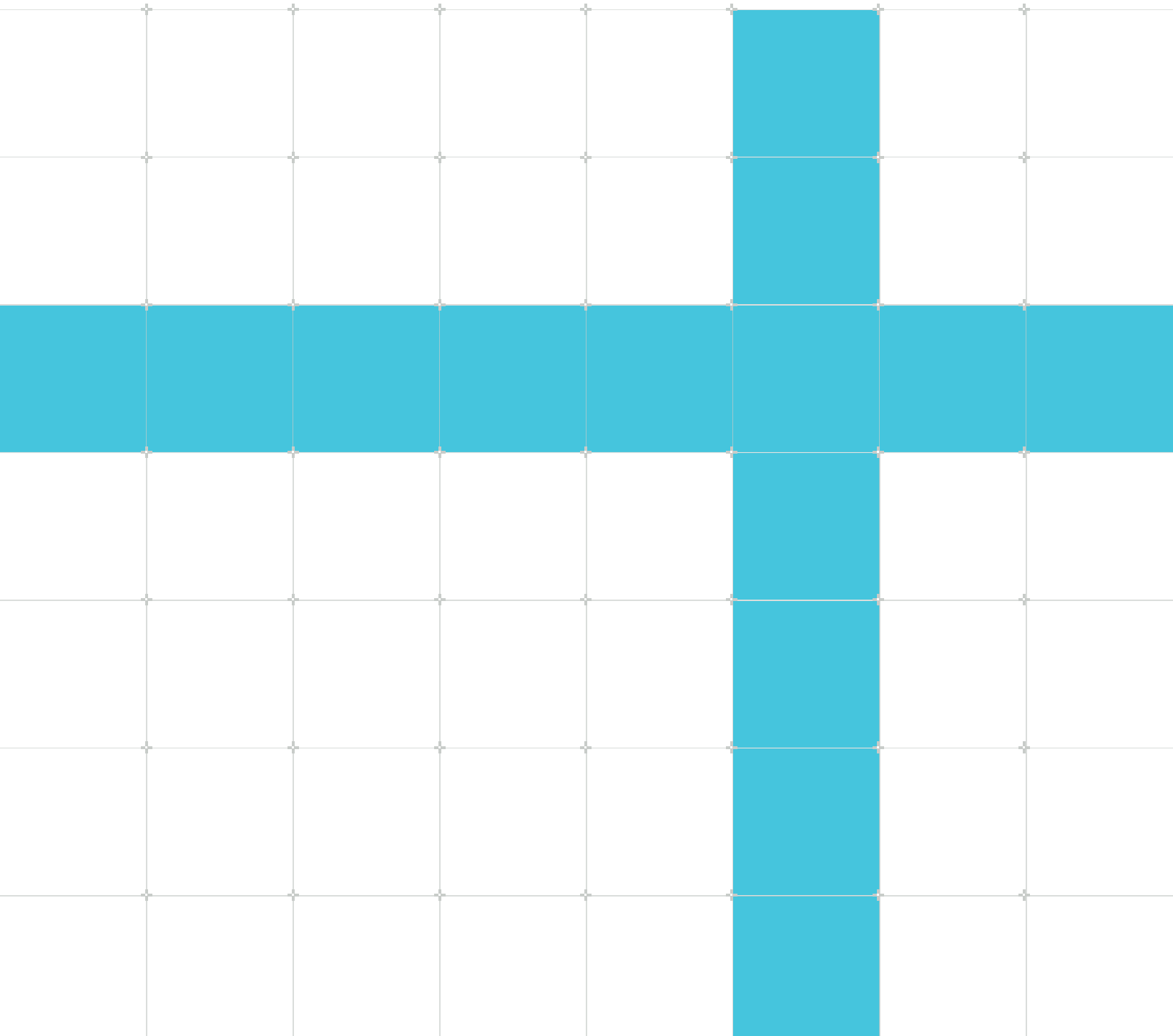
Real-time 3D Art Best Practices - Texturing

Version 2.0

guide

Non-Confidential
Copyright © 2020, 2023 Arm Limited (or its affiliates).
All rights reserved.

Issue 01
102449_0200_01_en



Real-time 3D Art Best Practices - Texturing guide

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	1 March 2020	Non-Confidential	First release
0200-01	1 August 2023	Non-Confidential	Merged with document 102539

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be 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.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at https://www.arm.com/company/policies/trademarks.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

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.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

1. Overview....6
2. Texture atlasing....7
3. Texture filtering....8
4. Mipmapping....12
5. Texture size, color space, and compression....14
6. UV unwrapping....18
7. Visual impact....20
8. Texture channel packing....23
9. Alpha channels and normal map best practices....25
10. Normal map best practice....27
11. Normal map baking best practice....28
12. Editing texture settings....31
13. Check your knowledge....32
14. Related information....33
15. Next steps....34

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Overview

Texture mapping takes a 2D surface image and maps it onto a 3D polygon.

This guide covers several texture optimizations that can help your games to run more smoothly and look better.

At the end of this guide you can Check your knowledge. You will have learn:

- Best practices for texturing targeting mobile devices.
- What is texture atlasing.
- How mipmapping is used.
- Normal map best practices.

This guide is also available in the format of a Unity Learn Course - Arm & Unity Presents: 3D Art Optimization for Mobile Applications.

- Arm Guide for Unity Developers: Korean
- Arm Guide for Unity Developers: Chinese
- Arm Guide for Unity Developers: Japanese

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Texture atlasing

A texture atlas is an image that contains data from several smaller images that have been packed together. Instead of having one texture for one mesh, you have a larger texture that several meshes share.

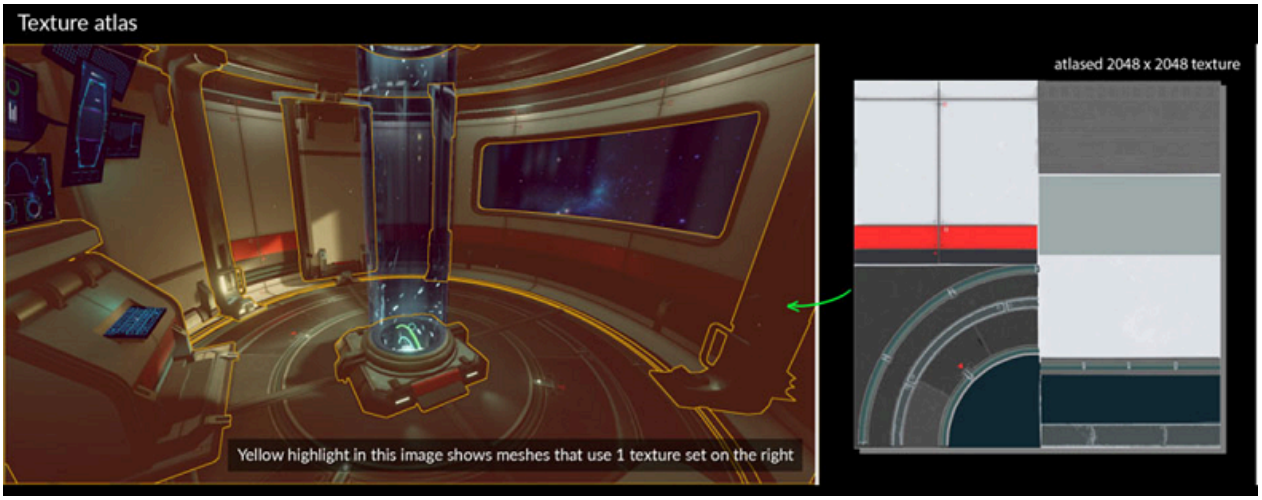
A texture atlas can be created before making the asset, which means that the asset is UV unwrapped according to the texture atlas. This requires some early planning when creating the texture.

The texture atlas can also be created after the asset is finished by merging textures in painting software. However, this also means that the UV islands must be rearranged according to the texture.

A UV island is a connected group of polygons in a texture map.

