Fast Shadow Receiver (Version 1.5.9)

Website

Nyahoon Games : http://nyahoon.com/
Document : http://nyahoon.com/products/fast-shadow-receiver
Document (Japanese): http://nyahoon.com/products/fast-shadow-receiver-ja
Demo : http://nyahoon.com/demos/fast-shadow-receiver
FAQ : http://nyahoon.com/products/fast-shadow-receiver/faq

Release Note

Version 1.5.9

[Bug fix]

- Fix an issue whereby Offset parameter of projector material was not editable in Unity 2020 or later.

Version 1.5.8

[Bug fix]

- Fix an issue whereby a temporarily instantiated prefab object was not destroyed after mesh tree build.

Version 1.5.7

[Bug fixes]

- Fix view clipping issue of Projector Manager.
- Support scaled Projector For LWRP

Version 1.5.6

[Bug fix]

- Fix shaders to support VR single pass rendering and GPU instancing.

Version 1.5.5

[Bug fixes]

- Migrate to Unity Job system from ThreadPool (Unity 2019 or newer only).
- Migrate to [ExecuteAlways] attribute from [ExecuteInEditMode] (Unity 2018.3 or newer).
- Support Terrain Holes (Unity 2019.3 or newer)

Version 1.5.4

[Bug fixes]

- Remove GC Alloc from frame rendering loop that occurred when using MeshShadowReceiver with lightmaps.
- Add “Offset Slope Factor” property to all projector shaders. Now both of the depth offset parameters (Factor and Units) can be zero (default values are both -1).

Version 1.5.3

[Bug fixes]

- Mesh trees stop ignoring IgnoreProjector materials by default because URP shaders have IgnoreProjector tag.
- Use local shader keywords instead of global ones.

Version 1.5.2

[Bug fixes]

- Fixed CBUFFERs so that shaders are compatible with SRP batcher.
- Shaders which use lightmaps are now available in Universal RP.

Version 1.5.1

[Bug fix]

- Fixed duplicated CBUFFER names which can cause "m_BuffersToBind[shaderType][bind].buffer == NULL" errors.

Version 1.5.0

[New features]

- Unity 2017 or later required.
- Lightmaps and Shadowmasks are available in Projector materials (example: Blob Shadow Projector with mixed lighting).
- Projector for LWRP support (https://nyahoon.com/products/projector-for-lwrp).

Version 1.4.6

[Bug fix]

- Fixed a bug that caused IndexOutOfRangeException when using TerrainMeshTree.
- Unity 4.7 support.

Version 1.4.5

[Bug fix]

- Supports a case where Mesh Shadow Receiver has a scaled parent.
- Improved visibility test of AutoProjector.

Version 1.4.4

[Bug fix]

- Fix Unity 5.3 warnings.
- Windows Store App (.NET Core) support.
- Fix an issue that shadows were incorrectly clipped out by ProjectorManager.

Version 1.4.3

[Bug fix]

- Fix the problem that "Cleanup leaked objects…" is shown when a scene is saved.
- Remove FogMacro.cginc file from the package for Unity 5.

Version 1.4.2

[Bug fix]

- Update demo scenes for Unity 5
- Optimize mesh tree search.
- Improve Dynamic Shadow Projector support.

Version 1.4.1

[New features]

- Shadow Receiver and Projector Manager now call "UpdateTransform()" method before using transform of a projector, if any components of the projector object have such public method.

[Bug fix]

- Unity 5 fog support
- Fixed a bug of Easy Setup Wizard whereby a MeshShadowReceiver was created with null material.
- Fixed a null pointer exception which could happen when building a mesh tree.

Version 1.4.0

[New features]

- Easy Setup Wizard for ProjectorManager.
- Create Mesh Tree Wizard.
- Added some error check in Inspector View.
- No longer support Unity older than 4.6.2.

[Bug fix]
- Fixed some Unity 5 compatibility issues with Demo.
- There was a little chance that raycast against BinaryMeshTree could return non-closest point.
- It is now possible to setup MeshShadowRenderer at runtime. It is not required to set some MeshTree and Mesh Transform in Editor.
- Fixed some issues which might have happened when trying to build MeshTree at runtime.

Version 1.3.0

[New features]
- Thread-safe raycast against a MeshTree (See "Multithreaded Raycast" demo).

[Bug fix]
- Fixed some Unity 5 compatibility issues.

Version 1.2.2

[Bug fix]
- Fixed the issue that MeshShadowReceiver did not work correctly if the mesh model was scaled.

Version 1.2.1

[Bug fix]
- Fixed the issue that near clip and far clip of a Light Projector was affected by the directional light position.

Version 1.2.0

[New features]
- Light Projection support (Light Projector + Fake Shadows)
- Bullet Marks demo
- Fake Projector (Project Shadows/Lights/Bullet Marks without Projector)
- Draw Gizmos when shadow receiver is selected.

[Bug fixes]
- Fixed incorrect back face culling
- Fixed broken layout in this document
- Shader optimization

Version 1.1.0

[New features]
- Normal vectors are now available in shadow shaders.
- Added back face culling option to MeshShadowReceiver.
- New projector shadow shaders which can avoid backside projection problem. ("FastShadowReceiver/Projector/Multiply Diffuse" and "FastShadowReceiver/Projector/Multiply Diffuse Without Falloff")

Overview

Shadows are very important aspects in 3D space. Shadows enhance spatial awareness of objects in 3D space and give better user experience for players. However, shadow rendering is GPU intensive process. For low-end mobile devices, it will slow down your application. “Fast Shadow Receiver” will make shadow rendering much faster by minimizing the area where shadows are drawn.

“Fast Shadow Receiver” provides a useful mesh search tree called “Mesh Tree” which is used to search for polygons which are receiving shadows. Mesh Tree is useful not only for shadows but also for various purpose such as light projection, bullet marks, AI, and so on.

Features

- Unity 2017 or higher is required.
- Lightmaps and Shadowmasks are available in Projector materials (example: Blob Shadow Projector with mixed lighting) (Version 1.5.0)
- It works with Lightweight Render Pipeline (LWRP) (*)
- It works with Blob Shadow Projector or shadow-map.
- It can cast shadows on non-flat surface.
- Terrain object is also supported (Only Blob Shadow Projector is available)
- Really fast, even on low-end mobile devices!
- Accompanied with fast (but no falloff) projector shadow shader.
- It can handle multiple projectors or shadow casters.
- Optimized for multi-core processors.
- Light Projector / Bullet Marks are also supported.

(*) LWRP does not support Projector. However, Fast Shadow Receiver objects can receive Projector shadows. For more details, see https://nyahoon.com/products/fast-shadow-receiver/use-projectors-in-lwrp-projects

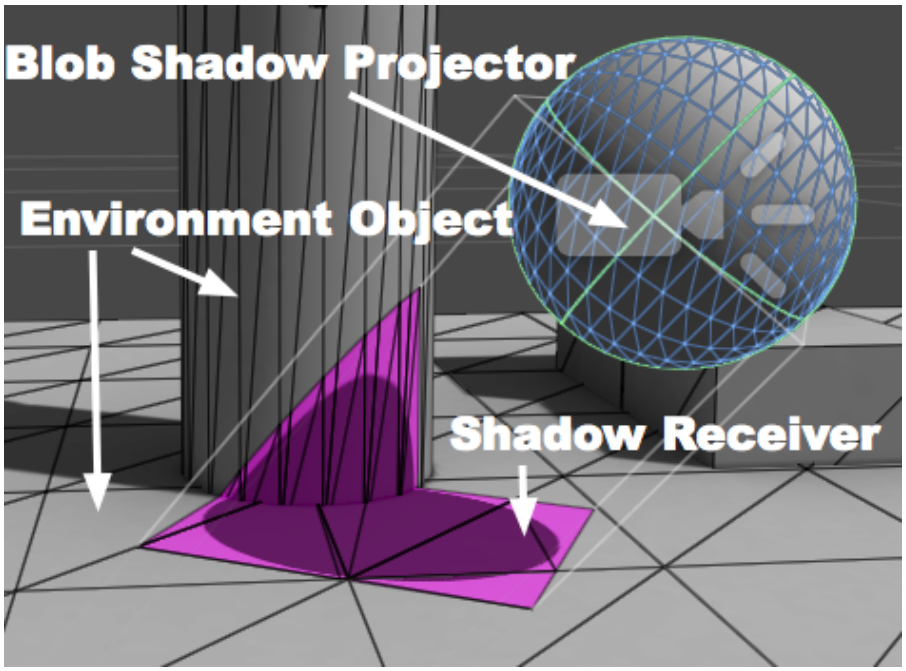
How it works

Usually in a scene that has a large environment object like a terrain, shadow rendering will causes a performance problem. The large environment object will occupy most of pixels on the screen, rendering shadows on it will use a lot of GPU resource.

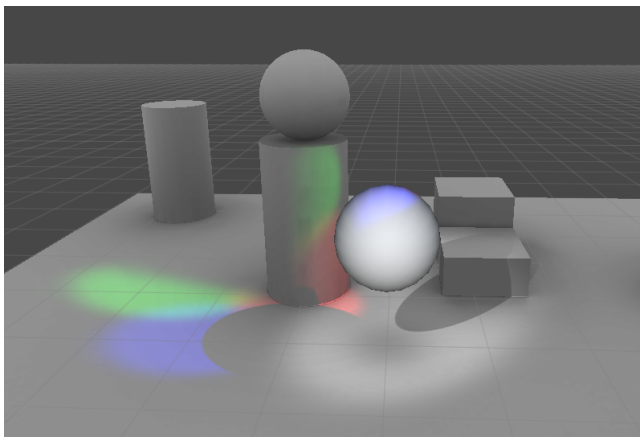
So, it is better for you to remove shadows from the large object, replace each shadow with a pair of “shadow receiver” and “blob shadow projector”. “Shadow receiver” will generate a minimum mesh to receive a shadow based on associated projector and large environment object (See the image below).

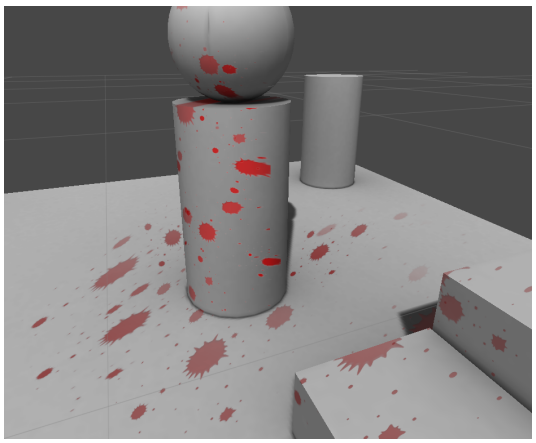
“Fast Shadow Receiver” is designed to work with Unity built-in Projector.

Additionally, “Fast Shadow Receiver” can also work with shadow-map. However, the shadow-map is not used with lighting. It is just alpha-blended with the scene, which makes the shadows not realistic.



“Fast Shadow Receiver” is an asset for optimizing Projector performance. It is applicable not only to Blob Shadow Projector but also to Light Projector and Bullet Marks.



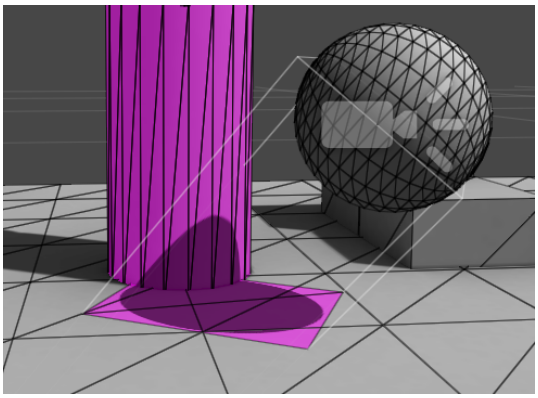


Types of Shadow Receiver

Fast Shadow Receiver provides three types of shadow receiver, InfinitePlaneShadowReceiver, MeshShadowReceiver and RaycastPlaneShadowReceiver. If you are using lightmaps and want to use mixed lighting blob shadows, please use MeshShadowReceiver.

InfinitePlaneShadowReceiver

If your scene consists of one big flat plane and small objects (I mean, not so big, for example: buildings, trees and so on), you can use InfinitePlaneShadowReceiver. You can still cast shadows on the small objects, because rendering shadows on small objects will not take so much time. The following image shows an example of InfinitePlaneShadowReceiver. In this case, the plane floor object is the large environment object, and InfinitePlaneShadowReceiver creates a rectangle mesh on the floor. The cylinder object is not counted as the large environment object and it is also receiving the shadow from the projector.



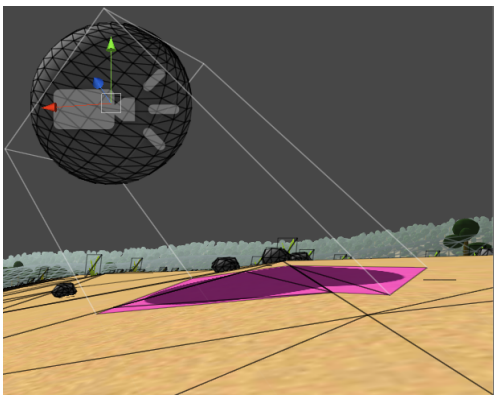
MeshShadowReceiver or RaycastPlaneShadowReceiver

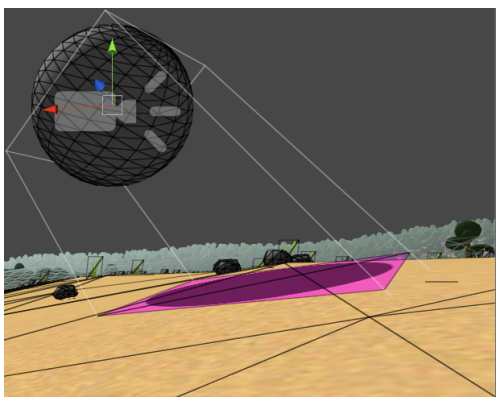
If the large environment object in your scene has non flat surface, these two receivers are suitable to be used.

