Qualcomm Technologies, Inc.

Qualcomm® Snapdragon™ Occlusion Culling

User Guide

80-NB295-15 Rev. AB

January 20, 2022

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
AA	April 2021	Initial release
AB	January 2022	Updated document to reflect Snapdragon Occlusion parameters and instructions Updated patch and library instructions Added SDOC API, settings recommendations

Contents

1 Introduction

1.1 Purpose

Occlusion culling is a widely used technique in game development. It reduces the workload that the GPU must process by limiting the number of objects that are drawn and submitted to the GPU every frame.

Qualcomm® Snapdragon Occlusion Culling (SDOC) is a mobile-optimized version of the Open Source project https://github.com/rawrunprotected/rasterizer. SDOC resolves the workload issue in a way that is performant across the mobile device spectrum, while yielding good power and performance characteristics.

This document explains basic concepts and algorithms of SDOC, details information on how to use SDOC and integrate with Unreal Engine 4 (UE4).

1.2 Visibility problem

Occlusion Culling solves visibility problems that arise in a game scene. The key goal is to determine whether an occludee is visible. The following terms help differentiate various objects in a given scene:

- **Camera View**: Defines the position, orientation and parameters that define the point of view of a camera overlooking a game scene.
- **Occluder**: Object/Geometry in scene that is fully or partially covering/occluding other objects in the scene given a camera viewpoint.
- **Occludee**: Object/Geometry in the scene that is fully or partially covered/occluded by occluders in the scene given a camera viewpoint.

1.3 Occlusion Culling comparison

1.3.1 Algorithms

The following algorithms are available for Occlusion Culling in UE4:

- CPU based
  - UE4 OC
  - SDOC
- GPU based
  - Hardware Occlusion (HWOC)

1.3.2 Hardware Occlusion latency issue

HWOC in UE4 allows occlusion to be computed in the GPU. HWOC execution is asynchronous because it is performed using occlusion queries. These queries are usually performed on a given frame, and results are collected and used in subsequent frames.

HWOC is a great solution for situations where games run at high FPS and have GPU cycles to spare. One of the shortcomings of this method arises from its asynchronous nature since it introduces latency in its calculations.

This latency can be one or more frames which can lead to incorrectly culled objects when the camera moves too fast. This artifact can result in user perceived flashes or flickering.

In SunTemple HWOC demo, occludees are culled away incorrectly if the camera moves too quickly.

Compared to HWOC, SDOC can run in the render thread with zero frame latency. If Coherent mode is on, this latency can increase by less than 50% of frame latency.

1.3.3 Unreal Engine 4 Occlusion small resolution issue

UE4 can also compute Occlusion Culling in the CPU. This reduces the workload on the GPU. Computing this on the CPU can be costly and lead to simplifications in the culling resolution which result in imprecise culling.

HWOC can use the full image view resolution depth map to query object visibility. This is prohibitive on the CPU. If the depth map is not high enough, there can be negative culling due to lack of resolution.

The following image shows a UE4 Occlusion small resolution issue, where the default Occlusion Culling has a depth map resolution of 384x192. The white region between A and B is misculled. The two pillars (A and B) are rendered as connected in the small depth map.

Compared to UE4 Occlusion Culling, this culling issue would not be present on SDOC as a higher resolution depth map can be used. The following image shows an example depth map with a resolution of 1152x192. The depth of pillar A and pillar B are rendered separately.

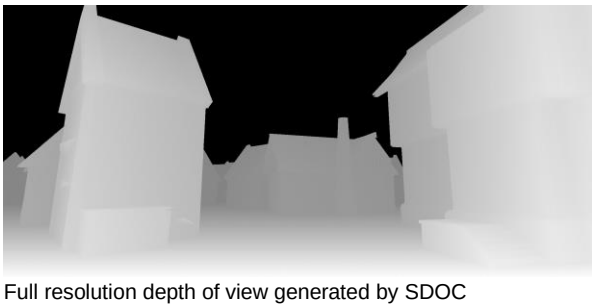
With a resolution of 1152x192, SDOC running in render thread could provide good culling results for the SunTemple demonstration.

