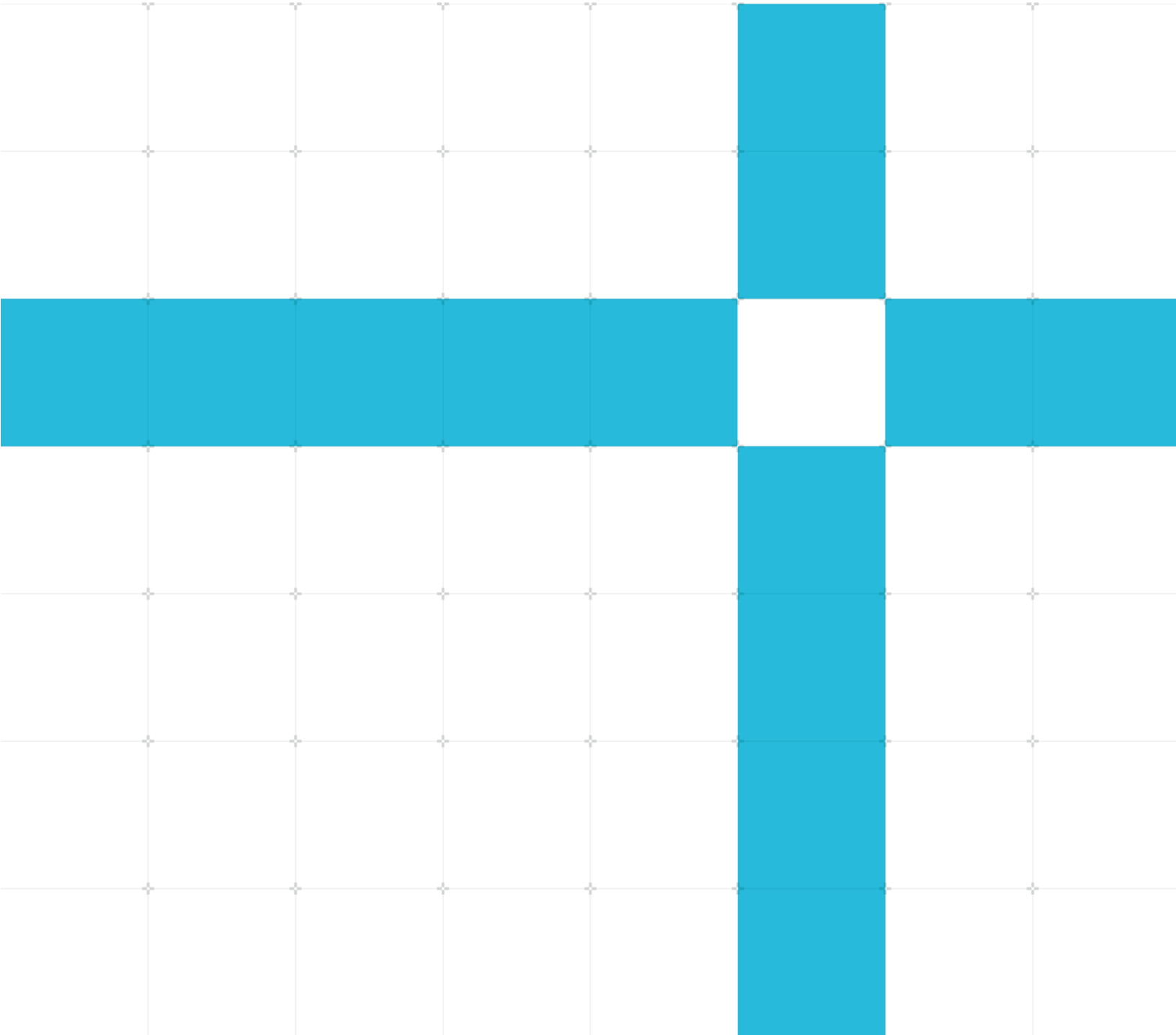
arm

Key project-wide settings in Unity

Non-Confidential

Issue 02
102312



Key project-wide settings in Unity

Release information

Document history

Issue	Date	Confidentiality	Change
01	12 November 2020	Non-Confidential	First release
02	25 January 2021	Non-Confidential	Player settings addition

Non-Confidential Proprietary Notice

translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Web Address

www.arm.com

Contents

1 Overview

This guide explains the key options that Unity provides so that you can balance image quality and performance for your application.

