Unlocking Native Performance

A developer's guide to maximizing performance and battery life in games for Windows laptops

By Dustin Graves, Jeremy Brennan, John Parsaie, and Nathan Frost

Contents

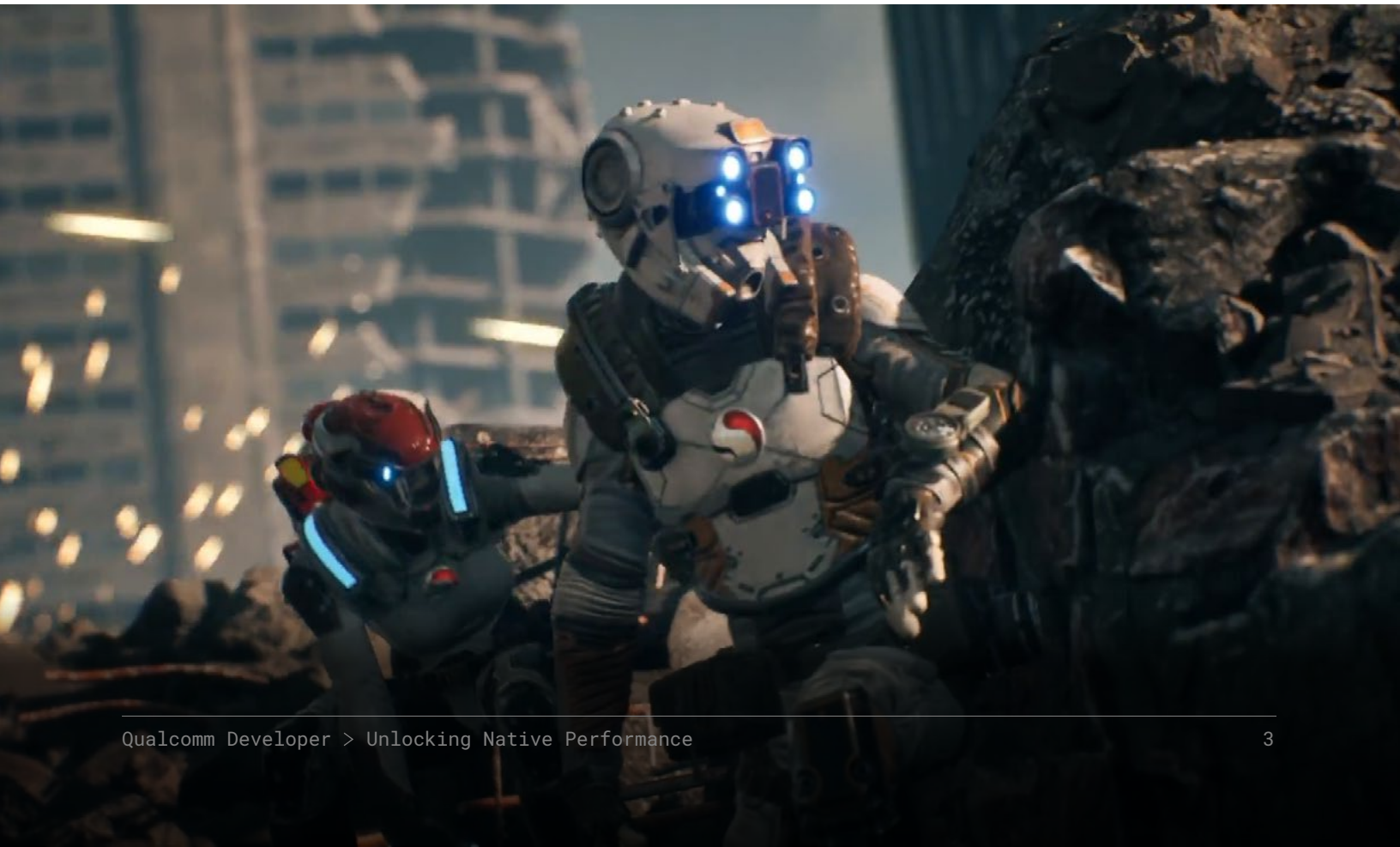
Introduction

If you develop games, chances are you're also a gamer. You push the software and the hardware to maximize performance and battery life while minimizing thermal profile and latency.

The longstanding, ongoing collaboration between Qualcomm Technologies and Microsoft has resulted in a blossoming developer ecosystem around Windows on Snapdragon. Compared to what Windows developers have known on traditional architectures, they now enjoy high levels of support on Snapdragon for their go-to tools. Those include development environments like Visual Studio and Eclipse, game engines like Unreal Engine and Unity, and an array of compilers, profilers and debuggers.

And commercially, Windows on Snapdragon is a compelling hardware platform on which to develop. With the Snapdragon® X Series processor, Qualcomm Technologies has extended its **performance pedigree to multiple laptop tiers**, so that your games will appeal regardless of hardware price point. All major game stores support Windows on Snapdragon, so your gamers can find, buy, install and run your titles easily.

This guide shows game developers how to create new Windows games and modify existing ones to take maximum advantage of the engines built into the Snapdragon X Series processors for laptops.



You'll learn

- What it takes to get your games running on Windows on Snapdragon
- How to enable Easy Anti-Cheat
- How to submit your titles to the most popular game stores

The next platform is now ready

Why should you go to the additional effort of getting your game to run on Windows on Snapdragon?

For one thing, your customers and gamers will find validated support for **thousands of Windows-based applications and games** (see Figure 1).

Figure 1: Representative game titles - Windows on Snapdragon

Run your game on WoS

They'll have a wide range of hardware choices; **laptops powered by Snapdragon** are available from top-tier OEMs in everyday retail channels at competitive prices.

Furthermore, as a developer you'll likely find that it's not that much additional effort.

The graphics processing unit (GPU) inside the Snapdragon latest X Series chips is the Qualcomm Adreno X2.

The Adreno X2 includes native driver support for DirectX 12.2 Ultimate, Vulkan 1.4, OpenCL 3.0 and SYCL, making more than 90 percent of the most popular game titles playable at launch.