The following image highlights several 3D objects using one texture set:

Figure 2-1: 3D objects using one texture set



Texture atlasing enables batching for several static objects that share this texture atlas and the same material. Batching reduces the number of draw calls. Fewer draw calls results in better performance when the game is CPU-bound.

The Unity game engine has a feature that does batching when objects are marked static. This is done without having to manually merge the objects. Further information on this can be found on Unity's website.

Texture atlasing also requires fewer textures inside the game as they are packed together. With compression, this helps to keep the memory cost of textures down.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. Texture filtering

Texture filtering is a method that is used to improve the texture quality in a scene. Without texture filtering, artifacts like aliasing generally look worse.

Texture filtering makes textures look better and less blocky. Usually, this makes the game look better.

However, texture filtering can also degrade performance. This is because better quality often means that more processing is required. Texture filtering can account for up to half of the GPU energy consumption. Choosing simpler texture filters can reduce the energy demands of an application.

There are several options available in Unity for texture filtering:

- **Nearest, or Point filtering**

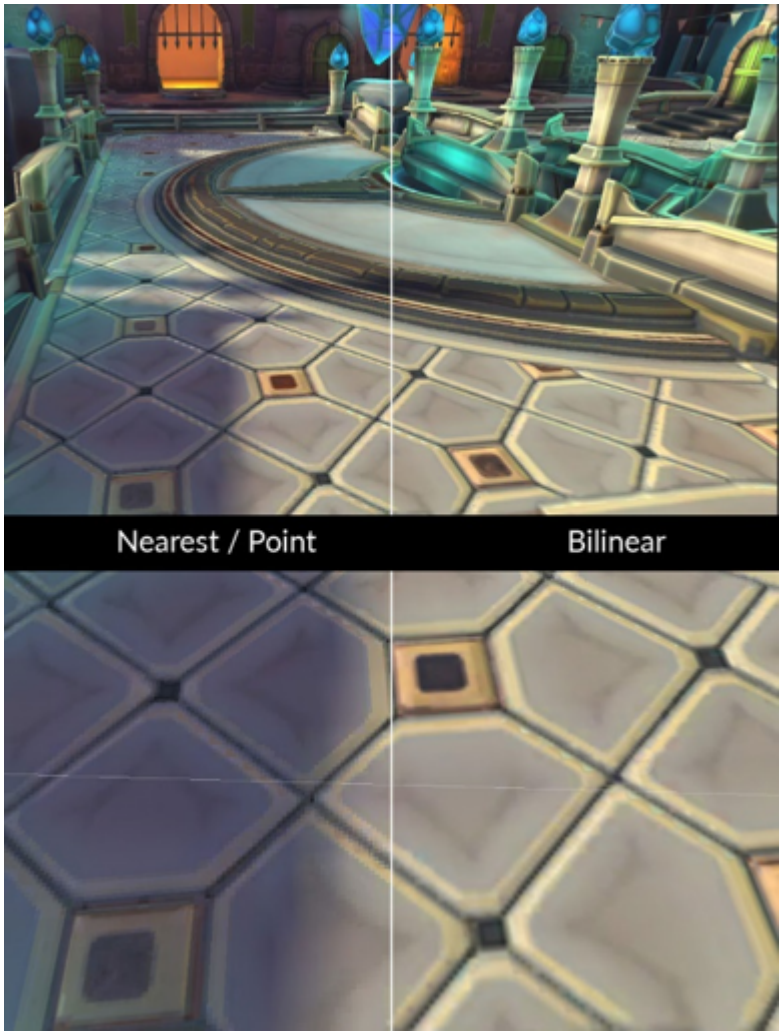
  When seen up close, nearest filtering makes the texture looks blocky. This is the simplest and cheapest texture filtering.

- **Bilinear filtering**

  The texture is blurrier up close with bilinear filtering. The four nearest texels are sampled and then averaged to color the main pixel. Unlike nearest filtering, bilinear filtering results in less blocky pixels as the pixels have a smooth gradient, as shown in the following image:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-1: Bilinear filtering

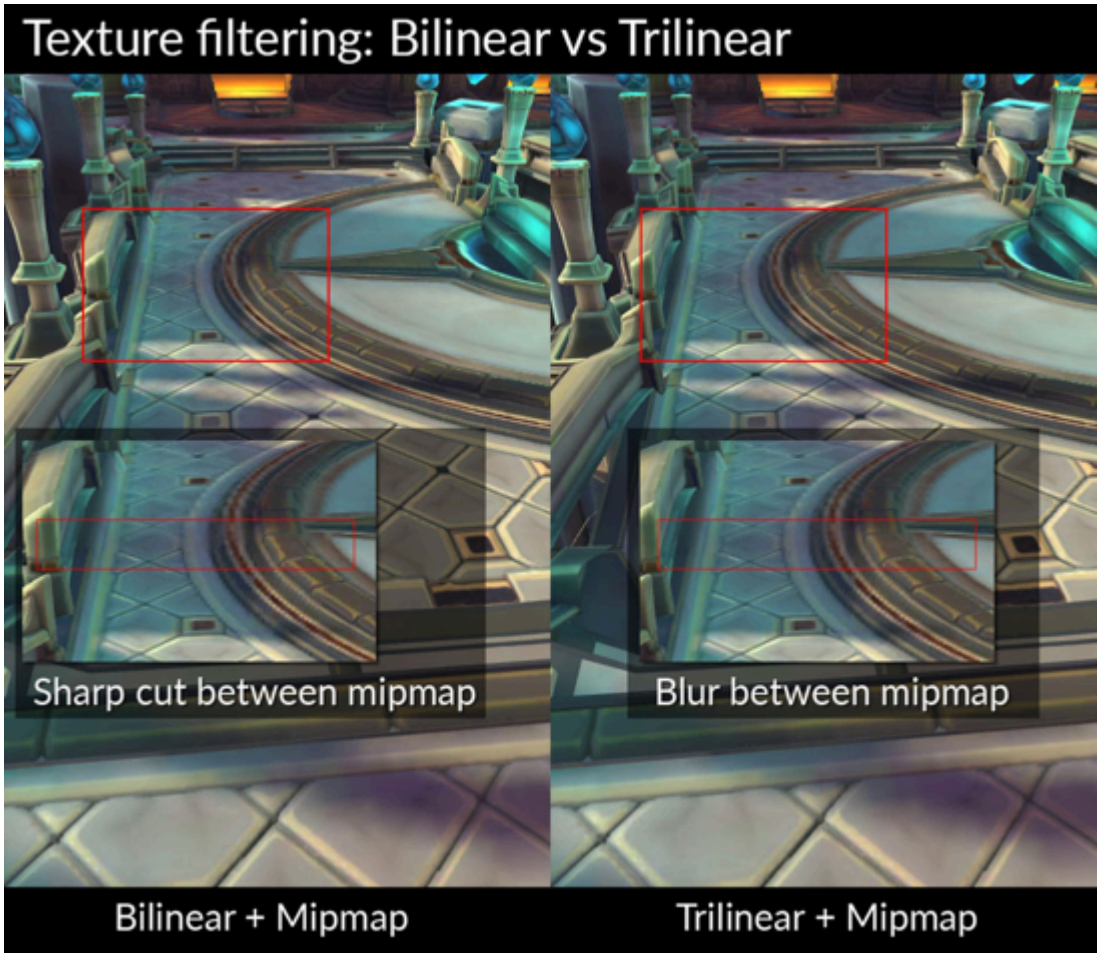


- **Trilinear filtering**

  Trilinear filtering is like bilinear filtering, but with added blend between mipmap levels. Trilinear filtering removes noticeable changes between mipmaps by smoothing the transition between mipmaps, as shown in the following image:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-2: Trilinear filtering



Note

Bilinear and trilinear filtering requires sampling more pixels and, therefore, more computation.