MeshShadowReceiver will pick up polygons from the mesh of the large environment object. It can cover not only a single MeshRenderer object but also multiple MeshRenderer objects and Unity Terrain object. This method will take some CPU time.

On the other hand, RaycastPlaneShadowReceiver will generate a quadrangle mesh at the place where a raycast hits, which is much faster than MeshShadowReceiver method. However, the quadrangle mesh will not fit the actual environment surface if it has large curvature.

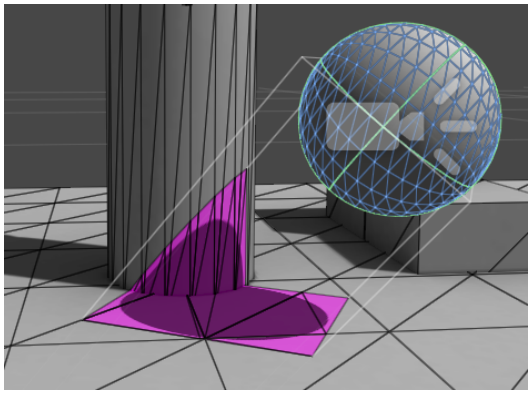
Images below are examples of both methods. In these examples, the environment object is a terrain. The left image is using MeshShadowReceiver, and the right image is using RaycastPlane-ShadowReceiver. As you can see, MeshShadowReceiver can make a mesh which perfectly fits the terrain surface. On the other hand, the mesh created by RaycastPlaneShadow-Receiver does not fit the surface, since it is just a rectangle polygon. If the surface was concave, the rectangle polygon would intersect with the terrain surface.





MeshShadowReceiver vs InfinitePlaneShadowReceiver

The following image shows an example of MeshShadowReceiver compared to InfinitePlaneShadowReceiver in the pervious example. The cylinder object is counted as the large environment object, and the mesh created by the MeshShadowReceiver contains a part of the cylinder surface. This is just an example to show how MeshShadowReceiver works. In this situation, using InfinitePlaneShadowReceiver is recommended.



Getting Started

Fast Shadow Receiver is a bit complicate because it works with an environment object and shadow projectors. You need to set up the relationships among them. Here is the outline of what you need to do:

1. Choose a large environment object(s).
2. Remove shadows from the large environment object(s).
3. Create a shadow receiver object and give it some information of the environment object(s).
4. Assign a projector reference to the shadow receiver object.

If you already have blob shadow projectors in your scene, Easy Setup Wizard is available (from Version 1.4.0). You just need to follow the instructions in the wizard. Please refer to "**Easy Setup Wizard for Projector Manager**" section for details.

If you are using shadow-map, or you don't want to use the Wizard for some reasons, you can do the setup manually in Inspector View. Please refer to the following sections for details. Even if you are going to use Easy Setup Wizard, the following sections will be helpful to understand the details.

- If you have only a few projectors, see "**Steps to use Shadow Receiver with Projector**".
- If you want to use shadow-map, see "**Steps to use Shadow Receiver with Shadow-map**".
- If you want to have many projectors or to spawn projectors at runtime, see "**Steps to use Shadow Receiver with Massive Projectors**".

For light projection, the steps are almost similar to the steps for shadows. The difference is only a shader assigned to a Projector material. You can use "FastShadowReceiver/Projector/Light x Shadow" shader. Additionally, you can add "Fake shadows". Add "Light Projector Shadow Receiver" component to Light Projector objects, and add "Light Projector Shadow Caster" component to shadow caster objects. For more details, please have a look at "SpotLights + Fake Shadows" demo scene.

For bullet marks projection, you can also follow the steps for shadows. Just change the Projector shaders. However, you might not want to use lots of Projectors for bullet marks. BulletMarkReceiver script in “BulletMarks” demo scene is an example for generating bullet marks from a single Custom Projector. Please have a look at the demo scene, and customize the script to make it fit your application.

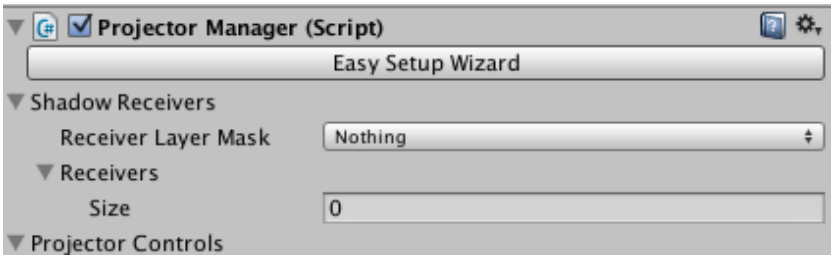
If you want to use a Mesh Tree for multithreaded raycast, please have a look at "Multithreaded Raycast" demo. Also, see “**Create a Mesh Tree**” section below.

Easy Setup Wizard for Projector Manager (from Version 1.4.0)

If you already have blob shadow projectors in your scene, Easy Setup Wizard makes it easy to setup everything. If you want to use shadow-map, you cannot use this wizard.

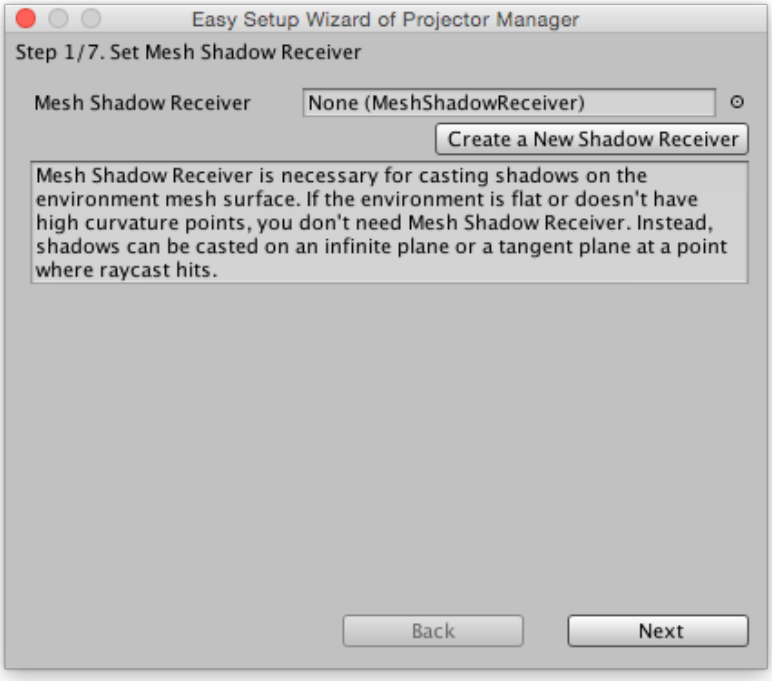
To open the wizard, you need to create a Projector Manager object.

1. Create an empty object, and rename the object as you like.
2. Press “Add Component” button in Inspector View of the object, and select
   Scripts > FastShadowReceiver > Projector Manager.
3. After step 2 is done, you can find "Easy Setup Wizard" button in Inspector View. Press the button to open Easy Setup Wizard.



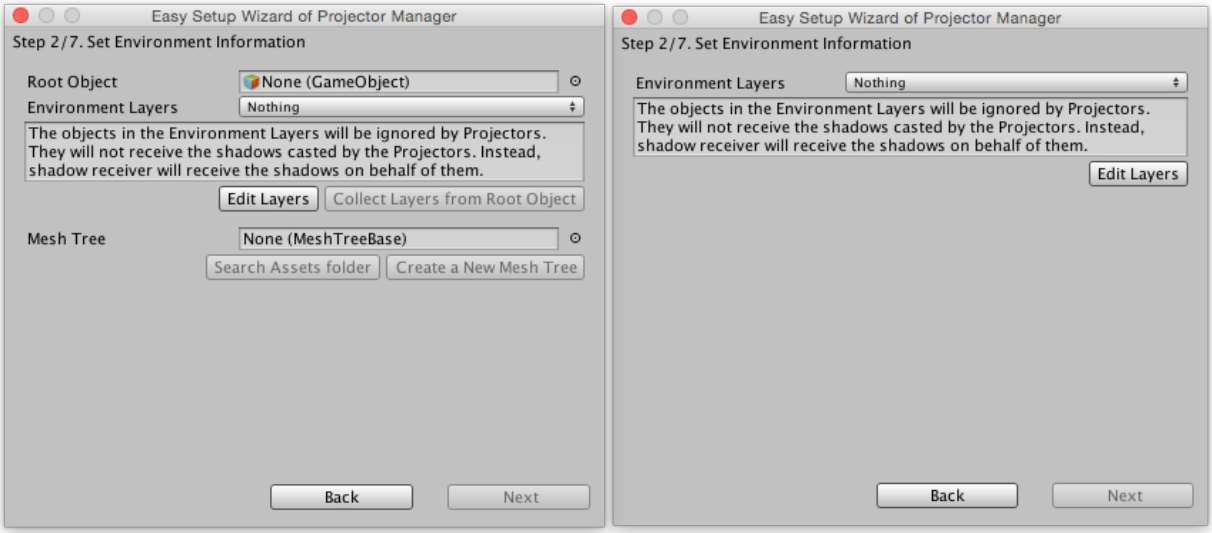
Easy Setup Wizard consists of seven pages, and some of them will be skipped depending on what you need.
This is the first page you can see just after the wizard open.

Page 1/7. Set Mesh Shadow Receiver



In this page, you just need to set a Mesh Shadow Receiver. You can create one by clicking "Create a New Shadow Receiver". If you don’t need to cast shadows on the meshes of the environment objects, you can skip this page.

Page 2/7. Set Environment Information

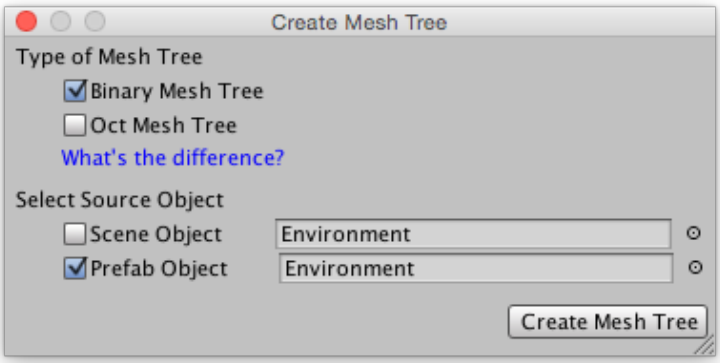


If you set a Mesh Shadow Receiver in page 1, you will see a page like the left image above, if not, a page like the right image above will appear.

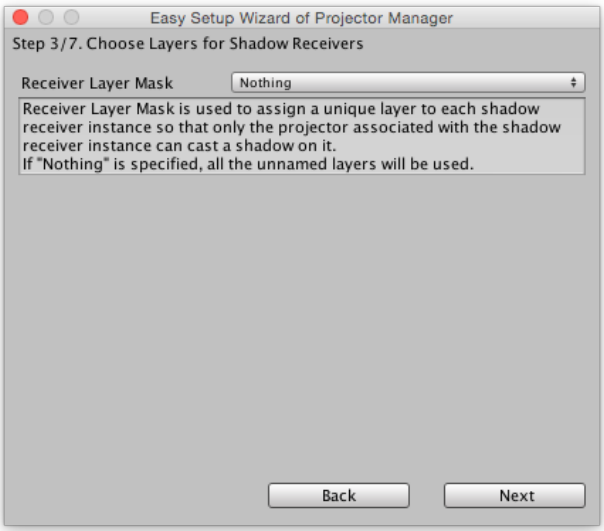
First of all, you need to choose a layer (or some layers if you want) for the environment objects so that blob shadow projectors can ignore the environment objects.

If you set a Mesh Shadow Receiver in page 1, you also need to set the root object of the environment. Once you set the root object, "Search Assets folder" button and "Create a New Mesh Tree" button will be enabled.

If you already have a Mesh Tree object created from the root object, press "Search Assets folder" button or manually drag the Mesh Tree object to the "Mesh Tree" field in the wizard window. If you don't have a Mesh Tree yet, or want to create a new one, press "Create a New Mesh Tree" button. It will open another wizard as shown below. Please refer to "**Create a Mesh Tree**" section for details of Mesh Tree.