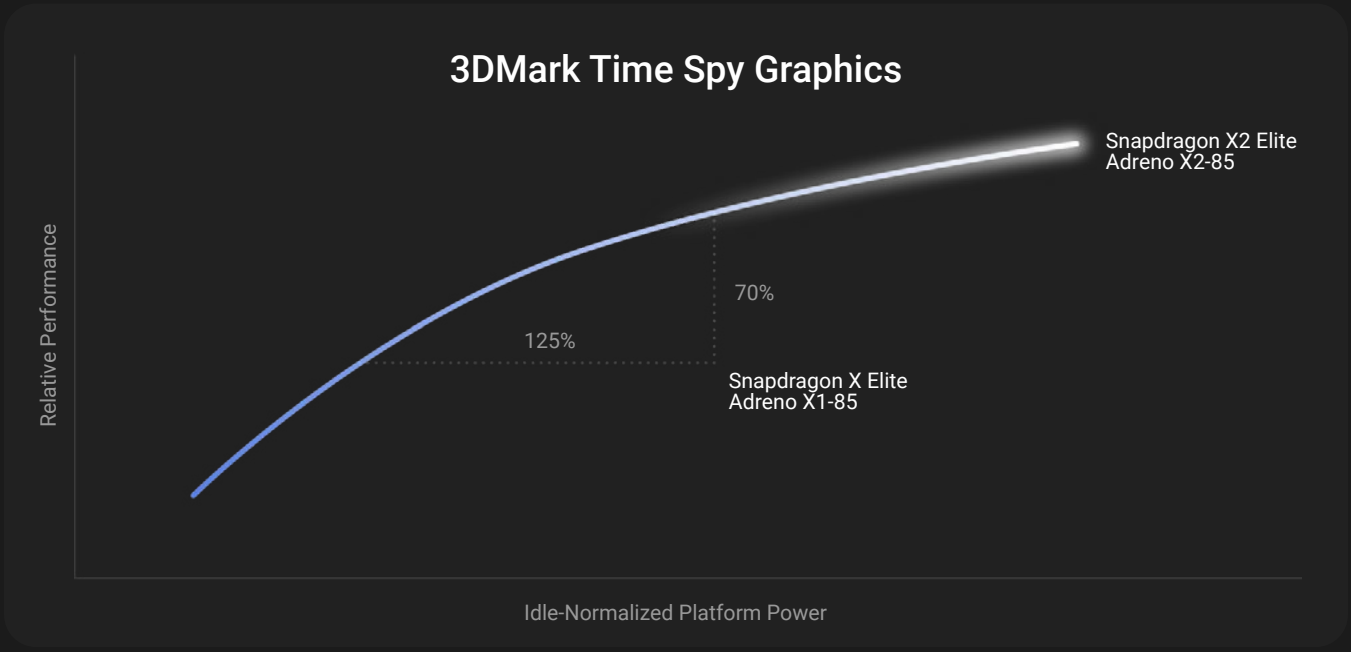
You'll work with the amenities you've come to expect on PCs, including regularly updated, native (even when running under emulation) drivers and a graphics control panel. Qualcomm Technologies is continually engineering Snapdragon and Adreno to expand app compatibility, both internally and with independent software vendors (ISVs) worldwide.

PROOF POINTS

High performance with power efficiency

Most of all, you can offer the gaming experience you want your customers and gamers to have, with high frame rates, low thermal profile and long battery life.

As shown in Figure 2, the Adreno X2 GPU in the Snapdragon X2 Elite can deliver up to 70 percent higher speed than the Adreno X1 GPU while consuming the same amount of power. That means up to 125% better performance per watt than Adreno X1.



Up to

70%

faster than Adreno X1 at the same power

Up to

125%

better performance per watt than Adreno X1

Figure 2: Adreno X2 GPU power efficiency

GPU performance & power is based on 3DMark Time Spy on Windows 11 OS, run in September 2025. Snapdragon X2 Elite (X2E-88-100) was tested using a Qualcomm reference design. Snapdragon X Elite (X2E-84-100) was tested using a Qualcomm reference design. Power and performance comparison reflects results based on measurements and hardware instrumentation of given devices. Lowest power and performance figures may not represent the lowest achievable platform power and performance.

Similarly, the Adreno X2 GPU in the Snapdragon X2 Elite Extreme delivers competitive frame rates on popular, graphics-intensive titles, as shown in Figure 3:

Those gains come at relatively low development cost to you. To make it as easy as possible for you to develop games for Windows on Snapdragon, Qualcomm Technologies has published the **Snapdragon Game Toolkit**. The toolkit is a set of guides (including the **Game Developer Guides**), components and tutorials for getting the most out of Snapdragon. Focused on optimized graphics and cross-functional development, the toolkit gives you the resources and documentation to create visually stunning, battery-efficient video games.