- **Anisotropic filtering**

  Anisotropic filtering makes textures look better when viewed at an angle. It is good for ground level textures, for example, as shown in the following image:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-3: Anisotropic filtering



Texture filtering best practices

Arm recommends that you try the following texture filtering tips:

- Use bilinear filtering for a balance between performance and visual quality.
- Use trilinear filtering selectively. This is because trilinear filtering requires more memory bandwidth than bilinear filtering.
- Use bilinear and 2x anisotropic filtering instead of trilinear and 1x anisotropic. This is because this combination of filtering techniques can both look and perform better.
- Keep the anisotropic level low. Only use a level higher than two for critical game assets.

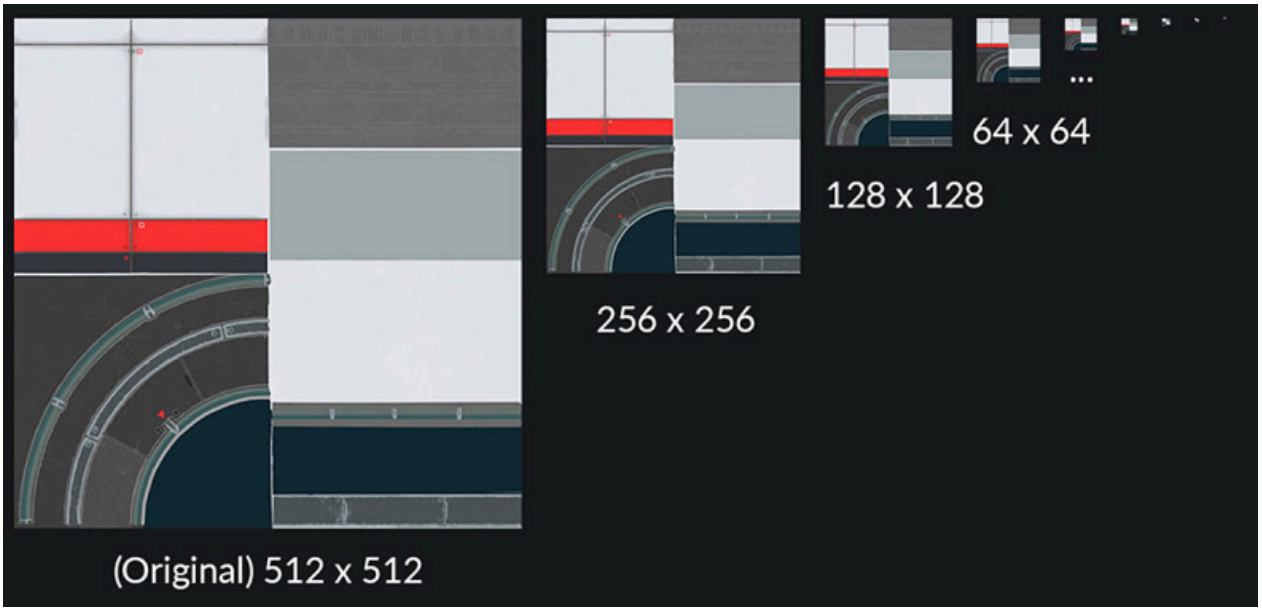
Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. Mipmapping

Mipmaps are copies of the original texture that are saved at lower resolutions. You can think of mipmapping like the equivalent of Level of Detail (LOD), but for textures.

Based on how much texture-space a fragment occupies, you can select an appropriate level for sampling. When an object is further from the camera, a lower resolution texture is applied. A higher resolution texture is applied when an object is closer the camera. The following image shows a mipmap collection containing the same texture at different resolutions:

Figure 4-1: Mipmapping



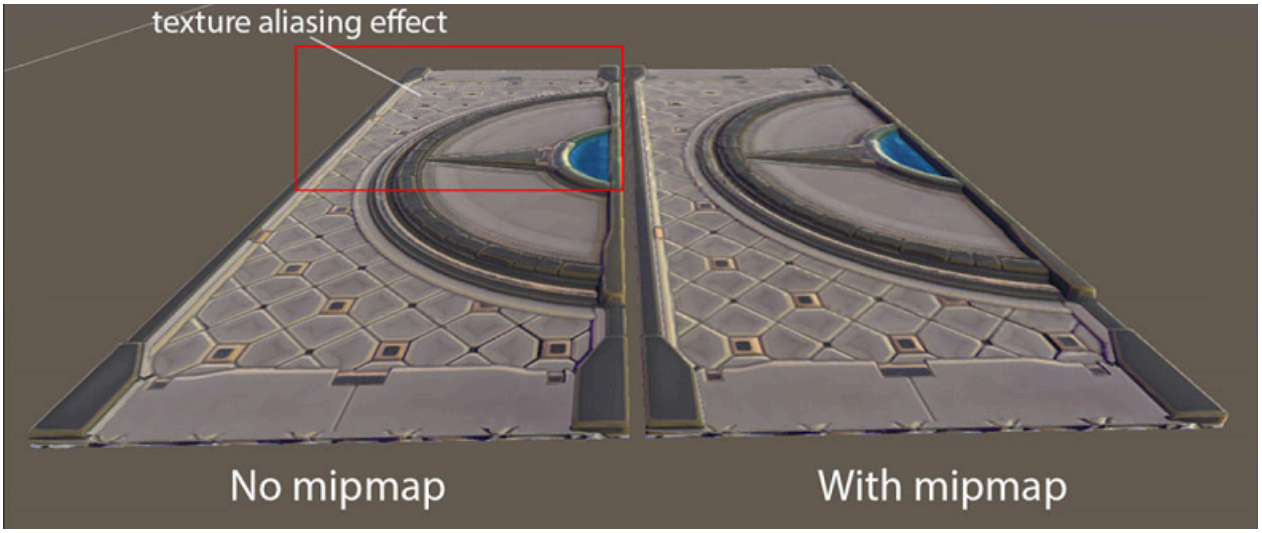
Best practices

Make sure that you use mipmapping. This is because mipmapping improves both performance and quality. Mipmapping improves the performance of the GPU. This is because the GPU does not need to render the full resolution texture on an object that is far away from the camera.

Mipmapping also reduces the texture aliasing and improves the final image quality. Texture aliasing causes a flickering effect on areas that are further from the camera. The following image shows texture aliasing on the same texture both with and without mipmapping:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-2: Texture aliasing with and without mipmapping



Unity automatically creates mipmaps as they are needed on import. Unity then rescales textures that are not power of two. More information on this can be found on Unity's website.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. Texture size, color space, and compression

Texture size, color space, and compression can all have an impact on the performance of your game.

Only create textures that are large enough to meet the required quality, but no bigger. It is also best practice to use a large-size texture atlas that contains several textures that are shared between many meshes.

Texture size

Textures can be different sizes. Reducing the size of certain textures that require less detail helps to reduce bandwidth levels. For example, the diffuse texture can be set to 1024x1024 pixels and the roughness, or metallic, map to 512x512 pixels.

Selectively reduce the texture size and check whether any of the visuals have been degraded afterwards.

Texture color space

Most texturing software, for example Adobe Photoshop or Substance Painter, works and exports using the sRGB color space.

We recommend that you try the following:

- Use diffuse textures in the sRGB color space.
- Textures that are not processed as color must not be in the sRGB color space. Examples include metallic, roughness, and normal maps. This is because of the following reasons:
  - Maps are used as data, and are not used as color.
  - Using sRGB in these maps results in the wrong look, or visual, on the material.

