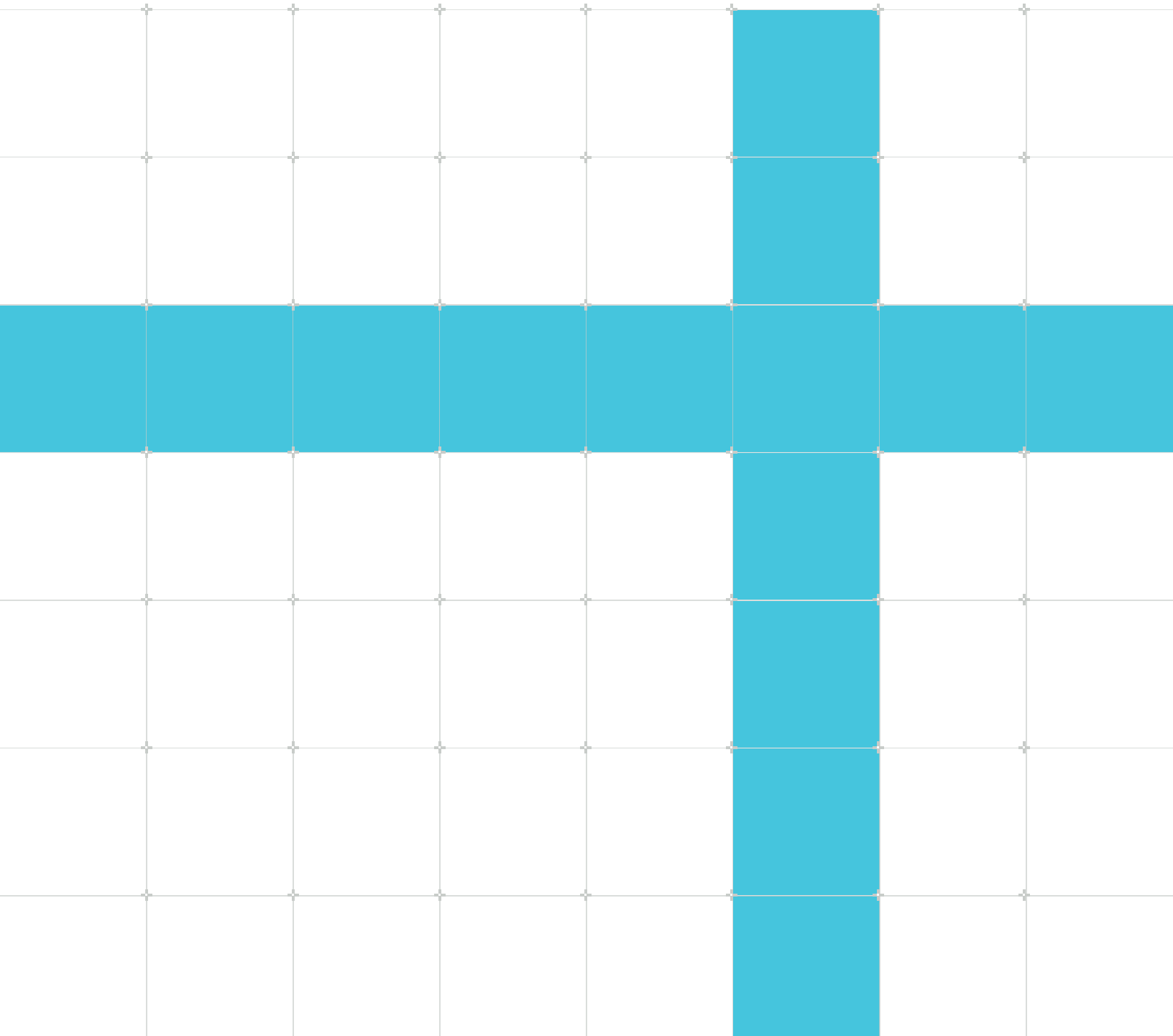
Unreal Engine

Version 1.0

Geometry best practices for Unreal Engine

Non-Confidential
Copyright © 2021, 2024 Arm Limited (or its affiliates).
All rights reserved.

Issue 02
102695_0100_02_en



Unreal Engine

Geometry best practices for Unreal Engine

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	28 September 2021	Non-Confidential	First issue
0100-02	14 February 2024	Non-Confidential	Editorial changes

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 © 2021, 2024 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 © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Overview

In this guide, you will learn about how to make geometry optimizations using the Epic Unreal Engine game engine.

Geometry is one of the main components when you create a 3D game. To ensure a game runs well on all devices, geometry must be optimized as much as possible. This guide tells you how to make geometry optimizations for 3D assets to make more efficient games that perform better on mobile platforms.

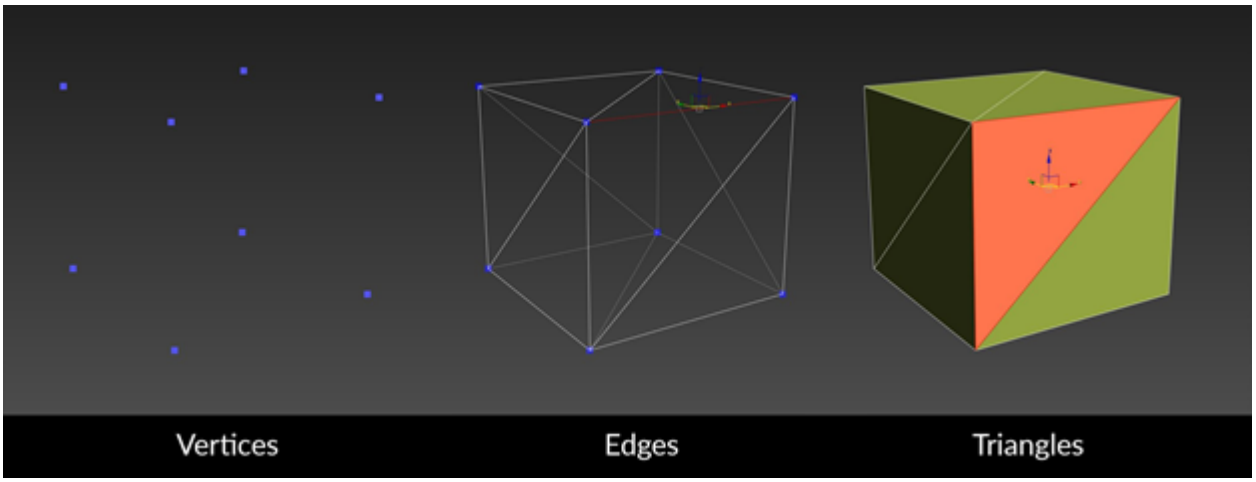
By the end of this guide, you will understand how to do the following:

- Optimize geometry for mobile devices
- Effective triangle and polygon usage
- Level of Detail (LOD) tips and tricks
- Other geometry best practices

What is Geometry?

Geometry, also known as a polygon mesh, is a collection of vertices, edges, and faces that make up the shape of a 3D object. This can be any asset in a video game like a car, weapon, environment, or character. The following image shows the three points that make up geometry:

Figure 1-1: Geometry example



Vertices are the points that make up the surface of a 3D object. An edge occurs when you connect two vertices with straight line. Three vertices connected to each other by three edges is a triangle, and is sometimes referred to as a polygon or face.

Within 3D software such as Max, Maya, or Blender, we usually work using quads (a four-sided polygon) because they are easier to change and work with. When rendered on screen, polygons are displayed as triangles, so we use this term in this guide.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. Triangle and polygon usage

We recommend that you use the fewest number of triangles possible to get the quality required to create your object. The number of triangles should be one of the first things to consider when creating content for mobile platforms.