+50%

faster gaming vs.
Intel Core Ultra Series 2

+29%

faster gaming vs.
AMD Ryzen AI 9

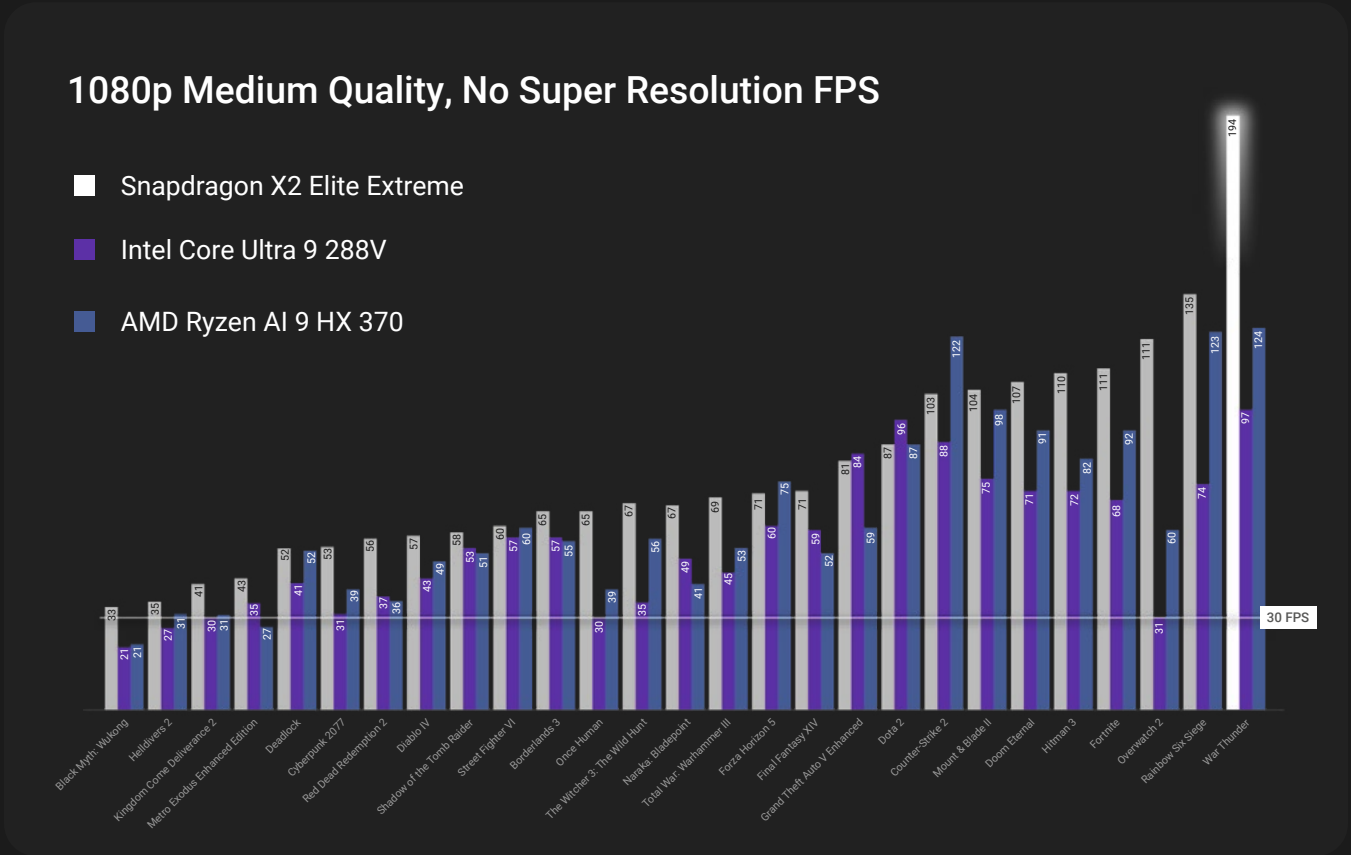


Figure 3: Adreno X2 GPU competitive frame rates

Based on average FPS results of the listed games run by Qualcomm Technologies in October 2025 using Qualcomm reference design with Snapdragon X2 Elite Extreme (X2E-96-100), Dell XPS 13 with Intel Core Ultra 9 288V, and ASUS Vivobook S14 with AMD Ryzen AI 9 HX 370.

Alien: Rogue Incursion

EVOLVED EDITION

Survios Game Studios has taken Alien: Rogue Incursion – Evolved Edition from a VR title to gaming consoles and the desktop.

This marks a major milestone not only for the Alien franchise but also for game development on Windows on Snapdragon.

Survios packaged their binaries using ARM64EC, a hybrid ABI that allows native ARM64 code to interoperate with x64 components. This approach simplifies development and ensures compatibility with existing Windows libraries and tools.

Improvements to the game include NEON intrinsics support, device profile integration, bootstrap package enhancements and driver awareness.

1

How do I get my game to run on Windows on Snapdragon?

The **Snapdragon Game Toolkit** includes a tutorial titled **Get Started with Windows on Snapdragon Development** with instructions on setting up and enabling your environment.

First, consider the build target you envision for your game.

CPU PATH

Do I choose x64 Emulation, ARM64EC or ARM64?

You have options for how your game will run on CPU. Here are some general guidelines:

1. You can get high performance and extensive, robust support with emulation. More than 1,000 apps and games running in emulation are currently supported.
2. Emulation may already be more than sufficient for your game. Many (though not all) games are GPU-bound. If your game is one of them, then optimizing CPU workloads with ARM64 or ARM64EC (“native”) will not help you increase frames per second.
3. Choosing native ARM64 can also help in anticipating advances in GPU hardware that may make your game CPU-bound in the future.
4. Note that not all of the popular libraries required by games have been ported to ARM64. Therefore, until ALL libraries (internal and external) that your game uses have been ported to ARM64, the best way to minimize development risk and shorten your time to market is to choose ARM64EC.

With those guidelines in mind, consider the following options.

x64 Emulation

Microsoft's Prism provides a stable and performant environment for executing x64 code on Windows on Snapdragon devices. It supports common CPU extensions, including SSE up to SSE4 and AVX up to AVX2. However, there are some limitations. Emulation effectively executes at the user level and will be incompatible with x64 kernel drivers, which may be used by anti-cheat software.

Recommendations

- Run preliminary tests to gauge your game's performance in emulation. Well-threaded applications tend to perform better.
- **Detect the Adreno** GPU in the Snapdragon chipset and set the initial settings for a good user experience. The vendor ID is 0x4d4f4351, and the user experience depends on your default setting for an "unknown" GPU. Note that `D3D12_FEATURE_DATA_ARCHITECTURE1` contains a Boolean value to indicate the underlying GPU supports a unified memory architecture (UMA), which will be set to TRUE for integrated Adreno GPUs.
- Prepare to port any kernel drivers to ARM64; for example, if your game uses an anti-cheat system at the kernel level (more on anti-cheat later).
- Integrated GPU (UMA) optimization work. In most cases, the optimizations you make to run your games on other vendors' integrated GPUs will work on the Adreno GPU.

ADVANTAGES AND LIMITATIONS

ARM64EC

ARM64EC (Emulation-compatible) is a hybrid application binary interface (ABI) that can mix instruction sets (see Figure 4), allowing native ARM64 code to interoperate with x64 components. An ARM64EC module uses ARM64 native instructions, and is interoperable with x64 emulated code, but does not execute within the emulation system.

If you're trying to future-proof your game, you can use ARM64EC to take the code in your existing app from emulated to native in increments over time. At each step, you can enjoy good performance without the effort and risk of recompiling your game in one fell swoop. The ABI simplifies development and ensures compatibility with existing Windows libraries and tools. (See the sidebar on Alien: Rogue Incursion – Evolved Edition. Vendor Survios packaged the binaries for the game using ARM64EC.)

All Windows libraries and Qualcomm Technologies drivers have an ARM64EC-compatible version. The ABI is designed for nearly native CPU performance and using it means you don't need to port dependencies on outside components. And, most compellingly in the world of game development, there currently is no direct path to publish a fully native ARM64-compatible game through popular game stores due to x64 runtime incompatibilities.

ARM64EC provides interoperability with existing game store runtimes, providing a way to publish a game with nearly native CPU performance.

