AMD OpenGL® Extension
TECHNOLOGY PAPER

AMD FirePro™

DISPLAY OUTPUT POST-PROCESSING

Display Output Post-Processing (DOPP) is an AMD OpenGL® extension that lets users of AMD FirePro™ workstation cards grab the desktop directly as a texture and manipulate it in an almost infinite number of ways before it is output to a display system.

Contents

23 September 2013
Version 1.4

Christopher Mayer

Key Features

- DOPP extension to grab the desktop as a texture.
- Full OpenGL API to transform the surface.
- DOPP extension to send the transformed (present) surface to video out (including wirelessly).

DOPP Related Terms

- Desktop texture: the texture containing the desktop image.
- Present texture: the texture that will be displayed on the following present.

Arbitrary Transformations

Display output post-processing (DOPP) can be used to perform arbitrary transformations such as warping, blending, color correction/calibration and desktop capturing.

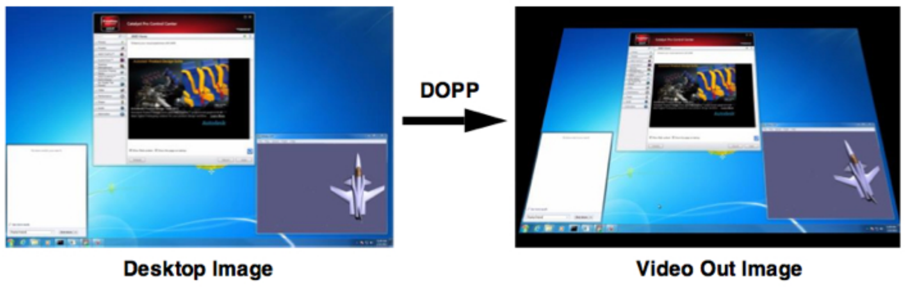


Figure 1

Arbitrary Warps Using DOPP

Desktop before and after DOPP warping.

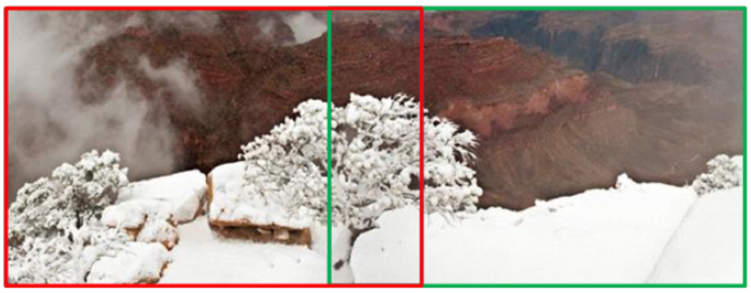


Figure 2

Arbitrary Blending Using DOPP

Desktop with edge blending.

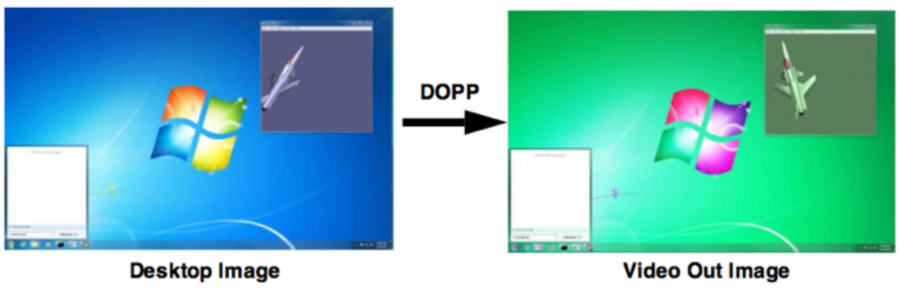


Figure 3

Arbitrary Pixel Transformations Using DOPP

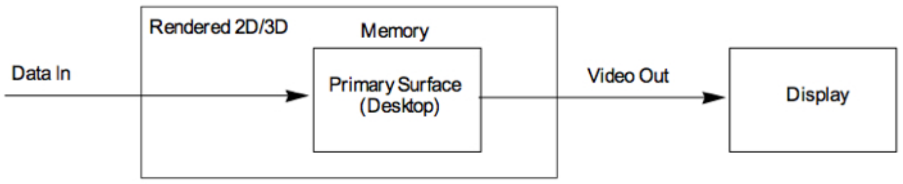
Desktop before and after arbitrary pixel transformations.