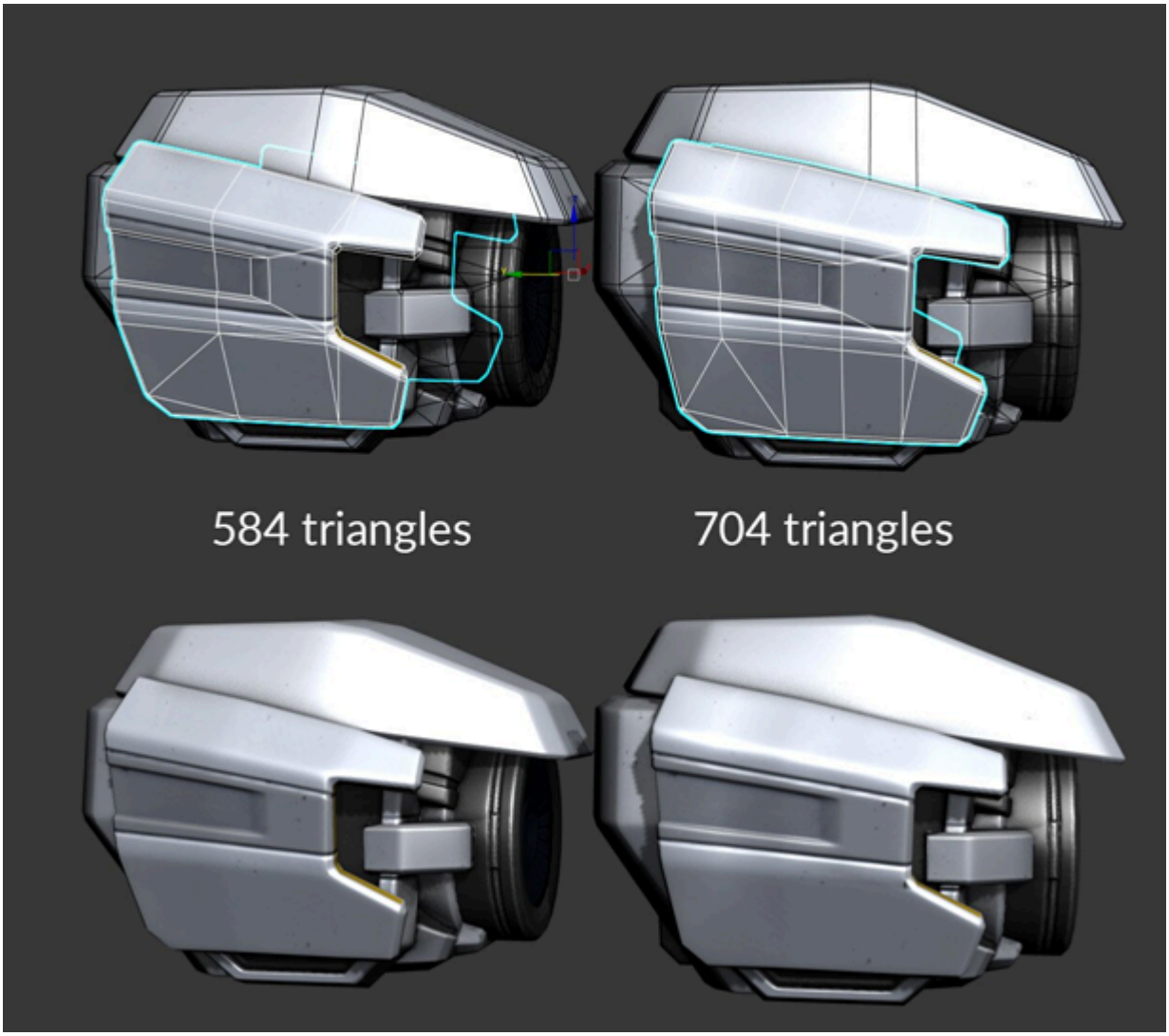
Fewer triangles will help boost performance in the following ways:

- Fewer vertices are processed by the GPU (Graphics Processing Unit)
- Processing vertices is expensive. The fewer number of vertices that get processed, the overall performance improves.

Having fewer triangles means the game can be released on more devices, not only devices that have the most powerful GPUs. The following image shows a scaled down version of two objects using fewer triangles:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-1: Triangle comparisons



In this image, edges are removed that do not contribute to the silhouette. As a result, the two objects look the same in shaded mode.

On mobile platforms, the maximum vertices on each mesh is 535 because GPU support for 32-bit indices is not guaranteed on all platforms. For example, Android devices with Mali-400 GPU do not support 32-bit indices and the mesh will not render.

View and test the game on your target devices and do not use a monitor display. Some details created with many triangles might not be visible on a phone.

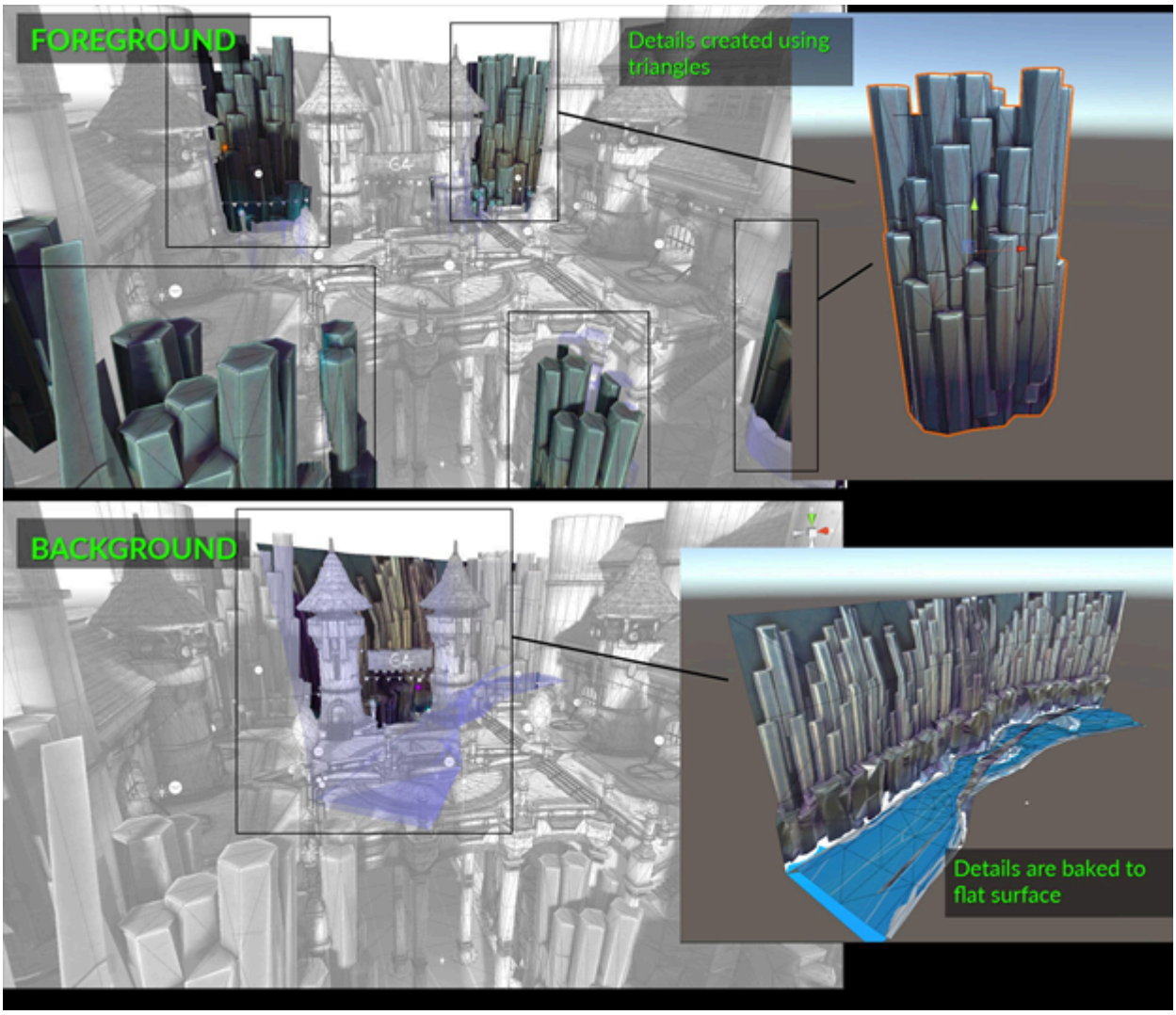
Foreground and background objects

Use more triangles on foreground objects that are closer to camera and less on background objects.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

