Creating Full Screen Applications Across Multiple Displays in Extended Mode

October, 2012
Executive Summary

This document presents a summary of methods to help developers enable full screen applications across multiple displays in Extended mode on Intel Architecture (IA) platforms.

To be competitive, enabling full screen applications across multiple displays is critical for delivering a great user experience.

One of the key objectives for gaming customers to is create fantastic user experience through wide screen and high quality displays. There are many requests from the gaming and amusement industry to display 3D contents in resolution greater than 1080p.

Most monitors are designed for 1080p resolution; high end monitors that support resolution greater than 1080p are very expensive. To display content at more than 1080p resolution, a customer application needs to pan across multiple monitors. Due to normal OS and application behavior, when an application turns into full screen mode, it will display only on one monitor.

To be competitive, enabling full screen applications across multiple displays is critical for delivering a great user experience.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel’s tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today.

## Contents

Business Challenge ........................................................................................................... 4
  Default Extended Mode Behavior .................................................................................. 4
Solution ............................................................................................................................. 5
  Overview ....................................................................................................................... 5
  Context of Solution ....................................................................................................... 5
The CreateWindow(..) API .............................................................................................. 6
Graphics Device Interface (GDI) Example ......................................................................... 7
Direct X Examples ........................................................................................................... 9
Simple DirectX Example ................................................................................................. 11
Complex DirectX Example ............................................................................................ 13
  CreateWindow Modification ......................................................................................... 14
  DXUTChangeDevice ...................................................................................................... 15
Conclusion ...................................................................................................................... 16
  Advantages .................................................................................................................. 16
  Limitations .................................................................................................................. 17
Acronyms and Terminology ........................................................................................... 18
Business Challenge

Default Extended Mode Behavior

Modern graphics adapters allow users to plug in multiple display panels into the same hardware, creating a multi-display configuration for the system. One of the multi-display configurations is Extended mode. Extended mode uses multiple framebuffers, one for each display and the operating system can display content on any of the displays. Extended mode gives the user a single large display using multiple display panels.

Figure 1. Extended Mode Configuration in Microsoft Windows 7

When a Windows application is maximized in an extended desktop configuration, the application maximizes into only one of the displays, as illustrated in the example in Figure 2. Some embedded applications need to be maximized across multiple displays. The application should appear as one seamless application across the Extended mode displays.
The solution proposed in this paper is a quick way for customers to modify their application so that it spans across multiple Extended mode displays. However, the solution proposed in this paper is intended only for a single application running on the system and therefore it has limitations when used outside the intended context.

Solution

Overview

The solution recommended in this paper is for the user to modify the application’s `CreateWindow(..)` API to make the application look like it is in full screen mode and spanning across Extended displays. The user only needs to modify the style, width, and height parameters of `CreateWindow(..)` API in the user application to create the illusion of an application spanning across multiple displays. This paper contains modified code examples available from Microsoft to demonstrate the solution.

Context of Solution

For the purpose of this paper, two nearly identical panels are used to demonstrate Extended mode displays. The displays are both set to 1920 x 1200 resolution so that they have the same screen area. Both displays also have the same bit depth (32 bits), although this setting does not affect the end result. Using the same resolution in both displays is important because the solution assumes that the application will be divided evenly across the extended displays.
Creating Full Screen Applications Across Multiple Displays in Extended Mode

This solution also assumes that only one application is running at the same time. The example modification suggested in this paper does take into consideration multiple applications switching among each other. Although it is possible to run a background application underneath the main application, Intel does not recommend such usage mode as the background application may draw over the main application.

It is up to the application developer to disable switching between applications, for example, by limiting the keyboard inputs.

This paper is limited to demonstrating the capabilities in Microsoft Windows 7 for 3D and 2D applications. This solution does not cover other applications such as media playback, although the concept should be applicable. The intention of this paper is to demonstrate the concept of changing application codes so that the application can span across multiple Extended mode displays. The exact method may be different for every application but the concept should be the same.