Unity has several options that affect the image quality of your game. Some of these options have a high computational cost and can have a negative impact on the performance of your game. Other options can increase the image quality of your game with only a small trade-off in performance.

For example, if the frame rate of your game is low, the GPU might be processing too much information when performing a complex graphical effect. You can perform less complex versions of graphical effects, like shadows and lighting, for a relatively small impact on the graphical quality. Simpler effects can reduce the load on the GPU significantly, providing a higher frame rate.

The default settings for lighting are sometimes too complex for a mobile device. This means that some games that are written for mobile platforms avoid complex techniques or use game-specific techniques. These techniques include pre-baking lighting into light maps or projecting textures instead of casting shadows.

1.1 Before you begin

This guide was last updated against Unity 2019.3.

This guide refers to the Universal Render Pipeline (URP). In previous versions of Unity this was called the Lightweight Render Pipeline.

When creating a new project, choose the URP template to set up your project.

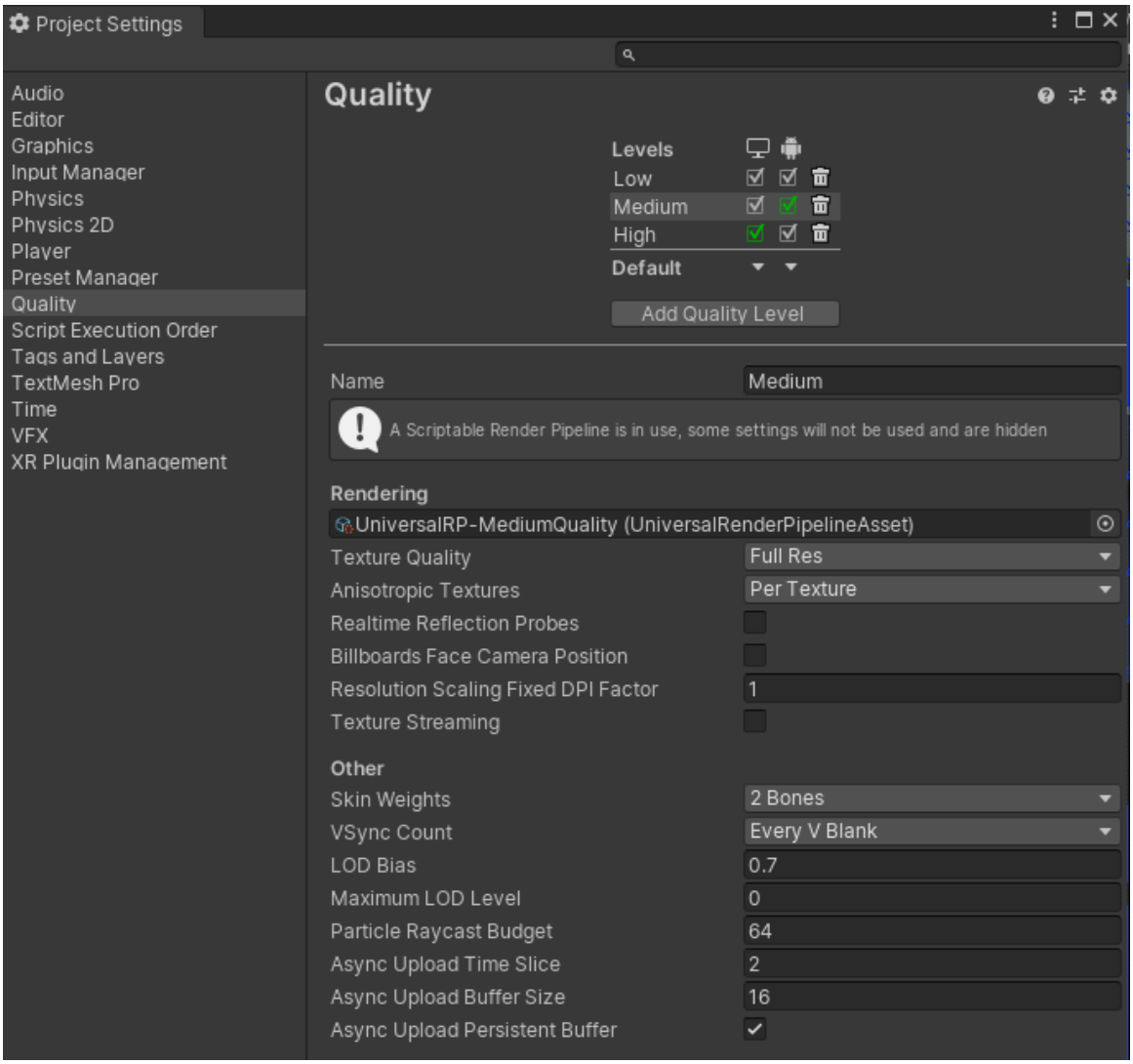
2 Project settings: quality

This section of the guide explains the quality settings that Unity provides so that you can select the correct settings for your application.

To see the Unity quality settings, follow these steps:

1. Click **Edit** > **Project Settings** on the Unity main menu to display the **Project Settings** dialog.
2. Click **Quality** in the category list on the left.

The details pane on the right shows the quality settings, as shown in the following image:



The Unity Project Settings dialog, Quality category

When using the Universal Render Pipeline (URP) template, some settings that would otherwise appear in the Quality category move to the URP asset.

The following options in the Quality category can have a large impact on the performance of your game:

Texture Quality