The following example shows a game with a static camera point of view using more triangles on foreground objects:

Figure 2-2: Foreground and background objects example



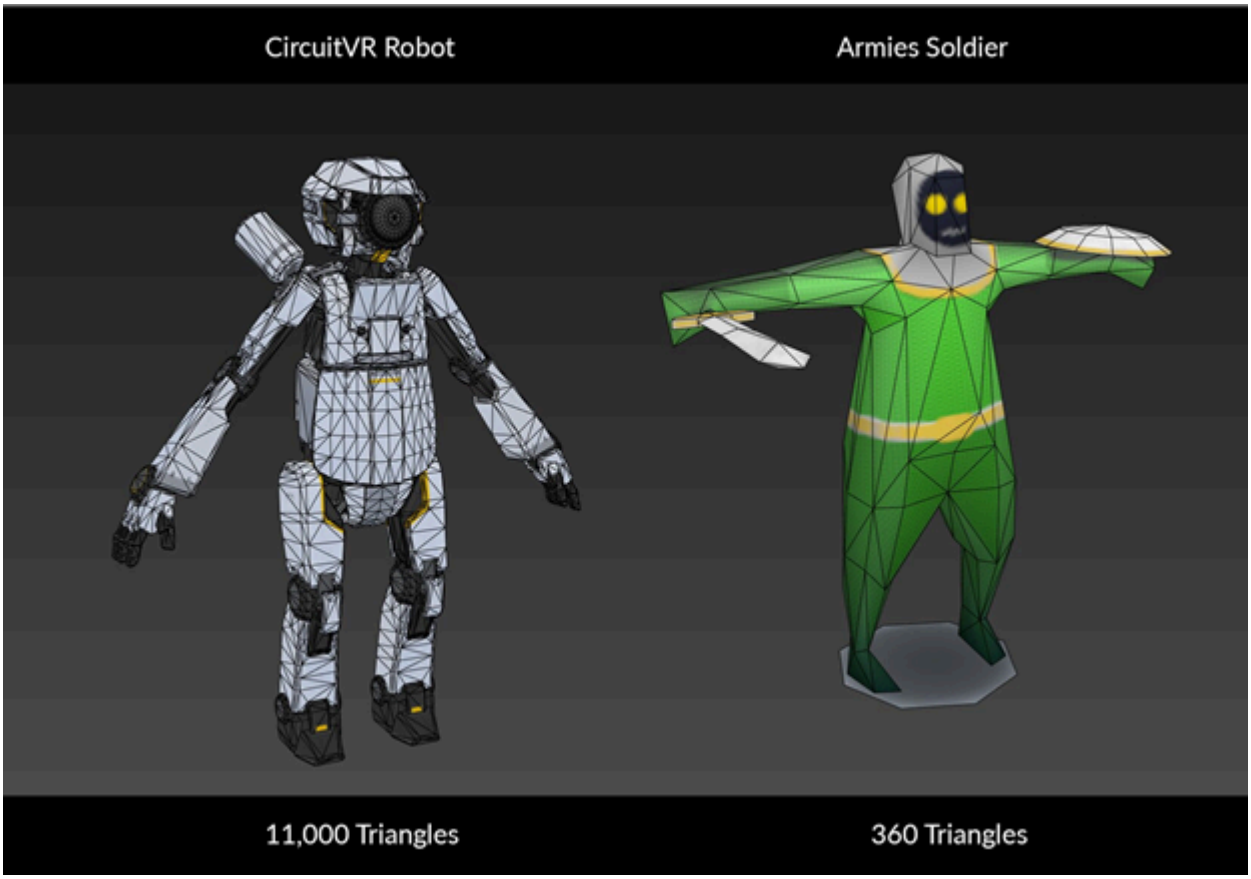
The maximum number of triangles used on a model depends on the following factors:

- The number of objects that are visible at a time in the game. Multiple visible objects have a lower triangle count, but if there are only two or three objects visible, we can use a more triangles.
- The target device. Newer smartphones such as the Samsung Galaxy S series can handle more complex geometry than an older mobile device.

The following example shows a robot character with a higher polycount, and a game with hundreds of soldiers in one frame using fewer triangles:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-3: Triangle count comparison

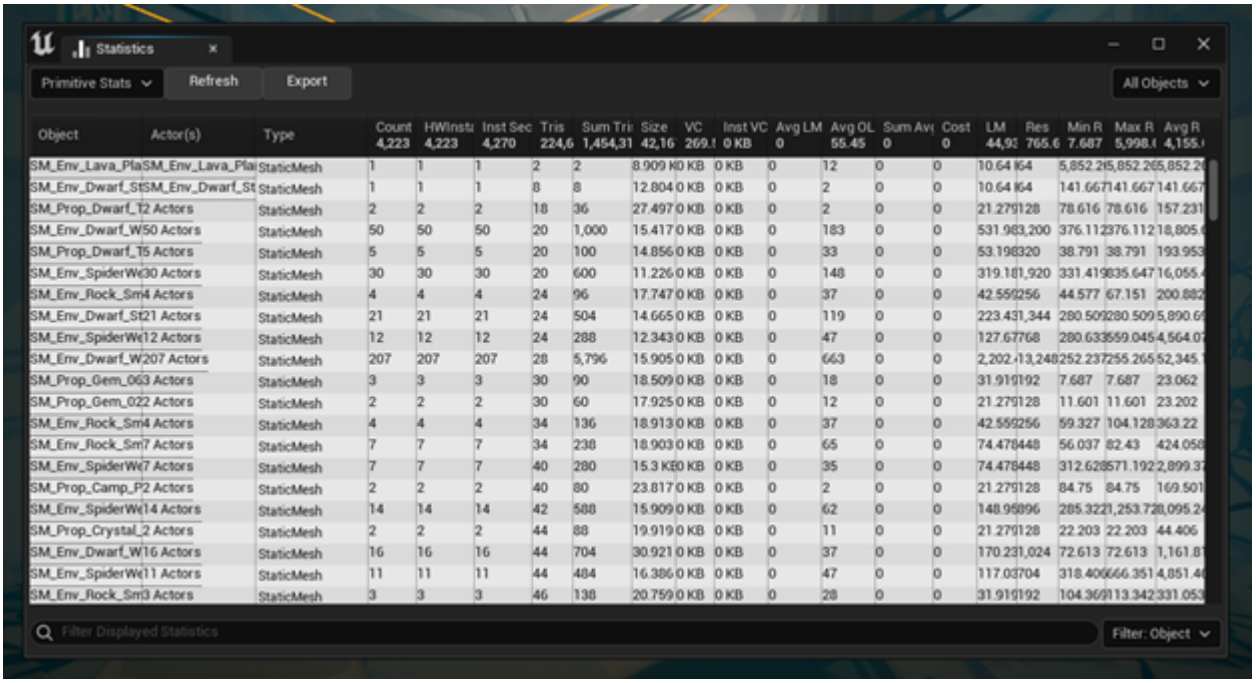


Manage assets

Unreal Engine offers tools to help manage your assets, such as the Statistics screen. The following screenshot shows the primitive mesh statistics in a project:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-4: Primitive mesh statistics example

