

Sample This is a sample driver that shows how to create a

Indirect Display Driver

have a NEAR address: moopaloo.near

extension driver. **Installation**

Windows Indirect Display Driver using the IddCx class

If you want me to build on this donate eth or similar here: 0xB01b6328F8Be53c852a54432bbEe630cE0Bd559a I now

Thanks to https://github.com/akatrevorjay/edid-generator

Scoop (recommended)

for the hi-res EDID.

If you have Scoop, you can easily install this driver in one go. In an elevated prompt, run:

ſО scoop bucket add extras scoop bucket add nonportable

The driver should be automatically installed and should be working out of the box.

scoop install iddsampledriver-ge9-np -g

1. Download the latest version from the releases page, and extract the contents to a folder.

Manually

before installing the driver (important!). 3. See the guide in roshkins repo for the rest of the

2. Copy option.txt to C:\IddSampleDriver\option.txt

- installation steps.
- Configuration

Configure C:\IddSampleDriver\option.txt to set the number of monitors and resolutions. See option.txt

Background reading Start at the Indirect Display Driver Model Overview on

The sample driver code is very simplistic and does

Customizing the sample

code, there are TODO blocks with important information on implementing functionality in a production driver. Code structure Direct3DDevice class

nothing more than enumerate a single monitor when its device enters the D0/started power state. Throughout the

Contains logic for enumerating the correct render GPU from DXGI and creating a D3D device.

- Manages the lifetime of a DXGI factory and a D3D
 - device created for the render GPU the system is using to render frames for your indirect display device's swap-chain.
 - SwapChainProcessor class Processes frames for a swap-chain assigned to the monitor object on a dedicated thread. • The sample code does nothing with the frames,
- but demonstrates a correct processing loop with error handling and notifying the OS of frame
 - completion. IndirectDeviceContext class
 - Processes device callbacks from IddCx. Manages the creation and arrival of the sample monitor.

creating a Direct3DDevice and handing it off to

Handles swap-chain arrival and departure by

a SwapChainProcessor. First steps

Consider the capabilities of your device. If the device supports multiple monitors being hotplugged and removed at runtime, you may want to abstract the monitors further from the IndirectDeviceContext class.

The INF file included in the sample needs updating for production use. One field, DeviceGroupId, controls how the UMDF driver gets pooled with other UMDF drivers in the same process. Since indirect display drivers tend to be more complicated than other driver classes, it's highly recommended that you pick a unique string for this field

which will cause instances of your device driver to pool in

a dedicated process. This will improve system reliability in case your driver encounters a problem since other drivers will not be affected.

Ensure the device information reported to IddCxAdapterInitAsync is accurate. This information determines how the device is reported to the OS and what static features (like support for gamma tables) the device will have available. If some information cannot be known immediately in the EvtDeviceD0Entry callback, IddCx allows the driver to call IddCxAdapterInitAsync at any point after D0 entry, before D0 exit.

Careful attention should be paid to the frame processing loop. This will directly impact the performance of the user's system, so making use of the Multimedia Class Scheduler Service and DXGI's support for GPU prioritization should be considered. Any significant work should be performed outside the main processing loop, such as by queuing work in a thread pool. See SwapChainProcessor::RunCore for more information.

License

made, check with Microsoft for their license), whichever is least restrictive -- Use it

AS IS - NO IMPLICIT OR EXPLICIT warranty This may break

License MIT and CC0 or Public Domain (for changes I

your computer, it didn't break mine. It runs in User Mode which means it's less likely to cause system instability like the Blue Screen of Death.

Acknowledgements

See the original repo below:

for the hi-res EDID.

https://github.com/roshkins/IddSampleDriver

Thanks to https://github.com/akatrevorjay/edid-generator

Terms Privacy Security Status Community Docs Contact Manage cookies

Do not share my personal information

© 2025 GitHub, Inc.