

SimpleFrontPanel Sample

*This sample is compatible with the Microsoft Game Development Kit (June 2020)*

# Description

The SimpleFrontPanel sample demonstrates the XFrontPanelDisplay API covering the basic functionality that you will need to get started programming for the Xbox One X Devkit and the Project Scarlett Devkit Front Panel displays. The sample demonstrates how to operate the front panel and handles the case when there is no front panel, such is the case for an Xbox One or Xbox One S devkit. The sample also covers basic functionality such as polling for front panel button states, getting/setting front panel light states, and drawing simple bit patterns to the front panel LCD display. The sample also shows how to save the front panel display buffer to a .dds texture file.

# Building the sample

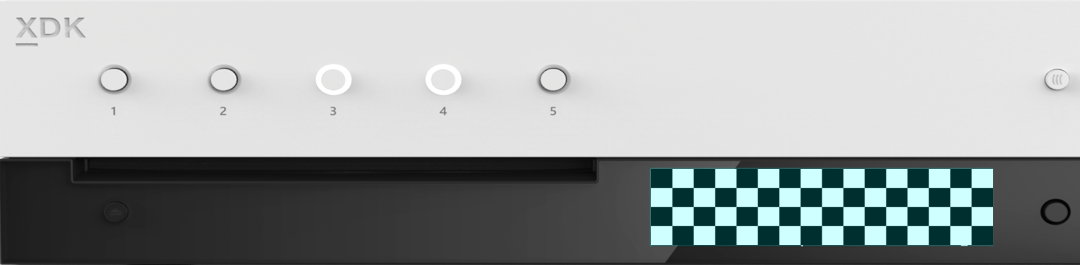
If using an Xbox One devkit, set the active solution platform to Gaming.Xbox.XboxOne.x64.

If using Project Scarlett, set the active solution platform to Gaming.Xbox.Scarlett.x64.

*For more information, see* Running samples*, in the GDK documentation.*

# Using the sample

The sample is intended for the Xbox One X Devkit and the Project Scarlett Devkit with the integrated front panel. When you start the sample, it will render a checkerboard pattern to the front panel display. Use the front panel DPAD (left, right) to change the display bit pattern and to change the brightness of the pixels (up, down). The DPAD button can also be pressed (select) to capture the buffer for the front panel display. Each of the five front panel buttons has an integrated LED associated with it. When you press a button, it will toggle the light on or off.



1 LED per Button

5x Programmable buttons

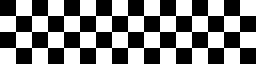
256 x 64 x 4bpp OLED display

DPAD + Select

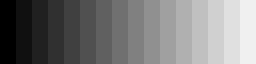
The sample performs all input and output using the integrated front panel and does not interact with gamepad(s) or the connected display. The sample will run on an Xbox One or Xbox One S but has no interesting functionality on these devkits since they do not have front panel displays.

## Checkerboard Screen

|  |  |
| --- | --- |
| Action | Front Panel |
| Previous Screen | DPAD Left |
| Next Screen | DPAD Right |
| Increase Brightness | DPAD Up |
| Decrease Brightness | DPAD Down |
| Capture Front Panel | DPAD Select |
| Toggle Button Lights | Front Panel Buttons |



## Gradient Screen



Implementation notes

* On a Xbox One X Devkit or Project Scarlett Devkit, ::XFrontPanelIsAvailable() will return true and the full API will be available. Otherwise, ::XFrontPanelIsAvailable() will return false and other ::XFrontPanel\*() functions will return a failed HRESULT code. (e.g. on an Xbox One, Xbox One S, or any retail console without a physical front panel.)
* It is not necessary to present to the front panel on every frame (::XFrontPanelPresentBuffer()). Instead, you only need to present when one or more pixels has changed. Therefore, the sample has an m\_dirty member that will be set whenever there are changes to the display buffer.
* It is also only necessary to set the light states whenever there are changes.
* ::XFrontPanelGetScreenPixelFormat() returns DXGI\_FORMAT\_R8\_UNORM, however the screen itself only supports 16 shades of gray. By convention, you should encode the grayscale values using only the four high bits for each 8-bit pixel. The low bits will be ignored. For example, see Sample::CheckerboardFillPanelBuffer() and Sample::GradientFillPanelBuffer().
* The API does not support changing the brightness of the display. The sample supports this by simply incrementing/decrementing each pixel by amount 0x10. For example, see Sample::BrightenPanelBuffer() and Sample::DimPanelBuffer().
* You cannot directly access the front panel buffer. Instead, you must manage your own buffer and pass the address of your buffer to ::XFrontPanelPresentBuffer(). Sample::CaptureFrontPanelScreen() simply uses the contents of m\_panelBuffer as the pixel payload for a DDS surface.

# Update history

April 2019, first release of the sample.

November 2019, support for the Project Scarlett Devkit.

# Privacy Statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).