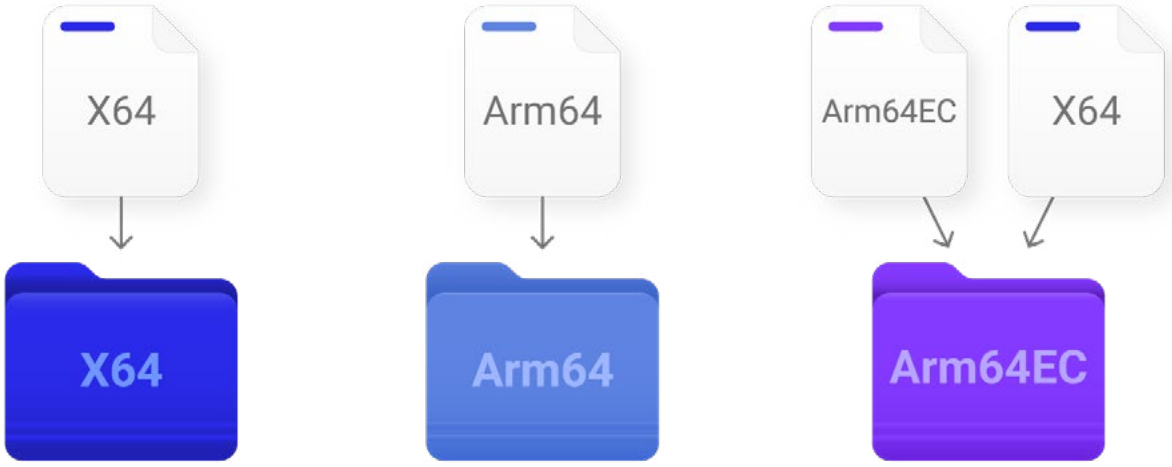


Figure 4: ARM64EC

However, ARM64EC cannot be used for kernel drivers; any X64 kernel driver must be ported to ARM64. (See below for information on using anti-cheat systems with Windows on Snapdragon.)

Also, in the hybrid ARM64-x64 environment, your build pipeline can become more complex. And you'll need to convert SSE/AVX instructions to NEON code.

Recommendations

- ARM64EC requires Visual Studio 2022 or later. Use the latest version for best support.
- Be mindful of the cost to transition from x64 to ARM64EC. You will likely see a small performance penalty during the transition from x64 to ARM64EC, but the code inside an ARM64EC module uses ARM64 instructions and runs at nearly native speed. Try to avoid frequent-back-and forth transitions between x64 and ARM64EC code, especially inside tight loops or on hot code paths.
- The **Windows SDK** includes faster SSE/AVX/AVX2 software intrinsics for even easier porting to Windows on Snapdragon. The intrinsics replace SSE instruction with C code or NEON instructions. Still, you may see better performance if you convert SIMD code to NEON.

ARM64

Porting your game to native ARM64 code is the most ambitious CPU path for running on Windows on Snapdragon. With native code, of course, you and your gamers can substantially reduce CPU use and power consumption.

But, as noted above, not all the most popular libraries required by games have been ported to ARM64. Unless you are certain that ALL required libraries (internal and external) have been ported to ARM64, Qualcomm Technologies recommends choosing ARM64EC for nearly native performance.

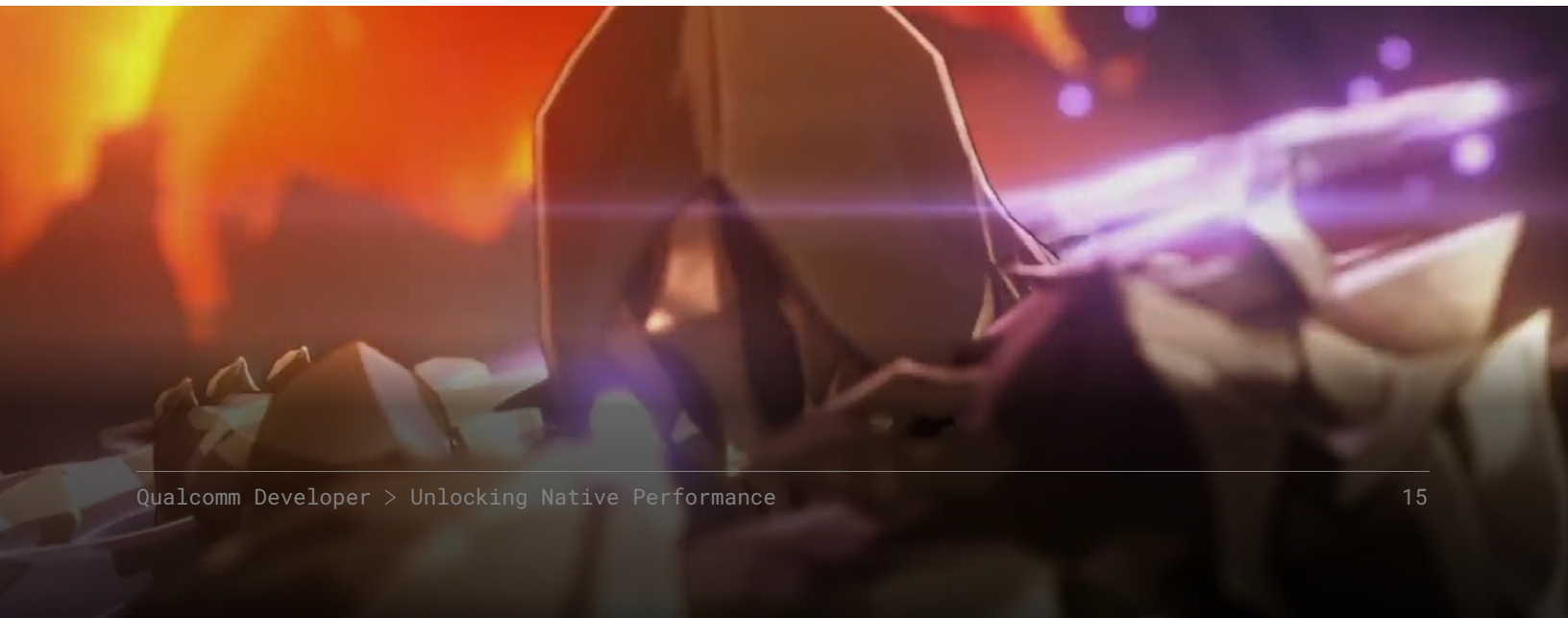
In the short term, ARM64EC will help give you the CPU performance gain you need without the development risk of being blocked by a library that has not yet been ported to ARM64. Once all the libraries you need have been ported, you can move from ARM64EC to ARM64.

For highest performance on GPU, you'll still need to convert SSE/AVX instructions to NEON code, though it's likely that you'll see a performance boost only if overheating was your main problem.

Recommendations

- First, make sure that your game is CPU-bottlenecked.
- To help you convert your SSE code to NEON, use a library like **SIMDe**. The best performance will require that you perform your conversion manually, but the library will help you target the part to be converted.