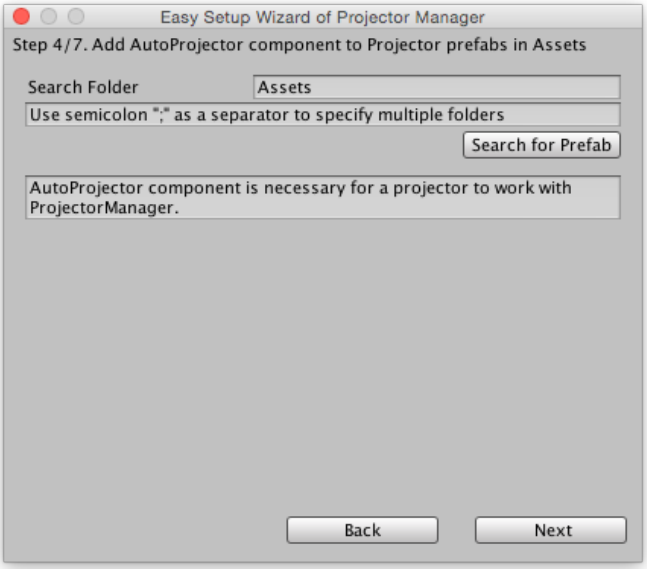


Page 3/7. Set Environment Information



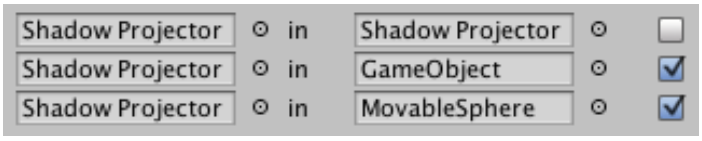
If Mesh Shadow Receiver was set in page 1, you will see this page. In this page, you can set Receiver Layer Mask which is used to assign a unique layer to each shadow receiver instance. You can leave it "Nothing" to use all the unnamed layers.

Page 4/7. Add Auto Projector component to Projector prefabs in Assets



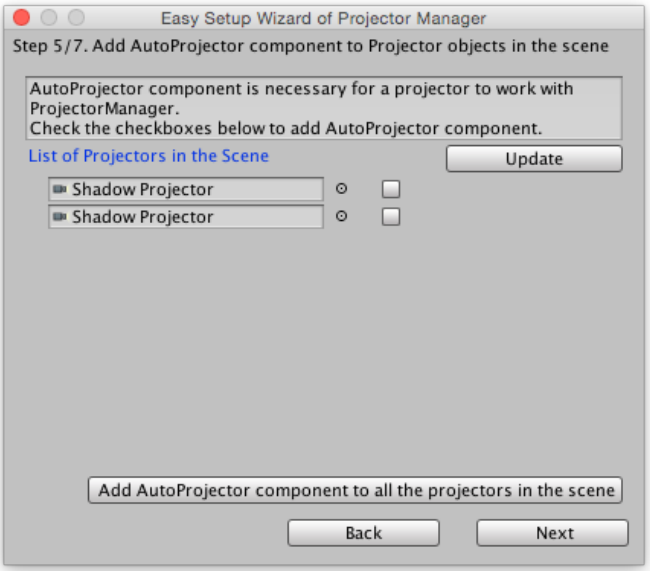
In this page, you can add Auto Projector component to Projector objects in prefabs. If a projector object doesn't have Auto Projector component, it cannot work with Projector Manager. That means, the projector cannot project shadows on shadow receivers, resulting in bad performance.

Press "Search for Prefab" button to get the list of Projector objects in prefabs as shown in the below image.



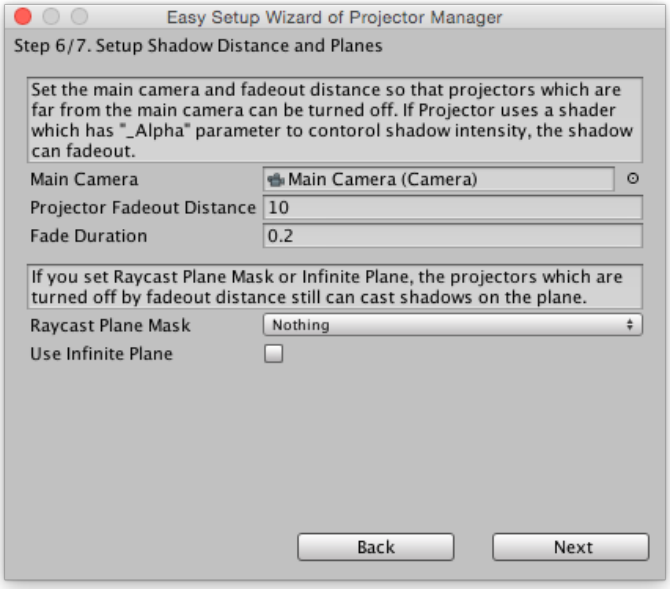
The left most column shows the projector objects, and the next columns shows the prefab objects containing the projector object in the first column. You can click the fields in those columns to select the object in Project window. Each checkbox in the right most column indicates whether the projector object has Auto Projector component or not. You can check/uncheck the checkboxes to add/remove Auto Projector components.

Page 5/7. Add Auto Projector component to Projector objects in the scene



In this page, you can add Auto Projector component to Projector objects in the current scene. Check/uncheck the checkboxes to add/remove Auto Projector components.

Page 6/7. Setup Shadow Distance and Planes



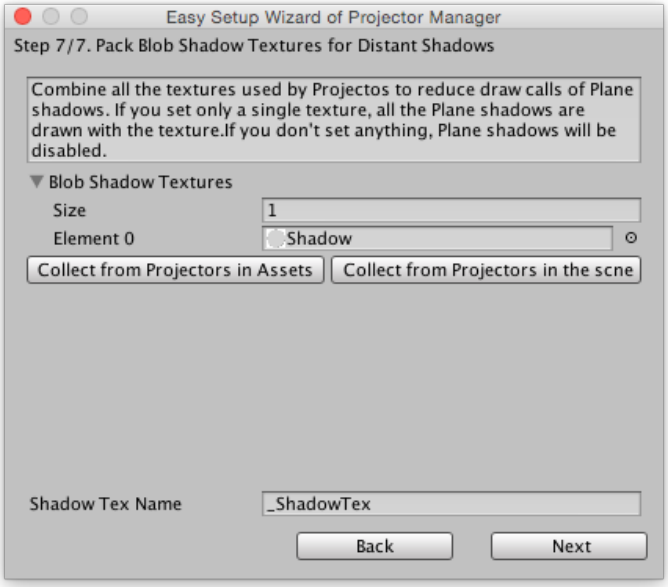
In this page, you need to set the Main Camera and Projector Fadeout Distance. The projectors whose distance from the camera are longer than Projector Fadeout Distance will be turned off. If you want projectors to cast shadows always, set a large value (e.g. far clip plane of the main camera) to Projector Fadeout Distance.

Additionally, you can set Raycast Plane Mask and Infinite Plane. Those parameters are used for the projectors which are turned off by Projector Fadeout Distance. Even a projector which is turned off can generate a shadow represented by a quadrangle polygon. If Raycast Plane Mask is not Nothing, a raycast is used to put the quadrangle polygon. If Infinite Plane is set, the quadrangle polygon will be casted on the plane. If both are specified, raycast is tested first, and if the ray hit an object in Raycast Plane Mask layers, the shadow will be placed at the hit point. If the ray doesn't hit, the infinite plane will be used.

For more details, please refer to "**Setup Projector Manager**" section.

If you setup neither Raycast Plane Mask nor Infinite Plane, the wizard will end at this page.

Page 7/7. Setup Shadow Distance and Planes



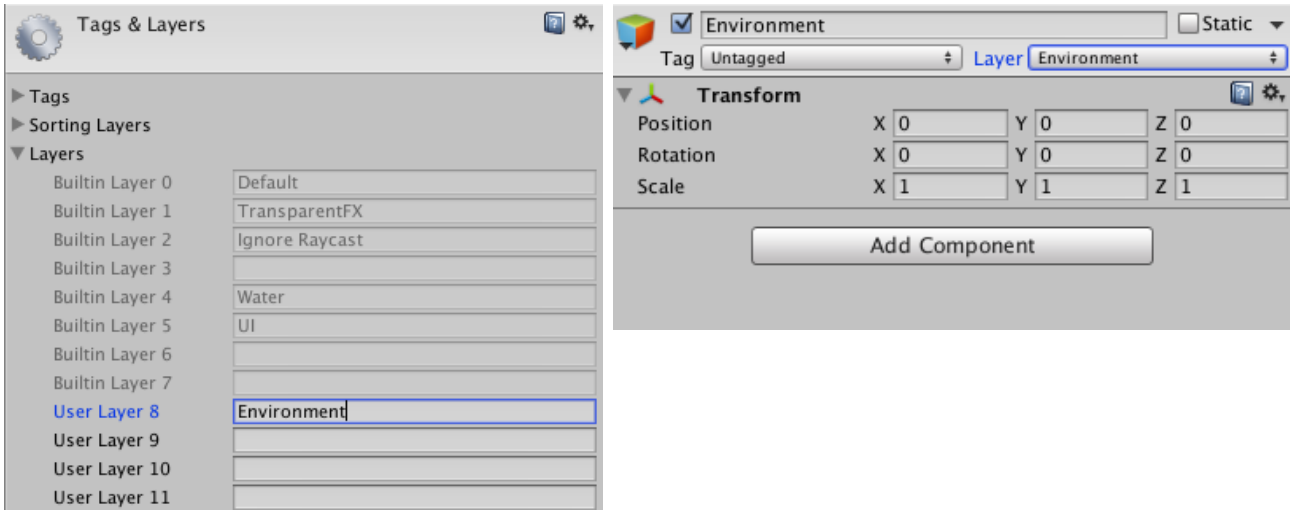
In the last page of the wizard, you need to set textures of blob shadows and combine them into a single packed texture so that the quadrangle polygon shadows can be drawn by a single draw call. If you didn't setup either Raycast Plane Mask or Infinite Plane, this page will be skipped.

You can collect all the blob shadow textures which are used by Projector objects by clicking "Collect from Projectors in Assets" button and "Collect from Projectors in the scene" button. Please be noted that only the textures used by projector objects with Auto Projector component can be collected.

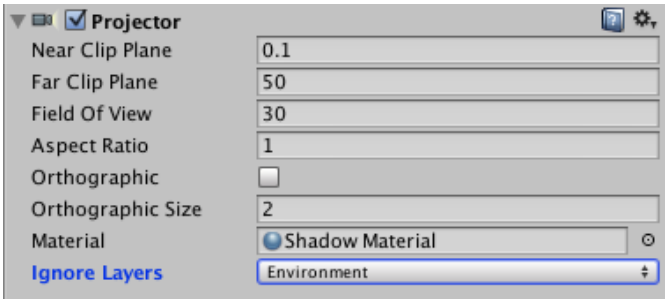
If more than one textures are collected, "Pack Blob Shadow Textures" button will appear. Click this button to create a packed texture. It is not recommended to manually set a packed texture created before. If you want to share a packed texture with other Projector Managers, please create a prefab of a Projector Manager, and create instances from the prefab.

Steps to use Shadow Receiver with Projector

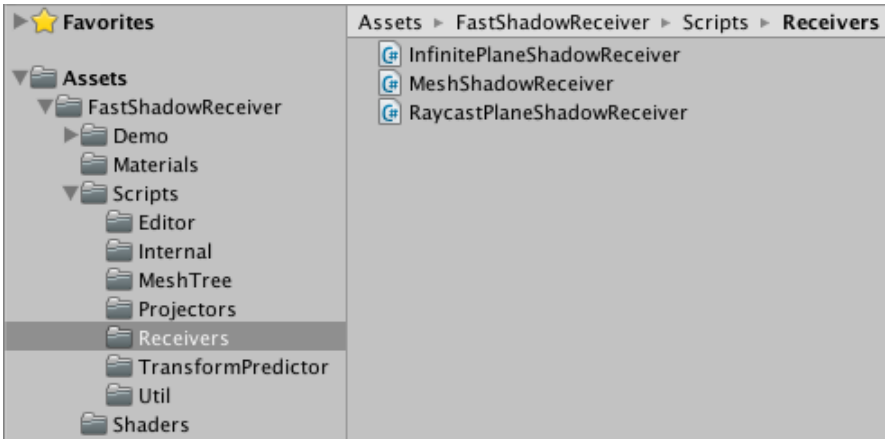
1. If you don't have a "Layer" for your large environment object yet, add a new layer (let's say "Environment" layer) and assign it to the environment object. You can open "Tags and Layers Manager" by selecting "Edit > Project Settings > Tags and Layers" menu.



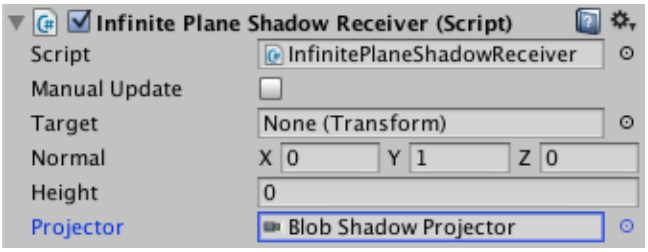
2. Add "Environment" layer to "Ignore Layers" of your projector object(s). If you don't have any projector objects yet, you can add 'Effect > Projector' component to an gameobject by clicking 'Add Component' button in the Inspector View.



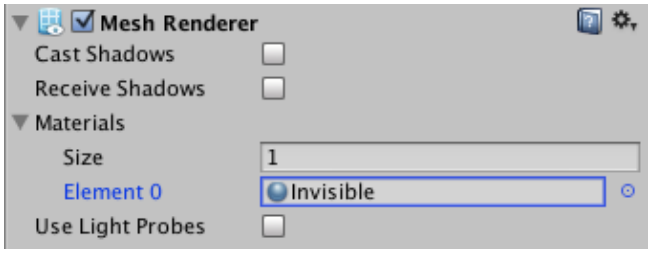
3. Create an empty game object and attach one of the shadow receiver components.
The scripts of shadow receiver components are located in "FastShadowReceiver/Scripts/Receivers" folder.



4. Assign a projector object to the shadow receiver component created at step 3.



5. Assign “Invisible” material to MeshRenderer component of the game object created at step 3. “Invisible” material can be found in “FastShadowReceiver/Materials” folder. This step will be done automatically at step 4.