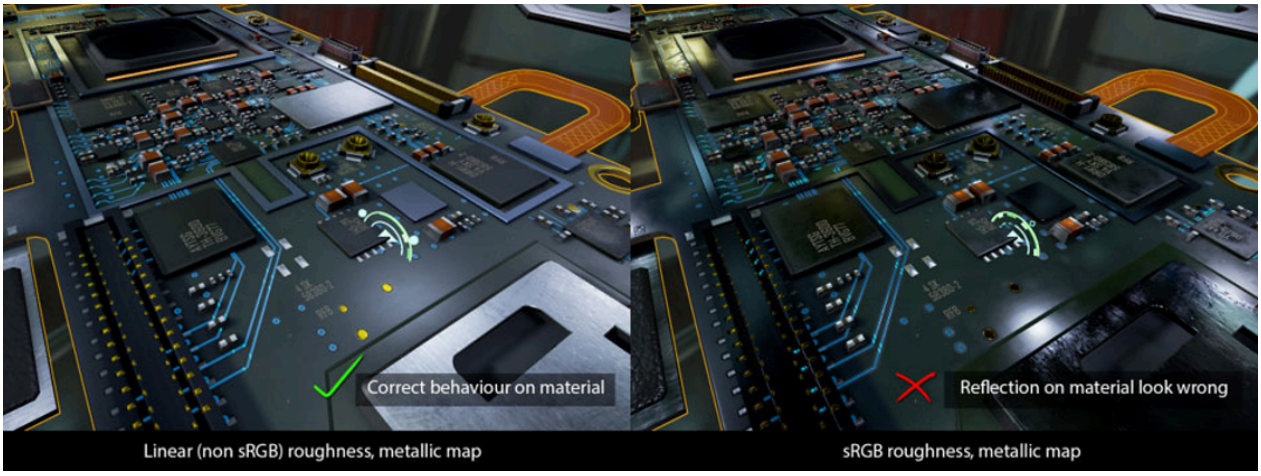


The sRGB (Color Texture) setting in the Inspector window for the texture must not be ticked for roughness, specular, normal maps, or similar.

The following screenshot shows you what happens when sRGB is incorrectly applied to such a texture:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 5-1: Incorrectly applied sRGB



Texture compression

Texture compression is an image compression that is applied to reduce texture data size, while minimizing the loss in visual quality. In development, we export textures using a common format, like TGA, or PNG. These formats are more convenient to use, and major image software programs support them.

These formats must not be used in final rendering. This is because they are slower to both access and sample, when compared to specialized image formats. For Android, there are several options, like Adaptive Scalable Texture Compression (ASTC), Ericsson Texture Compression (ETC) 1, or ETC2.

Texture compression best practices

We recommend that you use the ASTC technology that Arm created. Here several reasons to use ASTC:

- ASTC gives better quality with the same memory size as ETC.
- ASTC produces the same quality with a smaller memory size than ETC.
- ASTC takes longer to encode than ETC. This means that game packaging process can take more time. If this is an issue, then it is better to use ASTC on the final packaging of the game.
- ASTC allows for more control in terms of quality, because ASTC allows the block size to be set. While there is no single best default for block size, setting the block size to 5x5 or 6x6 pixels is a good starting point.

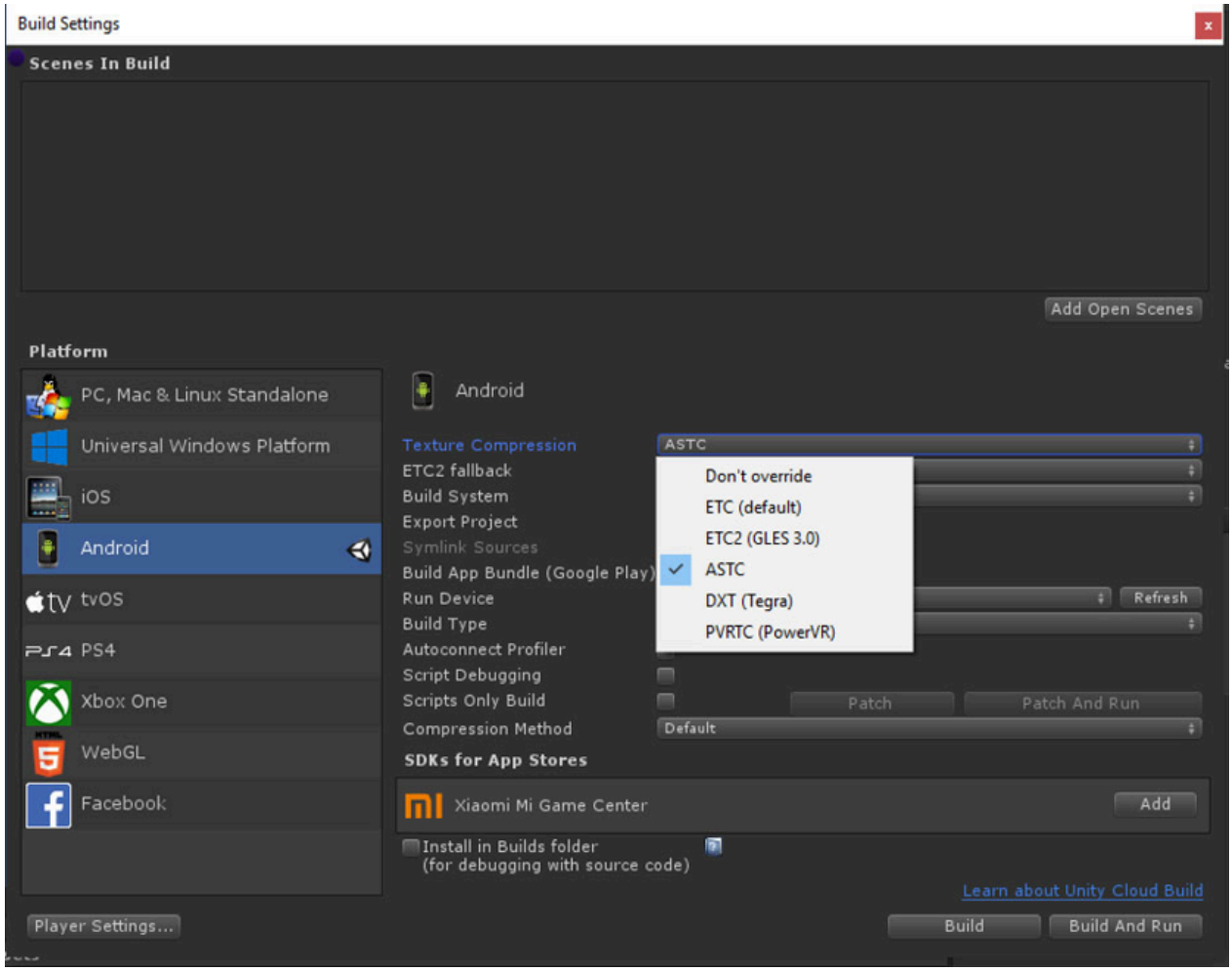
Sometimes, it might be faster to use ETC for development if you have to quickly deploy your game on a device. You can use ASTC with fast compression settings to get around the increase in deployment time. When encoding, there are options to trade off speed against both quality and size. For the final build, ASTC is the best option in terms of balance between visual quality and file size.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The game engine handles the texture compression when you package the game. However, you can choose which compression technique to use. You must choose the format to use, so it is difficult to skip this step.