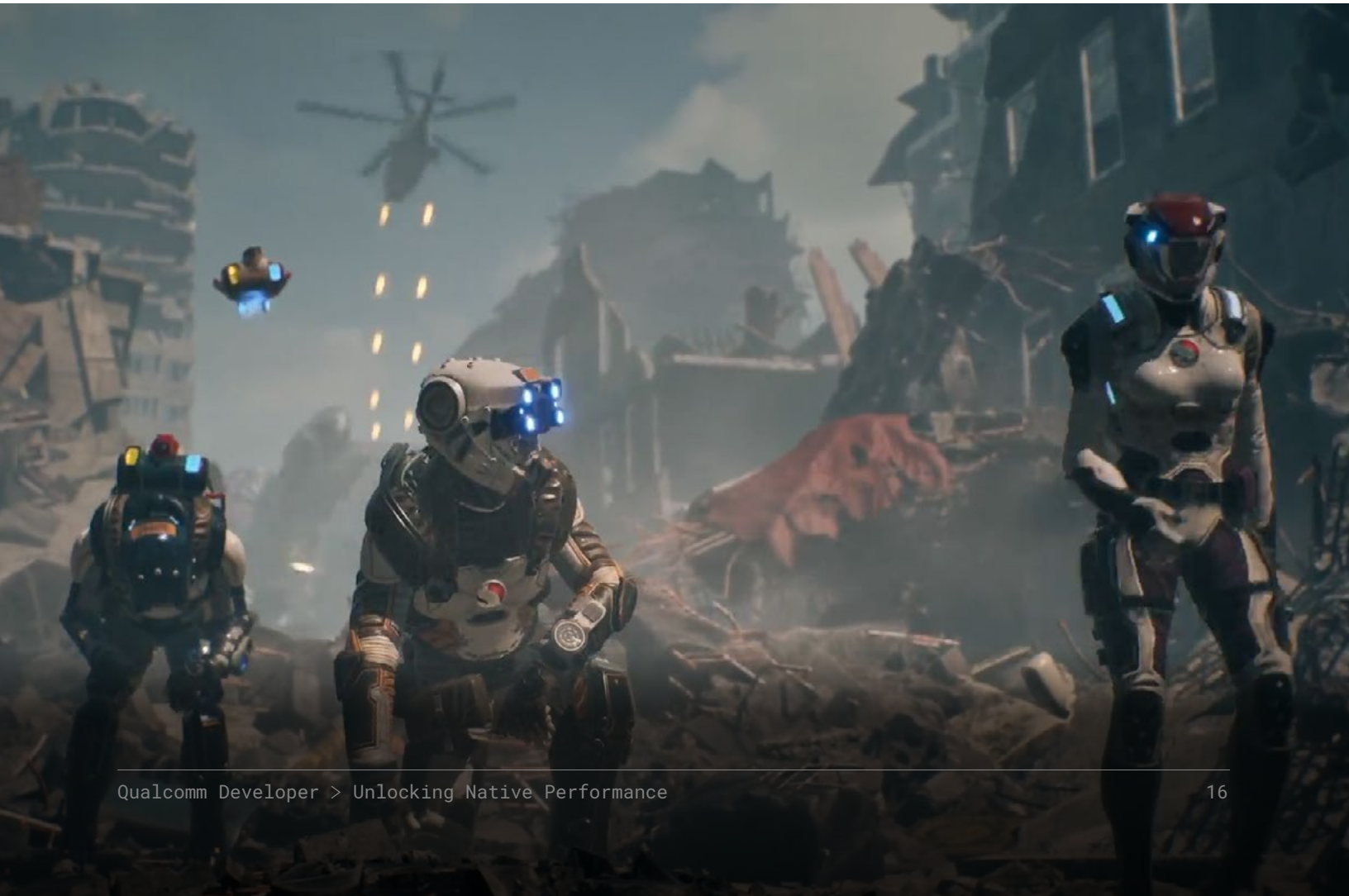
Based on those advantages, limitations and recommendations, you can choose the CPU path for your game.



No changes required to code or assets

You follow a single GPU path for laptops powered by Snapdragon, with APIs, ongoing support and native drivers for DirectX 12.2 Ultimate, Vulkan 1.4, OpenCL 3.0 and SYCL. If your game uses a supported API, then you can rely on your graphics code and path working, with no changes required to your traversal code.

You won't need to modify your shaders; you can use the same shader assets you use on other GPUs. And with DirectX, you can take advantage of **Render Passes for a performance boost**.



TOOLSET

Unreal Engine

Surveys show **strong preferences for Unreal Engine**. Windows on Snapdragon runs games built with Unreal Engine without a hitch.

Prerequisites

When building for Windows on Snapdragon with Epic Games's Unreal Engine, note these prerequisites:

- Install the set of **Visual Studio Installer** components for ARM64-compatible systems that match your Unreal engine version, such as "MSVC v143 - VS 2022 C++ ARM64 build tools (v14.44-17.14)"
- **Unreal Engine** – First, get the appropriate access and install toolchains as described in **Unreal Engine Getting Started Prerequisites**. Then follow the guide on how to **build Unreal Engine from source** for the latest in support for Windows on Snapdragon.
  Note that experimental support for Windows on Snapdragon began in Unreal 5.5; as such we recommend beginning your project with version 5.5 or with the latest officially supported version. If you don't want to update from Unreal 5.4 to a more recent engine, the documentation for the Snapdragon Game Toolkit includes "**Unreal Engine 5 for Windows on Snapdragon**," with links to Epic's onboarding steps and a branch on GitHub to a Snapdragon-specific build.
- **ARM64EC** – Microsoft Learn has released "**Build and port apps for native performance on Arm**," with a walkthrough on getting started with ARM64EC-compatible systems.

Unreal Engine

Project setup

Select “Windows on Arm” in the Build window of Unreal Editor.

Plugins

One component of the Snapdragon Game Toolkit is the collection of **Snapdragon Game Plugins for Unreal Engine**. Plugins currently support **Snapdragon® Game Super Resolution (GSR)**, the Qualcomm® Neural Processing SDK and the Qualcomm® Shadow Denoiser.

Optimizing

Epic Games provides tools for measuring the performance of your game as you move to Windows on Snapdragon. The **Memory Insights feature** offers detailed information about memory allocation and deallocation. **Profiler** monitors game performance by collecting and tracking data you can use to identify possible sources of bottlenecks.

Detecting Snapdragon

Device Profile Fragments can be set up for any CVars or specific behaviors that differ on Snapdragon hardware. The code below implements a check for generic Snapdragon hardware (may not be accurate for all models), Snapdragon X Series hardware, and Snapdragon X2 Series hardware.

