Evaluation environment for Windows games 2.0 README

Overview

The evaluation environment for Windows games is now a subset of the expanded Game Porting Toolkit. The evaluation environment helps game developers try out their existing Windows games right on Apple Silicon Macs running macOS 14 or higher. Using community projects that bundle the evaluation environment, or building your own environment using the included binaries and installation and configuration instructions, you can run your game to get a sense of how it can feel and play right away, even before you begin your porting journey.

Requirements

- The evaluation environment for Windows games only runs on Apple Silicon Macs running macOS 14 Sonoma or higher.
- Translated games require more resources, so developer-focused Macs with 16GB of RAM or more are recommended.
- The provided macOS graphics bridge libraries must be configured with a custom version of the Wine translation environment in order to create your evaluation environment.
- Instructions and scripts for building and configuring the custom version of Wine are included here.

Installation and Setup

Option 1: Use a pre-built evaluation environment

While simple to use once configured and installed, the evaluation environment for Windows games can take time to configure and use correctly depending on your proficiency and comfort with command-line tools and your experience using translation tools like WINE. The following free and commercial products incorporate the supplemental evaluation layers from this distribution within a pre-built WINE environment.

- Dean Greer's (aka GCenX) **homebrew-wine** (https://github.com/Gcenx/homebrew-wine) and **game-porting-toolkit** (https://github.com/Gcenx/game-porting-toolkit) casks - use **`brew install --cask --no-quarantine gcenx/wine/<XYZZY>`** to easily install either of the more complete environments and the graphical translation layer.
- Isaac Marovitz's **Whisky** (https://getwhisky.app and https://github.com/Whisky-App/Whisky) - Whisky is a community supported easy-to-user graphical interface that creates an evaluation environment and installs and configures the graphical translation layer.

- CodeWeaver's **CrossOver**. (https://www.codeweavers.com/crossover) CodeWeavers offers a 14-day free trial of CrossOver, which includes integration of the graphical translation layer.

Note: early in the macOS 15 beta period these pre-built tools may still be carrying the 1.1 version of D3DMetal. You can temporarily update these tools to use the 2.0 version as follows.

- **homebrew-wine** and **game-porting-toolkit** casks - you can replace the copies of the D3DMetal.framework and libd3dshared.dylib found at /Applications/Game\ Porting\ Toolkit.app/Contents/Resources/wine/lib/external/ with the libraries from this distribution:

```
cd /Applications/Game\ Porting\ Toolkit.app/Contents/Resources/wine/lib/external

mv D3DMetal.framework D3DMetal.framework-old; mv libd3dshared.dylib libd3dshared.dylib-old

ditto /Volumes/Evaluation\ environment\ for\ Windows\ games\ 2.0/redist/lib/external/ .
```

- **Whisky**: replace Whisky's copies of the D3DMetal.framework and libd3dshared.dylib found at ~/Library/Application Support/com.isaacmarovitz.Whisky/Libraries/Wine/lib/external/ with the libraries from this distribution:

```
cd ~/Library/Application\ Support/com.isaacmarovitz.Whisky/Libraries/Wine/lib/external

mv D3DMetal.framework D3DMetal.framework-old; mv libd3dshared.dylib libd3dshared.dylib-old

ditto /Volumes/Evaluation\ environment\ for\ Windows\ games\ 2.0/redist/lib/external/ .
```

- **CrossOver**: replace CrossOver's copies of the D3DMetal.framework and libd3dshared.dylib found at /Applications/CrossOver.app/Contents/SharedSupport/CrossOver/lib64/apple_gptk/external with the libraries from this distribution:

```
cd /Applications/CrossOver.app/Contents/SharedSupport/CrossOver/lib64/apple_gptk/external

mv D3DMetal.framework D3DMetal.framework-old; mv libd3dshared.dylib libd3dshared.dylib-old

ditto /Volumes/Evaluation\ environment\ for\ Windows\ games\ 2.0/redist/lib/external/ .
```

Option 2: Build your own evaluation environment from scratch

1. Setup your development and Homebrew environment

- Ensure that you are usihg Command Line Tools for Xcode 15.1. Visit https://developer.apple.com/downloads to download this older version of the tools. Note: there is a header file layout change preventing the use of newer versions of the macOS SDK which we are well aware of.

- Open Terminal.

- The evaluation environment for Windows games runs under Rosetta 2. Ensure that Rosetta 2 is installed.

```
softwareupdate --install-rosetta
```

- Enter an x86_64 shell to continue the following steps in a Rosetta environment. All subsequent commands should be run within this shell.

```
arch -x86_64 zsh
```

- Install the x86_64 version of Homebrew if you don't already have it.

```
/bin/bash -c "$(curl -fsSL https://
raw.githubusercontent.com/Homebrew/install/HEAD/
install.sh)"
```

- Make sure the brew command is on your path:

```
which brew
```

  If this command does not print `/usr/local/bin/brew`, you must either modify your PATH to put `/usr/local/bin` first, or fully specify the path to `brew` in the subsequent commands.