The following screenshot shows how to select ASTC when building an Android package in Unity:

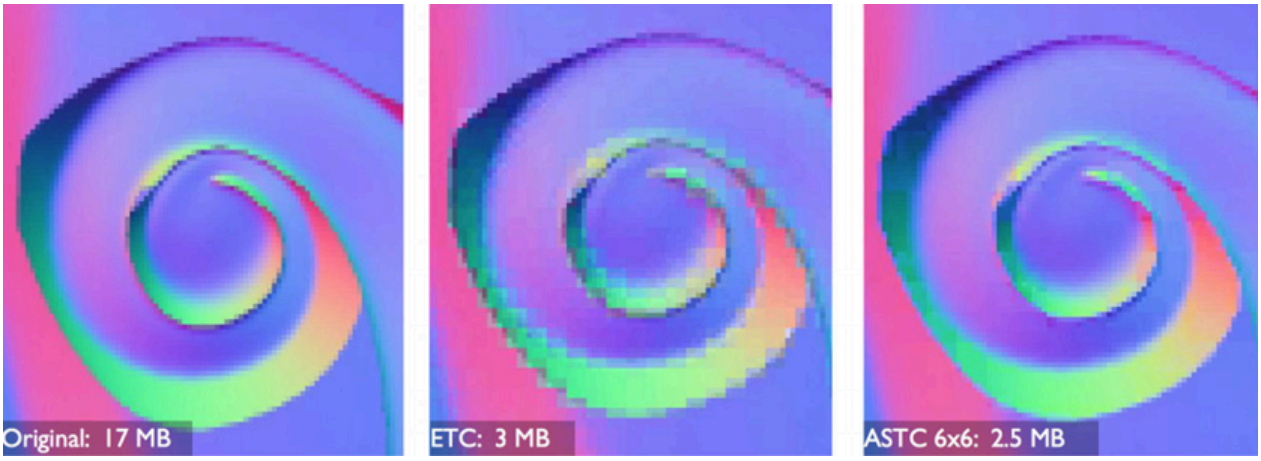
Figure 5-2: Selecting ASTC



The following image shows both the difference in quality, and the respective file size between ETC and ASTC compression:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 5-3: File size between ETC and ASTC compression



Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

6. UV unwrapping

UV unwrapping is the process of creating a UV map. A UV map projects 2D textures onto the surface of a 3D model.

It is best practice to keep UV islands as straight as possible.

Note A UV island is a connected group of polygons in a texture map.

The reasons for keeping UV islands as straight as possible are:

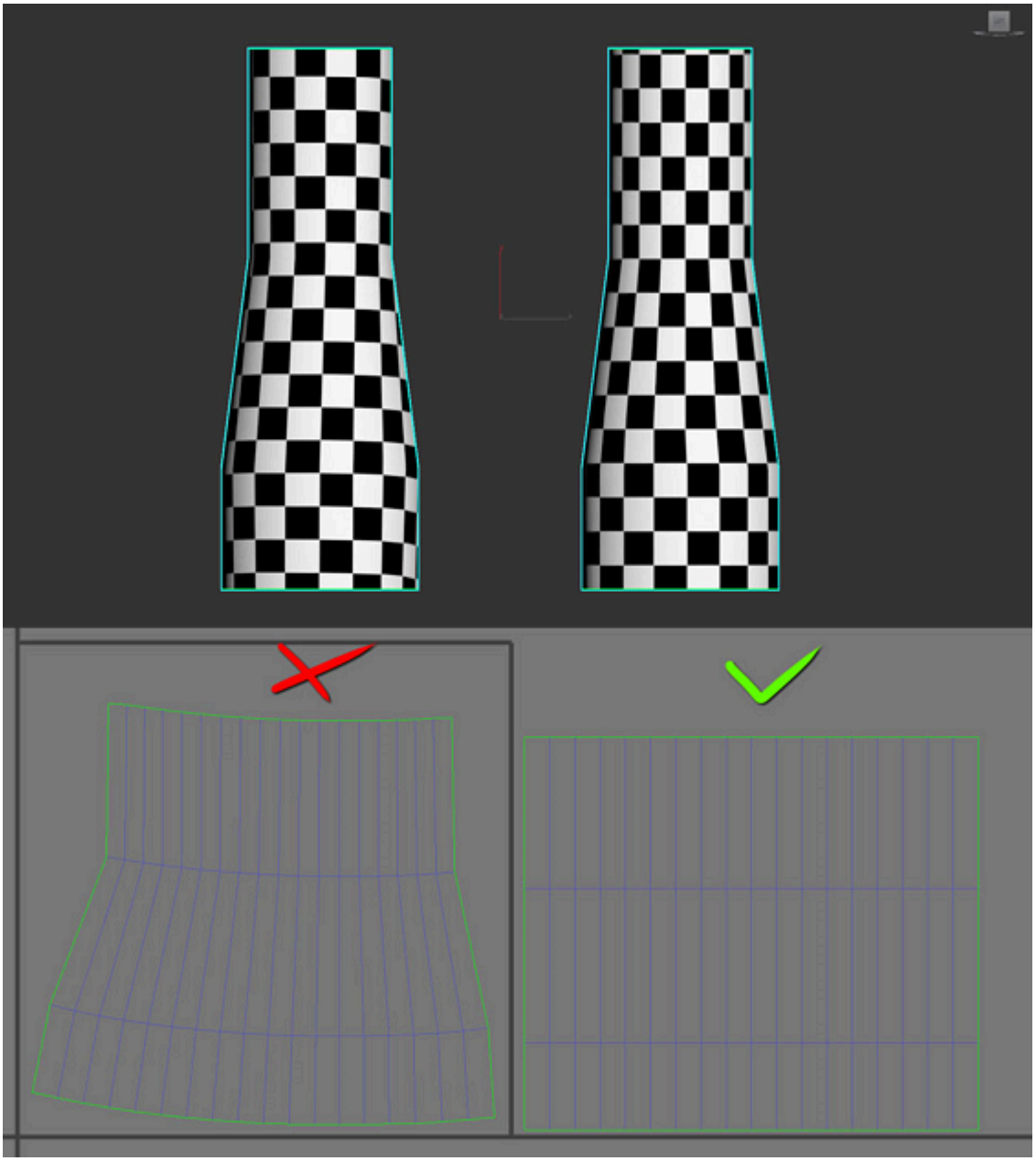
- It makes packing UV islands easier and less space is wasted.
- A straight UV helps to reduce the staircase effect happening on textures.
- On mobile platforms, texture space is limited. This is because the texture size is usually smaller on a mobile platform than on a games console or a PC. Good UV packing ensures that you get the best resolution from your texture.
- You might consider having a slightly distorted UV by keeping the UV straight, in order to have better quality texture overall.

Place UV seams in places that make them not too visible, for visual quality purposes. This is because the texture seam can look bad on a model. Therefore, split UV islands where the edges are sharp and have a small space between the UV islands. This helps later on to create better normal maps through the baking process.

The following image shows an example of how to use UV unwrapping to maximize texture space:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 6-1: Maximizing texture space with UV unwrapping



Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

7. Visual impact

The detail you include when creating textures should be proportionate to the visual impact of that detail. Make sure that you only create details that can be seen.

Phone screens are small, therefore fine-grained detail is not visible. Take this into account when creating textures. For example, you do not need a 4K texture with lots of details for a chair that is barely visible in the corner of the room.

The following screenshot shows an example. The small 256x256 pixel texture on the left has a low level of detail to be used on soldier characters. The larger image on the right shows the entire scene, and shows that the low level of texture detail is sufficient for the required amount of detail:

Figure 7-1: Example screenshot



In certain cases, you need to exaggerate and highlight edges and shading to improve shape readability. Because mobile platforms generally use smaller textures, it might be hard to capture all of the detail that is needed within this small texture.

Use fewer textures and bake in any extra details into one texture. This is important because:

- Phone screens are small, and some details are better to be baked onto the diffuse texture itself to make sure that those details are visible.
- Elements like ambient occlusion and small specular highlights can be baked in and then added to the diffuse texture.

This approach means that you do not have to rely too much on shader and engine features to get specular and ambient occlusion.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The following screenshot shows an example of details that have been baked into a texture, for example the highlights on roof tiles:

Figure 7-2: Example screenshot



When possible, use grayscale textures that allow color tinting in the shader. This saves texture memory at the cost of creating a custom shader to perform the tinting.

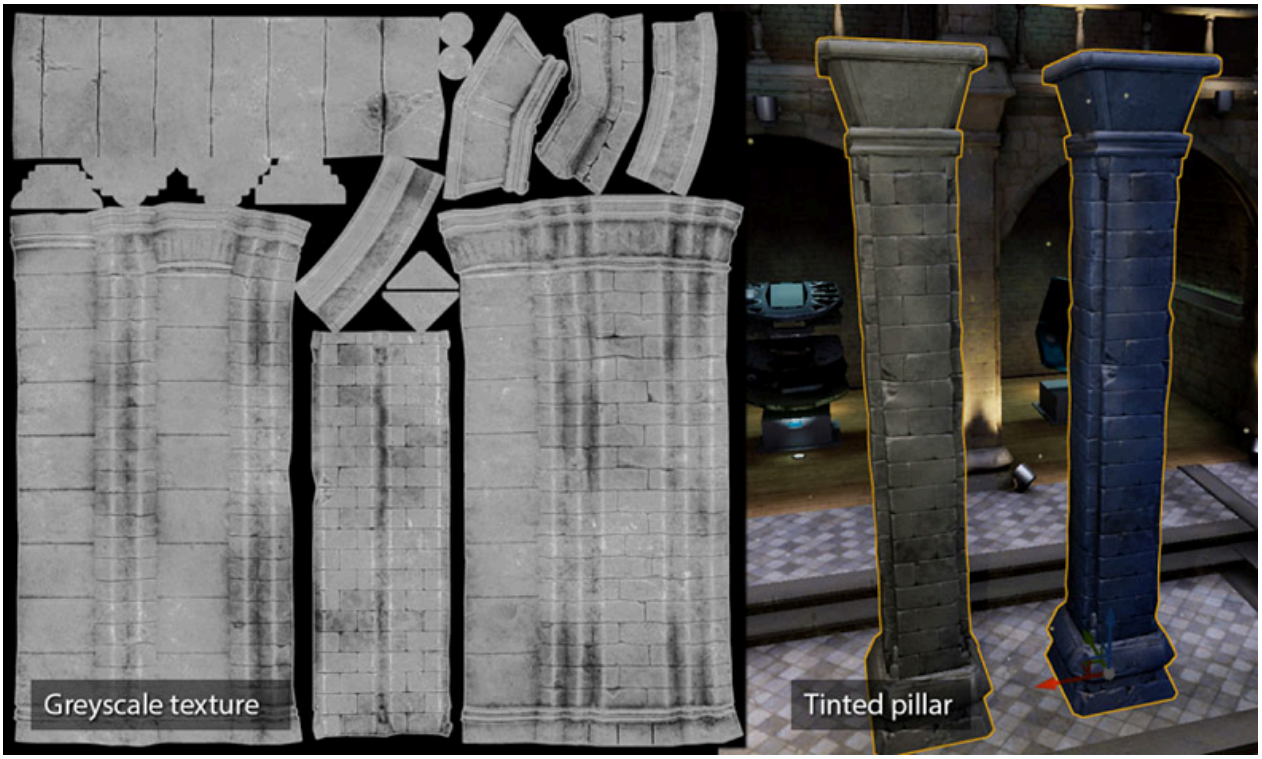
Be selective with this technique, because not all objects look good using this method. It is easier to apply this technique to an object that has a uniform, or similar, color.

You can also use RGB masks and then apply textures that are based on the color range of the mask to achieve this effect.

The following image shows an example of a grayscale texture that is being applied to a tinted pillar:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 7-3: Greyscale texture example



Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

8. Texture channel packing

Use texture channels to pack multiple textures into one.

Arm recommends texture channel packing because packing helps to save texture memory. Packing saves memory because you can get three maps into one texture using this technique. This means that fewer texture samplers are required.

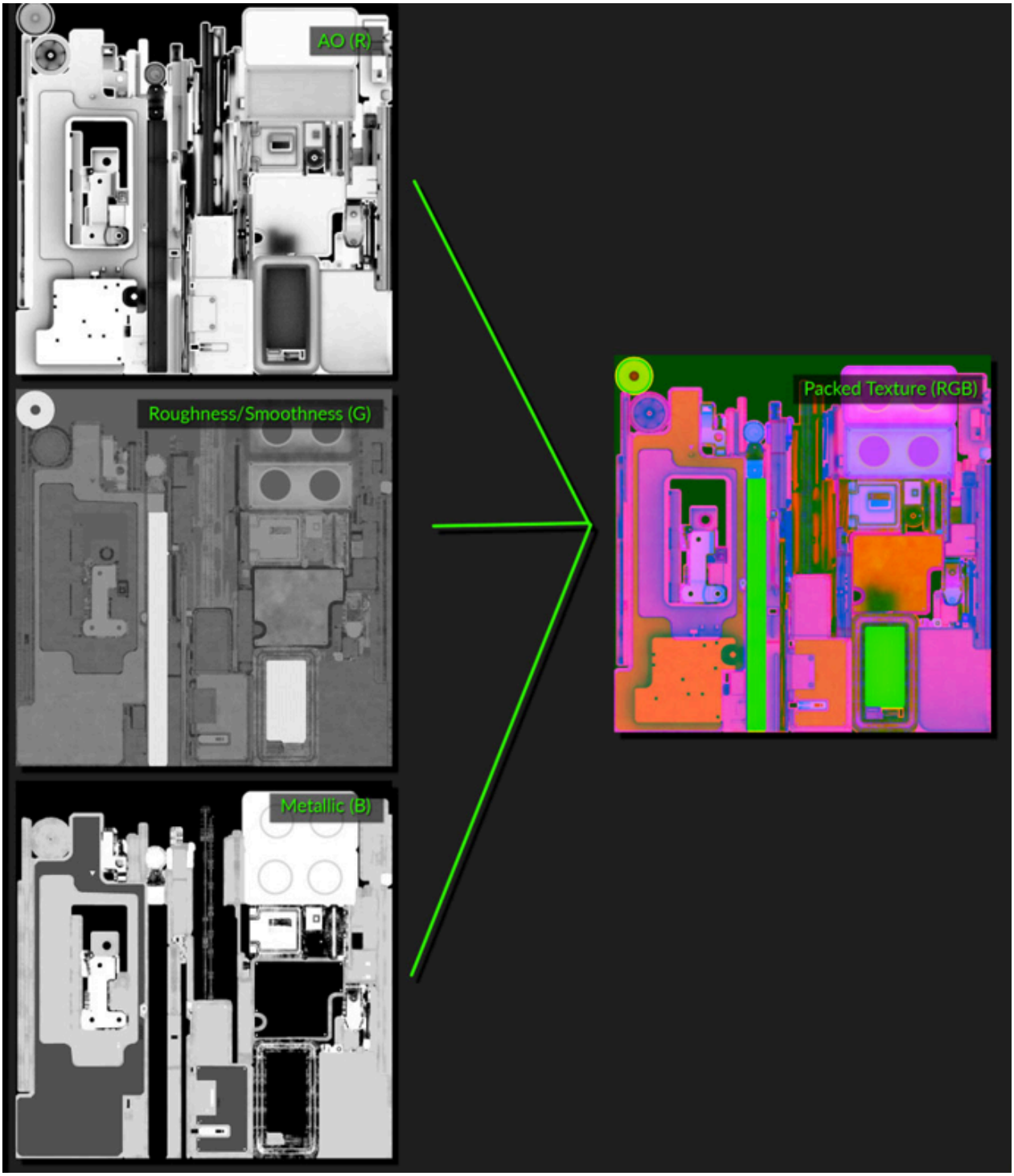
This texture packing technique is commonly used to pack roughness, or smoothness, and metallic into one texture. But it can be applied for any texture mask.

Use the green channel to store the more important mask. The green channel usually has more bits. This is because our eyes are more sensitive to green and less sensitive to blue.

The following image shows an example of texture packing:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 8-1: Texture packing example



Further reading on the color sensitive cones can be found here.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

9. Alpha channels and normal map best practices

In this section of the guide, we look at some more best practices that you can try if you want to use alpha channels in your game.

Be selective when adding an alpha channel to a texture. Adding transparency often makes the texture larger in file size because images are converted to a 32-bit format, increasing the overall bandwidth use.

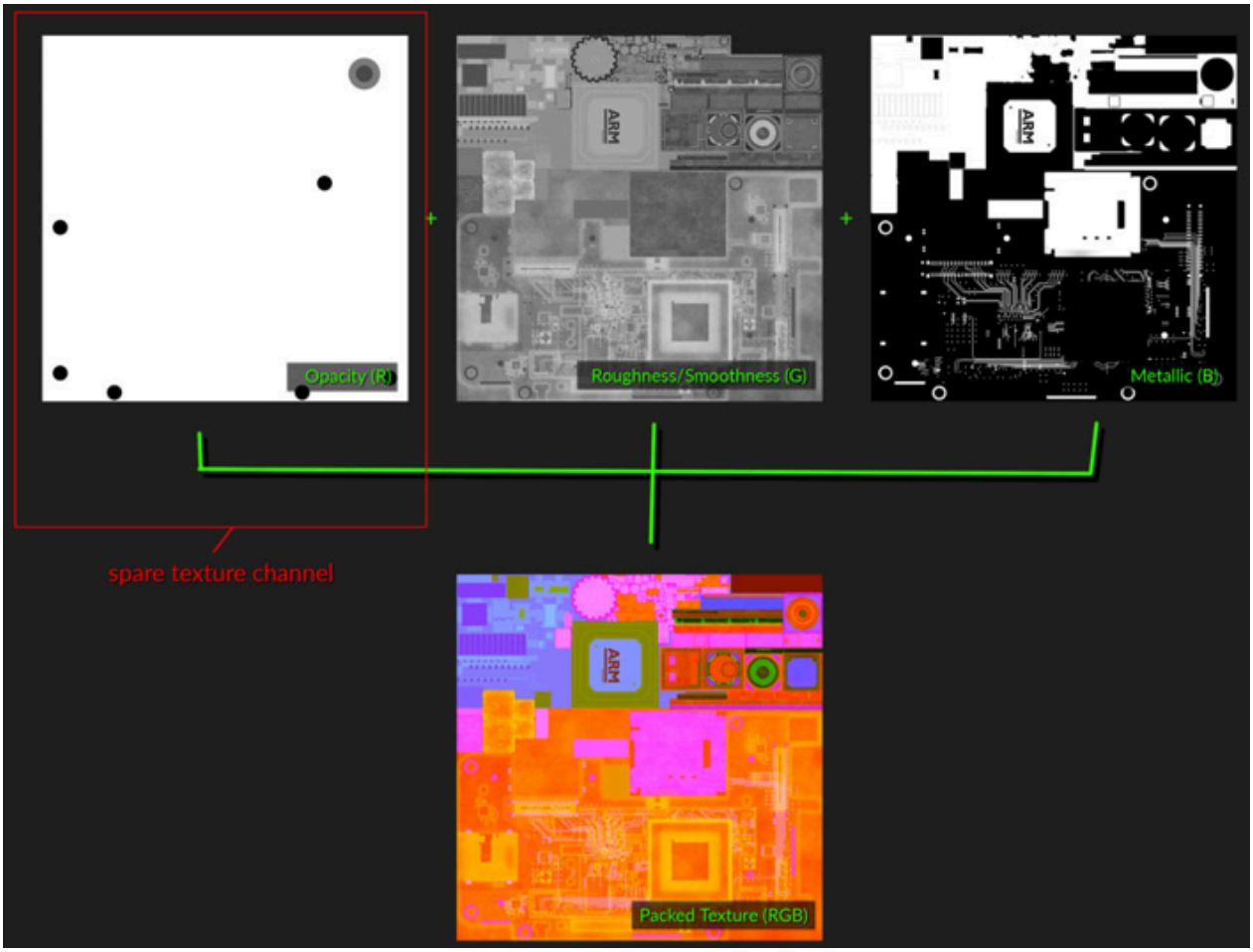
Another way to store an alpha channel is by using the extra channel in roughness, or metallic textures. In Unity, this texture sometimes uses two channels out of three, roughness (G) and metallic (B), leaving the (R) channel free for you to use.

By using the free channel to store the alpha mask, you can keep the diffuse texture at 16-bit, halving the file size. An ambient occlusion map usually can be baked in the diffuse map.

The following image shows an example of how you could store an opacity map in the red channel:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 9-1: Storing opacity map in the red channel



Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

10. Normal map best practice

A normal map is a good way to make a 3D object appear to have more detail. Normal mapping is best used to add smaller details like wrinkles, bolts, and other details that need lots of triangles to model.

The usage of normal mapping can depend on the type and art direction of a game.

In most of our internal projects, we use normal mapping with no noticeable degradation in performance. Because we target high-end devices for most of our demos, low-end devices might have different results.

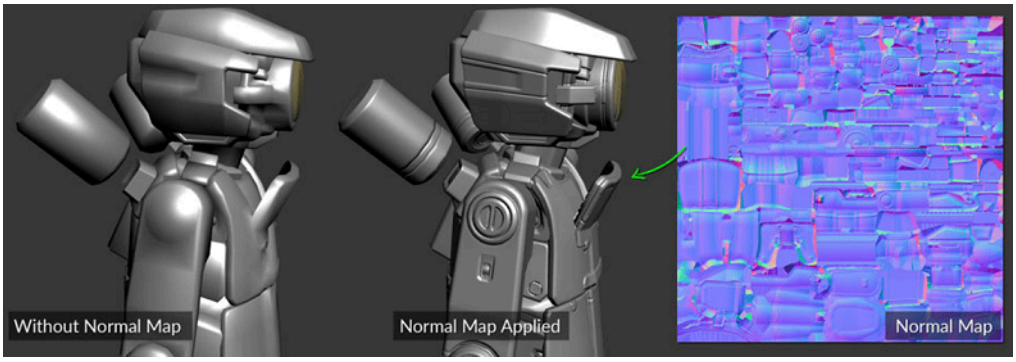
Using normal mapping does come with a cost, even if the cost is small. Remember:

- A normal map is an extra texture. This means more texture fetches, which results in more bandwidth being used.
- Use normal maps sparingly when targeting lower-end devices.

However, performance can be improved if geometry has fewer triangles as a result of normal map use.

The following image shows an example of how you could use a normal map and textures for the smaller details:

Figure 10-1: Using normal map and textures for the smaller details



Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

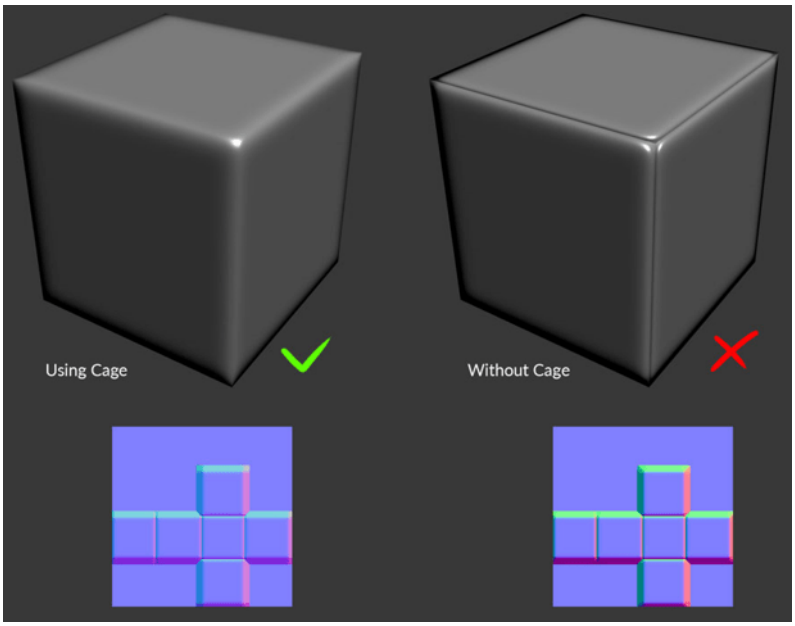
11. Normal map baking best practice

Using a cage is a great way to get a high-quality normal map, regardless of the surface that you are baking.

Most normal mapping software can make your cage automatically. However, you can make one from copying your low polygon model and then increasing its scale slightly.