Setting texture quality to a higher resolution can load the GPU but typically does not cause performance problems. Reducing texture quality can negatively impact the visual quality of your games, so only reduce the quality if you must. In the **Ice Cave demo**, **Texture Quality** is set to full resolution.

If textures are causing performance problems, try using **mipmapping**. Mipmapping reduces compute and bandwidth requirements without impacting image quality.

Anisotropic Textures

The **Anisotropic Textures** option is a technique that removes distortion from textures that are drawn at high gradients. This technique improves image quality but is computationally expensive.

Avoid using the **Anisotropic Textures** option unless distortion is especially noticeable.

Realtime Reflection Probes

The **Realtime Reflection Probes** option can have a significant negative impact on runtime performance.

When a reflection probe is rendered, every face of the cubemap is calculated separately by a camera at the origin of the probe. If inter-reflections are considered, this process happens for every reflection bounce level. For glossy reflections, the cubemap mipmaps are also used to apply a blurring process.

The following factors influence rendering of reflection probes with cubemaps:

- Cubemap resolution
  - Higher resolution cubemaps increase rendering time. Use the lowest resolution cubemap possible for the quality that you require.
- Culling mask
  - Use the culling mask when rendering the cubemap to avoid rendering any geometry that is not relevant in the reflections.
- Cubemap update frequency

  The **Refresh Mode** option defines the update frequency for a cubemap. This option has the following settings:

- **Every Frame** renders the cubemap every frame. This option is the most computationally expensive mode, so avoid using it unless you require it.
- **On Awake** renders the cubemap at runtime only once, when the scene starts.
- **Via Scripting** lets you control when the cubemap is updated. With this mode, you can limit the use of runtime resources by specifying the conditions when an update takes place.

Skin Weights, **LOD Bias**, **Particle Raycast Budget**

These settings are all options that you should consider adjusting to balance quality against performance. Please refer to the **Unity Documentation** for more information.