**The CreateWindow(..) API**

The key to spanning across extended displays is to change the size and style of the application window. On Microsoft Windows 7 operating system, this is achieved through the `CreateWindow` API.

This API creates an overlapped, pop-up, or child window. It specifies the window class, window title, window style, and (optionally) the initial position and size of the window. The function also specifies the window’s parent or owner, if any, and the window’s menu.

For more details on the CreateWindow function refer to the following MSDN definition:

Graphics Device Interface (GDI) Example

GDI is used in Microsoft Windows to draw simple 2D objects and text. To demonstrate a simple GDI application rendering across multiple monitors, the "Multi Monitor Example" from Microsoft will be used. The sample is available from


Based on the "Multi Monitor Example" project, CreateWindow is called in DemoApp.cpp line 113. Without modification, the function call and parameters appear as follows:

```c++
// Create window.
hwnd_ = CreateWindow(
    TEXT("DemoApp"),
    TEXT("Simple DirectWrite Hello World"),
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    static_cast<int>(640.0f / dpiScaleX_),
    static_cast<int>(480.0f / dpiScaleY_),
    NULL,
    NULL,
    HINST_THISCOMPONENT,
    this
);
```

The third parameter, WS_OVERLAPPEDWINDOW, is defined as a window that contains a system menu, minimize button, maximize button, caption, thick frame, and is able to be overlapped.

The sixth and seventh parameters set the application size to a ratio of 64dx480 pixels.

Changing the parameters as below, we are able to obtain a simulated full screen application across the extended displays. In the example below, both screen resolutions are 1920 x 1200. The modified parameters are marked in red.

```c++
// Create window.
hwnd_ = CreateWindow(
    TEXT("DemoApp"),
    TEXT("Simple DirectWrite Hello World"),
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    static_cast<int>(1920.0f / dpiScaleX_),
    static_cast<int>(1200.0f / dpiScaleY_),
    NULL,
    NULL,
    HINST_THISCOMPONENT,
    this
);
```
// Create window.
hwnd_ = CreateWindow(
    TEXT("DemoApp"),
    TEXT("Simple DirectWrite Hello World"),
    WS_POPUP,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    1920 * 2,
    1200,
    NULL,
    NULL,
    HINST_THISCOMPONENT,
    this
);

The WS_POPUP style creates a window on top of all other running applications. It does not have menus, borders, minimize buttons, or maximize buttons.

Figure 3. Full screen multi-monitor example spanning across extended desktop
DirectX Examples

The DirectX Software Development Kit (DirectX SDK) is required to develop DirectX applications. This kit contains the required libraries, header files, documentation, and tools to develop DirectX applications. The DirectX SDK also comes with an extensive DirectX sample source code for reference. The sample source code will be available with the installation of the DirectX SDK.

Tips

Remember to select the DirectX samples when performing custom installation of the DirectX SDK. The samples should be installed when performing full installation of DirectX SDK.

The June 2010 DirectX SDK is the latest SDK available from Microsoft and can be downloaded from the link below.


The examples used in this paper are taken from the June 2010 SDK. Sample source code provided in the future DirectX SDK may be different from that available in the June 2012 DirectX SDK.

After the DirectX SDK is installed with the samples, a "DirectX Sample Browser" is available in the start menu (Start->All Programs->Microsoft DirectX SDK (June 2010)->DirectX Sample Browser)

Figure 4. Location of the DirectX Sample Browser in the Start menu
Creating Full Screen Applications Across Multiple Displays in Extended Mode

Figure 5. DirectX Sample interface

Tip
From the DirectX Sample Browser, choose “InstallProject” to copy the solution file and source codes into your desired library. After that, you may edit and compile the codes from Microsoft Visual Studio.
Simple DirectX Example

The DirectX sample “Tutorial 2” is used to demonstrate the modification of the `CreateWindow(..)` API to achieve a full screen DirectX application across Extended mode displays.