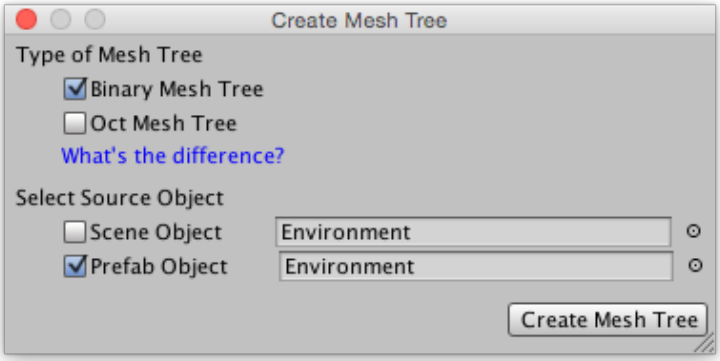


6. Assign the large environment object to the shadow receiver:
   - as “Target” property , if you created InfinitePlaneShadowReceiver at step 3.
   - as “Mesh Transform” property, if you created MeshShadowReceiver at step 3.
   - set “Raycast Mask” to “Environment” layer, if you created RaycastPlaneShadowReceiver at step 4.

   For more details, see “**Setup Shadow Receiver**” section.

7. If you created MeshShadowReceiver at step 3, you need to create a Mesh Tree, and set it to “Mesh Tree” property of MeshShadowReceiver component.
   If no “Mesh Tree” has been set, and “Mesh Transform” property is set as described in step 6, you can find “Create a new MeshTree” button.

   This button will open Create Mesh Tree Wizard. For more details, see “**Create a Mesh Tree**” section.

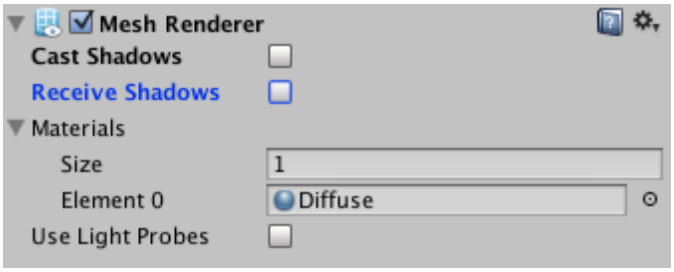


8. Repeat steps from 3 to 6 for any projector objects which are not yet assigned to any shadow receiver objects.

9. If you have multiple projectors, assign a unique layer to each shadow receiver object, and modify “Ignore Layers” property of each projector so that each shadow receiver can receive only the shadow from the projector assigned to it. For more details, please have a look at “Multi-Projector Plane” and “Multi-Projector Mesh” demo.

Steps to use Shadow Receiver with Shadow-map

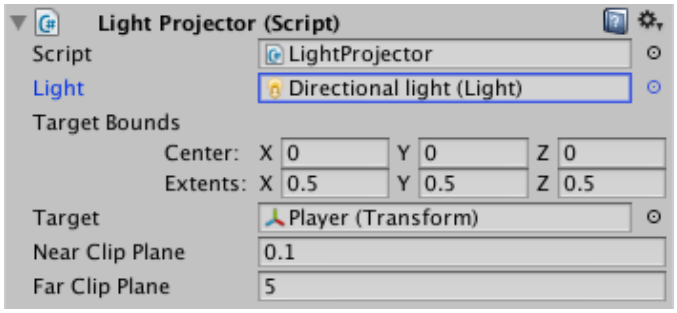
1. Select the large environment object(s) and uncheck “Receive Shadows” in MeshRenderer component. (Unity Terrain object doesn’t have “Receive Shadows” property, so you cannot use Unity Terrain object with shadow-map)



2. Repeat steps from 3 to 11 for each shadow caster objects.

3. Create an empty game object and attach “LightProjector” component. You can find the script file in “FastShadowReceiver/Scripts/Projectors” folder.

4. Assign the main light object to “Light” property of LightProjector component created at step 3.

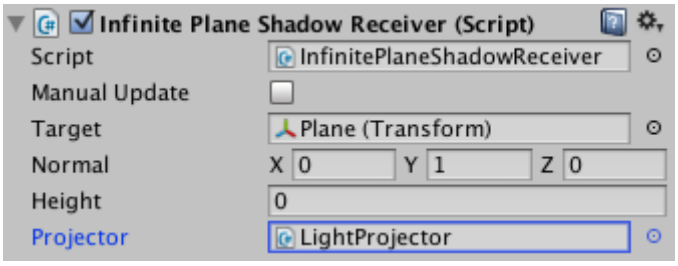


5. Assign the caster object to “Target” property of the light projector object.

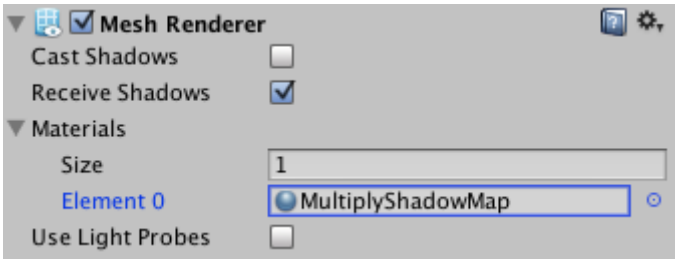
6. Specify “Target Bounds” of the LightProjector object so that the bounding box can enclose the caster object.

7. Create an empty game object and attach one of the shadow receiver components.
The scripts of shadow receiver components are located in “FastShadowReceiver/Scripts/Receivers” folder.

8. Assign the LightProjector object created at step 3 to “Projector” property of the shadow receiver object created at step 7.



9. Assign “MultiplyShadowMap” or “BlendShadowMap” material in “FastShadowReceiver/Materials” folder to MeshRenderer component of the game object created at step 7.

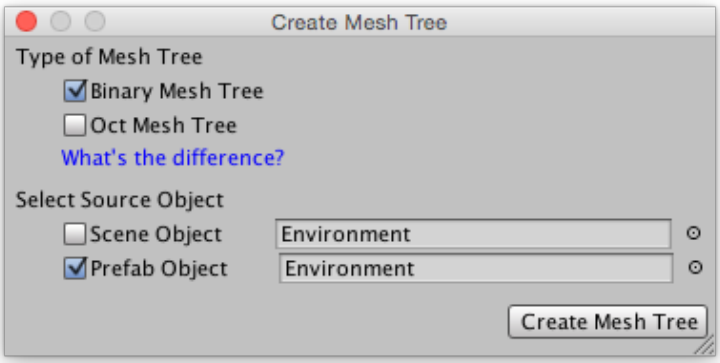


10. Assign the large environment object to the shadow receiver:
    - as “target” property , if you created InfinitePlaneShadowReceiver at step 7.
    - as “MeshTransform” property, if you created MeshShadowReceiver at step 7.
    - set “Raycast Mask” to “Environment” layer (same layer as the environment object), if you created RaycastPlaneShadowReceiver at step 7.

    For more details, see “**Setup Shadow Receiver**” section.

11. If you created MeshShadowReceiver at step 7, you need to create a Mesh Tree, and set it to “Mesh Tree” property of MeshShadowReceiver component.
    If no “Mesh Tree” has been set, and “Mesh Transform” property is set as described in step10, you can find “Create a new MeshTree” button.

    This button will open Create Mesh Tree Wizard. For more details, see “**Create a Mesh Tree**” section.



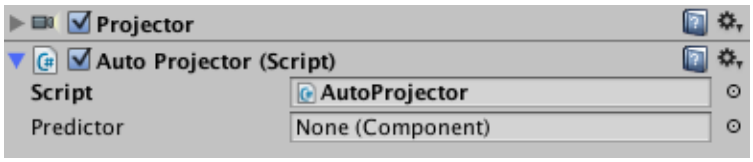
Steps to use Shadow Receiver with Massive Projectors

1. Follow step 1 and step 2 in "**Steps to use Shadow Receiver with Projector**".

2. Create an empty game object and attach "ProjectorManager" script to it. You can find the script file in "FastShadowReceiver/Scripts/Util" folder.

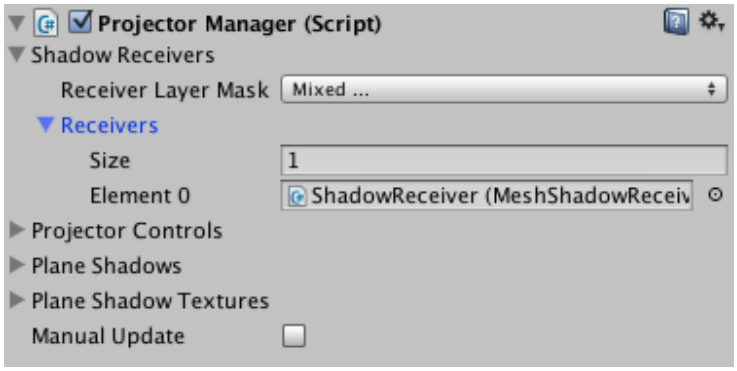
3. Setup the ProjectorManager component as in "**Setup Projector Manager**" section below.

4. Attach "AutoProjector" script to your Projector object (or prefab), so that it can be automatically registered to the ProjectorManager at runtime.



5. Create MeshTree and MeshShadowReceiver as described on "**Steps to use Shadow Receiver with Projector**" (steps 3 to 7), if you want to project shadows on non-flat surface. You can skip this step if you don't need shadows on non-flat surface of the large environment object. ProjectorManager has similar functionality as InfinitePlaneShadowReceiver and RaycastPlaneShadowReceiver. It can create quadrangle polygons to draw shadows. So, you don't need to create a shadow receiver object unless the large environment object has non-flat surface.

6. If you created a MeshShadowReceiver at step 5, register the MeshShadowReceiver object to ProjectorManager, and assign some layers to 'Receiver Layer Mask'. The MeshShadow-Receiver will be duplicated at runtime accordingly to match active projector instances. Also, one of the layers in 'Receiver Layer Mask' will be assigned to each duplicated shadow receiver. If you have many projectors in your scene, It is better to assign as many layers as possible to 'Receiver Layer Mask'. For more details, please have a look at "RandomSpawn - Mesh" demo.



Create a Mesh Tree

MeshShadowReceiver component requires a Mesh Tree object in order to search for polygons which are intersecting with the shadow volume. You can create a Mesh Tree object from the context menu of Project window. Right click a folder in which you want to create a Mesh Tree, and select “Create > FastShadowReceiver > XxxxMeshTree”, where Xxxx is Binary, Oct, or Terrain. Also, you can create a Mesh Tree with a wizard window which can be opened from Easy Setup Wizard or Inspector View of MeshShadowReceiver. A Mesh Tree must be built from mesh object(s) before use as described below, and if the mesh object or the layout of the mesh objects is changed, need to be rebuilt.

BinaryMeshTree and OctMeshTree

BinaryMeshTree and OctMeshTree have almost same functionality. Both are used for either a single mesh object or a game object which consists of many child MeshRenderer objects. Basically, the difference between BinaryMeshTree and OctMeshTree is performance, such as search speed, memory usage and scissoring cost. Roughly speaking, OctMeshTree can save memory but slower for small scene and search result will have larger extra area.

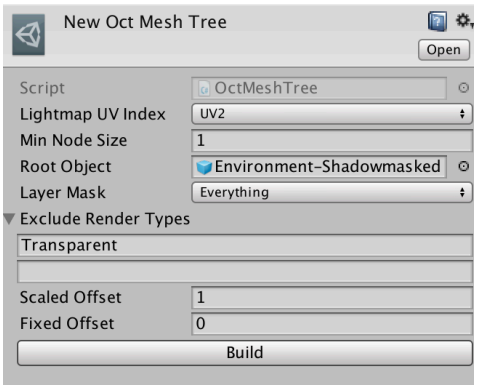
Large extra area will incur a performance penalty. MeshShadowReceiver has “Scissor” option. If you enable it, the extra area can be cut out and GPU performance will be improved. However, it will take some CPU time to scissor polygons. If you use “Scissor” option with “Scissor Margin” property, you can reduce scissoring cost, but it is only applicable for BinaryMeshTree. For more details, see “**Setup Shadow Receiver**” section below.

Table 1: The differences between BinaryMeshTree and OctMeshTree

	BinaryMeshTree	**OctMeshTree**
Search Speed	Faster for small scene	Faster for huge scene
Memory Usage	More	Less
Search Result	Smaller extra area	Bigger extra area
Scissoring	Faster	Slower

If you are not sure which Mesh Tree is better, simply try both and choose the one which has better performance.

Before you use a Mesh Tree object, you need to build it from mesh object(s). Set a Mesh object or a root game object of the environment object(s) at ‘Root Object’ in Inspector View. Then, ‘Build’ button will be enabled. It is recommended to set a prefab object as the Root Object instead of a scene object. Otherwise, the reference to the Root Object cannot be serialized. You will need to set the Root Object again when you rebuild the Mesh Tree.



Optionally, you can specify ‘Layer Mask’ and ‘Exclude Render Types’ when you set a root game object. If you are building OctMeshTree, you can also specify ‘Min Node Size’ before build. Smaller ‘Min Node Size’ will make search result smaller, but memory usage will be bigger.

'Scaled Offset' and 'Fixed Offset' are used to push each vertex of the mesh along its normal vector. This is necessary for avoiding z fighting problem, especially if the environment is large. The offset value is calculate for each vertex v as follows:

offset = 0.00000025 x |v| x <Scaled Offset> + <Fixed Offset>.