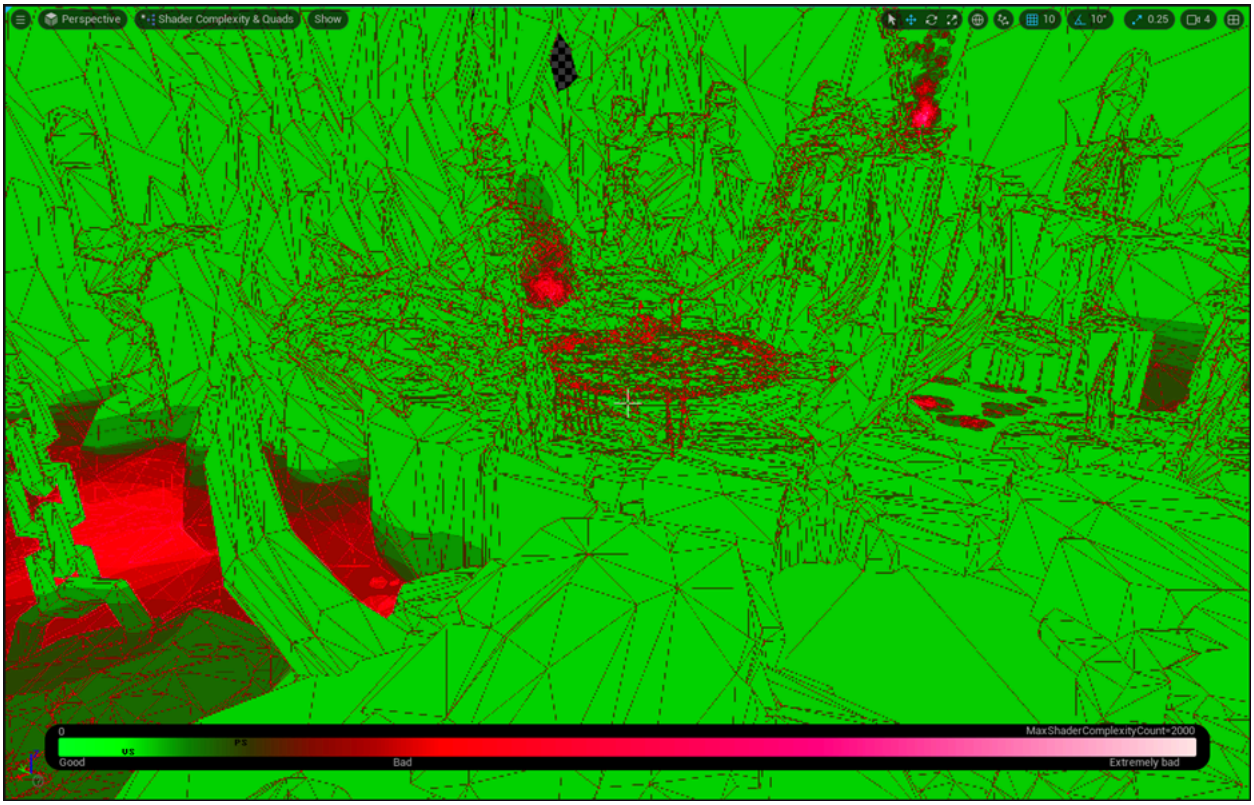


These statistics help you check that your assets stay within budget and quickly identifies assets that are not.

Another powerful tool is the Shader Complexity and Quads view mode. This mode is used to visualize the number and shader instructions and helps you see areas where your geometry is too complex. In the following screenshot, red areas are particle effects and red vertices towards the middle of the scene are visible:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-5: Shader Complexity and Quads



To bring this mesh back into the green, you can lower the number of triangles to reduce the mesh complexity. Additionally, you can use LOD, as described in Level of Detail.

Distribute triangles

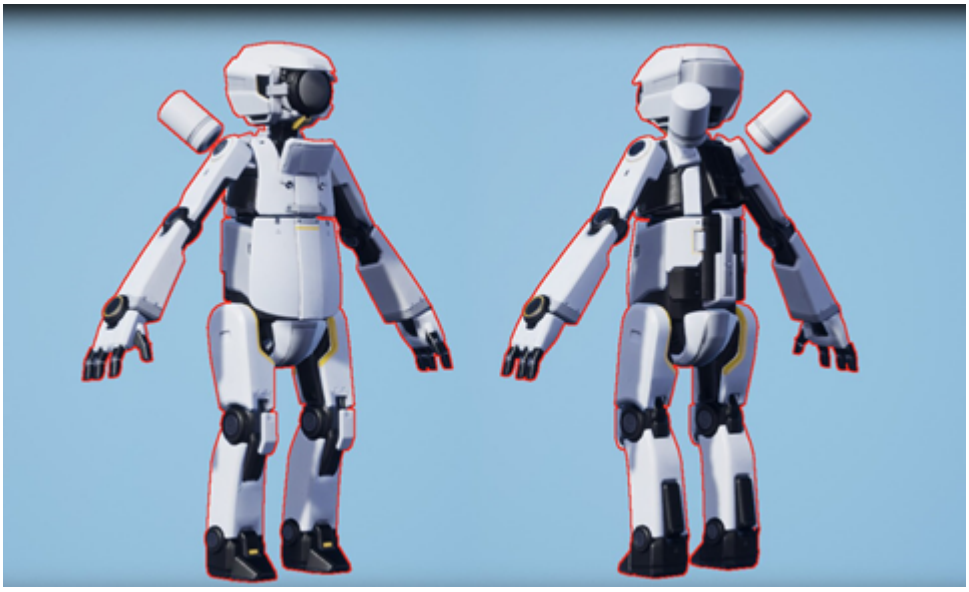
Polygons or vertices are very expensive on mobile platforms. By placing polygons in areas that contribute to the visual quality of the game, processing budget is not wasted.

When you are creating details, it is important to consider that small triangle details on a 3D object will not be visible on the final screen in the game. These details are affected by a combination of a small phone screen size and the placement of 3D objects in the game level.