3 Project settings: graphics

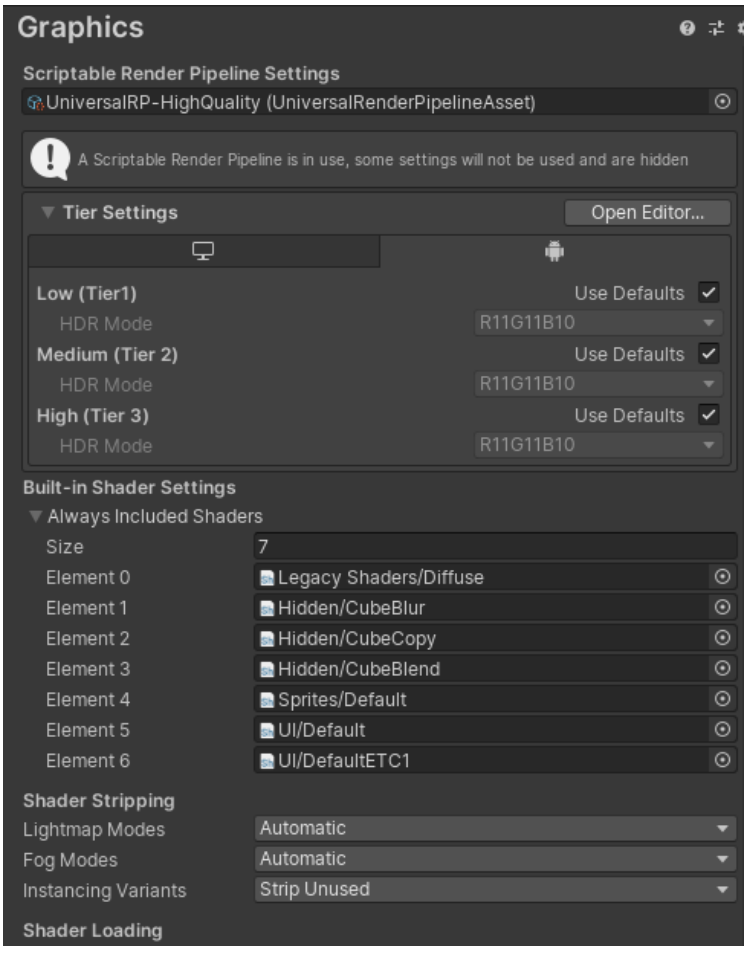
This section of the guide explains the graphics settings that Unity provides so that you can select the correct settings for your application.

Unity has several options that affect the image quality of your game. Some of these options have a high computational cost and can have a negative impact on the performance of your game. Other options can increase the image quality of your game with only a small trade-off in performance.

To see the Unity graphics settings, follow these steps:

1. Click **Edit** > **Project Settings** on the Unity main menu to display the **Project Settings** dialog.
2. Click **Quality** in the category list on the left.

   The details pane on the right shows the quality settings, as shown in the following image:



The Unity Project Settings dialog, Graphics category

3. If you created your project as a URP project, the URP asset appears in the **Scriptable Render Pipeline Settings** field.

   Otherwise, you can **create and add a URP asset** to put into the **Scriptable Render Pipeline Settings**.

The following setting on the **Graphics Project Settings** dialog can have a large impact on the performance of your game:

HDR Mode

Under the **Android** settings, **HDR Mode** is set to R11G11B10 by default. This mode is usually an appropriate choice because colors will fit in 32 bits instead of the 64 bits of FP16, halving bandwidth.

Consider carefully whether you need the extra color resolution before changing this setting to a higher mode.