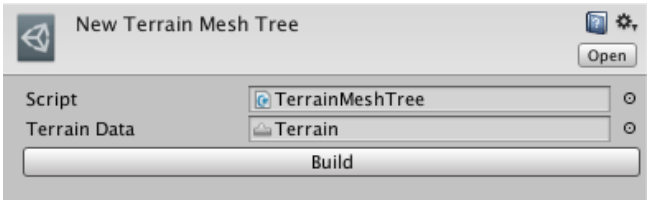
The reason why it has Scaled Offset is, z fighting problem is caused by floating point rounding error, and maximum possible value of the error is roughly proportional to the absolute value of the operands.

If you want to use lightmaps with a shadow receiver, you need to use "UV2" for Lightmap UV Index. Then, the root object or an instance of the root object prefab must exist in the current scene. Lightmap indices and lightmap scale offsets for the current scene are stored in the mesh tree. Please rebuild the mesh tree every time when these lightmap parameters are changed.

If the prefab object contains lightmap data, and lightmap indices may change when the prefab is instanciated, please add "FastShadowReceiver.LightmapDataInPrefab" component to the root object in the prefab.
If "FastShadowReceiver.LightmapDataInPrefab" is added to the root object, build the mesh tree in the scene where the lightmaps are baked.
If you don't use "FastShadowReceiver.LightmapDataInPrefab" component but you want to reuse the lightmaps, a mesh tree must be built for each scene where the prefab is used (if the lightmap indices can be different among the scenes).

TerrainMeshTree

TerrainMeshTree is used for a Terrain object. Set a TerrainData asset to 'Terrain Data' in Inspector View, and press 'Build' button. There are no other build options for TerrainMeshTree.



Building a Mesh Tree at runtime

It is possible to build a Mesh Tree at runtime, if the meshes of the environment objects are readable. It will take long time though, building process is done in background thread. It will not stop your application. However, shadows are not seen until the build process is completed. There is an example scene in FastShadowReceiver/Demo/SimpleExamples/ RandomLevelGeneration. Also, you can find C# source code for the runtime build in FastShadowReceiver/Demo/Scripts/RandomLevelGeneration.cs.

Setup Shadow Receiver

Common Settings

- **Manual Update**
  If 'Manual Update' is checked, the Shadow Receiver will not be updated automatically. If you want to control update timing yourself, use this option. This is useful for MeshShadow-Receiver because it uses multi-thread to update its mesh (if available).

  If 'Manual Update' is unchecked, multi-threaded task will start in LateUpdate function, and join the main thread in OnWillRenderObject function. If you can start the multi-threaded task just after Projector position is changed, you can save waiting time in OnWillRenderObject function.

  To start the multi-threaded task manually, call UpdateReceiver function of FastShadow-Receiver.ReceiverBase class.

  From version 1.4.1, shadow receiver will call "UpdateTransform()" on LateUpdate if any components of the associated projector object have such public method. So, you don't need to care about script execution order if a component which controls the projector transform has UpdateTransform method.

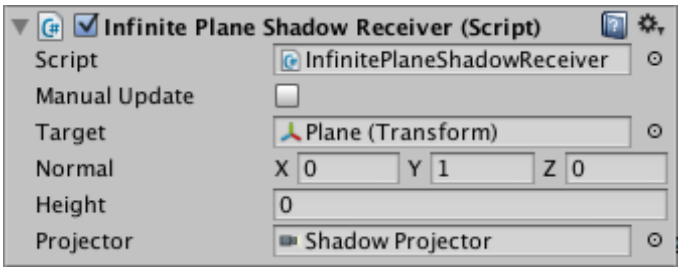
- **Projector**
  Set a Projector (UnityEngine.Projector) or FastShadowReceiver.ProjectorBase object here. This projector is used to calculate the intersection of the shadow receiver and the projection volume.

  If the shadow receiver is receiving a shadow from a Projector, simply set the Projector here.

  If shadow-map is used to cast shadow on the shadow receiver, set a LightProjector here (See "**Steps to use Shadow Receiver with Shadow-map**" section above).

  If you want to calculate the projection volume manually, you can use FastShadow-Receiver.CustomProjector.

InfinitePlaneShadowReceiver

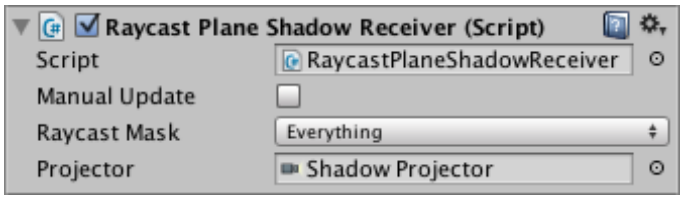


- **Target**
  Set the environment object which will be represented by a plane here. If nothing is specified, no transform will be applied to the plane.

- **Normal**
  Input the normal vector of the plane here. It must be represented in local space of Target.

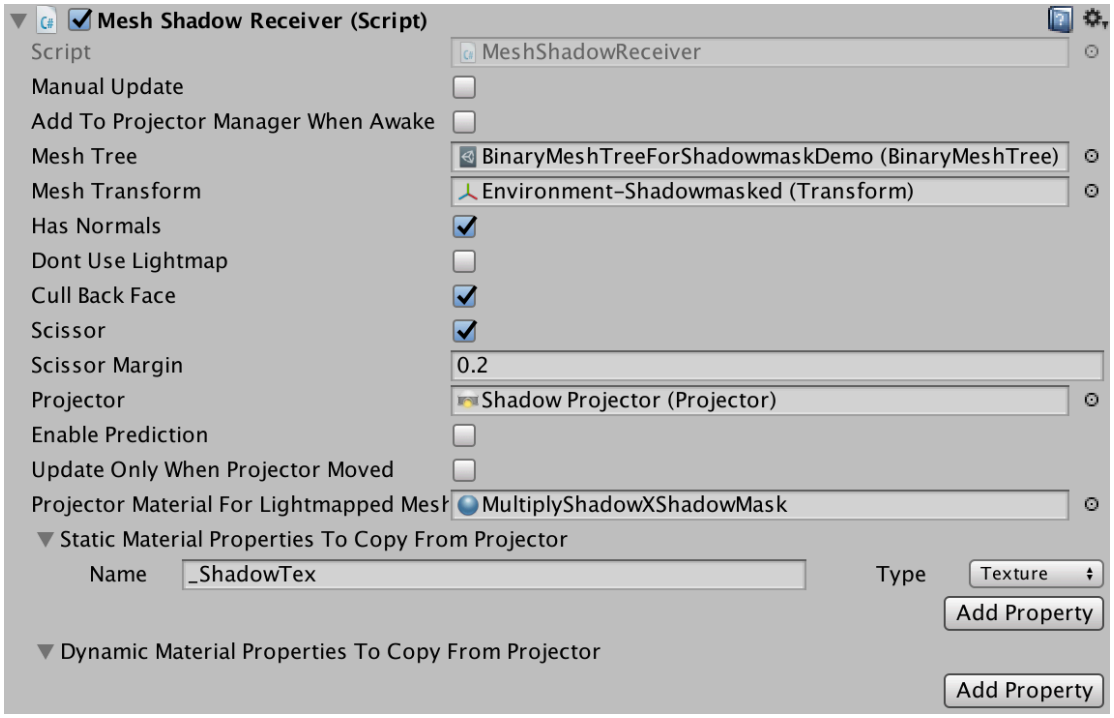
- **Height**
  Height is the position of the plane in Normal direction. In other words, distance from the position of Target to the plane.

RaycastPlaneShadowReceiver



- **Raycast Mask**
  This Layer Mask will be passed to Physics.Raycast function when RaycastPlaneShadow-Receiver tries to find a position to place a quadrangle mesh. The environment object should have collider(s) and one of the layers in Raycast Mask.

MeshShadowReceiver



- **Add To Projector Manager When Awake**
  If checked, this mesh shadow receiver will be added to the ProjectorManager when awake. This is useful if the mesh shadow receiver is instantiated from a prefab. For more details, see "**Setup Projector Manager**" section below.

- **Mesh Tree**
  Set a Mesh Tree here (see "**Create a Mesh Tree**" section above).

- **Mesh Transform**
  A mesh generated from the Mesh Tree will be transformed by 'Mesh Transform'. So, it should be the same object from which the Mesh Tree was created. If you create the Mesh Tree from a single mesh, it should be the object which has MeshRenderer of the mesh.

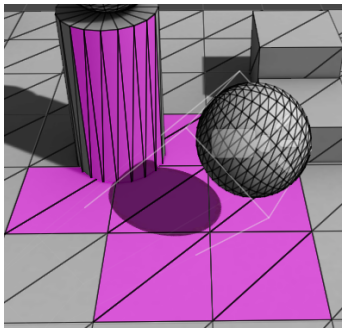
- **Has Normals**
  Check this option if the projector uses a shader which requires normal vectors. For more details, see "**Projector Shaders**" section below.

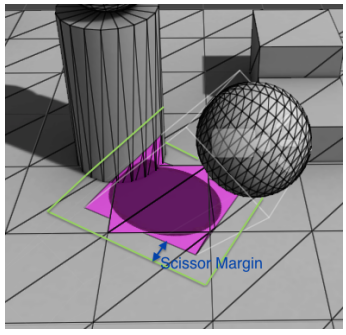
- **Dont Use Lightmap**
  If checked, this mesh shadow receiver will never use lightmaps regardless of whether the mesh tree has Lightmap UVs or not. For more details, see "**Mixed Lighting Blob Shadow**" section below.

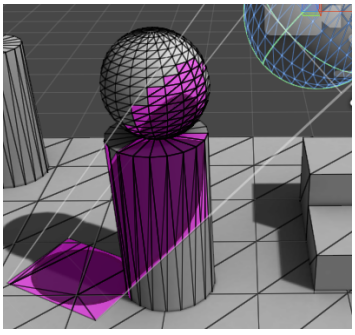
- **Cull Back Face**
  If this option is checked, polygons which are not facing to the projector are removed from the mesh.

- **Scissor**
  MeshShadowReceiver will pick up polygons which intersect with the projection volume from the MeshTree (See the left figure below). However it is still larger than the actual intersection. If 'Scissor' is checked, extra area can be cut out (the center figure below), but it will take some CPU time.

- **Scissor Margin**
  It is waste of CPU time to scissor small polygons which stick out just a bit from the bounds. The center figure below shows how 'Scissor Margin' works. The right figure below is an effective example of 'Scissor Margin'. In this scene, the polygons of the plane and the cylinder are scissored but the small polygons of the sphere are not scissored.
  'Scissor Margin' is only applicable for BinaryMeshTree.







- **Enable Prediction**
  If 'Enable Prediction' is checked, MeshShadowReceiver will prepare a mesh for next frame in advance so that it can take advantage of multi-core processors. To enable prediction, you need to set 'Predictor' as well. Prediction will be turned off if it runs on a device which does not have multi-core processor.

- **Predictor**
  Predictor is a component inherited from ITransformPredictor interface. It predicts next frame position of the projector. For more details, please have a look at "MultiProjector - Mesh" demo.

- **Update Only When Projector Moved**
  If the Projector does not keep moving, you can check this option to save CPU time for updating the mesh. The mesh will not be updated as long as the projection volume stays within 'Margin'.

- **Margin**
  Applicable only when ‘Update Only When Projector Moved’ is checked.

- **Projector Material For Lightmapped Mesh**
  Specify a material to be applied to the lightmapped receiver mesh. If not specified, the projector material will be used. However, usually, the projector material are applied to dynamic objects as well as the lightmapped static meshes. It is better to set non-lightmapped material to the projector, and set lightmapped material here. For more details, see “**Mixed Lighting Blob Shadow**” section.

- **Static Material Properties To Copy From Projector**
  Specify material properties which are needed to copy from the projector material to the “Projector Material For Lightmapped Mesh”. Copy is done only once when created. If the value of the property will change, add the property to “Dynamic Material Properties To Copy From Projector” instead.

- **Dynamic Material Properties To Copy From Projector**
  Specify material properties which are needed to copy from the projector material to the “Projector Material For Lightmapped Mesh”. Copy will be done every frame.
  If the mesh shadow receiver is used with a ProjectorManager which has effective “Projector Fadeout Distance” and “Fade Duration”, please add “_Alpha” property with Float type. For more details, please see “Setup Projector Manager” section below.

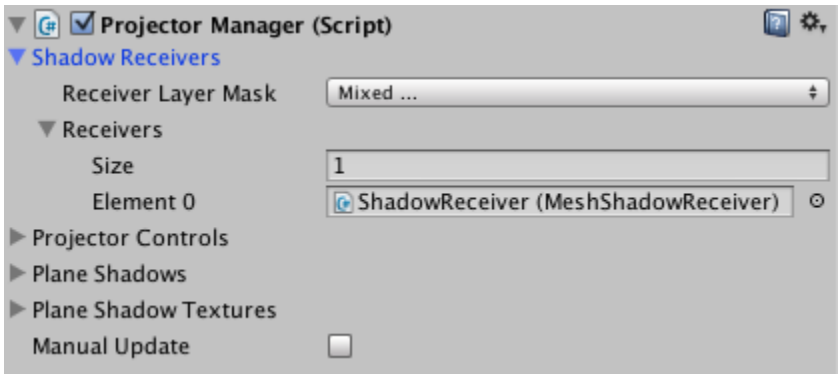
Setup Projector Manager

Projector Manager is a singleton component which manages projectors in a scene. It will activate the projectors which are close to the main camera, and deactivate the projectors which are far from the main camera or not visible from the main camera. To activate/deactivate projectors, you need to setup "**Projector Controls**" properties.

For the deactivated projectors, Projector Manager will create quadrangle polygons to draw shadows. So, even when a projector is deactivated, it can project a shadow on a plane. For this purpose, you might need to setup "**Plane Shadows**" properties and "**Plane Shadow Textures**" properties.