Focus on big shapes and forms that contribute to the silhouette of the object instead of the detail, as shown in the following example:

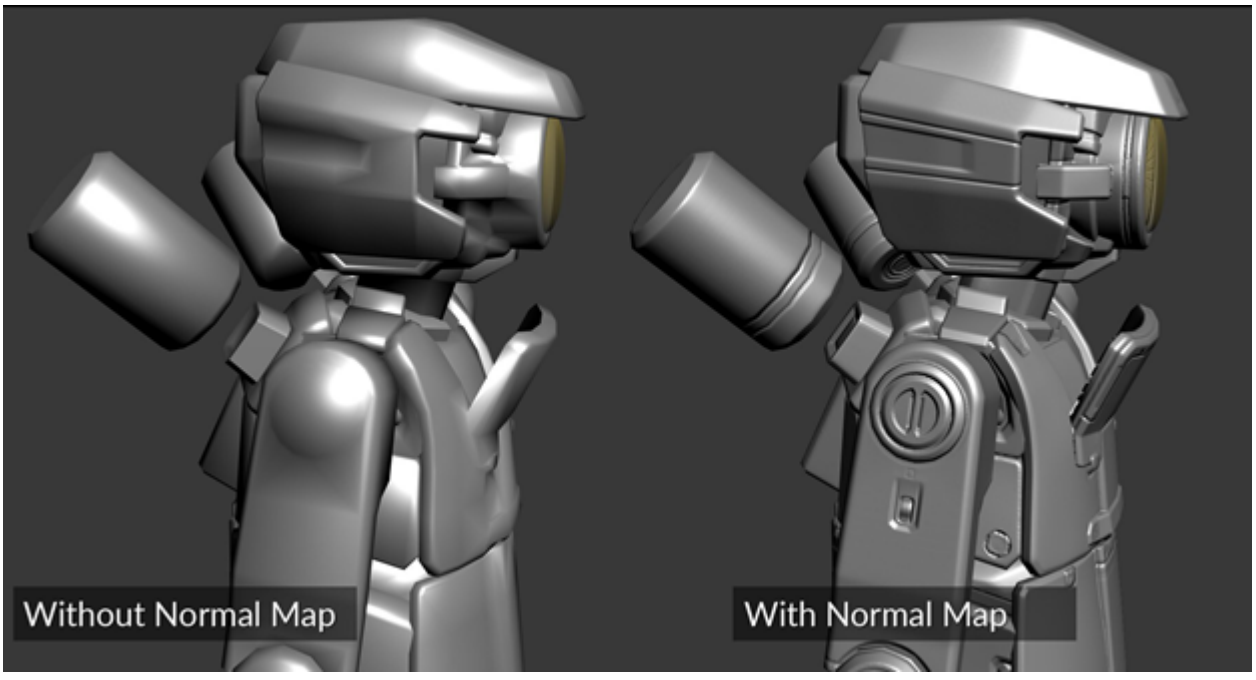
Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-6: Silhouette example



Do not model intricate details using high density triangle meshes and use textures and normal maps for fine detail. The following image shows an example of a mesh with and without normal map:

Figure 2-7: Map examples



Use fewer triangles on areas that are not often seen from the camera point of view, like the bottom of a car or the back of a wardrobe. You can also delete these areas, however, delete areas carefully

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

in case you need to reuse the scene. For example, if you delete the bottom part of a table mesh, it cannot be placed upside down.

Do not use micro triangles

Micro triangles are triangles that are too small to contribute to the visual of an object or final scene. Micro triangles are smaller than 1 to 10 pixels on a phone screen. The GPU processes these triangles, even though they do not benefit the object. We recommend that triangles are above 10 pixels in area.

Vertices are expensive to process, and small triangles cause more vertices. This processing also affects memory bandwidth, because there is more data that needs to be sent to the GPU. On a mobile device the GPU processing affects battery life, so the user cannot play the game for as long.

Additionally, 3D objects with a high polygon count experience problems with micro triangles when these objects are moved further away from the camera.

Micro triangles can be caused by the following:

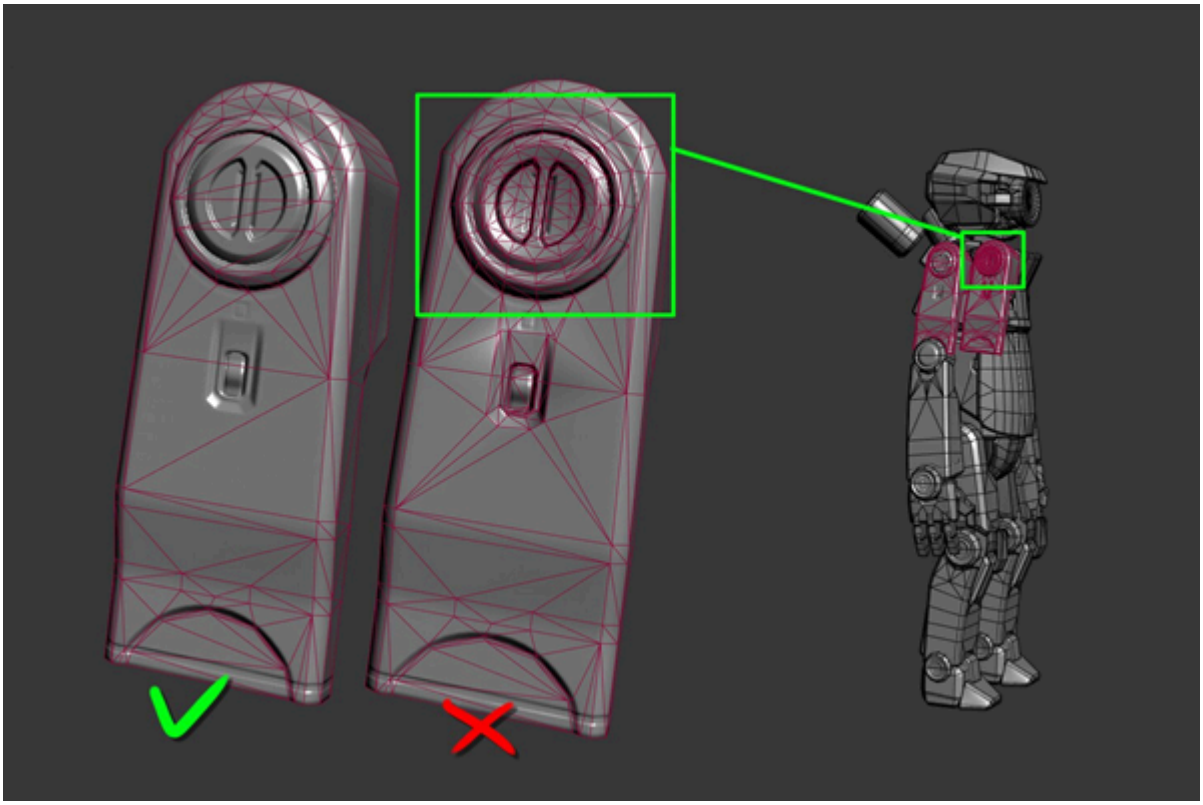
- Details that are too small and consist of many triangles
- Objects further from camera with many triangles

Do not model details using many polygons. Instead, use textures and normal maps for this type of fine detail. You can also merge vertices or triangles that are too small.

In the following image, the triangles in the area in the green square are too small to be visible on phone screens:

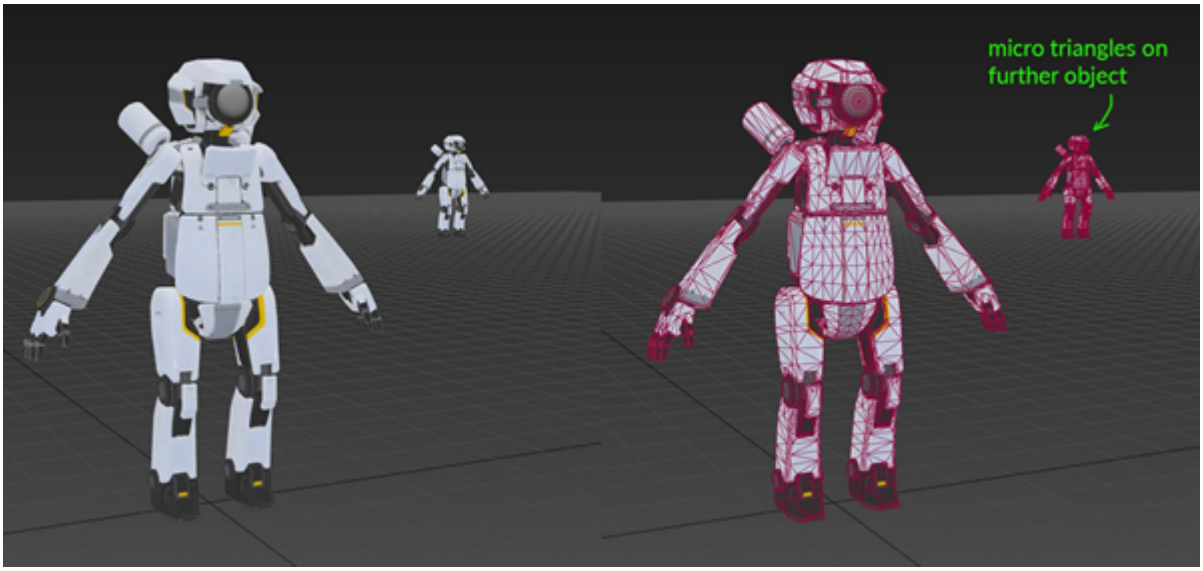
Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-8: Micro triangle example



If an object is further from the camera, use LOD to reduce complexity. LOD makes objects simpler and have less dense triangles. In the following image, the character on the left uses less triangles and utilizes normal map for finer detail:

Figure 2-9: Micro triangles on objects that are further from the camera



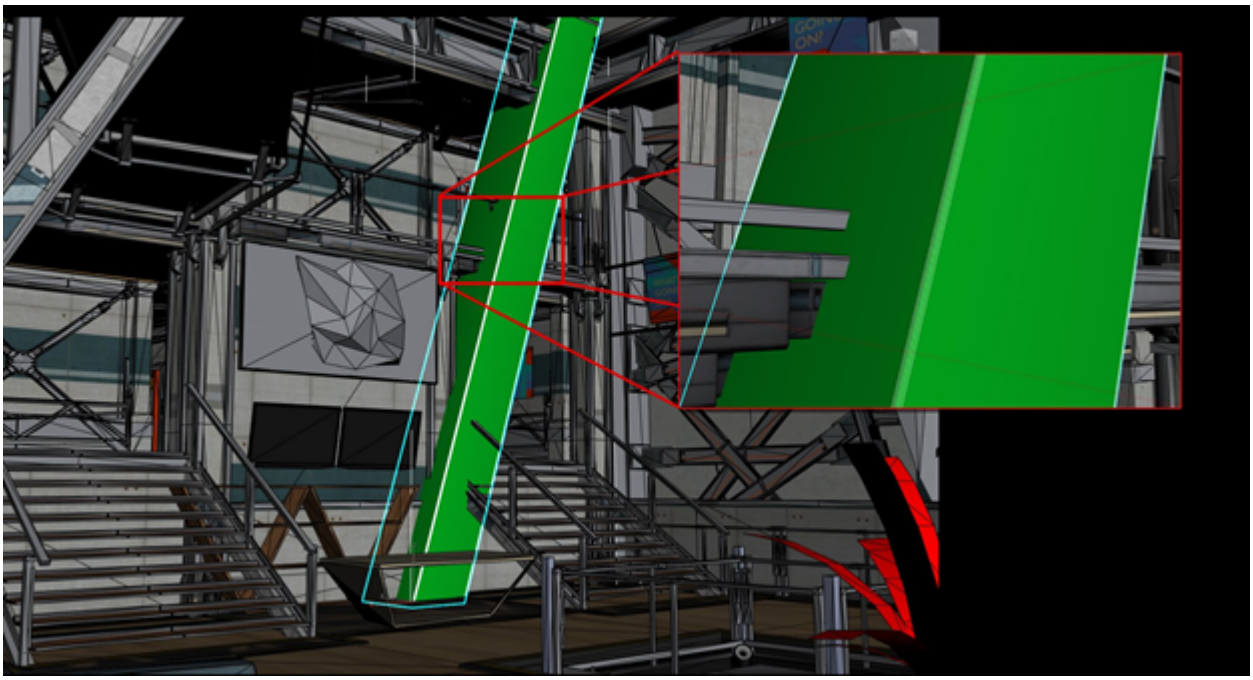
Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Do not use long thin triangles

Long thin triangles are smaller than 10 pixels and span the screen when rendered in a final image. Do not use long thin triangles because it is generally more expensive to process compared to normal triangles.

For example, a long thin triangle is a bevel on a pillar when viewed from a distance. These bevels are not a problem if they are viewed close, like in the following image:

Figure 2-10: Long thin triangle example



We recommend the following best practices for triangles:

- Remove long thin triangles on objects, if possible.
- Do not use shiny material on an object with long thin triangles, because this will cause flickering.
- Use LOD and remove long thin triangles when they are further away.
- Keep the triangles close to equilateral so that objects have more inside area and fewer edges. For more information, see Triangulation.

Because the triangle is very thin, when the camera moves slightly it either makes the triangle touch the sampling point or miss this point. As a result, the fragment is not always rendered and appears to flicker in the final image.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. Level of Detail

As an object moves further from camera, we can see less detail in that object. For example, it is difficult to see the difference between an object that consists of 200 triangles and an object with 2000 triangles from 20 meters away. Using fewer triangles boosts performance because fewer triangles need to be processed. The following image shows an example of objects with different triangle counts in the distance:

Figure 3-1: Example of triangle counts at different distances

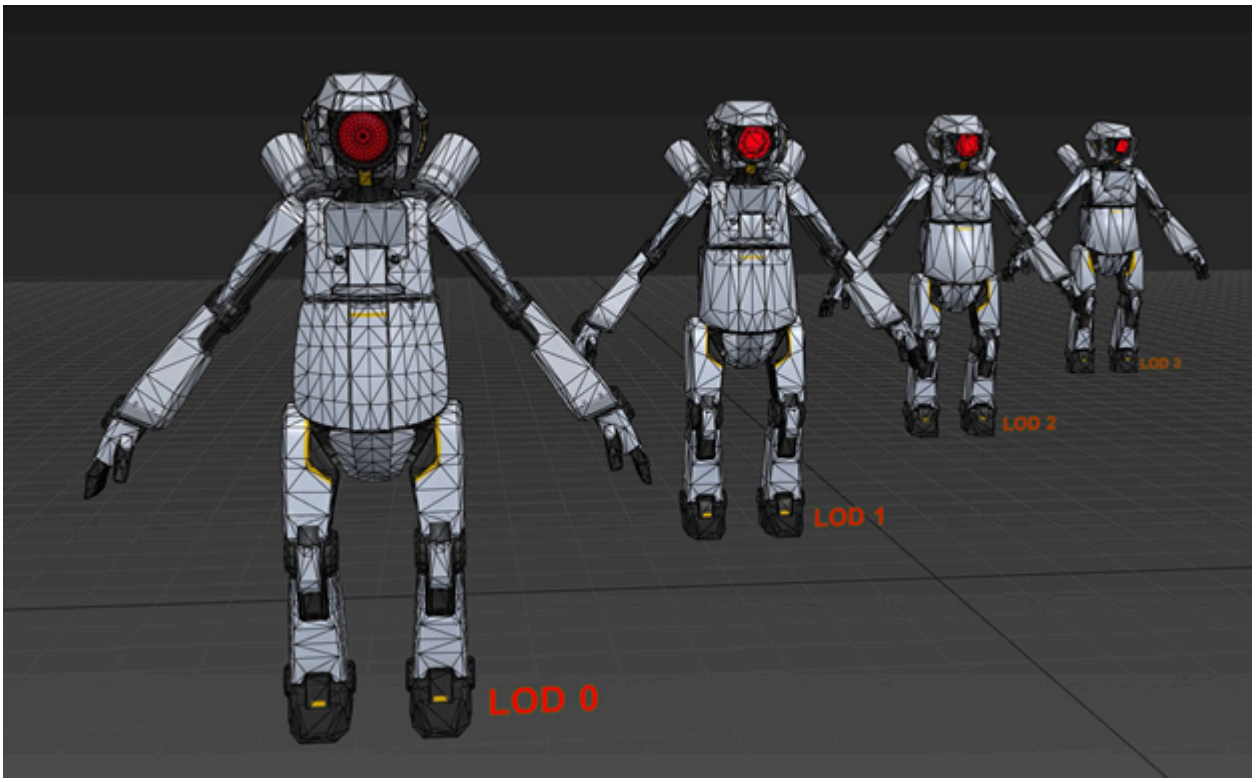
Level of Detail (LOD) provides the following benefits:

- Reduces mesh complexity as objects become more distant from the viewer
- Reduces the number of vertices that need to be processed and avoids micro triangles.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- Helps mitigate problems caused by micro triangles, as explained in Do not use micro triangles.
- Helps objects placed further away in the scene look better, as shown in the following image:

Figure 3-2: Level of detail example



When using LOD, focus on the silhouette of the objects. LOD can also apply to shader complexity, to optimize shader and material for objects that are further away. For example, by reducing the number of textures used. Remove polygons on flatter areas and use mipmap as LOD for texture.

We recommend that you do not use LOD in the following situations:

- On a game where the camera view is static and the objects are also static, as shown in the following image:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-3: Scene without LOD

In this example, a different method of mesh optimization is used, such as removing parts that are not visible from the camera.

- On an object that is already simple or an object that has a low triangle count

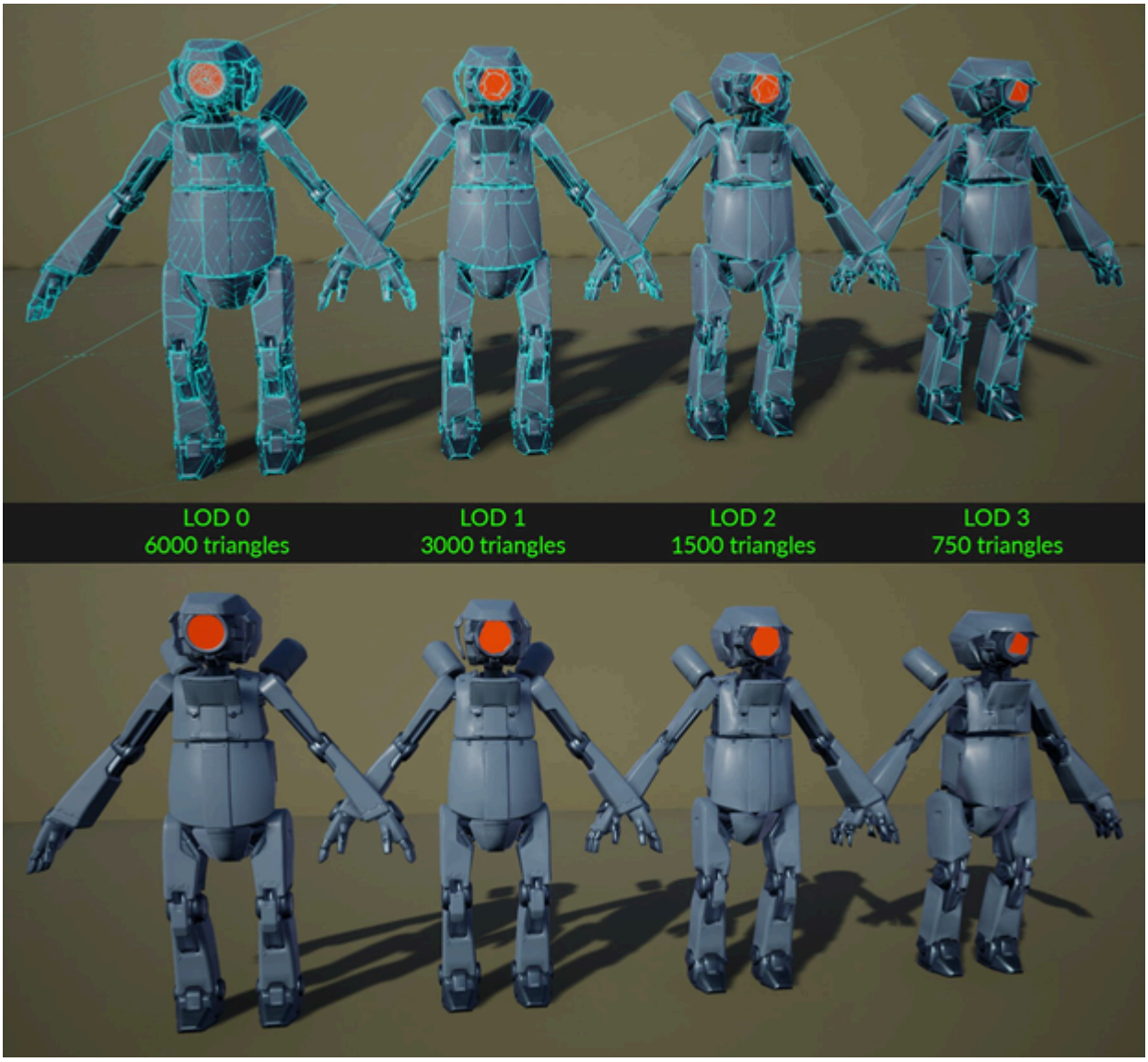
LOD comes with a memory overhead and therefore, a bigger file size. The LOD mesh data will need to be saved in memory.

Reduce triangles on each LOD

We recommend that you reduce the number of triangles by 50% between levels, as shown in the following image:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-4: Triangle reductions