1.3.4 Snapdragon Occlusion Culling

SDOC is a fast solution to the 3D visibility problem optimized for mobile platforms. Key highlights of SDOC include:

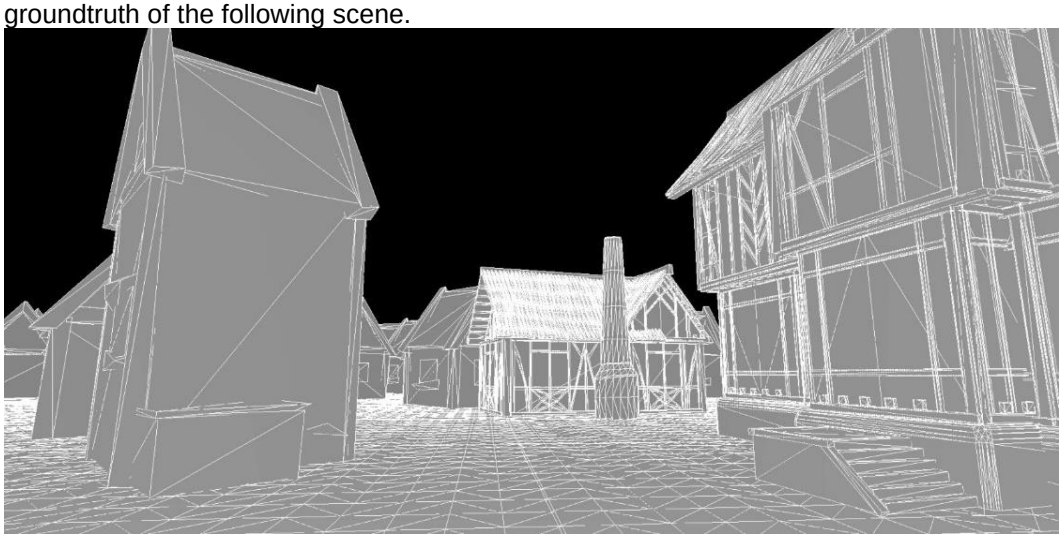
- Fast CPU approach exploring ARM NEON
  - Renders depth map accurately and quickly
    - 4-16 times faster compare with UE4 OC
  - Coherent/CoherentFast mode for temporal coherence
- Compared with GPU HWOC:
  - Could achieve zero frame latency while GPU HWOC causes at least 1 frame latency
  - Smaller draw call amount


Village pack demo view


Full resolution depth of view generated by SDOC

SDOC has the following advantages:

- Accurate
  - SDOC is a pixel level rasterizer that can generate highly accurate depth maps. The error of SDOC full mode is ~1 pixel. Bresenham's line mode are provided to show the groundtruth of the following scene.



- Fast
  - SDOC is fast enough to run in the render thread with a higher resolution. The memory usage is just around the resolution of the depth buffer.
- Low memory usage
  - SDOC memory usage is approximately the size of the depth buffer image. For example, if the depth buffer is 1024x256, the memory usage would be 1024x256x2, e.g., 512 KB.
- Adaptive
  - Developers can use baked occluder to fast the rasterization or use the coherent mode to reduce workload with the given SDOC APIs.
    - Pre-bake occluder can reduce workload by 20-30%.
    - Coherent/CoherentFast render mode can reduce workload, i.e., ~60-80% of Full render mode workload.
    - According to CPU capability, different resolutions can be set, e.g., 384x256, 512x256, 1024x256

2 User guide

2.1 Overview

1. Start SDOC.
   ```
   sdocInit(1024, 256, 1.0f);
   ```
   This command checks whether SDOC is supported on the device. SDOC is active for NEON supported ARM chips such as ARMv7 and ARMv8, SSE supported chips. It is optimized and work for little endian device only.
2. For every frame:
   a. Provide camera information for new frame.
      ```
      sdocStartNewFrame
      ```
   b. Submit occluders.
      ```
      sdocSubmitOccluder
      ```
   c. Query visibility of occludees with the occluder depth map.
      ```
      sdocQueryOccludees
      ```
3. End SDOC.
   ```
   sdocSetConfig(SDOC_DestroySDOC, 1);
   ```

2.2 Package contents

- /lib: Includes Android 32bit/64bit dynamic library, iOS 64bit static library, Mac(Intel CPU) 64bit dynamic library, Windows 64bit dynamic library
- UE4.25/UE4.26/UE4.27 patch

2.3 Unreal Engine integration

2.3.1 Apply patch and library

To integrate SDOC into Unreal Engine, apply the respective git patch to the UE4 codebase and rebuild the engine. SDOC ships with Unreal Engine patches but these can be applied and ported to other versions as needed.