4 Project settings: player

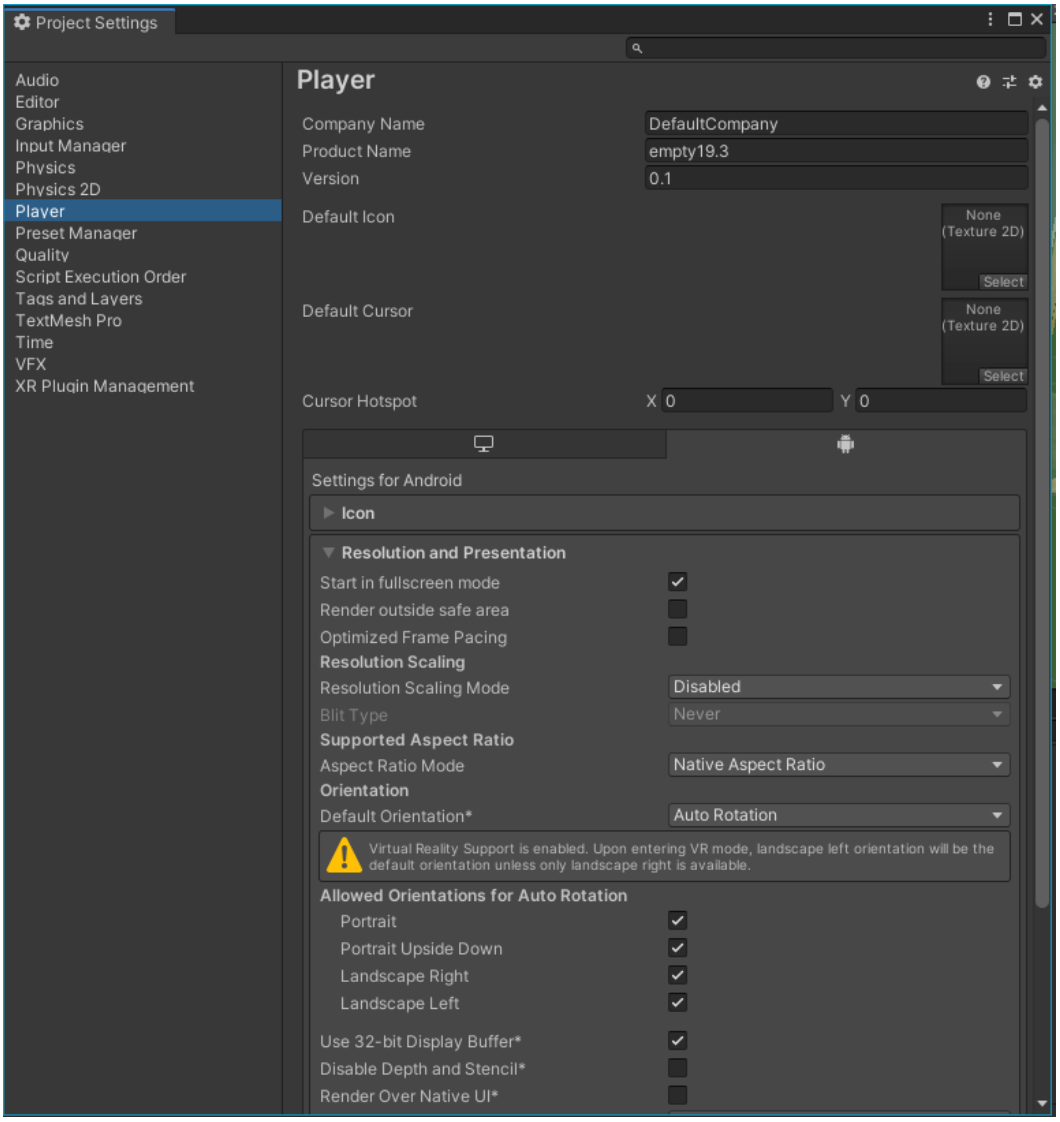
This section of the guide explains the player settings that Unity provides so that you can select the correct settings for your application.

To see the Unity player settings, follow these steps:

1. Click **Edit** > **Project Settings** on the Unity main menu to display the **Project Settings** dialog.

2. Click **Player** in the category list on the left.

The details pane on the right shows the quality settings, as shown in the following image:



The Unity Project Settings dialog, Player category

Blit Type

Blitting is the process of copying graphical data from a frame buffer and displaying it on screen. **Blit Type** provides three available settings which control whether graphical data renders directly to the system frame buffer or renders to an offscreen buffer first. These settings are Auto, Never and Always.

To reduce requirements on the GPU, select the settings Auto or Never.