Figure 6: Tutorial 2 description in the DirectX Sample Browser

The `CreateWindow(../)` API is called in the `Tutorial02.cpp` file line 117:

```cpp
    g_hWnd = CreateWindow( L"TutorialWindowClass", L"Direct3D 11 Tutorial 2: Rendering a Triangle", WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, CW_USEDEFAULT, rc.right - rc.left, rc.bottom - rc.top, NULL, NULL, hInstance, NULL );
```

The default application creates a windowed application with a yellow triangle on a blue background. This is a simple DirectX application and therefore uses the `CreateWindow(../)` API almost the same way as the GDI application.
Using the Tutorial 2 source code as an example, the `CreateWindow(..)` API is modified as below. Parameters 4, 6, and 7 are modified in red. Like the GDI example, `WS_POPUP` creates a borderless window that does not contain a menu, status bar, or title bar. This gives the application the look of a full screen application. The sixth and seventh parameters set the size of the application to span across the extended mode displays. In this example, the monitors are both set to 1920 x 1200 resolution. The sixth parameter is the width of one display multiplied by 2.

```c
ghWnd = CreateWindow( L"TutorialWindowClass", L"Direct3D 11 Tutorial 2: Rendering a Triangle",
    WS_POPUP,
    CW_USEDEFAULT,
    CW_USEDEFAULT,
    1920 * 2,
    1200,
    NULL, NULL, hInstance,
    NULL );
```

The code modification creates a full screen application across Extended mode displays.
Complex DirectX Example

More complex DirectX samples use the DXUT framework to create the application window. DXUT makes Direct3D samples, prototypes, and tools as well as professional games more robust and easier to build. It simplifies the Windows and Direct3D APIs for typical usage. The `CreateWindow(..)` API is called in the DXUT framework. In addition, DXUT is able to clip the contents of the application into a single adapter. This ability causes the application to always render onto a single display regardless of the size of the application.

To demonstrate the changes needed to span a DirectX application that make use of DXUT into extended mode displays, this paper uses the BasicHLSL11 example. Figure 9 shows the sample browser "application" where you see the section that shows the overview of the BasicHLSL11 sample.

Figure 9. Extended mode spanning changes on BasicHLSL11 application
CreateWindow Modification

Following the previous examples, the first modification to span the BasicHLSL11 across extended display is to change the CreateWindow call. Unlike previous examples, CreateWindow is implemented deep in the DXUT API. Therefore we need to change the use of DXUT API rather than directly modifying the CreateWindow API that is buried deep in the DXUT framework.

For the BasicHLSL11 application, we make the change in the file BasicHLSL11.cpp, line 150.

The original sample code looks like this:

```cpp
DXUTCreateDevice (D3D_FEATURE_LEVEL_9_2, true, 800, 600);
```

Parameters 3 and 4 are changed to the size of the combined extended screen space. On the setup that is used in this paper, two screens with the size of 1920 x 1200 are used. The final API will look like:

```cpp
DXUTCreateDevice (D3D_FEATURE_LEVEL_9_2, true, 1920*2, 1200 ) ;
```
DXUTChangeDevice

Further analysis of DXUT framework that is used in BasicHLSL11 application shows that the rendering is clipped to a single adapter. There are two adapters for each screen in an Extended display mode, one adapter for each screen. Therefore, regardless of the width and height parameters passed into the CreateWindow API, the application uses only one display.

The bClipWindowToSingleAdapter parameter in the DXUTChangeDevice function limits the application from spanning more than one display panel.

Changes to DXUT.cpp line 1865:

```cpp
hr = DXUTChangeDevice( &deviceSettings, NULL, NULL, false, false );
```

This change removes the clipping of the window to one display panel.

Figure 11. BasicHLSL11 spanning across Extended mode
Conclusion

The solution proposed in this paper is a quick way for the customer to modify their application so that it spans across multiple Extended mode displays. However, the solution proposed in this paper is only intended for a single application running on the system and therefore has limitations when used outside the intended context.