- Tap the Apple Homebrew tap, which can be found at `https://github.com/apple`:

```
brew tap apple/apple http://github.com/apple/homebrew-
apple
```

- Install the `game-porting-toolkit` formula. This formula downloads and compiles several large software projects. How long this takes will depend on the speed of your computer.

```
brew -v install apple/apple/game-porting-toolkit
```

- If during installation you see an error such as "`Error: game-porting-toolkit: unknown or unsupported macOS version: :dunno`", your version of Homebrew doesn't have macOS Sonoma support. Update to the latest version of Homebrew and try again.

```
brew update
```

```
brew -v install apple/apple/game-porting-toolkit
```

2. Create a new Wine prefix for your evaluation environment for Windows games:

A Wine prefix contains a virtual C: drive. You will install the toolkit and your game into this virtual C: drive.

- Run the following command to create a new Wine prefix named **my-game-prefix** in your home directory.

  ```
  WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-toolkit`/bin/wine64 winecfg
  ```

  A “Wine configuration” window should appear on your screen.

- Change the version of Windows to Windows 10.

- Choose Apply and then OK to exit winecfg.

If the “Wine configuration” window does not appear, and no new icon appears in the Dock, verify that you have correctly installed the x86_64 version of Homebrew as well as the `game-porting-toolkit` formula.

3. Install the redistributables from the evaluation environment into the Wine prefix

The graphics bridge libraries need to be placed inside your Wine prefix in order to finalize your evaluation environment. These instructions assume you have mounted the evaluation environment package at `/Volumes/Evaluation Environment For Windows Games-2.0`.

- Copy the Game Porting Toolkit library directory into Wine’s library directory.

  ```
  ditto /Volumes/Evaluation\ Environment\ For\ Windows\ Games-2.0/redist/lib/ `brew --prefix game-porting-toolkit`/lib/
  ```

Launch your game

Open your Wine prefix’s virtual C: drive in Finder (open `~/my-game-prefix/drive_c`) and copy your game into an appropriate subdirectory of your choosing.

The provided `bin/ee4wg*` scripts can be copied onto your path to facilitate different forms of logging and launching. You can run these scripts from any shell; you don’t need to switch to the Rosetta environment first.

A. Standard launching:

```
gameportingtoolkit ~/my-game-prefix 'C:\Program
```

```
Files\MyGame\MyGame.exe'
```

This launches the given Windows game binary with a visible extended Metal Performance HUD and filters logging to just show output from the D3DMetal graphics layer.

B. Launching without a HUD

```
gameportingtoolkig-no-hud ~/my-game-prefix 'C:\Program
Files\MyGame\MyGame.exe'
```

Launches your game without the extended Metal Performance HUD visible.

C. Launching with Wine ESYNC disabled

```
gameportingtoolkit-no-esync ~/my-game-prefix 'C:\Program
Files\MyGame\MyGame.exe'
```

Finally, the no-esync versions of the script disable Wine's ESYNC option, a common compatibility flag. If your game experiences issues with multithreading, or you get an error message about running out of files, you can try launching your game without this Wine environment variable enabled to see if disabling esync clears the problem.

Environment Variables

Environment variables can be used to control some aspects of translation and emulation in the evaluation environment.

`D3DM_SUPPORT_DXR` - Defaults to 0 (OFF). On M3 Macs, setting this environment variable to 1 (ON) enables DirectX Raytracing (aka DXR) features in D3DMetal's DirectX 12 translation layer, so games querying for DXR support will find the support level and expected interfaces of DXR.

`ROSETTA_ADVERTISE_AVX` - Defaults to 0 (OFF). On macOS 15 Sequoia, setting this environment variable to 1 (ON) causes the CPU instruction translation layer to publish cpuid information to translated applications when running in the evaluation environment, so games querying instruction set extension capabilities before utilizing them can conditionally control their use of instruction extensions. This setting does not modify the availability of the instruction set in Rosetta; it only controls whether the processor advertises its support for these extensions.