For the activated projectors, you can use shadow receiver objects. However, it is meaningless to use InfinitePlaneShadowReceiver or RaycastShadowReceiver, because Projector Manager can also create quadrangle polygons like these two shadow receivers. If you want to use Mesh-ShadowReceiver, you need to setup "**Shadow Receivers**" properties.

Shadow Receivers



- **Receiver Layer Mask**
  Projector Manager will duplicate a set of shadow receiver objects in 'Receivers' array for each active projector object at runtime, and assign one of the layers in 'Receiver Layer Mask' to a duplicated set so that only one projector object can project a shadow onto the set of the receiver instances. If 'Receiver Layer Mask' doesn't have enough layers, more than one projector objects will project shadows onto the same shadow receiver instance. Usually, it won't cause a problem, but if the projection volumes intersect each other, some shadows will be drawn more than once, and a part of the shadow will be darker than the normal shadow.

- **Receivers**
  Specify a set of shadow receivers which will be duplicated for each active projector object at runtime. Usually, the set must have only one MeshShadowReceiver. Even if the environment model consists of multiple objects, it is better to combine these objects into one Mesh Tree object for performance reason. However, if the scene has an open field, you might want to create environment models at runtime. In such cases, you can add multiple MeshShadowReceiver objects from script using this following code:

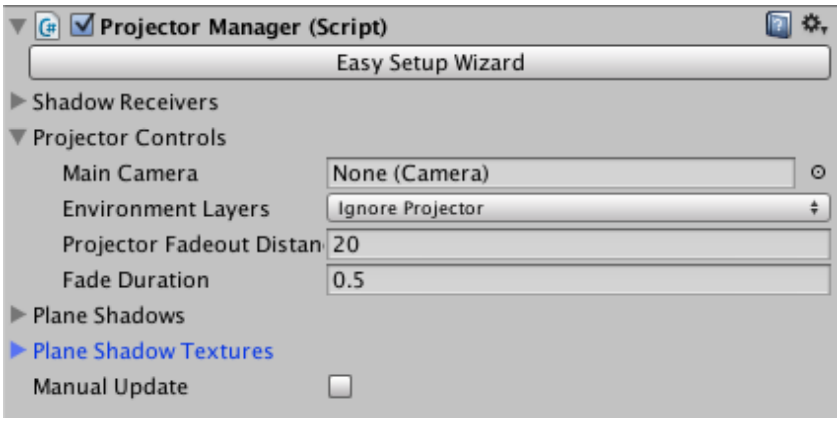
```
ProjectorManager.Instance.AddReceiver(shadowReceiver);
```

  Also, you can remove a shadow receiver with this code:

```
ProjectorManager.Instance.RemoveReceiver(shadowReceiver);
```

  If the array of 'Receivers' is empty, 'Plane Shadows' are used for activated projectors as well as deactivated projectors.

Projector Controls



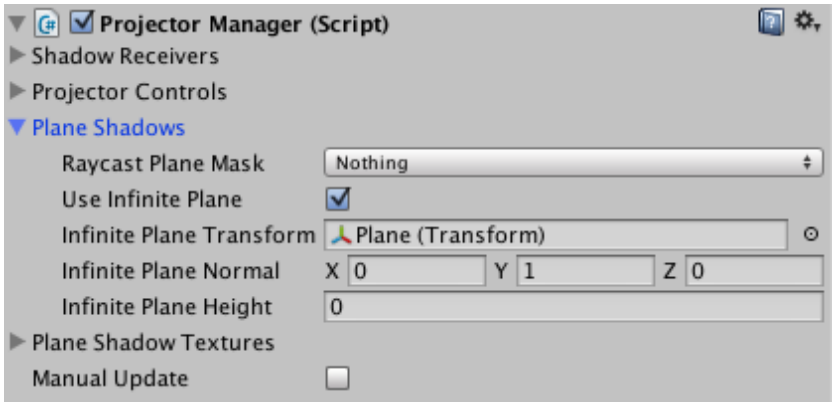
- **Main Camera**
  Specify a main camera in your scene. If not specified, Camera.main will be used. This camera is used to activate/deactivate projectors. Only the projectors which are close and visible to the camera will be activated.

- **Environment Layers (from Version 1.4.0)**
  Specify layers for the environment objects. This layer mask is used for projectors to ignore the environment objects. If you specify this layer mask properly, you don't need to set the environment layers to Ignore Layers of the projectors so that you can see shadows in Editor.

- **Projector Fadeout Distance**
  Specify a range where the projectors can be active around the main camera.

- **Fade Duration**
  Specify fade in/out duration. To enable fade in/out, the material of a projector should have "_Alpha" property to control the darkness of shadows.

Plane Shadows



- **Raycast Plane Mask**
  Specify a layer mask which will be passed to Physics.Raycast function. Raycast is performed for each projector to create a quadrangle polygon at the position where a ray from the projector hit. If 'Nothing' is specified, no raycast will be performed.

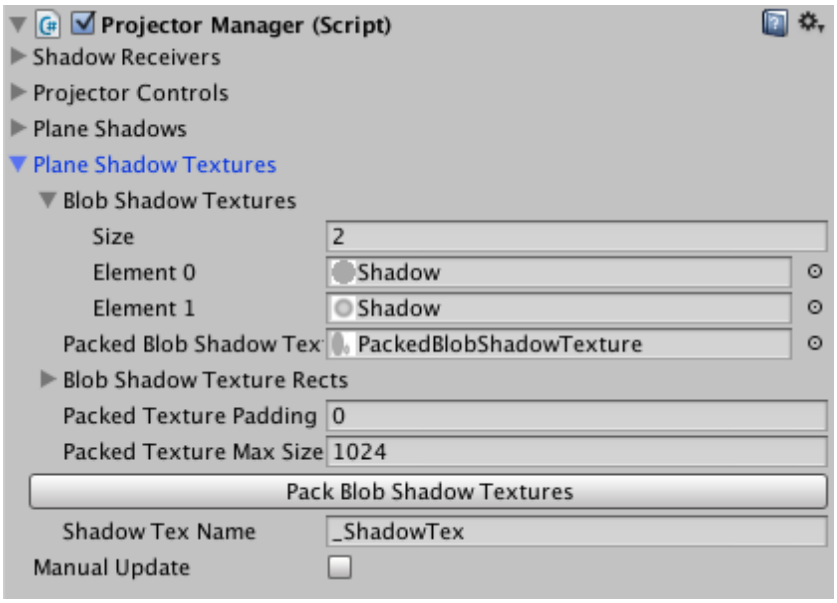
- **Use Infinite Plane**

Specify if a infinite plane is used to create plane shadows or not. Raycast Plane has a higher priority than Infinite Plane. A quadrangle shadow polygon will be created on the infinite plane only when a raycast hit nothing.
If this option is checked, the following three parameters will be shown.

- **Infinite Plane Transform**
  Specify a transform of the infinite plane. If not specified, the infinite plane will be placed in world space.

- **Infinite Plane Normal**
  Specify a normal vector of the infinite plane.
- **Infinite Plane Height**
  Specify a distance from the position of 'Infinite Plane Transform' to the plane.

Plane Shadow Textures



- **Blob Shadow Textures**
  Set all the blob shadow textures used in the scene here.

- **Packed Blob Shadow Texture**
  Specify a combined texture which includes all the textures in 'Blob Shadow Textures' array. You can generate a combined texture by clicking "Pack Blob Shadow Textures" button.

- **Packed Shadow Texture Rects**
  Specify a uv rect in the combined texture for each 'Blob Shadow Texture'. The uv rects will be generated in conjunction with the combined texture when "Pack Blob Shadow Textures" button is clicked.

- **Packed Texture Padding**
  Specify a padding size in pixels between the packed textures.

- **Packed Texture Max Size**
  Specify a maximum size of a texture which will be generated by "Pack Blob Shadow Textures" button.

- **Shadow Tex Name**
  Specify a texture name used in both Projector's material and Projector Manager's material. By default, it is "_ShadowTex".

- **Manual Update**
  If ‘Manual Update’ is checked, Projector Manager will not be updated automatically. If you want to control the update timing, use this option and call

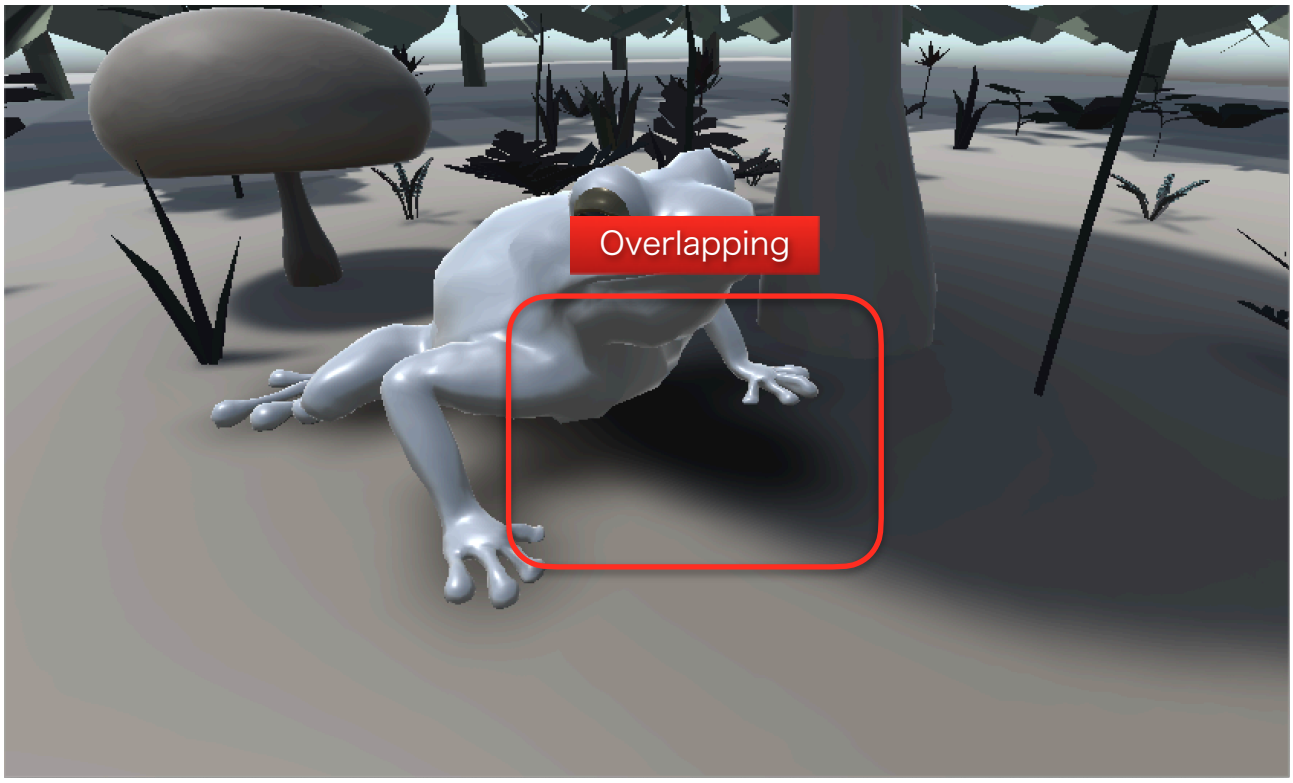
  ```
  ProjectorManager.Instance.UpdateScene();
  ```

  at the right timing.
  If ‘Manual Update’ is unchecked, Projector Manager will be updated on LateUpdate.

  From version 1.4.1, Projector Manager will call "UpdateTransform()" if any components of a projector object have such public method. So, you don’t need to care about script execution order if the component which controls projector transform has UpdateTransform() public method.

Mixed Lighting Blob Shadow (New feature in Version 1.5.0)

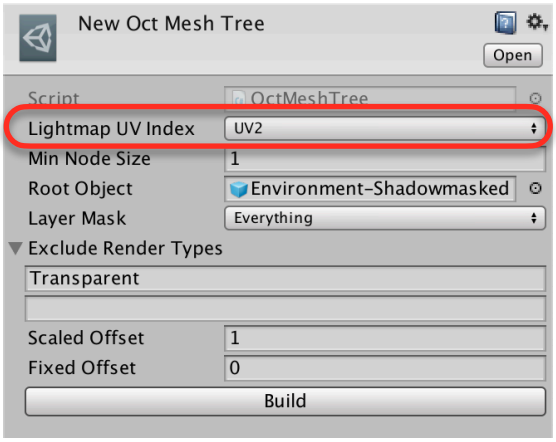
Before version 1.5.0, there was a problem that blob shadows were overlapping with static lightmap shadows.



From version 1.5.0, Mesh Shadow Receiver can have lightmap UVs, that makes Lightmaps and Shadowmasks available in Projector shaders. As a result, blob shadows can be properly mixed with lightmap shadows like the image below.

There are a few things to do for mixed lighting blob shadows.

1. Build a mesh tree with lightmap UVs



   Lightmap index and lightmap scale offset are stored in the mesh tree. It must be rebuilt every time when the lightmaps are changed.