Advantages

Portability is the main advantage of modifying the application codes to span across Extended mode screens. Implementing the feature in the application removes the dependency on the hardware and device driver. The modified application can then work the same across all Intel processors and integrated graphics adapters from Intel® Core™ to Intel® Atom™ product lines. Therefore, the customer would only have to modify the application once and then it can be used in other Intel platforms.

By removing the dependency on the driver, the application developer also enjoys a quicker turn-around time to deploy the application. The customer could write the application on one Intel platform and easily deploy it into the production system.

The portability of the solution proposed in this paper can also scale to using PCI-E* graphics adapter for the extended monitor. Using PCI-E graphics adapter, the system can be configured so that it can have more displays than limits imposed by the integrated graphics adapter. In such cases, the application can span across four displays as long as the system is configured correctly in the operating system.

The solution proposed in this paper provides flexibility and control to the application developer to quickly design the application according the functionality of the application. For example, the application can be made full screen for 2 out of 3 monitors, so that the right-most monitor can be used for other purposes.
Limitations

The method proposed in this paper does not allow an application to be used in exclusive mode. In DirectX exclusive mode, the application will own the display adapter and all other 3D applications will lose their Direct3D devices. The solution in this paper modified the application window fill up the display but does not set the application into an exclusive mode. It is therefore possible for other applications to be drawn over the main full screen application.

In the proposed method, the screen size needs to be fixed when creating the application. When the application is maximized in a normal application, it will fill up the display regardless of the monitor resolution. However, this application needs the monitor resolution to match the application window size when it is programmed so that the application will fit the screen. This may not be a problem for intelligent systems applications such as arcade gaming machines and kiosks.
Creating Full Screen Applications Across Multiple Displays in Extended Mode

Acronyms and Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>2 Dimension</td>
</tr>
<tr>
<td>3D</td>
<td>3 Dimension</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>Direct3D</td>
<td>Microsoft’s DirectX Application Programming Interface</td>
</tr>
<tr>
<td>DirectX</td>
<td>Microsoft’s Application Programming Interface Collection</td>
</tr>
<tr>
<td>DXUT</td>
<td>DirectX Utilities</td>
</tr>
<tr>
<td>Extended Mode</td>
<td>Display mode to dual or multiple monitors with different contents</td>
</tr>
<tr>
<td>Framebuffer</td>
<td>Video Display Memory</td>
</tr>
<tr>
<td>GDI</td>
<td>Graphics Device Interface</td>
</tr>
<tr>
<td>IA</td>
<td>Intel Architecture</td>
</tr>
<tr>
<td>MSDN</td>
<td>Microsoft Developer Network</td>
</tr>
<tr>
<td>Panels</td>
<td>Monitor</td>
</tr>
<tr>
<td>PCI-E</td>
<td>Peripheral Component Interconnect Express</td>
</tr>
</tbody>
</table>


**Authors**

**Anthony See** is a Gaming Platform Solution Architect with Intelligent System Group at Intel Corporation

**Ho Nee Shen** is a Software Engineering Manager with Intelligent System Group at Intel Corporation
INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A “Mission Critical Application” is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL’S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS’ FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to:
http://www.intel.com/design/literature.htm%20

Intel, Intel Atom, Intel Core, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2012 Intel Corporation. All rights reserved.
Software Source Code Disclaimer

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license:

Intel Sample Source Code License Agreement

This license governs use of the accompanying software. By installing or copying all or any part of the software components in this package, you ("you" or "Licensee") agree to the terms of this agreement. Do not install or copy the software until you have carefully read and agreed to the following terms and conditions. If you do not agree to the terms of this agreement, promptly return the software to Intel Corporation ("Intel").

1. Definitions:
   A. "Materials" are defined as the software (including the Redistributables and Sample Source as defined herein), documentation, and other materials, including any updates and upgrade thereto, that are provided to you under this Agreement.
   B. "Redistributables" are the files listed in the "redist.txt" file that is included in the Materials or are otherwise clearly identified as redistributable files by Intel.
   C. "Sample Source” is the source code file(s) that: (i) demonstrate(s) certain functions for particular purposes; (ii) are identified as sample source code; and (iii) are provided hereunder in source code form.
   D. "Intel's Licensed Patent Claims” means those claims of Intel's patents that (a) are infringed by the Sample Source or Redistributables, alone and not in combination, in their unmodified form, as furnished by Intel to Licensee and (b) Intel has the right to license.