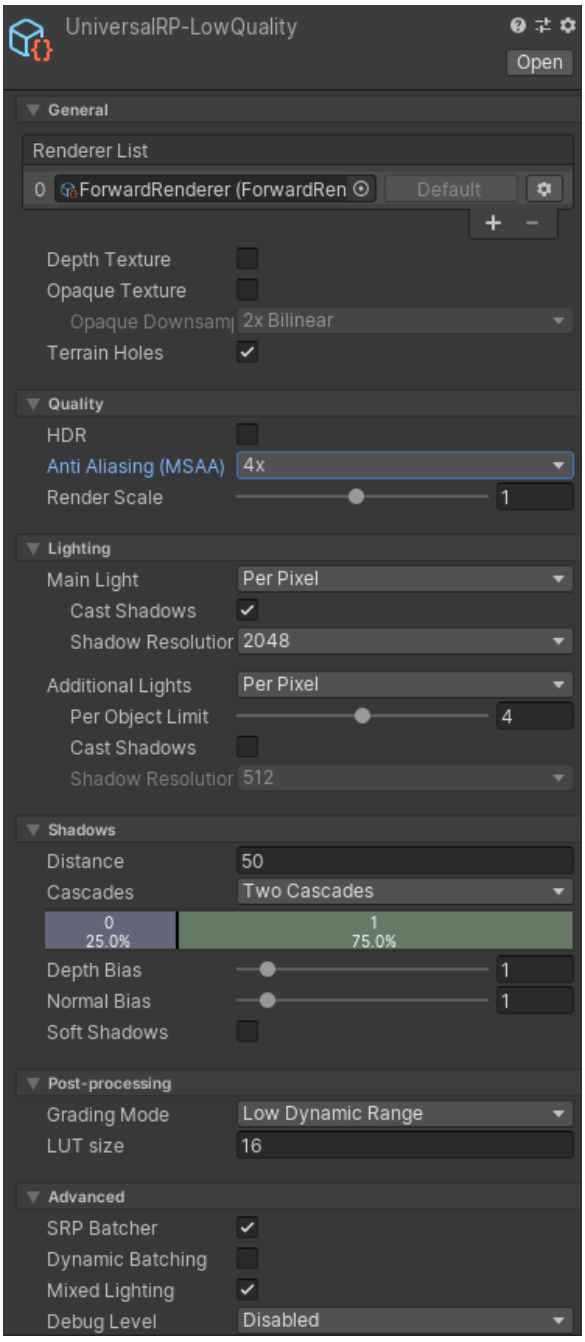
5 URP asset settings

This section of the guide explains the settings that are associated with the Universal Rendering Pipeline (URP) asset. The URP asset affects the image quality and performance of your game.

To see the URP asset settings, follow these steps:

1. Select your URP asset in the **Project** view.

   The **Inspector** view shows the settings associated with your URP asset, as shown in the following image:



Inspector window showing URP asset settings

The following URP asset settings can have a large impact on the performance of your game:

AntiAliasing (MSAA)

Anti-aliasing is an edge-smoothing technique that blends the pixels around triangle edges. This technique provides a noticeable improvement to the visual quality of your game. There are several anti-aliasing methods, for example Multi-Sampled Anti-Aliasing (MSAA). Setting

this option to **4x MSAA** is very low-cost operation on Mali GPUs, so you should use this setting whenever possible.

Lighting

Generally, the **Per Pixel** setting, rather than the **Per Vertex** setting, gives the required quality.

The **Additional Lights** option specifies the number of lights that can affect a given pixel. A high light count per pixel requires many calculations. Most games can use very few dynamic and real-time lights with minimal impact on image quality. Consider using techniques like light maps and projected textures in your game if lighting is causing performance problems.

The **Cast Shadows** and **Shadow Resolution** options also add to the performance cost, so these settings should be considered carefully.

Shadows

High-quality shadows can be computationally intensive. If shadows cause performance problems, try simple shadows, or switch them off. If shadows are important in your game, consider using simple dynamic shadowing techniques like projected textures.

The **Distance** option lets you reduce load by limiting shadows to objects that are close to the camera.

The **Cascades** option lets you balance quality and processing time. A higher number of cascades produces better quality but increases the processing overhead.

The **Soft Shadows** option adds a smoothing filter to the shadow map. This option has a computational cost, so consider whether this smoothing is necessary.

Dynamic Batching

Unity performs dynamic batching transparently, but the computational overhead becomes too large for objects that contain many vertices. Apply static batching to objects that do not move during rendering.

Soft Particles

When using the URP asset, you can turn on soft particles by enabling the **Depth Texture** option and altering the settings under individual particle materials. Soft particles will then be rendered to the depth texture.

Using soft particles increases the load on the GPU, but is sometimes worth the cost to achieve realistic visuals on your particles. On mobile platforms, rendering to depth textures uses valuable bandwidth.

Future versions of the Unity URP will introduce a deferred renderer that you can use to achieve soft particles. However, deferred rendering means that you cannot access MSAA. Consider whether soft particles are important enough to your game to use them.

6 Related information

Here are some resources that are related to material in this guide:

- **Arm Guide for Unity Developers**
- **Arm Guide for Unity Developers Optimizing Mobile Gaming Graphics**
- **Ice cave demo video**
- **Unity at Arm**
- **Unity User Manual**

7 Next steps

This guide introduced the key Unity options that let you balance image quality and performance for your application.

You can now investigate other performance optimizations.

You can use evaluation tools like **profilers and graphics debuggers** to test how your game performs on a mobile device.

Other optimization areas that you can investigate include **application processor optimizations**, **GPU optimizations**, and **asset optimizations**.