SAMPLE CODE

DefaultDeviceProfiles.ini

```
[Windows DeviceProfile]
DeviceType=Windows
BaseProfileName=PC_Base

;-------------------------------------------------------------------------------
; Windows [Device Profile Matching Rules]
;-------------------------------------------------------------------------------
+MatchingRules={
(
        RuleName="SnapdragonX1",
        IfConditions=((Arg1="$(SRC_PrimaryGPUDesc)", Operator="CMP_Regex", Arg2="(?i).*Xl\-\d*.*")),
        OnTrue=((AppendFragments="Snapdragon_Profile"))
),
(
        RuleName="SnapdragonX2",
        IfConditions=((Arg1"$(SRC_PrimaryGPUDesc)", Operator="CMP_Regex", Arg2="(?i).*X2\-\d*.*")),
        OnTrue=((AppendFragments="Snapdragon_Profile"))
),
(
        RuleName="SnapdragonGeneric",
        IfConditions=((Arg1="$(SRC_PrimaryGPUDesc)", Operator="CMP_Regex", Arg2="(?i).*X\d*\-\d*.*")),
        OnTrue=((AppendFragments="Snapdragon_Generic"))
)}

;Example Profile
[Snapdragon_Profile DeviceProfileFragment]
+CVars=r.RayTracing=0
+CVars=r.FidelityFX.FSR3.ForceDX12RHI=1
+CVars=r.FidelityFX.FSR3.Prefer16x8GroupSize=1
[Snapdragon_Generic DeviceProfileFragment]
+CVars=r.RayTracing=0
```

Also implemented is a function you can call from C++ that queries whether the current device matches the device profile fragment.

SAMPLE CODE

SVRUtils.cpp

```
bool USVRUtils::IsSnapdragon()
{
#if !(UE_BUILD_SHIPPING)
	if (!!CVarSpoofSnapdragon.GetValueOnAnyThread())
	{
		return true;
#endif

	static UDeviceProfileManager& DeviceProfileManager = UDeviceProfileManager::Get();
	// e.g. “Fragmentl,Fragment2, [tag]Fragment3”
	FString MatchedFragments = DeviceProfileManager.
GetActiveDeviceProfileMatchedFragmentsString(true, false, false);

	return MatchedFragments.Contains(TEXT(“Snapdragon”), ESearchCase::IgnoreCase);
}
```

Profiling/Tuning

Besides the toolset-specific profiling tools given in the “Optimizing” section above, several prominent profiling tools can help you identify performance bottlenecks and find their root cause:

- **Snapdragon® Profiler** – For small triangle processing, complex shaders and pipeline bubbles, with analysis of CPU and GPU performance. Snapdragon Profiler is designed to give you insight into performance of the operating system, Qualcomm Oryon™ CPU, Adreno GPU, Qualcomm® Hexagon™ NPU and Qualcomm Spectra™ image signal processor.
- **Superluminal** – Recommended as CPU profiler tool when first analyzing performance. For instrumentation and high-frequency sampling, Superluminal offers source and disassembly view along with multi-threading analysis. The tool works with ARM64EC and ARM64-compatible systems.
- **PIX** – For profiling expensive materials, excessive scalability settings and complex lighting setups. PIX supports ARM64-compatible systems and includes a plugin specifically for profiling your game’s performance on Adreno GPU.

2

How do I enable Easy Anti-Cheat on Windows on Snapdragon?

Easy Anti-Cheat (EAC) is part of **Epic Online Services** from Epic Games, whose goal is to make it easier and faster for developers to manage and scale high-quality games. EAC is designed to thwart hacking and cheating in multiplayer PC games through the use of hybrid anti-cheat mechanisms.

To enable Easy Anti-Cheat on Windows on Snapdragon, follow these steps:

1 Download the Epic Online Services (EOS) SDK

It contains the prerequisites for EAC, whether you develop with Unreal Engine, Unity or another gaming engine. Snapdragon support for x64 binaries that run using emulation is available from EOS 1.17.1.3 and beyond.

2 Update your game

At a minimum, update the anti-cheat bootstrapper, `start_protected_game.exe`. If possible, update all of your game's EOS SDK files as you normally would, including all `AntiCheatTools` files.

3 Test and activate the anti-cheat client module

With your game files updated as in step 2, follow the instructions to test the module and activate it in the EOS Developer Portal.

3

How do I post my Windows on Snapdragon game to the game stores?

Your gamers' perspective

Using Windows 11, gamers can find, buy, install and run games from all major game stores – such as Steam, Epic Games Store, Xbox App, itch.io, Battlenet, GOG, EA Play, Ubisoft Connect – as they always have.

All features of the game stores, including in-game overlay and social, function normally for Windows on Snapdragon games.

Your development perspective

Most game stores support x64 executables in emulation (x64 emulation). As noted above, that option shortens your development path to reaching the growing Windows on Snapdragon user base.

Recommendation

Regardless of the game store, publish a single package that contains binaries for both x64 and ARM64EC. Note that that does not increase the size of your game significantly. In our experience, binary sizes for ARM64EC- are within 5 to 10 percent (plus or minus) of the x64 binaries.

A small bootstrap launcher that has implemented **Windows on Snapdragon Detect** is then used to determine whether the user's system is running Windows on Snapdragon and whether the application is running in x64 emulation. The launcher then runs the appropriate binary.

Update packaging scripts

The following instructions will depend on how your project is built.

You can trigger the internal build scripts below from the command line or from CI/CD. The scripts are updated to call UAT with `-architecture={arch}`, where `{arch}` can be `[ x64 , arm64ec , arm64 ]`. Also added is a new build target that builds the Win64 (x64) package then builds the ARM64EC binaries and copies them into the same binaries directory.

The updated scripts support the following optional features:

- Upload ARM64EC symbols to our symbol store
- Add support to Gauntlet for launching the correct binaries on ARM64-compatible systems

Modify bootstrap launcher

By default, Unreal Engine creates a bootstrap executable during packaging for Win64 for launching a packaged game. The code is in `Engine\Source\Programs\Windows\BootstrapPackagedGame\Private\BoostrapPackagedGame.cpp` . The script below is modified to check whether the game is running on an ARM64-compatible platform and to launch the correct executable based on the detected platform.