2. License Grant: Subject to all of the terms and conditions of this Agreement:
   A. Intel grants to you a non-exclusive, non-assignable, copyright license to use the Material for your internal development purposes only.
   B. Intel grants to you a non-exclusive, non-assignable copyright license to reproduce the Sample Source, prepare derivative works of the Sample Source and distribute the Sample Source or any derivative works thereof that you create, as part of the product or application you develop using the Materials.
   C. Intel grants to you a non-exclusive, non-assignable copyright license to distribute the Redistributables, or any portions thereof, as part of the product or application you develop using the Materials.
   D. Intel grants Licensee a non-transferable, non-exclusive, worldwide, non-sublicenseable license under Intel’s Licensed Patent Claims to make, use, sell, and import the Sample Source and the Redistributables.

3. Conditions and Limitations:
   A. This license does not grant you any rights to use Intel’s name, logo or trademarks.
   B. Title to the Materials and all copies thereof remain with Intel. The Materials are copyrighted and are protected by United States copyright laws. You will not remove any copyright notice from the Materials. You agree to prevent any unauthorized copying of the Materials. Except as expressly provided herein, Intel does not grant any express or implied right to you under Intel patents, copyrights, trademarks, or trade secret information.
   C. You may NOT: (i) use or copy the Materials except as provided in this Agreement; (ii) rent or lease the Materials to any third party; (iii) assign this Agreement or transfer the Materials without the express written consent of Intel; (iv) modify, adapt, or translate the Materials in whole or in part except as provided in this Agreement; (v) reverse engineer, decompile, or disassemble the Materials not provided to you in source code form; or (vii) distribute, sublicense or transfer the source code form of any components of the Materials and derivatives thereof to any third party except as provided in this Agreement.
4. No Warranty:
THE MATERIALS ARE PROVIDED "AS IS". INTEL DISCLAIMS ALL EXPRESS OR IMPLIED WARRANTIES WITH RESPECT TO THEM, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR ANY PARTICULAR PURPOSE.

5. Limitation of Liability: NEITHER INTEL NOR ITS SUPPLIERS SHALL BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR OTHER LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

6. USER SUBMISSIONS: You agree that any material, information or other communication, including all data, images, sounds, text, and other things embodied therein, you transmit or post to an Intel website or provide to Intel under this Agreement will be considered non-confidential ("Communications"). Intel will have no confidentiality obligations with respect to the Communications. You agree that Intel and its designees will be free to copy, modify, create derivative works, publicly display, disclose, distribute, license and sublicense through multiple tiers of distribution and licensees, incorporate and otherwise use the Communications, including derivative works thereto, for any and all commercial or non-commercial purposes.

7. TERMINATION OF THIS LICENSE: This Agreement becomes effective on the date you accept this Agreement and will continue until terminated as provided for in this Agreement. Intel may terminate this license at any time if you are in breach of any of its terms and conditions. Upon termination, you will immediately return to Intel or destroy the Materials and all copies thereof.

8. U.S. GOVERNMENT RESTRICTED RIGHTS: The Materials are provided with "RESTRICTED RIGHTS". Use, duplication or disclosure by the Government is subject to restrictions set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor. Use of the Materials by the Government constitutes acknowledgment of Intel's rights in them.

9. APPLICABLE LAWS: Any claim arising under or relating to this Agreement shall be governed by the internal substantive laws of the State of Delaware, without regard to principles of conflict of laws. You may not export the Materials in violation of applicable export laws.

* Other names and brands may be claimed as the property of others.

Intel and Intel Atom are trademarks of Intel Corporation in the U.S. and/or other countries.

Copyright (C) 2008 – 2012, Intel Corporation. All rights reserved.