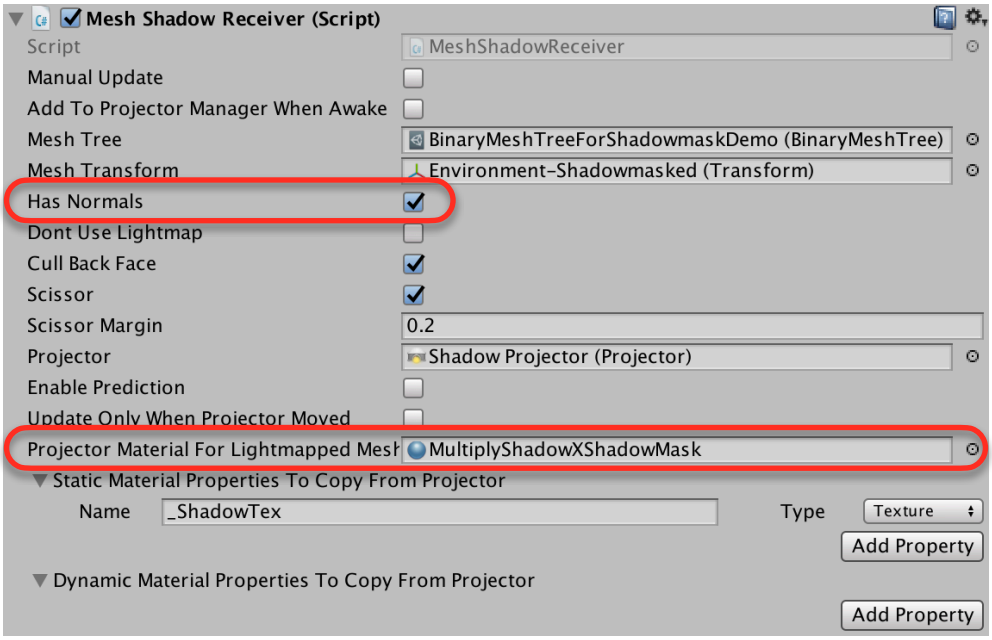
   Terrain mesh tree doesn't have this field, because Lightmap UVs are generated from vertex positions.

2. Create a new material and choose a correct shader
   Fast Shadow Receiver provides the following 3 shaders for mixed lighting.

Shader Name	Lighting mode	Perfromance and Quality
Multiply Lightmap Subtractive Shadow	Subtractive	This mode uses lightmap textures only. It has the highest performance. However you need to manually adjust Ambient Color in material editor.
Multiply Lightmap Shadowmasked Shadow	Shadowmask	Since this mode uses lightmap textures and Shadowmask texutres, it has the lowest performance, but the highest quality.
Multiply Diffuse x ShadowMask	Shadowmask	Simpler version for shadowmask mode. It still uses lightmap textures and shadowmask textures for drawing static objects and the performance is lower than subtractive mode. You need to manually adjust Ambient in material editor.

   For more details, see "Projector Shaders" section below.

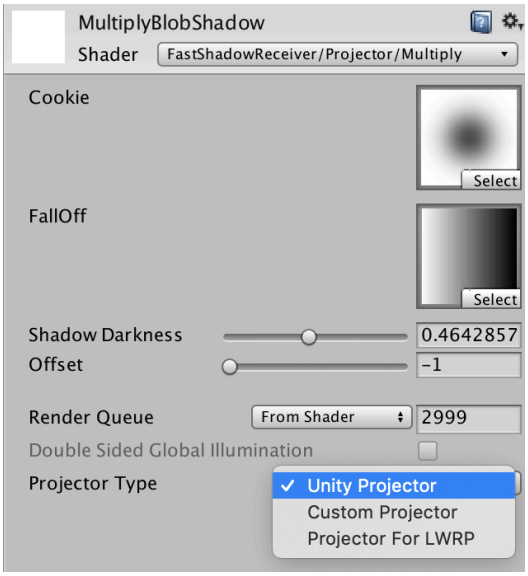
3. Use the new material as a 'Projector Material For Lightmapped Mesh' (see MeshShadowReceiver settings above). Also don't forget to check 'Has Normals', because the mixed lighting shaders use normal vectors for lighting computation.


Mesh Shadow Receiver (Script)
Script
MeshShadowReceiver
Manual Update
Add To Projector Manager When Awake
Mesh Tree
BinaryMeshTreeForShadowmaskDemo (BinaryMeshTree)
Mesh Transform
Environment-Shadowmasked (Transform)
Has Normals
Dont Use Lightmap
Cull Back Face
Scissor
Scissor Margin
0.2
Projector
Shadow Projector (Projector)
Enable Prediction
Update Only When Projector Moved
Projector Material For Lightmapped Mesh
MultiplyShadowXShadowMask
Static Material Properties To Copy From Projector
Name
_ShadowTex
Type
Texture
Add Property
Dynamic Material Properties To Copy From Projector
Add Property

Projector Shaders

Material Editor

When you choose one of the Projector Shaders for a material, you can see “Projector Type” property in the material editor.



Usually, you can just use “Unity Projector” type (default value).

“Custom Projector” is used with “ProjectionReceiverRenderer” component. If the projector material doesn’t have “Custom Projector” type, it will duplicate the material to change the type. If you chose “Custom Projector” type in advance, duplicate will not happen.

“ProjectionReceiverRenderer” is used for lightmapped mesh receiver (see Mixed Lighting Blob Shadow section above). It is better to use “Custom Projector” type for the mixed lighting blob shadow materials.

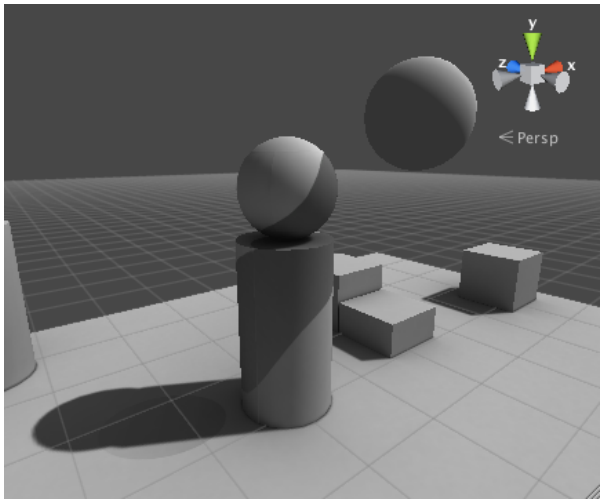
“Projector For LWRP” type should be used when you use Projector For LWRP (https://nyahoon.com/products/projector-for-lwrp).

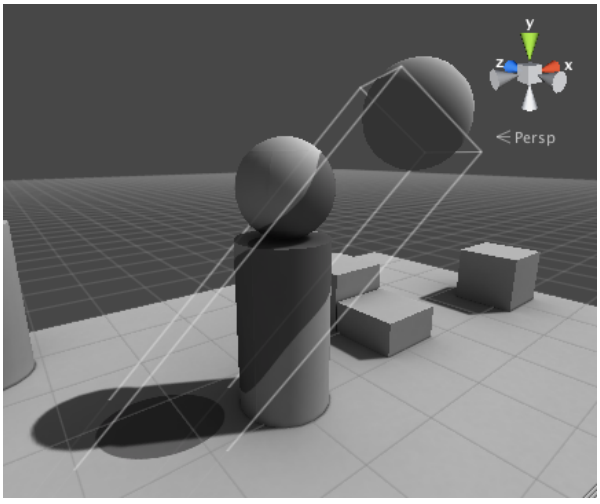
FastShadowReceiver/Projector/Multiply
FastShadowReceiver/Projector/Multiply Without Falloff

These shaders are similar to “Projector/Multiply Shadow” shader which is included in Standard Assets (Projectors). The difference is both of the FashShadowReceiver shaders have “Shadow Darkness” property to control shadow darkness.

“FastShadowReceiver/Projector/Multiply” have FallOff texture like “Projector/Multiply Shadow” shader (the left image). On the other hand, “FastShadowReceiver/Projector/Multiply Without Falloff” doesn’t have FallOff texture. So, the shadow which is far from the projector is still dark (the right image), and this shader is faster than the others.

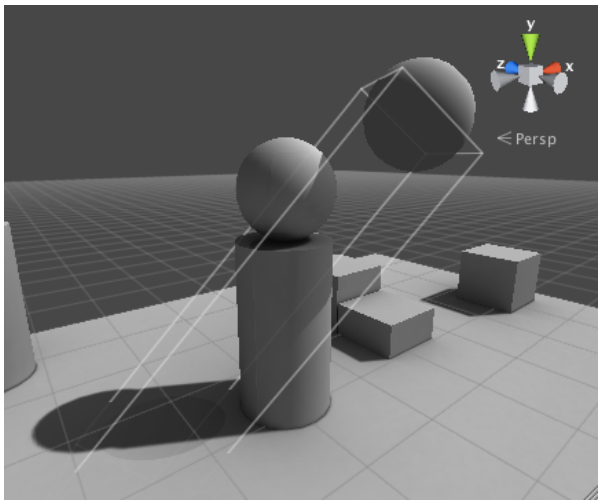
FastShadowReceiver/Projector/Multiply Diffuse
FastShadowReceiver/Projector/Multiply Diffuse Without Falloff

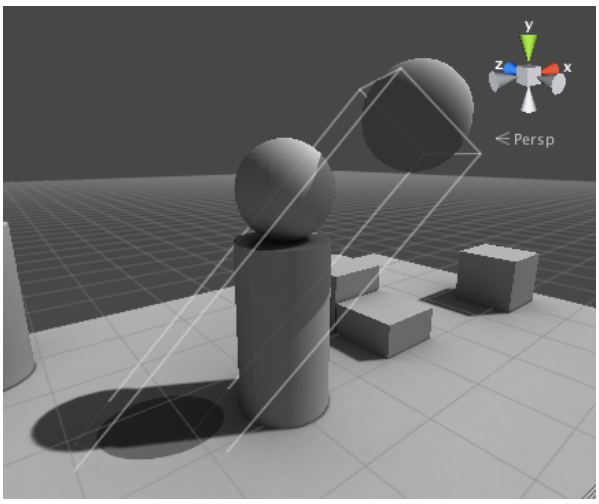




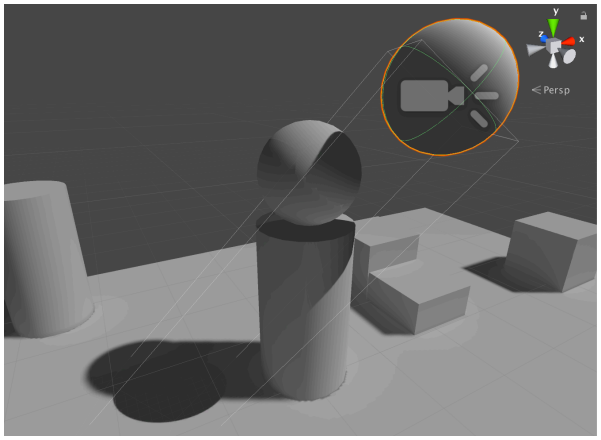
These shaders take into account normal vectors of a surface like diffuse lighting so that shadows can be more realistic. Also, these shaders can avoid the backside projection problem whereby a shadow is projected on a surface which is not facing to the projector.

These shaders require normal vectors. Don't forget to check 'Has Normals' property in MeshShadowReceiver.





FastShadowReceiver/Projector/Multiply Lightmap Subtractive Shadow
FastShadowReceiver/Projector/Multiply Lightmap Subtractive Shadow Without Falloff



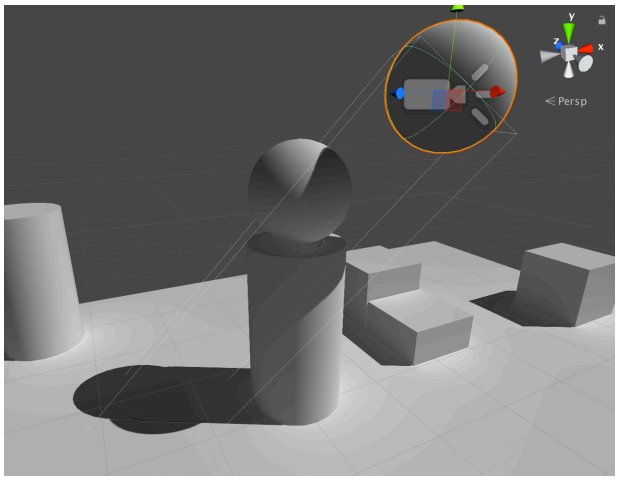
These shaders can be used when you are using mixed lighting with subtractive mode. The shadows casted by these shader are mixed with the shadows in the lightmaps to remove the overlapping shadows which can be seen in the screenshots above.

In subtractive mode, a lightmap has both direct and indirect light contributions in a single texture, and there is no way to find out how

much the direct light is shadowed by the static objects. So, these shaders requires ‘Ambient Color’ property to be set as the darkest color of the shadows.

If you want to use these shaders, please have a look at “Mixed Lightmap Blob Shadow” section above.

FastShadowReceiver/Projector/Multiply Lightmap Shadowmasked Shadow
FastShadowReceiver/Projector/Multiply Lightmap Shadowmasked Shadow Without Falloff



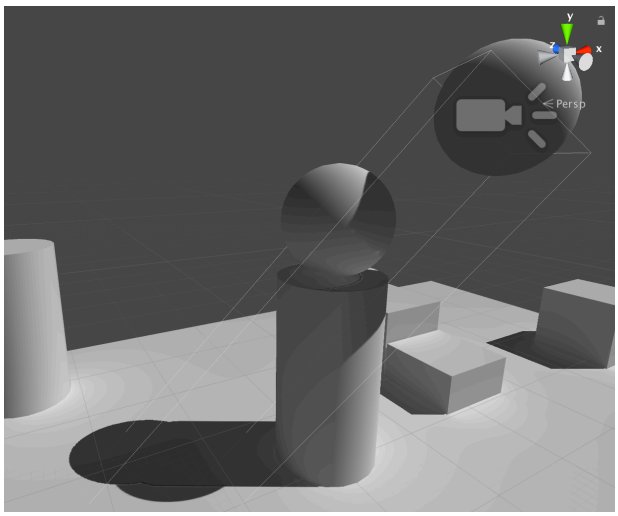
These shaders have the same features as the previous shaders, but for ‘shadowmask’ mixed lighting mode.