SAMPLE CODE

BootstrapPackagedGame.cpp

```
//#SurviosBeginChange
static bool ShouldLaunchARM64EC()
{
            HANDLE hProcess = GetCurrentProcess();
            USHORT ProcessMachine = 0;
            USHORT NativeMachine = 0;

            // Use IsWow64Process2 if available
            typedef BOOL(WINAPI* LPFN_ISWOW64PROCESS2)(HANDLE, USHORT*, USHORT*);
            LPFN_ISWOW64PROCESS2 fnIsWow64Process2 =
                        (LPFN_ISWOW64PROCESS2)GetProcAddress(GetModuleHandleW(L"kernel32"), "IsWow64Process2");

            if (fnIsWow64Process2)
            {
                        if (fnIsWow64Process2(hProcess, &ProcessMachine, &NativeMachine))
                        {
                                    switch (NativeMachine)
                                    {
                                    case IMAGE_FILE_MACHINE_ARM64:
                                                return true;
                                    default:
                                                MessageBoxW(NULL, L"Unknown or unsupported architecture", L"Architecture", MB_OK);
                                                break;
                                    }
                                    return false;
                        }
            }

            SYSTEM_INFO SysInfo = {};
            GetNativeSystemInfo(&SysInfo);

            switch (SysInfo.wProcessorArchitecture)
            {
            case PROCESSOR_ARCHITECTURE_ARM64:
                        return true;
            default :
                        MessageBoxW(NULL, L"Unknown or unsupported architecture", L"Architecture", MB_OK);
                        break;
            }

            return false ;
}
//#SurviosEndChange

int WINAPI wWinMain(_In_ HINSTANCE hinstance, _In_opt_ HINSTANCE hPrevinstance, _In_ LPWSTR Cmdline, _In_ int ShowCmd)
{
            (void)hPrev instance;
            (void)ShowCmd;

            //Get the current module filename
            WCHAR CurrentModuleFile[MAX_PATH];
            GetModuleFileNameW(hInstance, CurrentModuleFile, MAX_PATH);

            //Get the base directory from the current module filename
            WCHAR BaseDirectory[MAX_PATH];
            PathCanonicalize(BaseDirectory, CurrentModuleFile);
            PathRemoveFileSpec(BaseDirectory);

            //Get the executable to run
            WCHAR* ExecFile = ReadResourceString(hInstance, MAKEINTRESOURCE(IDI_EXEC_FILE));
//#Surviosbeginchange
            if (ShouldLaunchARM64EC())
            {
                        const WCHAR* ARM64EC_STR = L"arm64ec";
                        size_t ExecFileLen = wcslen(ExecFile);
                        //Find the last dot in the filename
                        WCHAR* Dot = wcsrchr(ExecFile, L'.');
                        size_t BaseLen = Dot ? (Dot - ExecFile) : ExecFileLen;
                        //Calculate new length: base + arm64ec + extension + null
                        size_t NewLen = BaseLen + wcslen(ARM64EC_STR) + (Dot ? wcslen(Dot) : 0) + 1;
                        WCHAR* NewExecFile = new WCHAR[NewLen];
                        wcsncpy_s(NewExecFile, NewLen, ExecFile, BaseLen);          //Copy base
                        wcscat_s(NewExecFile, NewLen, ARM64EC_STR);                 //Add arm64ec
                        if (Dot)
                        {
                                    wcscat_s(NewExecFile, NewLen, Dot);             //Add extension
                        }
                        //Check if file exists
                        DWORD Attributes = GetFileAttributesW(NewExecFile);
                        const bool bIsValidFile = (Attributes != INVALID_FILE_ATTRIBUTES) && !(Attributes & FILE_ATTRIBUTE_DIRECTORY);
                        if (bIsValidFile)
                        {
                                    delete[] ExecFile;
                                    ExecFile = NewExecFile;
                        }
                        else
                        {
                                    delete[] NewExecFile;
                        }
            }
//#SurviosEndChange
            ...
}
```

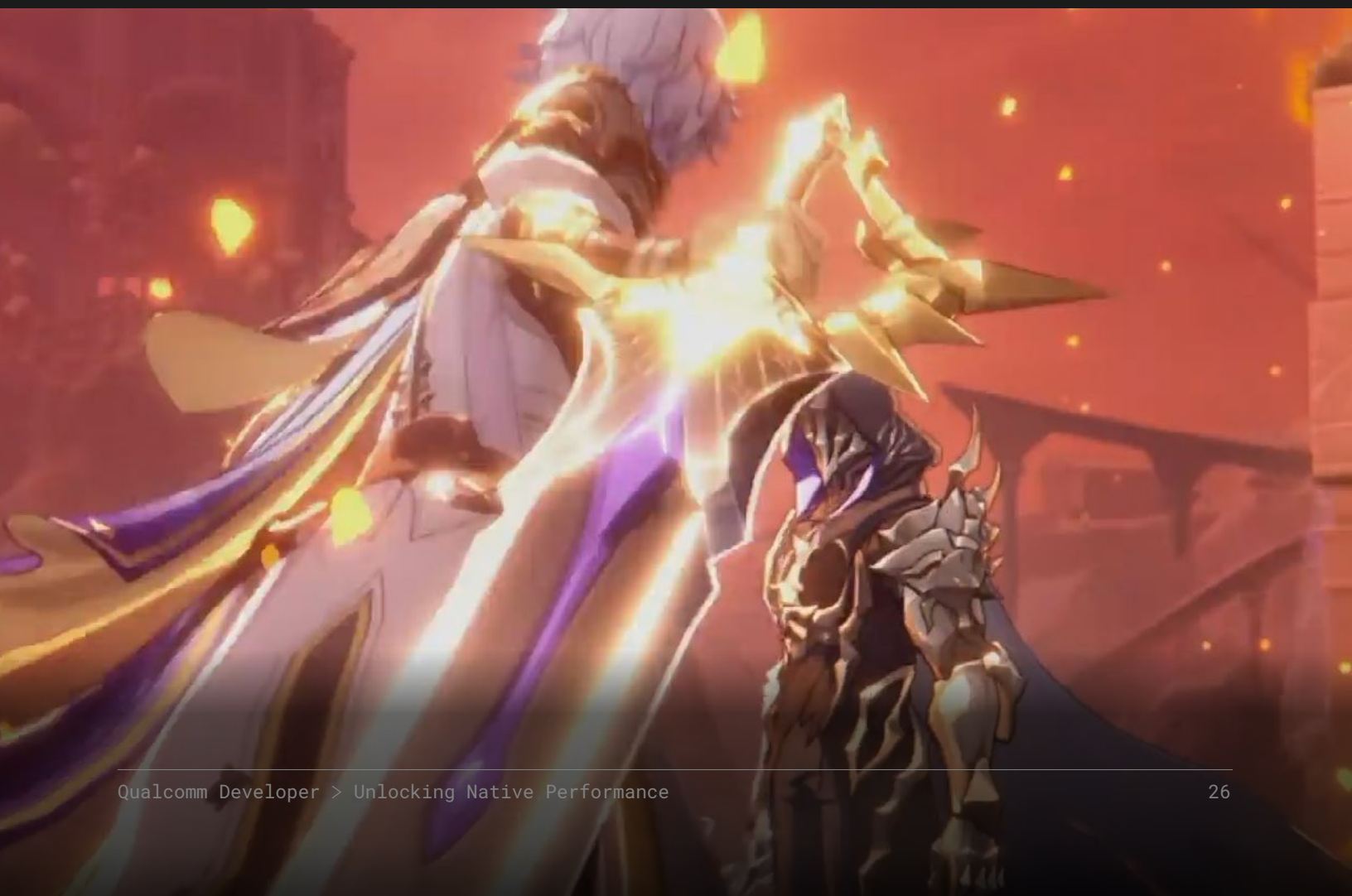
SAMPLE CODE

Arm64Ec.cmake

```
# Arm64Ec.cmake
set(CMAKE_GENERATOR_PLATFORM ARM64EC                    CACHE STRING “” FORCE)

# Force every cl.exe compile to /arm64ec
set(CMAKE_C_FLAGS_INIT             “/arm64EC”           CACHE STRING “” FORCE)
set(CMAKE_CXX_FLAGS_INIT    “/arm64EC”           CACHE STRING “” FORCE)

# Mark the library as ARM64X (link.exe /MACHINE:ARM64X)
set (CMAKE_STATIC_LINKER_FLAGS     “/MACHINE:ARM64X” CACHE STRING “” FORCE)
```