Logging

Logging output will appear in the Terminal window in which you launch your game as well as the system log, which can be viewed with the Console app found in Applications

‣ Utilities. Log messages from the evolution environment for Windows games are prefixed with **D3DM**. By default the `ee4wg*` scripts will filter to just the **D3DM**-prefixed messages. If you are experiencing an issue and want to send logging information through https://feedbackassistant.apple.com, please attach and send the full logs without filtering to **D3DM.**

Debugging your game with Metal debugger

> **Note: You will need to disable System Integrity Protection (SIP) (**https://developer.apple.com/documentation/security/disabling_and_enabling_system_integrity_protection) **to debug CrossOver's Wine processes. Reenable SIP after you finish debugging.**

- Compile your shaders with embedded debug information (https://developer.apple.com/metal/shader-converter/#shader) by passing `-Zi -Qembed_debug` to the DX Compiler.
- In CrossOver, select a bottle to launch your game from.
- Enable D3DMetal in the Advanced Settings for the bottle.
- Launch your game by clicking Run Command, choosing your game executable, and inserting the following environment variables to enable Metal debugging and processing of debug information: **`MTL_CAPTURE_ENABLED=1 D3DM_DXIL_PROCESS_DEBUG_INFORMATION=1`**
- In Xcode, click `Debug > Debug Executable…` from the menubar and select CrossOver.app (this is just to get a workspace window open)
- In the visible Scheme options, click the `Options` tab and change `GPU Frame Capture` from `Automatically` to `Metal`.
- Close Scheme.
- Click `Debug > Attach to Process` from the menubar and select your launched game process.
- After the debugger attaches to the process, you can capture your Metal workload (https://developer.apple.com/documentation/xcode/capturing-a-metal-workload-in-xcode#Capture-your-Metal-workload-while-debugging).

If lldb suspends the process due to handling `SIGUSR1`, you will need to run the following commands to ignore this signal and continue the process:

```
process handle -pass false -stop false -notify false
```

```
SIGUSR1
continue
```

Troubleshooting

My game won't run and crashes with an invalid instruction or complains about lack of certain instruction extensions

Invalid instruction crashes are sometimes caused when the Rosetta 2 instruction translation layer is unable to translate CPU instructions. You may be able to recompile a version of your game without certain instructions in order to evaluate its potential on Apple Silicon with the Game Porting Toolkit when you hit this error. You may also be able to use the `ROSETTA_ADVERTISE_AVX` environment variable to ensure your game recognizes available translation instruction extensions. When porting your code natively to Apple Silicon there are a variety or NEON and ARM instructions which offer high-performance replacements for AVX / AVX2, BMI, F16c and other less common instruction set extensions.

My game won't run because its anti-cheat or DRM software is incompatible with Wine translation.

You may be able to rebuild a custom version of your game in your Windows development environment with anti-cheat or DRM disabled for your own evaluation purposes. When porting your code natively to Apple Silicon and macOS, contact your anti-cheat or DRM provider—most have native Apple Silicon solutions for your native build, or you may find that existing macOS solutions like Hardened Runtime, Application Sandbox, and Application Attestation prevent forms of cheating or tampering that concern you.

My game won’t run because it thinks the version of Windows is too old.

First, make sure you have selected an appropriate Windows version in winecfg. This affects the major and minor Windows versions that are reported to your game.

If your game checks for a specific minimum or an exact build version, you can alter this value by changing the `CurrentBuild` and `CurrentBuildNumber` values of the `HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT` registry key. You must perform this step *after* selecting a Windows version in winecfg. Run the following commands, replacing «BUILD_NUMBER» with the specific build number your game checks for:

```
WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-
toolkit`/bin/wine64 reg add
'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion' /v CurrentBuild /t REG_SZ /d
«BUILD_NUMBER» /f
```

```
WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-
toolkit`/bin/wine64 reg add
'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion' /v CurrentBuildNumber /t REG_SZ /d
«BUILD_NUMBER» /f
WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-
toolkit`/bin/wineserver -k
```

The last command will shut down the virtual Windows environment to ensure that all components agree on the Windows version the next time you launch your game.

My game won't run because it requires Mono, .NET, or the MSVCRT runtime.

The evaluation environment for Windows games does not pre-install these runtime support packages. If your game makes use of one of these packages, consider searching for and downloading appropriate installers (.exe or .msi) and installing them to your evaluation environment. Additional runtime installers can be run on your environment by just launching the installer and following its installation instructions:

```
WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-
toolkit`/bin/wine64 <some-installer.exe>
```

And .MSI packages can be installed by launching the Windows **uninstaller** application and choosing to install a downloaded .msi package:

```
WINEPREFIX=~/my-game-prefix `brew --prefix game-porting-
toolkit`/bin/wine64 uninstaller
```

My game won't boot anymore even though I made no changes.

If the game stopped booting without being updated, you can try clearing the shader cache.
Run the following commands:

```
cd $(getconf DARWIN_USER_CACHE_DIR)/d3dm
cd «GAME_NAME»
rm -r shaders.cache
```

Do you have a different problem or other feedback?

Please let us know through https://feedbackassistant.apple.com.