1. Apply a git patch.
   ```
   cd [PATH_TO_UNREAL_ENGINE_SOURCE]
   git apply [PATH_TO_PATCH]/SDOCVxxx.patch
   ```
2. Copy Lib to UE Engine\Source\ThirdParty\Qualcomm\SDOC folder.

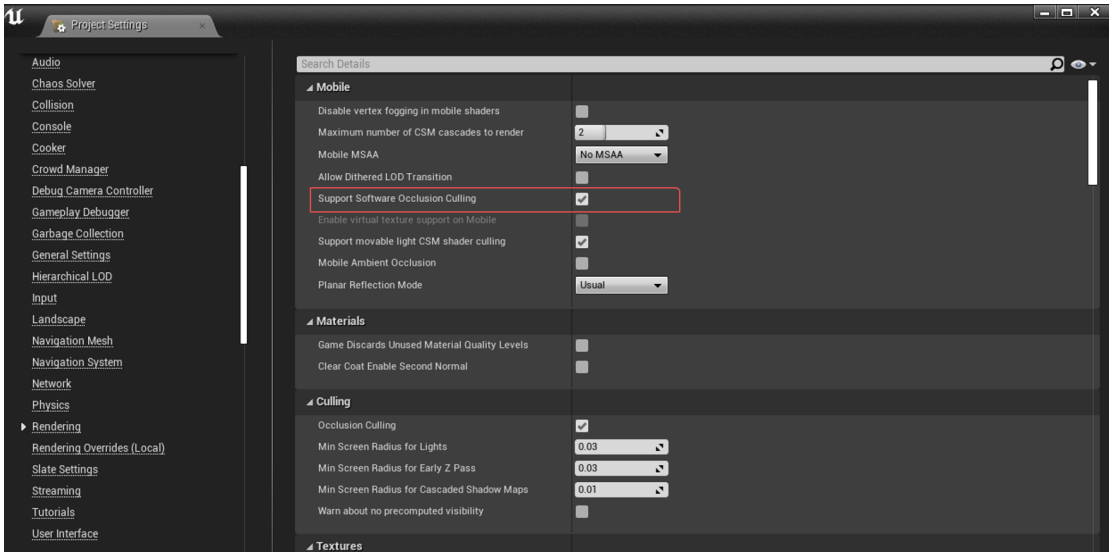
2.3.2 Enable SDOC in UE4

Enabling SDOC in UE4 follows the same instructions as the built-in UE4 Software Occlusion:

- Enable software occlusion culling

  In UE4, select **Project Settings** > **Rendering** > **Mobile** and select **Support Software Occlusion Culling**.

  Failing to check “Support Software Occlusion Culling” would create no occluders in mobile packages.

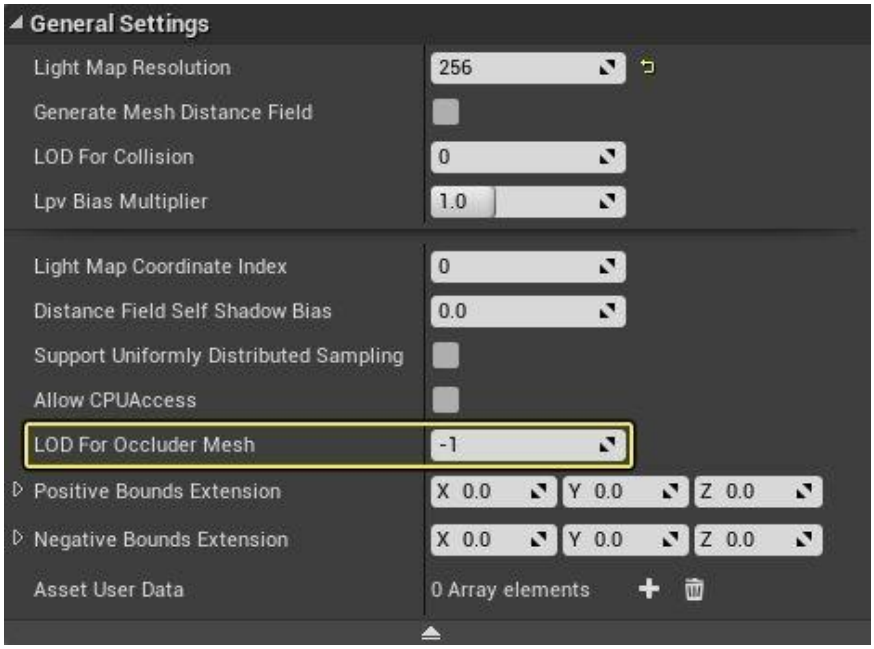


- Set up occluders in UE4

  SDOC requires developers to specify the occluder in a scene.

  In the Static Mesh Editor, use the **Details** panel to set the level of detail used for occlusion purposes.

  In UE4, the specified Level of Detail (LOD) will be treated as the occluder, and requires an **LOD for Occluder Mesh** value greater than zero.



- UE Integration tips

  In the UE Editor, select **UE Editor** > **Window** > **Developer Too**l > **Session Frontend** to send the following console commands.

Console command	Function
r.Mobile.AllowSDOC 1	Enables SDOC.
stat SDOC	Shows SDOC performance stats.
r.sdoc.VisualizeBuffer 1	Visualizes depth buffer, default 0 to turn off.
r.sdoc.RenderMode 0/1/2/12340	▪ If value is set to 0/1/2, different render mode is used. ▪ If value is set to 12340, different render type (line+mesh/point+mesh/line/point/mesh) would be toggled
r.sdoc.ShowMemStats 1	Shows memory usage which should be around depth buffer size
r.sdoc.CaptureFrame 1	Captures a frame for debugging
r.sdoc.Width 256/512	Updates depth buffer width on the fly. Changes width to height if the update target is height
r.sdoc.ShowOccludee 1	Shows occludee in depth map, default 0 to turn off
r.sdoc.ShowCulled 1	Shows culled occludees in the view, default 0 to turn off. This would disable depth map.
r.sdoc.SaveBuffer 1	Saves the depth buffer given that output path is set correctly

NOTE: For developer responsibilities, refer to SnapdragonOcclusionCulling.cpp and search "Developer todo".

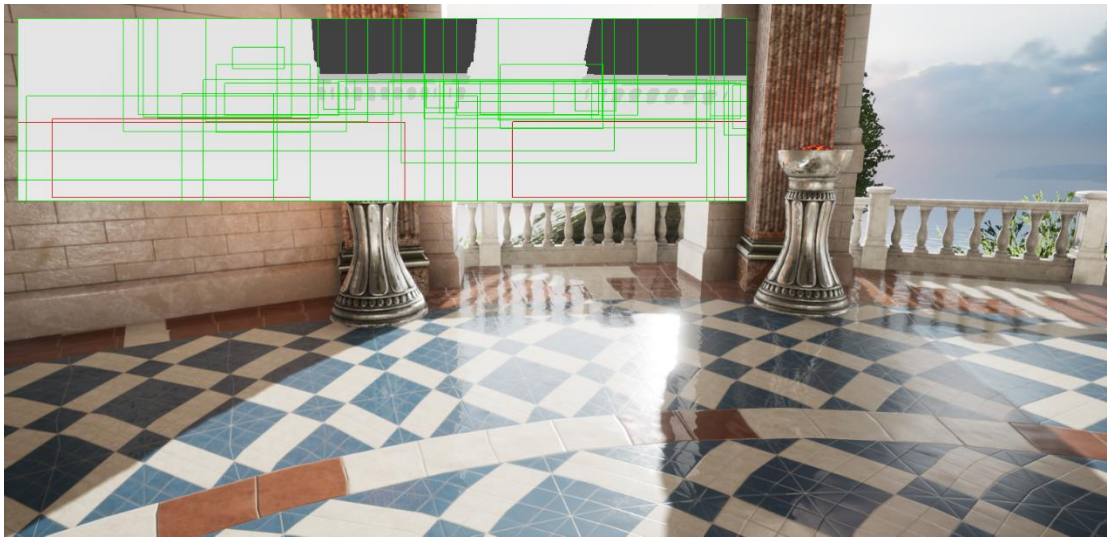
UE console commands example

1. Activate SDOC (r.mobile.AllowSDOC 1).
2. Visualize depth buffer (**r.sdoc.VisualizeBuffer 1**).

3. Show performance statistics (**stat SDOC**).

4. Show other line or point render type, such as Bresenham's line (**r.sdoc.RenderMode 12340**).

5. Show occlude in depth map (**r.sdoc.ShowOccludee 1**).



6. Show rejected occludees (**r.sdoc.ShowCulled 1**).

2.4 Snapdragon Occlusion API

API	Function
sdocInit	Starts SDOC.
sdocStartNewFrame	Starts new frame.
sdocRenderOccluder	Submits occluder drawcall.
sdocQueryOccludees	Batches query occludees' visibility.
sdocSet	General API to set configuration
sdocSync	General API to set/get information
sdocMeshBake	Pre-bakes mesh triangle to save calculation
sdocRenderBakedOccluder	Submits baked occluder drawcall.

2.4.1 General enum-related API

sdocSet and sdocSync are general API that communicate information between the game and SDOC. Developers can achieve a variety of features using these enumerations.

Enumeration	Example	Function
SDOC_RenderMode SDOC_RenderMode_Full	sdocSet(SDOC_RenderMode, SDOC_RenderMode_Full);	Sets Render mode to Full mode.
SDOC_RenderMode SDOC_RenderMode_Coherent	sdocSet(SDOC_RenderMode, SDOC_RenderMode_Coherent);	▪ Sets Render mode to Coherent mode. ▪ Explores interframe coherency.
SDOC_RenderMode SDOC_RenderMode_Coherent Fast	sdocSet(SDOC_RenderMode, SDOC_RenderMode_CoherentFast);	▪ Sets Render mode to Coherent Fast mode. ▪ Explores interframe coherency in Performance mode.
SDOC_RenderMode_Toggle RenderType	sdocSet(SDOC_RenderMode, SDOC_RenderMode_ToggleRender Type);	▪ Toggles among render type (line+mesh, vertices+mesh, line, vertices, mesh). ▪ For debug use.
SDOC_SetCCW	sdocSet(SDOC_SetCCW, 1);	States input mesh triangle in counter clockwise order, 0 for clockwise order.
SDOC_BackfaceCulling	sdocSet(SDOC_BackfaceCulling, 0);	Disable backface culling. Default is on with value 1, 0 to off.
SDOC_Get_IsSameCamera	sdocSync (SDOC_Get_IsSameCamera, pData);	▪ Format: pData bool *. ▪ Stores TRUE if camera pose is the same as previous frame, otherwise FALSE.
SDOC_Set_UsePrev DepthBuffer	sdocSet(SDOC_Set_UsePrevDepth Buffer, 1);	If camera and occluder state does not change, informs SDOC to use previously generated DepthBuffer.
SDOC_Get_Version	sdocSync(SDOC_Get_Version, pData);	▪ Format: pData format unsigned int * ▪ Gets SDOC version.
SDOC_Set_Coherent ModeSmallRotateDotAngle Threshold	sdocSync(SDOC_Set_CoherentMode SmallRotateDotAngleThreshold, pData);	▪ Format: pData float *. ▪ If cross-frame view direction dot value is smaller than this value, the camera is rotating at a slow pace. Default value: 0.9999f.

Enumeration	Example	Function
SDOC_Set_CoherentMode LargeRotateDotAngleThreshold	sdocSync(SDOC_Set_CoherentMode LargeRotateDotAngleThreshold, pData);	▪ Format: pData float *. ▪ If cross-frame view direction dot value is smaller than this value, the camera is rotating at a faster pace. Default value: 0.9995f.
SDOC_Reset_DepthMapWidth AndHeight	sdocSync(SDOC_Reset_DepthMap WidthAndHeight, pData);	▪ Format: pData unsigned int *, containing two values that indicate width and height. ▪ Resets width and height values.
SDOC_DestroySDOC	sdocSet (SDOC_DestroySDOC, 1);	Ends SDOC.
SDOC_Get_DepthBufferWidth Height	sdocSync(SDOC_Get_DepthBufferWidthHeight, pData);	Format: Input param of two element array unsigned int *. Gets width and height of occlusion buffer.
SDOC_Set_FrameCapture OutputPath	sdocSync(SDOC_Set_FrameCaptureOutput Path, pData);	▪ Format: pData char*. ▪ Sets output path in SDOC.
SDOC_Get_DepthMap	sdocSync(SDOC_Set_FrameCaptureOutput Path, pData);	▪ Format: Input param unsigned char* ▪ Dumps occluder depth map of SDOC.
SDOC_Save_DepthMap	sdocSync(SDOC_Save_DepthMap, pData);	Saves depth map to a file (input param format unsigned char*).
SDOC_ShowCulled	sdocSet(SDOC_ShowCulled, 1);	▪ Value = 0 or 1 ▪ Default = 0, 1 to reverse the query result.
SDOC_ShowOccludee InDepthMap	sdocSet(SDOC_ShowOccludeeInDepthMap, 1);	▪ Value = 0 or 1 ▪ Default = 0, hides/shows occludee information in depth map.
SDOC_Get_MemoryUsed	sdocSync(SDOC_Get_MemoryUsed, pData);	▪ Format: pData unsigned int *. ▪ Gets total memory used by SDOC (in KB).
SDOC_SetPrintLogInGame	sdocSync(SDOC_SetPrintLogInGame, pData);	▪ Format: pData unsigned int *, containing one value. ▪ If value=1, stores message for game to get later, else prints in SDOC.
SDOC_Get_Log	sdocSync(SDOC_Get_Log, pData);	▪ Format: pData char* of at least 256 char (uses first 256 bytes). ▪ Game gets log from SDOC. This works only if SDOC_SetPrintLogInGame Engine = 1.
SDOC_CaptureFrame	sdocSet(SDOC_CaptureFrame, 1);	Informs SDOC to capture one frame.
SDOC_CullAggressiveLevel	sdocSet(SDOC_CaptureFrame, 2);	Informs SDOC to cull occludee aggressively. Options 0/1/2/3, default 2.
SDOC_EnableOccluderPriority Queue	sdocSet(SDOC_EnableOccluderPrior ityQueue, 1);	Render occluders according to three priority queues. Options 0/1, default 0.

Enumeration	Example	Function
SDOC_FlushSubmittedOccluder	sdocSet(SDOC_FlushSubmittedOccluder, 1);	Force render submitted occluders.
SDOC_SimplifyMeshDuringBake	sdocSet(SDOC_FlushSubmittedOccluder, 1);	Apply planar neighbor quad merging before baking. Options 0/1, default 1.
SDOC_TerrainGridControlMergeDuringBake	sdocSet(SDOC_TerrainGridControlMergeDuringBake, 1);	Value = 0/1/2, default 2. 0/1/2 are corresponding to no grid guided merging/grid guided merging/grid guided aggressive merging.
SDOC_DebugSaveMeshSimplifiedModel	sdocSet(SDOC_DebugSaveMeshSimplifiedModel, 1);	Value = 0/1, default 0, for debug usage only. set to 1 to dump the simplified model

2.5 Setting recommendation

Based on recent experiments, resolution 1024x256 would be recommended.

The UE patch provides the option to enable SDOC from a render thread, where SDOC Full is selected as the Render mode setting, so that there would be zero latency. With SDOC Coherent Fast and SDOC Coherent modes, the performance would be better with approximately half frame latency.

Make sure bake mode is enabled as it would use minimal memory and provide 20%~30% performance improvement.

A References

A.1 Related documents

Title	URL
Resources	
Rasterizer	https://github.com/rawrunprotected/rasterizer

A.2 Acronyms and terms

Acronym or term	Definition
HWOC	Hardware Occlusion Culling
LOD	Level of Detail
SDOC	Snapdragon Occlusion Culling
UE4	Unreal Engine 4