Game store specifics

Following are procedures for posting games to the most prominent game stores.

Steam

As described above, package your ARM64EC binaries with the x64 game's SKU and update the build to dynamically load ARM64EC binaries on the detected Snapdragon platform. Then, follow the standard process to post your game to Steam.

Epic Games Store

Architecture Support: Epic Games Store primarily targets x64, but you can distribute ARM64 builds as optional downloads.

Requirements: Package your game as a standard installer (EXE/MSI) or zip. Epic does not enforce strict architecture rules, but you should clearly label the ARM64 build.

Submission: Use Epic Developer Portal.

Provide separate builds for x64 and ARM64-compatible systems if you want broader reach.

Install and test

The best way to ensure a smooth experience for your gamers is to test on the hardware they'll be using. Laptops with **Snapdragon X Series** are designed for Windows on Snapdragon and, as described in this paper, they support the features that you, your fellow developers and your gamers want.

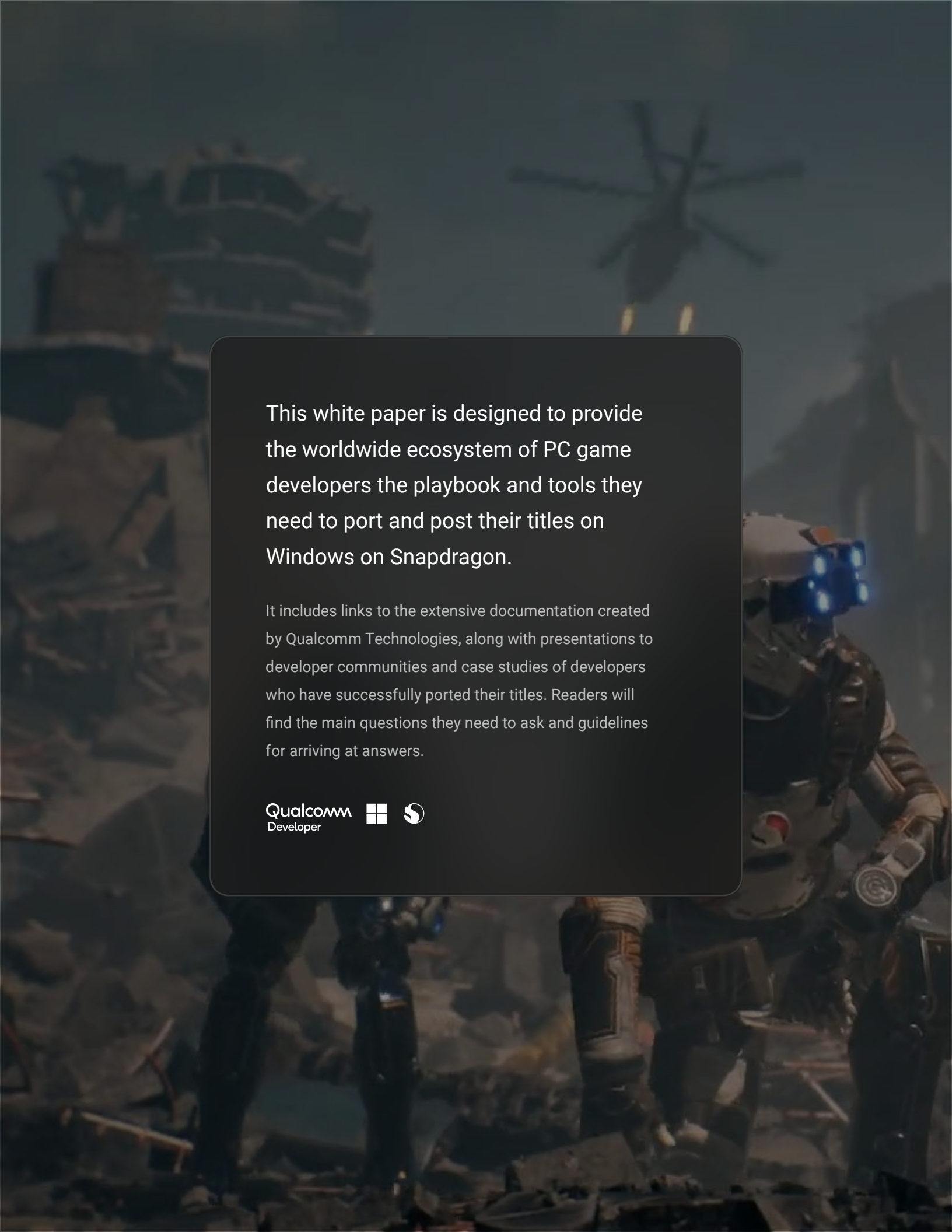
Conclusion

Ongoing collaboration between Qualcomm Technologies and Microsoft has brought a wide range of familiar developer tools and game engines to a new processor architecture. Windows on Snapdragon is a ripe opportunity for game developers seeking to maximize performance and battery life while minimizing thermal profile and latency on Windows laptops.

With the guidelines and recommendations in this paper, you can set off on the development path to your next platform. Besides the hardware, software and documentation, Qualcomm Technologies provides technical support through a broad spectrum of communities and channels including these:

- **Discord** →
- **Support forum** →
- **Blog** →
- **Newsletter** →
- **YouTube** →
- **GitHub** →

Let us know how we can help you succeed on Windows on Snapdragon



This white paper is designed to provide the worldwide ecosystem of PC game developers the playbook and tools they need to port and post their titles on Windows on Snapdragon.

It includes links to the extensive documentation created by Qualcomm Technologies, along with presentations to developer communities and case studies of developers who have successfully ported their titles. Readers will find the main questions they need to ask and guidelines for arriving at answers.

Qualcomm Developer