The purpose of using the cage is for the program to change the direction that is used to calculate the normal when baking. This produces far better results on split-normal and hard edges, as you can see in the following image:

Figure 11-1: Example of using cage

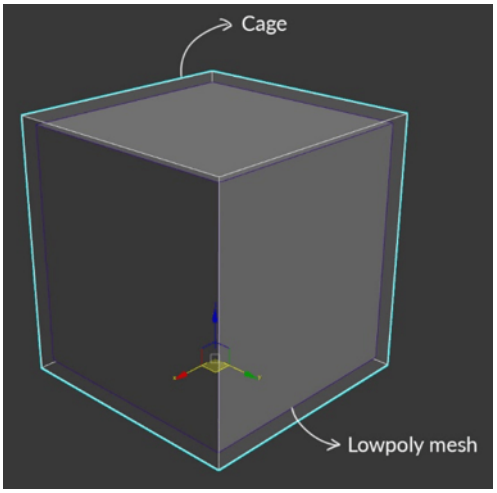


The cage is basically a larger, or pushed out, version of a low polygon count model. Encompass the high polygon count model for the baking to work well.

A mesh cage is used to limit the ray cast distance that is used during normal map baking. A cage can also solve problems with split-normal seams on the normal map, as demonstrated in the following image:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

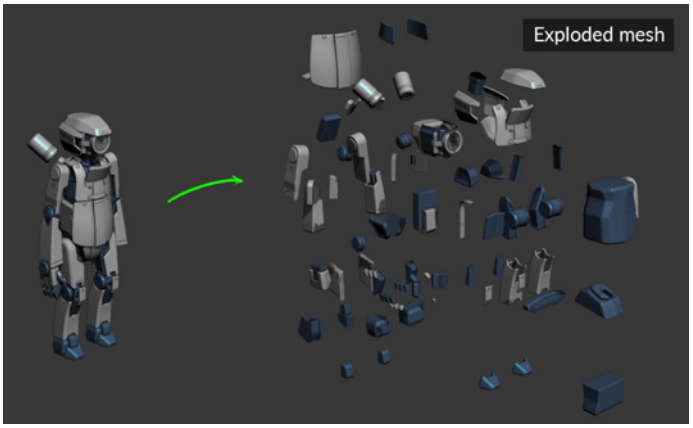
Figure 11-2: An example for mesh cage



If the baking software supports it, bake by matching the mesh names to mitigate the problem of creating a wrong normal map projection. When objects are too close to each other, they can unexpectedly project the normal map onto the wrong face. Matching the mesh names ensures that baking is only done on the right surface, with a matching name.

Further information on matching meshes by name is available on the substance3d website along with the Marmoset Toolbag tutorial.

Figure 11-3: Exploded mesh



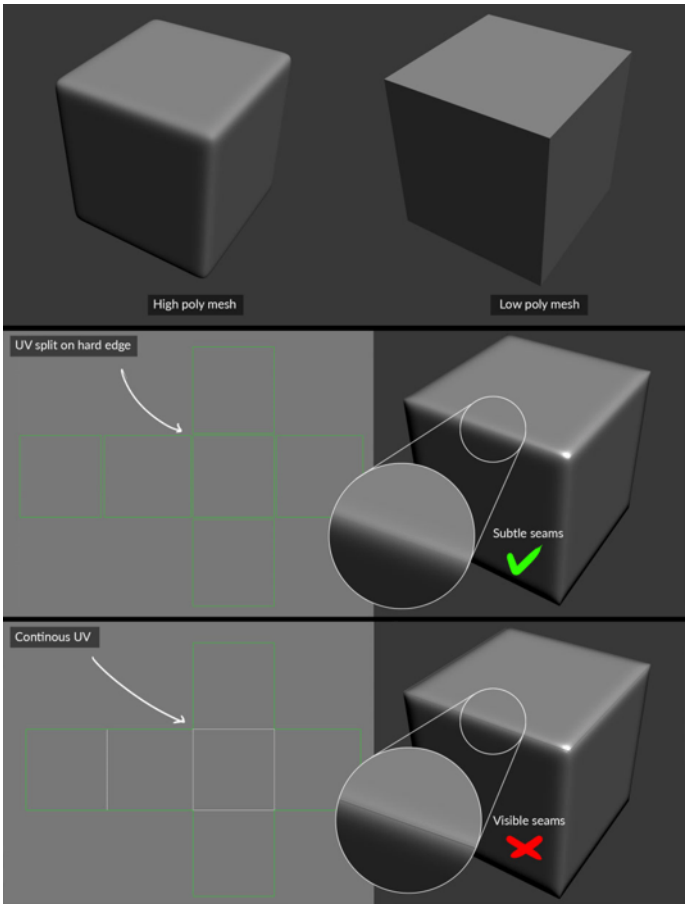
Mesh explode means moving parts away from each other, so the normal map don't project to unwanted surface. This will also help with wrong normal map projection

A separate bake for ambient occlusion can sometimes be required with this solution. This means that you should split UVs on hard edges, because a continuous UV on hard edges causes visible seams. The general rule is to keep the angle less than 90 degrees, or set it as a different smoothing group. Coincide UV seams with different smoothing groups on the triangles.

The following image shows an example of how a break UV looks on hard edges:

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 11-4: Break UV on hard edges



Set the smoothing group on mesh.

One simple rule of thumb would be to keep the angle less than 90 degree and it should be set as different smoothing group. UV seams need to have different smoothing group on the triangles.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

12. Editing texture settings

You can customize texture settings to provide fine-grained control over how Unity uses a texture.

Unity has made editing texture settings easy. Select the texture that you want edit, then the Inspector window displays the texture settings.

Here are some tips for when you are editing texture settings:

- Texture Type lets you control the type of texture, causing the texture to be used in different ways within the engine.
- Texture Shape allows you to choose cubemaps instead of 2D for some texture types.
- Depending on the Texture Type that is selected, there are two sections that appear:
- Texture Settings has specific controls for the texture to work as you need.
- Advanced Settings covers: sRGB, Alpha Source, Alpha Is Transparent, Read/Write Enabled, and Generate Mip Maps.
- Wrap Mode controls how the texture wraps your UVs. Here are the options that are available:
- Repeat tiles the texture, and is used for repeating patterns.
- Clamp locks the texture to the last pixel at the edge.
- Mirror works like Repeat, but mirrors every other repeat of the texture.
- Mirror Once mirrors the texture a single time, before locking the texture to the edge pixels.
- Filter Mode controls what texture filter is used on the texture.
- The Texture compression box gives you control over: Max Size, Resize Algorithm, Format, Compression, and Use Crunch Compression.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

13. Check your knowledge

Test your knowledge with the following questions:

What is a texture atlas, and what benefits does a texture atlas provide?

A texture atlas is an image that contains data from several smaller images that have been packed together. Texture atlasing reduces memory cost by reducing the total number of textures in a game, and increases performance by reducing the number of draw calls.

Which texture filter does Arm recommend for a balance between performance and visual quality?

Bilinear filtering

What is the recommended limit for non-critical asset anisotropic filtering levels?

Arm recommends keeping the anisotropic level low. Only use a level higher than two for critical game assets.

Which color channel should you use for the most important mask when packing textures?

Use the green channel to store the more important mask. The green channel usually has more bits. This is due to our eyes being more sensitive to green and less sensitive to blue.

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

14. Related information

Here are some resources related to material in this guide:

- Marmoset: The Toolbag Baking Tutorial
- Substance Bakers: Matching by Name
- Unity Manual: Draw Call Batching
- Unity Manual: Textures
- Unity Manual: Importing Textures

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

15. Next steps

This guide covers texture optimizations that can help your games to run more smoothly and look better.

You can continue learning about best practices for game artists and how to get the best out of your game on mobile by reading the other guides in our series:

- Real-time 3D Art Best Practices: Geometry
- Real-time 3D Art Best Practices: Materials and Shaders
- Real-time 3D Art Best Practices: Lighting

Copyright © 2020, 2023 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential