Software Solutions for Multi-Display Setups

January 2013

White Paper
Bruce Bao
Graphics Application Engineer
Intel Corporation
Executive Summary

Multi-display systems are growing in popularity. They provide bigger display areas and involve substantially lower system cost compared to high-end single display systems. How to enable one application to show its content across multi-display systems is a popular question. This article discusses how to go about doing that.

Contents

Business challenge and solution ................................................................. 4
Basic framework .......................................................................................... 7
Bezel compensation ...................................................................................... 13
Conclusions ................................................................................................. 15
References .................................................................................................. 15
Business challenge and solution

Developers use multiple methods to realize cross-monitor display effects, but these methods generally fall into two main categories. They are either implemented by modifying the display driver or implemented by modifying the application source code.

**Modifying the display driver:**

Independent Hardware Vendors (IHVs) need to modify the display driver to:

1. Get and shield the display information of each display monitor;
2. Generate display information of the virtual monitor which is combined by multi-display monitors;
3. Pass the virtual monitor display information to the OS and applications.

In this case, the driver handles all operations and is transparent to the OS and applications. The OS will be configured as a single or duplicated display mode, and the application only needs to be designed to work in the single display mode.

At the same time, IHVs also provide a GUI configuration tool for customers to configure the multi-display combination (such as, 3x1 or 1x3 layout) in order to generate the final virtual screen. The advantage of this modified display driver method is that it is easy to realize bezel compensation (or bezel correction). The disadvantage of this method is that each display monitor must be the same size / resolution setting and orientation.

**Figure 1 Example of IHV GUI to configure the 3x1 multi-display**

![Example of IHV GUI to configure the 3x1 multi-display](image)
Modifying the application:

Independent Software Vendors (ISVs) need to modify their applications’ source code to make it function in a multi-display system. First, the application needs to query the multi-display information, like the multi-screen layout, the width and height of the virtual screen etc. Then the application can display its content across the multiple connected monitors based on this information. In this method, the OS exactly knows how many display monitors are connected to the system, and will configure them work in extended mode.

All the information (including start point, resolution, direction etc.) for each display can be queried, which makes the software solution more flexible and can enable it to handle some special layouts. All the customer needs to do is to configure the layout of multi-display in the display control panel first, so the application can get this information by invoking the Windows API.

Because additional copies and synchronization operations may be invoked by the OS to simultaneously show different parts of content to different monitors, the performance of this solution will drop a little compared to the driver solution.

Figure 2 Special layout of multi-display example (usually done by application solution)
Figure 3 Special layout of multi-display example (usually done by application solution)

Table 1. Feature comparison between these two solutions

<table>
<thead>
<tr>
<th>Function/Request</th>
<th>Driver solution</th>
<th>Application solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware platform</td>
<td>Platform supported by this driver</td>
<td>Any hardware platform</td>
</tr>
<tr>
<td>Operation System</td>
<td>Single or Duplicate</td>
<td>Extend</td>
</tr>
<tr>
<td>Application mode</td>
<td>Full-screen or window mode</td>
<td>Window mode only</td>
</tr>
<tr>
<td>Display Topology</td>
<td>Simple:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Display setting of each monitor should be the same</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• The final virtual screen should be a rectangle.</td>
<td></td>
</tr>
<tr>
<td>Bezel</td>
<td>Easy to apply, benefits all software applications.</td>
<td>Difficult to apply, applications need do it by themselves.</td>
</tr>
<tr>
<td>ISV cost</td>
<td>Zero</td>
<td>Small</td>
</tr>
<tr>
<td>Performance</td>
<td>Better</td>
<td>Common</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Rely on IHV</td>
<td>ISV can take control</td>
</tr>
</tbody>
</table>
Basic framework

The application solution can have lots of methods, and we only discuss the simplest one here: create one window and have it span across all the displays, and enable the application to render its contents within this window. This method is simple to implement within the existing applications to make them suited to multi-display systems, while the performance may decrease a bit depending on different system configurations and software architecture: OS will do additional blt/copy operations to show different portions of the image across the different displays.

1. Configure the layout of multi-display
   a) Set extended mode in the Windows display control panel.

Figure 4 Display layout in system display panel.
b) Configure the display layout (drag the monitor to the proper position in display control panel) and set the corresponding resolution.

Figure 5 Display layout in system display panel.

2. Modify the application source code. The BasicHLSL example shown here is from the Microsoft® DirectX SDK.

a) Make sure the application is running in Windows* mode

b) Get/Query the current display settings from Windows, including the resolution of the virtual screen, the coordinate of the start point etc.

```c
// Get necessary information of Virtual Screen.
RES_X = GetSystemMetrics(SM_XVIRTUALSCREEN);
RES_Y = GetSystemMetrics(SM_YVIRTUALSCREEN);
RES_W = GetSystemMetrics(SM_CXVIRTUALSCREEN);
RES_H = GetSystemMetrics(SM_CYVIRTUALSCREEN);
```
c) Set the window size and position based on the information returned from step b.

```cpp
// Create window.
hwnd_ = CreateWindow(
    TEXT("DemoApp"),
    TEXT("Simple DirectWrite Hello World"),
    WS_POPUP, //"WS_POPUP" Window style
    RES_X, //Horizontal start point of the virtual screen
    RES_Y, //Vertical start point of the virtual screen
    RES_W, //Width of the virtual screen
    RES_H, //Height of the virtual screen
    NULL,
    NULL,0,
    HINST_THISCOMPONENT,
    this)
);
```

d) Remove the limitation to show content only in a single display: Disable the `bClipWindowToSingleAdapter` parameter in the `DXUTChangeDevice` function.

```cpp
HRESULT DXUTChangeDevice(DXUTDeviceSettings* NewDeviceSettings,
    IDirect3DDevice9* pd3d9DeviceFromApp,
    ID3D10Device* pd3d10DeviceFromApp,
    bool bForceRecreate,
    bool bClipWindowToSingleAdapter)
```
3. Additional work in abnormal display topology:

For abnormal display topology, nothing needs to be changed in the source code after step 2. All that is needed is to change the display topology in the Intel® Graphics and Media Control Panel. Here we take two monitors as an example and make one horizontal and the other vertical:

a) System display control panel:

Select the second display and choose “Portrait” from the “Orientation” drop down (Figure 6). Drag the vertical monitor to a suitable position (Figure 7), and click “OK” or “Apply”.

**Figure 6 Display layout in system display panel.**

![Display layout in system display panel](image1)

**Figure 7 Display layout in system display panel.**

![Display layout in system display panel](image2)
b) Intel® Graphics and Media Control Panel:

In the “General settings” tab, select the second display. Select the “Rotate To 90 Degrees” option from the “Rotation” drop down menu (Figure 8) and click “Apply”.

In the “Multiple Displays” tab, select the Primary and Secondary displays, drag the vertical monitor as needed to a suitable position in the “Positioning” column (Figure 9) and click “OK”.

Figure 8 Configurations in Intel® Graphics and Media Control Panel
Now the system has been successfully configured as one horizontal and one vertical display topology.

4. Running result

**Figure 10 Final result (horizontal layout)**
Bezel compensation

Each monitor has an edge where there is no picture. That part of the monitor is called the bezel. Bezels introduce big gaps which separate different parts of one picture, causing the image to look unnatural and not change smoothly. Bezel compensation refers to using the images to compensate for the bezel. In this case, the GPU will render on a virtual screen which is bigger than the real screen size, so some display content will not be seen on the real screen. From end user point of view, the missing contents are covered by the bezel and the total system is “Bezel compensated”.
Advanced and complicated application solution: Applications can be designed to create multiple windows and each one covers only one display monitor and binds to it. Thus different windows will be rendered as different parts of the image.

Performance of this architecture will be better because no additional blt/copy is needed. Bezel compensation can also be solved by rendering suitable content in each window. But the disadvantage is that the application needs to carefully handle the synchronization between different content part renderings.
Conclusion

The driver and application solutions can solve the request of one screen of content across a multi-display system. The software-based solution is more flexible and can be a good complement of a driver-based solution – a modified application will also run well on driver supported multi-display systems. Intel is providing driver solutions starting with the 3rd generation Intel® Core™ processors. Customers can easily design their Intel based system by choosing the technology which is right for them.

References


Author

Bruce Bao is a Graphics Application Engineer in the Intelligent Systems 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

Intel, 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 © 2013 Intel Corporation. All rights reserved.