Unlike subtractive mode, lightmaps have only indirect light information, and ‘Ambient Color’ property is not needed.

However, these shaders need another property ‘Shadowmask Channel’ because a single shadowmask texture can have shadows for up to 4 lights by using R, G, B and A channels. The default value of ‘Shadowmask Channel’ property is (1, 0, 0, 0) which means it uses only R channel.

If you want to use these shaders, please have a look at “Mixed Lightmap Blob Shadow” section above.

FastShadowReceiver/Projector/Multiply Diffuse x Shadowmask
FastShadowReceiver/Projector/Multiply Diffuse x Shadowmask Without Falloff



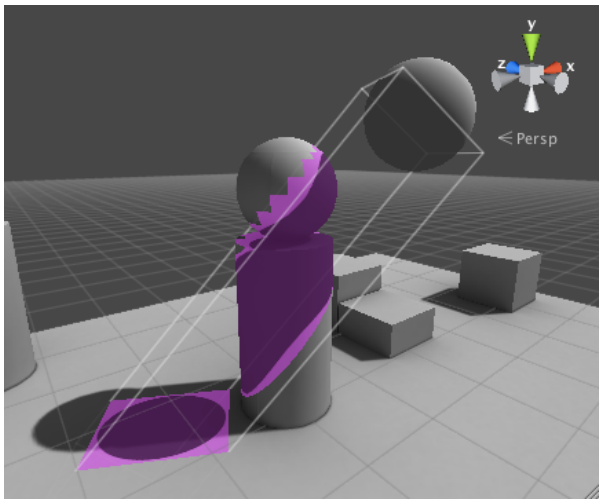
These shaders are simpler version of ‘Mulitply Lightmap Shadowmasked Shadow’ shaders above. They use only a shadowmask texture and they are faster to draw shadows. But usually, subtractive mode has better performance. You should use these shaders only when you have to use shadowmask mixed lighting mode, but don’t care very much about shadow quality.

These shaders requires ‘Ambient’ property to control shadow intensity, because they don’t use lightmaps which contain indirect light information.

If you want to use these shaders, please have a look at “Mixed Lightmap Blob Shadow” section above.

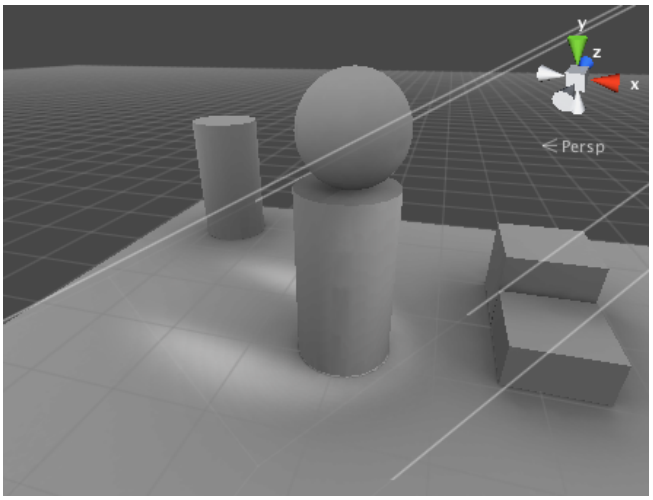
FastShadowReceiver/Projector/Visualize Receiver

This shader is used for debug purpose. It can visualize meshes which are receiving a shadow from the projector.



FastShadowReceiver/Projector/Light x Shadow

This shader can be used for projecting light texture. Additionally, it can cast shadow textures as well. To cast shadows, “Light Projector Shadow Receiver” component needs to be attached to a light projector, and “Light Projector Shadow Caster” component needs to be attached to shadow caster objects.



Demo

There are twelve demo scenes in “FastShadowReceiver/Demo/Scenes” folder. Additionally, there is another scene named “Demo” in “FastShadowReceiver/Demo” folder which can execute the other twelve scenes one by one. Before running the “Demo” scene, please add the twelve scenes into ‘Build Settings’, otherwise the scenes cannot be loaded from “Demo” scene.

In the demo scenes, some ‘Layers’ are used. If you add layers similarly as Table-2 below, it will be easier to understand how the demo scenes work.

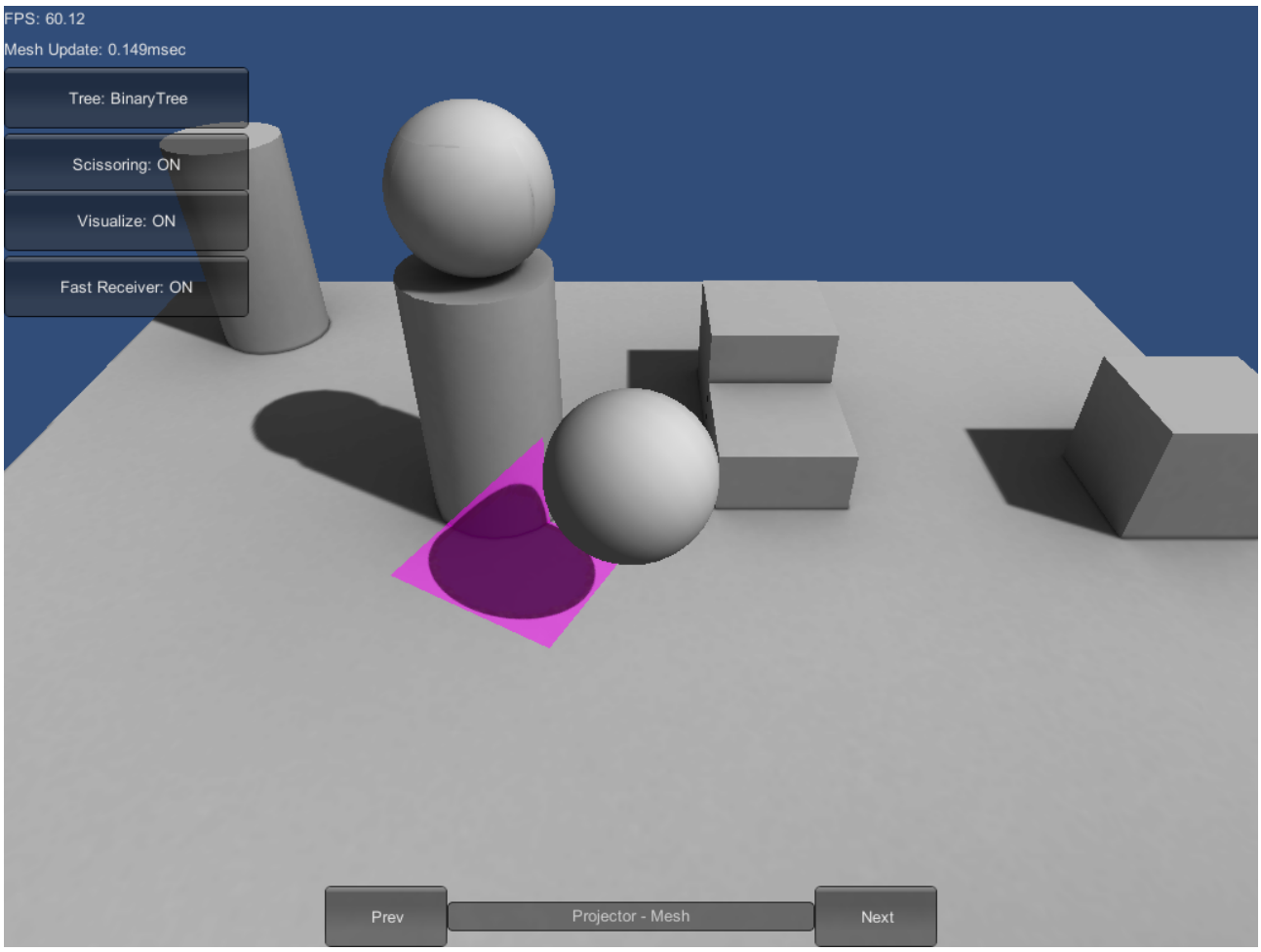
Table-2: Layer names used in Demo scenes

User Layer 8	Ignore Projector
User Layer 9	Projector
User Layer 10	Projector1
User Layer 11	Projector2
User Layer 12	Projector3
User Layer 13	Projector4
User Layer 14	ShadowReceiver
User Layer 15	ShadowReceiver1
User Layer 16	ShadowReceiver2
User Layer 17	ShadowReceiver3
User Layer 18	ShadowReceiver4

The following table shows what the content of each demo is.

Table-3: List of Demo scenes

Projector - Plane	Demonstrates how to use InfinitePlaneShadowReceiver with a Projector.
Projector - Mesh	Demonstrates how to use MeshPlaneShadowReceiver with a Projector.
Projector - Terrain	Demonstrates that MeshPlaneShadowReceiver is also available for Terrain object.
Shadowmap - Plane	Demonstrates how to use InfinitePlaneShadowReceiver with shadow-map.
Shadpwmap - Mesh	Demonstrates how to use MeshPlaneShadowReceiver with shadow-map.
MultiProjector - Plane	Demonstrates how to handle multiple projectors with InfinitePlaneShadow-Receiver.
MultiProjector - Mesh	Demonstrates how to handle multiple projectors with MeshShadowReceiver.
RandomSpawn - Plane	Demonstrates how to use Projector Manager without MeshShadowReceiver.
RandomSpawn - Mesh	Demonstrates how to use Projector Manager with MeshShadowReceiver.
SpotLights + Fake Shadows	Demonstrates how to project lights with shadows.
BulletMarks	Demonstrates how to generate bullet marks.
Multithreaded Raycast	Demonstrates how to cast rays against a Mesh Tree in multithread.



This is a screenshot of a demo scene. Only the shadow on the pink mesh is real-time shadow, others are baked shadows (Lightmap).

The demo has several buttons to switch options:

Tree: BinaryTree/Octree

It switches MeshTree between Binary Tree and Octree. The tree type displayed on the button is the used one. Only applicable when testing MeshShadowReceivers.

Scissoring: ON/OFF

It switches Scissor option on and off. Only applicable when testing MeshShadowReceivers.

Visualize: ON/OFF

If Visualize is on, the meshes which are receiving real-time shadows are drawn with pink color as in the above screenshot. (This is not null material!)

Fast Receiver: ON/OFF

It turns on and off Fast Shadow Receiver. If Visualize is on, you can see the difference.

Prev and Next

If you run “Demo” scene, you can see Prev and Next buttons to change the scenes back and forth. The text box between these buttons is showing the current scene name.

You can control the sphere object using A/W/S/D or arrow keys, and rotate the camera by click-drag left mouse button.

Performance

The following table shows the frame rates of Demo scenes, measured on some mobile devices. Each cell has two numbers, the left side indicates the frame rate when Fast Shadow Receiver is turned ON, and the right side indicates the frame rate when it is turned OFF.

	iPod touch 4th gen	iPhone 4S	Galaxy Nexus	Nexus 7 (2012)
Projector - Plane	60FPS / 55FPS	60FPS / 60FPS	60FPS / 60FPS	60FPS / 60FPS
Projector - Mesh	60FPS / 55FPS	60FPS / 60FPS	60FPS / 60FPS	60FPS / 60FPS
Projector - Terrain	11FPS / 9FPS	24FPS / 24FPS	30FPS / 27FPS	35FPS / 34FPS
Shadowmap - Plane	60FPS / 50FPS	60FPS / 60FPS	18FPS / 17FPS	N/A
Shadowmap - Mesh	53FPS / 50FPS	60FPS / 60FPS	18FPS / 17FPS	N/A
MultiProjector - Plane	60FPS / 18FPS	60FPS / 60FPS	60FPS / 50FPS	60FPS / 55FPS
MultiProjector - Mesh	40FPS / 18FPS	60FPS / 60FPS	60FPS / 50FPS	60FPS / 55FPS

The next table shows the frame rates when “Projector/Multiply” shader is used instead of “FastShadowReceiver/Projector/Multiply without Falloff” shader.

	iPod touch 4th gen	iPhone 4S	Galaxy Nexus	Nexus 7 (2012)
Projector - Plane	60FPS / 50FPS	60FPS / 60FPS	60FPS / 60FPS	60FPS / 60FPS
Projector - Mesh	60FPS / 50FPS	60FPS / 60FPS	60FPS / 60FPS	60FPS / 60FPS
Projector - Terrain	11FPS / 9FPS	24FPS / 24FPS	30FPS / 25FPS	35FPS / 33FPS
MultiProjector - Plane	60FPS / 15FPS	60FPS / 60FPS	60FPS / 45FPS	60FPS / 40FPS
MultiProjector - Mesh	40FPS / 15FPS	60FPS / 60FPS	60FPS / 45FPS	60FPS / 40FPS

As you can see, FastShadowReceiver was quite effective on iPod touch 4th gen. This device has hardware performance similar to iPhone 4. The performance differences are very obvious especially in “MultiProjector - Plane” scene and “MultiProjector - Mesh” scene.

On the other hand, we couldn’t see any performance difference on iPhone 4S, because we couldn’t measure higher frame rates than 60. If we could turn off this frame rate bounds, we might be able to see some performance differences, especially in “MultiProjector - Plane” scene and “MultiProjector - Mesh” scene. In actual game scenes which have more objects and particles, GPU will be busier and the performance differences might be visible even on this device.

On some old Android™ devices, we could also see some performance differences. If we could turn off the frame rate bounds, the difference might have been bigger than in the table.