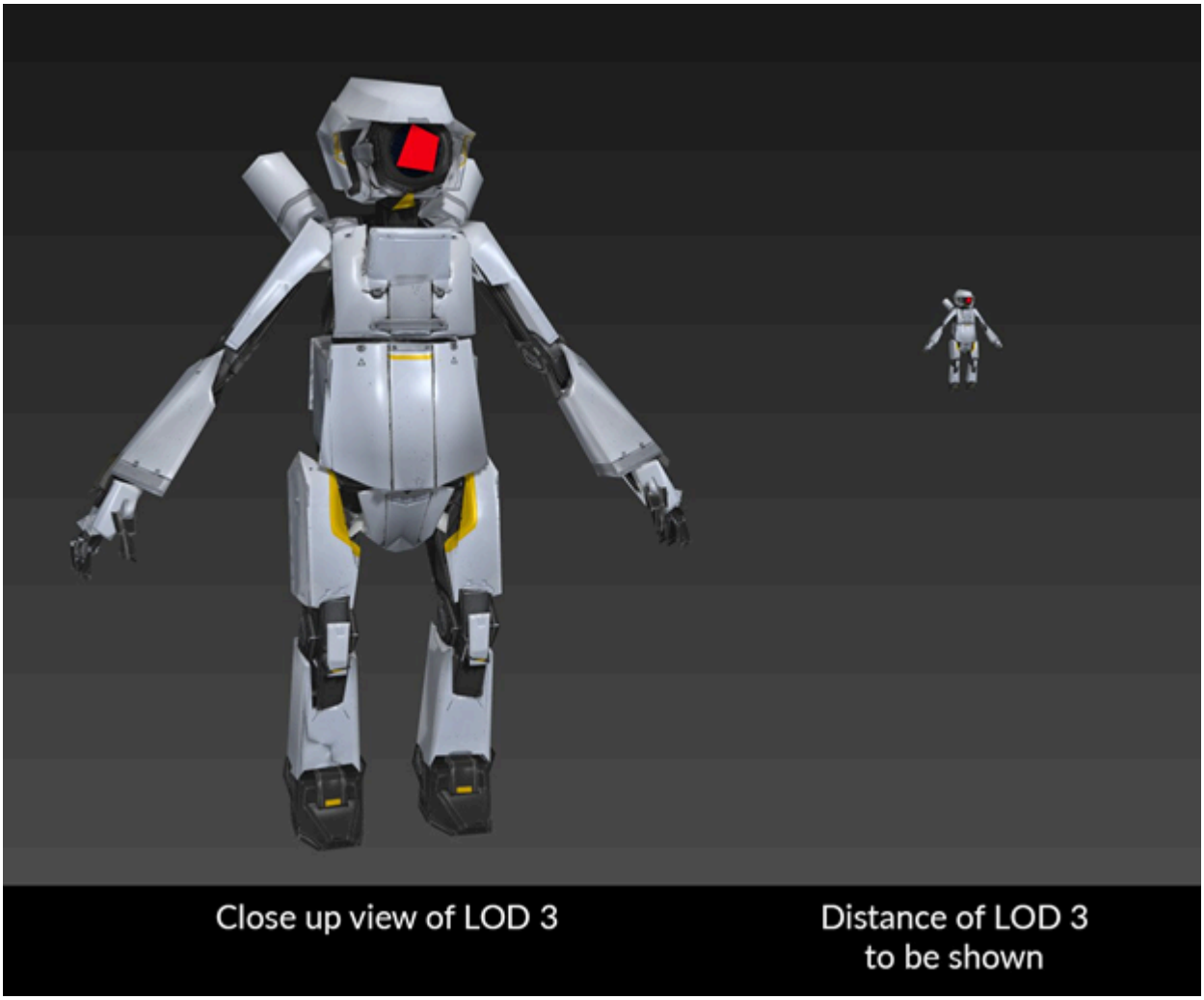


Do not use very dense triangles on lower LOD when objects are further away.

Test the LOD as they will be seen at the correct distance from camera. Lower LOD can look wrong when viewed at a close distance but appear fine at the distance intended, as shown in the following image:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-5: LOD distance example



Too little polygon reduction affects performance improvements because a similar number of triangles are rendered. Too much polygon reduction and popping are more noticeable on LOD switch.

Set the LOD on an object

The amount of LOD on an object depends on the size of the object and how important the object is. For example, a character in an action game or a car in a racing game has more LOD level than small background objects like a tree.

If the LOD is too low, the following occurs:

The performance gain is not noticeable if the polygon reduction is not substantial between levels The popping on LOD switch can be more noticeable.

If the LOD is too high, the following occurs:

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

- Extra CPU workload because processing is needed to decide which LOD is displayed.
- The LOD meshes cost memory to store the extra meshes and increase file size.
- Increased time needed to create and verify LOD models if they are created manually by an artist.

Create LOD meshes

To create LOD meshes manually in 3D software, remove edge loops or the number of vertices on a 3D object. This gives more control to the artist but may potentially take a longer time to do.

To create LOD meshes automatically, use one of the following methods:

- The Unreal LOD generation features to create and apply LOD meshes. These features enable you to apply LOD to a single mesh or automatically create LODs when you import a mesh. For more information about setting up these features, see Automatic LOD Generation.
- A modifier inside a 3D package, such as ProOptimizer in 3DSMax or Generate LOD Meshes in Maya.
- A LOD generation software such as Simplygon and InstaLOD.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

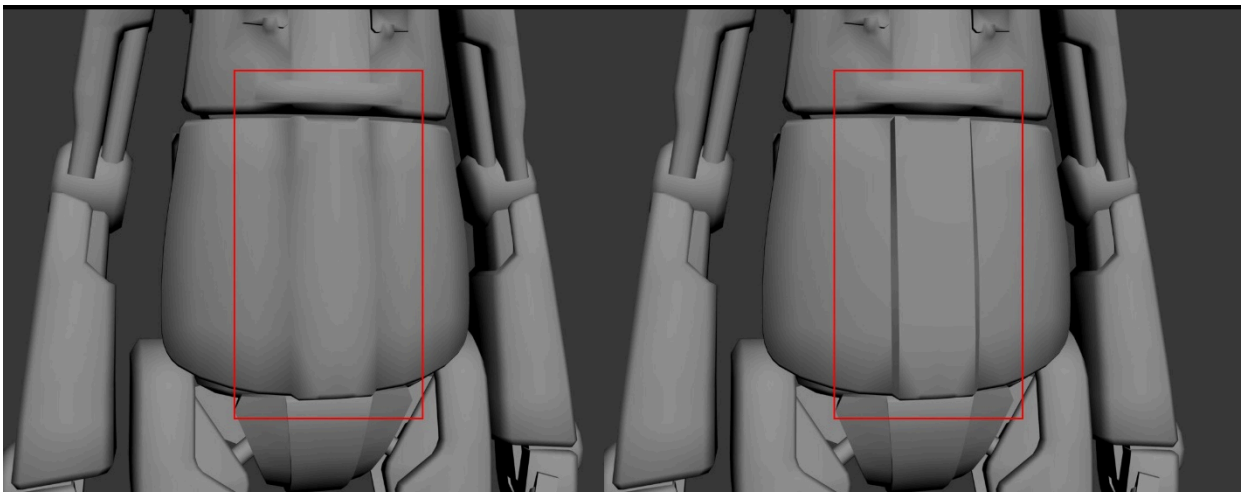
4. Other geometry best practices

This section provides recommendations for using smoothing groups and mesh topology with objects.

Smoothing groups

Use smoothing groups to define the hardness of an edge and alter the look of a model. Smoothing groups help create better shading on low polygon art direction, as shown in the following image:

Figure 4-1: Smoothing group application



Using smoothing group effects, UV islands split and can reduce the quality of a normal map when we bake the normal map. Extra care must be taken to prevent this.

When smoothing groups are used on a 3D model, the smoothing groups are exported from the 3D software and imported into the engine. Ensure smoothing groups are enabled in your 3D software to export these groups correctly.

Mesh topology

We recommend the following when creating mesh topology:

- Have clean topology when creating a 3D asset. Clean topology is essential for character or other objects that are deforming or animated in your scenes.
- Do not aim for perfect topology on a 3D model. Topology should be tidy, but not all objects need perfect edge flow.
- The end user does not see the wireframe of a 3D model
- A mesh has texture and material applied to it, which will have a bigger contribution to the look of a 3D model. In the following example, texture and material are more visible than topology in the final image:

The wireframe of the rock cliff mesh in this image shows simple geometry and topology.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

In the final scene, the rock cliff looks better with material applied and the problem with topology is not noticeable, as shown in the following image:

Figure 4-2: Texture and material result example

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. Related information

The following resources are related to material in this guide:

- Unreal Engine documentation
- Use Streamline to Optimize Applications for Mali GPUs

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

6. Next steps

This guide has introduced you to using geometry in the Unreal Engine game engine. You learned about triangles and polygon usage, using LOD, and other best practices for geography.

After reading this guide, you can use these best practices to optimize the performance of your apps on mobile devices that use Unreal Engine.

To learn more about unreal engine best practices, see the Arm Unreal Optimization Resources.

Copyright © 2021, 2024 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential