, AutoProbe explores space by picking a random direction from each point and raycasting through space *Ray Length* away from the point, stopping if geometry is struck. A light probe is only placed there if no other probe is visible within *Min Probe Distance*.



Be careful about setting *Min Probe Distance* too low. A small decrease in this value can dramatically increase the number of probe positions with the Spray method.

The number of rays cast from each point can be set with *Rays Per Point*. A good starting point for this is 10 rays. Raising to 100 rays will better explore the space, but takes longer to complete.

MESH GENERATORS



AutoProbe also allows for creation of light probes based on meshes. This is intended to be used as a way to both generate and limit light probes to a set of meshes, primarily for walkable spaces. You can add any number of meshes that are "floors", and anywhere the mesh exists within the "Constraint" volumes, points will be generated just slightly above the mesh and at a height specified by *Max Height Above Meshes*. Note, this *Max Height Above Meshes* property acts as a dynamic height restriction, so it follows the floor or terrain, without needing to fill the entire air space above the ground. This makes working on terrain much faster, since hills and mountains often have a lot of empty space that dynamic objects will not move through.

Here is an example of using a Mesh Generator indoors, with the *Max Height Above Meshes* set too low. The light probes did not get placed all the way to the ceiling because the height was limited lower than the bounding box. This is easily fixed, but be aware that it is an additional dynamic constraint.

Also, a helpful *Lock this Inspector?* toggle is there to let you quickly multi-select meshes from the hierarchy and drag them into the array. The way Unity's multi-select works, it would change the Inspector before you could drag, which is inconvenient sometimes.

For reference, Terrain is always sampled in a 100x100 grid. Feel free to change the code if this is too many samples (or too few). Also, you can generate light probes from any Mesh Filters, whether they are used as renderers or colliders. But again, be aware that no light probes will be generated unless they are within *Max Height Above Meshes* of a specified mesh.

MANUALLY PLACED PROBES

AutoProbe is intended to do everything for you, but sometimes it doesn't do exactly what you want. You can absolutely create a Light Probe Set and not attach an *AutoProbe* script to it, and it will continue to behave the way all light probes do. To make this even easier, there is a script called **ForceLightProbeHere** that you can attach to objects (such as point lights) that do not have collision but you absolutely want a light probe to be precisely located on it. Just add this script and it will make a light probe set in a self-documenting sort of way, which **AutoProbe** will not remove or optimize away.

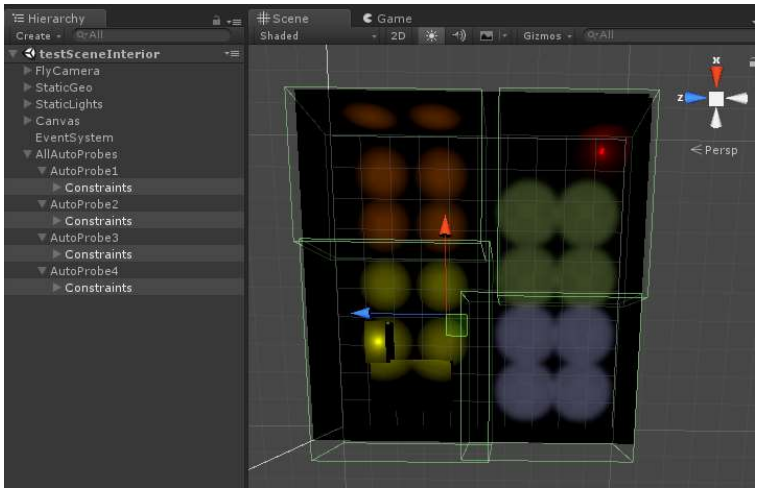
ADDITIVE WORKFLOW

Additionally, you can always take existing light probes that have been meticulously placed and add an AutoProbe script to that object, generate more probes *using your initial probes as a starting point* and it will do so happily. Or re-run the generator script on its own output, if you like, to add more points that the first pass might have missed (which can happen if you are using the Spray method with a low ray density).

Experimenting with generation or optimization is very easy, because **AutoProbe** supports Undo/Redo. Every action can be undone. Be careful, though, long optimization passes can be lost by a careless undo.

INTERIOR SPACES WORKFLOW

Many games will be mainly comprised of interior spaces. Building light probe sets with **AutoProbe** requires setting up bounding volumes that enclose each room, hallway, or alcove so as not to unnecessarily sample outside level geometry. It's ok for these bounding volumes to overlap. They don't even have to connect. Light probes will automatically fill all volumes and stop at collision geometry.

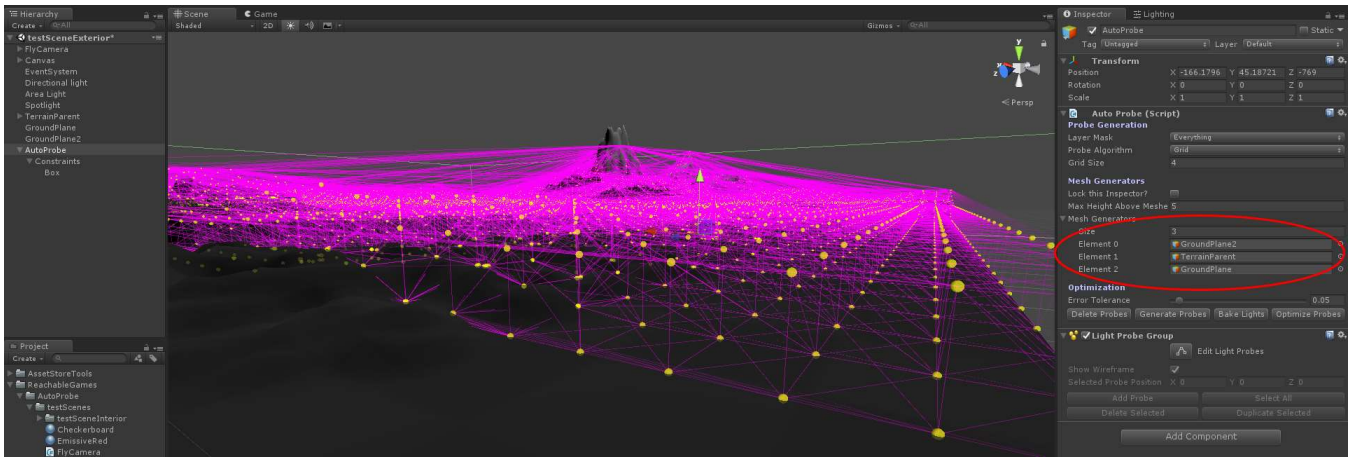


Notice in the testSceneInterior example, there are four distinct areas that are separately managed. You might find it beneficial to split large rooms if there is a lot of space to light, and AutoProbe is taking too long to calculate.

Most people find Grid does a very good job of capturing the geometry in their levels. However, a slightly higher quality sampling can be had by first generating a Grid, then following with a Spray sampling, where *Ray Length* is set to the same value as *Grid Spacing*, and a slightly smaller value for *Min Collision Spacing*, just to catch all the crevices that Grid misses.

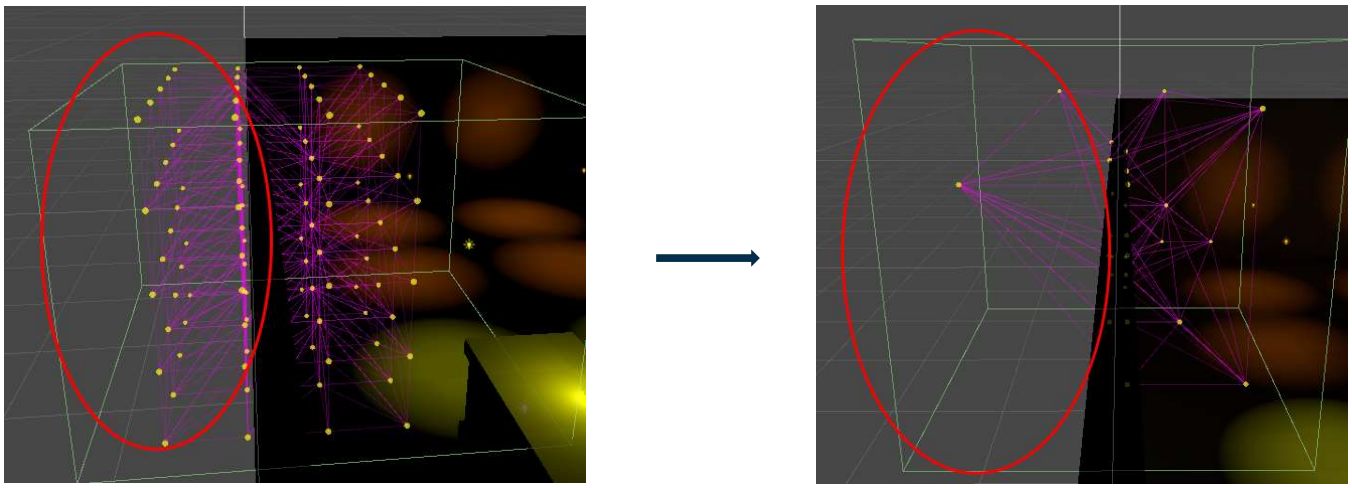
EXTERIOR SPACES WORKFLOW

For outdoor games, you will want to drag your Terrain objects into the *Mesh Generators* array. This feature will allow your characters to roam around the world with minimal unnecessary light probes. Note, there may be need for additional **AutoProbe** objects in places that players will go that are not on the terrain, such as in tall buildings or caves. You can add more than one and keep their probe sets separate. Generation and optimization are scoped to the Light Probe Set assigned to the AutoProbe object, and although they render as a connected mesh (and indeed, the runtime sees them that way), in the Editor you can treat them in isolation.



SAMPLING OUTSIDE LEVEL GEOMETRY

Be aware that you will get some samples behind walls, it's almost unavoidable, because geometry is not normally water tight. **Do not worry about it.** As long as you have light probes sampling the lit-sides of the walls, objects will light correctly.



FLICKERING LIGHTING ON OBJECTS

If you notice the lighting is flickering on something, it's usually one of two issues. Click on the AutoProbe object in that area of the level and look at the pink tetrahedrons.

If the triangles are long and thin, the values may be changing significantly from one end of the tetrahedron to the other, and neighboring tetrahedrons have different lighting data in them. Crossing between two thin tetrahedra might be a very short distance and cause some rapid lighting changes. In this case, the probe set may have been "over-optimized" and have too few points covering too large a space unevenly. Try increasing the *Probe Budget* slightly, then regenerate the probe set (click Delete, Generate, Bake, Optimize, Bake).

Alternatively, if there are plenty of probes in the area, you may have "over-sampled". If your objects are larger than the lighting tetrahedrons, perhaps because there are complex light situations that require many probes to retain the data correctly, try increasing the *Min Probe Distance* so that it is larger than the diameter of your dynamic objects. This will prevent there being so many probes so close together.

It's possible you will want to consider using a Light Probe Proxy Volume, which lets a single object be lit in multiple places by multiple light probes. That is beyond the scope of this document, as it has nothing to do with **AutoProbe**. But it's a helpful thing to know exists.

MERGING LIGHT PROBE SETS ACROSS SCENES

Older Unity runtimes (prior to 2019.3) did not support any kind of dynamic merging of light probe groups. The newer versions do, but you have to either use **LoadSceneMode.Single** or manually call **LightProbes.Tetrahedralize()** or **LightProbes.TetrahedralizeAsync()** after additive loading your scene. Alternatively, you can generate your light probes in a single scene with all the static geometry and lighting loaded and avoid the changes to your runtime loading sequence. For reference, you can additively load scenes and their lightmaps without any special handling.

In older versions of Unity, any scenes with light probes would clobber the previously loaded set. In current versions, additively loaded scenes do not change the light probes until you call the Tetrahedralize function. Keep that in mind when debugging multi-scene loading issues with lighting.

You might also find one of our other assets, **SplitScenes**, very handy as it simplifies and automates additive scene handling such that it works much closer to Photoshop layers with a handy GUI window for loading/unloading subscenes.

SUPPORT

Please read all the documentation. We put a lot of effort into it and hope that it exceeds your expectations. In the event you have further questions, please check the following web pages for more details about this asset:

Our Website: https://reachablegames.com/unity-assets/

Unity Forum: https://forum.unity.com/threads/520428/

If you find that none of the above can answer your question, you may contact us at support@reachablegames.com, but know that we always handle support requests first that include:

1. **Invoice Number**
2. **Unity Version**
3. **Asset Version**
4. Links to screenshots or small sample projects on DropBox or similar sharing site describing the problem.
5. Kindness. If you are mean, rude, harassing, or hateful, do not expect a response.

As a matter of common sense, we do not offer support to free customers except as time permits.

Finally, if you are looking for a feature that is not currently supported, understand that we are a business and get many such requests. If you need a custom feature that is important enough to your product to pay for its development, contact us about it.

ABOUT US

Reachable Games is located in the beautiful hill country of Austin, Texas USA. It was founded by Jason Hughes, who has been a professional game developer since 1995, at Origin Systems. He has worked on many AAA games and with many recognizable companies. As a generalist, Hughes has worked on nearly every kind of game platform in every capacity—from graphics tools to AI to UI/UX to game design to shader writing to database management to networking and server development. It turns out this is the right skillset to help improve the Unity development experience for other developers.

We currently have several other assets on the Asset Store. For what it's worth, they are all built because we are working on projects of our own and both need and use them on a daily basis. If you think this one is great, chances are the others will help you out too.

CHANGE HISTORY

V2.6 JUNE 2, 2021

- Changed from raycast to spherecast to make certain all colliders are considered when determining a light probe point is "too close" to a collider to use
- Fixed a bug in the collider cache that would throw errors if you deleted objects and rebuilt light probes

V2.5 JANUARY 1, 2021

- Added exhaustive detection algorithm for making certain no light probes end up inside colliders.
- Revised the error handling for UnityWebRequest now that the interface changed in 2020.2.

V2.4 DECEMBER 12, 2020

- Simplified version checking popup.
- Verified AutoProbe integrates with current version of Bakery properly.

V2.3 JUNE 3, 2020

- Found and fixed a complete Editor crash related to Unity 2019.3 removing a function in the WebView. No functionality change, but updating is strongly recommended.

V2.2 APRIL 30, 2020

- Cleaned up Bakery integration so that the user doesn't have to do anything for it to just work. If you want to use Unity lighting while Bakery is installed, the easiest way to accomplish that is to remove Bakery. AutoProbe's code keys off BAKERY_INCLUDED in the scripting definitions now. If you need a hybrid of Bakery + Unity lightmapping (a strange requirement), just change the AutoProbeGUI.cs to expose the buttons you want rather than using BAKERY_INCLUDED. The code is simple enough to follow.

V2.0 DECEMBER 3, 2019

- Added direct control over optimization with Probe Budget rather than error tolerance.
- Dramatic improvement in optimization performance.
- Fixed spray method to not put probes inside geometry as often, and massive speed improvement to both generation algorithms.
- Added CIE76 color differencing to improve the choices of the optimizer to Lab perceptual color space rather than RGB.
- Fixed long-standing floating point error problem with tetrahedralizer when using exact grid placement causing long thin tetras through slightly jittering of grid coordinates. Added 'deringing' as a default for all LightProbeGroup objects in 2018+.
- Fixed issue where constraints were checked before rays were cast, so sometimes probes would not reach floors or ceilings when tight bounds were applied.

V1.9 NOVEMBER 18, 2019

- Added better data feedback to Inspector that shows light probes selected, size of the current LightingData asset (light probe data size).

- Added enhanced interoperation with Bakery, which user can enable if Bakery is installed.
- Added an example scene using Bakery which includes configurations of various light sources and Bakery that shows how to get lightmapping and realtime shadows together.
- Improved documentation, store images, etc.

V1.8 SEPTEMBER 14, 2019

- Improved the optimization algorithm slightly. Sometimes 10% faster, other times less.
- Fixed a minor floating point epsilon issue where Grid method would not always place points it should have.
- Added an explicit Layer Mask control so individual **AutoProbe** instances can manage what geometry they consider for collision.
- Added better coloring for the Inspector in Pro/Plus skin mode.
- Added update window to show when new product updates are released.
- Fixed an unusual crash and added try/catch code to make sure they are well-behaved crashes instead of forcing you to close Unity.
- Added this sparkly new documentation for everyone to get a better idea how things work.

V1.7

- Added multi-selection for dealing with several **AutoProbe** nodes at once. All buttons work, all properties can be bulk edited.
- Added nice popups at the end of Generate and Optimize that give you stats on what was accomplished.
- Fixed Undo/Redo, again.
- Added collider constraints when using mesh generators to create light probes. Now you can control exactly where they are created in every mode.
- Updated interior and exterior test to show new features.

V1.6

- Fixed the issue with Light Probe Group components not correctly refreshing the purple tetrahedrons. That was really annoying!
- Added mesh spawning and height limits to specified colliders.
- Nicer looking inspector and a couple of helpful features to support mesh selection.
- New test scenes, new tutorial videos.
- Added ForceLightProbeHere.cs component, which simply allows you to stick probes on an object that AutoProbe will not remove.
- Verified asset works fine with 2018.2.11 and beyond.

V1.5

- Improved Inspector somewhat to use less memory and be faster, give better feedback on the buttons and tooltips, etc.
- Verified asset works fine with 2018.1.x, but will not work in 2018.2.x yet.

V1.4

- Massive 4x speedup in optimizer.
- Generally faster in generator, but the algorithm scales very well for large data sets.

- Fixed undo. It was only working for delete, and very slowly. Now works for all operations again.
- Slight improvement to custom inspector to give user feedback when their colliders are setup wrong.

V1.3

- **AutoProbe** generates constraint volumes automatically now.
- Added a walkthrough video showing how it works.

V1.2

- Added disabled colliders as spatial constraints for ray marching.
- Fixed issues with the custom inspector. It was throwing errors that came up in 2017.3.

V1.1

- Numerical stability fixes with tetrahedron detection
- Massive performance improvement that I promised in 1.0 has landed. New algorithm went from N^3 to NlogN, or thereabouts
- Fixed a couple of includes in the code that prevented builds from working

V1.0

- Initial Release