Implementation Details

Without DOPP, 2D or 3D rendered data is passed to the GPU for rendering into the primary surface. This is scanned out to one or more displays, as in figure 4 *Rendering and Output Without DOPP*.

Figure 4

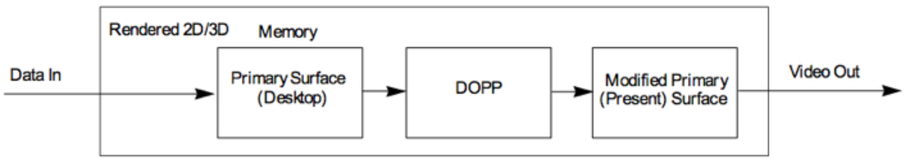
Rendering and Output without DOPP



When DOPP is enabled, the desktop image to be altered is rendered in the normal way by the operating system (OS). The surface on which it is rendered, usually called the primary surface, is then used as a texture by the DOPP application, which renders the final desktop image to a present surface. The DOPP application sends the present surface to the video output. This process is shown in Figure 5 *Rendering and Output Using DOPP.*

Figure 5

Rendering and Output Using DOPP



The DOPP primary surface is generated from the original primary surface. The DOPP control application transforms the desktop for different post processing usage requirements.

Since DOPP is an OpenGL extension, it allows developers to access the complete OpenGL API. This puts all OpenGL API calls to transform the primary surface to the present surface in multiple ways at the developer's disposal.

DOPP Functionality

<function>

`void wglEnablePostProcessAMD(bool enable)`
This function indicates whether post processing is enabled or disabled. A call to this function is only required if the application provides a present texture. If a DOPP application just wants to capture the desktop texture, this call is not needed.

This function should only be called by the application if a present texture was previously passed to the driver by calling `wglPresentTextureToVideoAMD`.

`GLuint wglGetDesktopTextureAMD()`
This function returns the texture name that identifies the desktop texture. The content of this texture changes asynchronously as the desktop changes. The application can use this texture for its rendering, eg to generate a present texture.

The origin of the desktop texture is at the upper left corner.

`GLuint wglGenPresentTextureAMD()`
This function returns a texture name identifying the present texture. The application can use different methods to get the pixels from the desktop texture into the present texture. One method is to bind the present texture to a framebuffer object (FBO) and render the desktop texture into the FBO using a shader for the processing.

The processed present texture is made available to the Operating System by calling `wglPresentTextureToVideoAMD`.

The origin of the present texture needs to be at the upper left corner.

`GLuint wglPresentTextureToVideoAMD(GLuint uiPresentTexture, const GLuint* attrib_list)`
This function sets the texture `uiPresentTexture` as present texture. If the post processing is enabled, this texture will be presented next.

`attrib_list` is a null terminated array of attributes. The only accepted attribute for now is `GL_WAIT_FOR_PREVIOUS_VSYNC`.

When the attribute `GL_WAIT_FOR_PREVIOUS_VSYNC` is passed, the call to `wglPresentTextureToVideoAMD` will not return until the previous present texture has been displayed. This flag can be used to achieve vsync-like effect.

A typical render loop of a DOPP application could look like this:

```
// Bind FBO that has the present texture attached
glBindFramebuffer(GL_FRAMEBUFFER, m_uiFBO);

// Render to FBO
updateTexture();

glBindFramebuffer(GL_FRAMEBUFFER, 0);
glBindTexture(GL_TEXTURE_2D, 0);

// Set GL_WAIT_FOR_PREVIOUS_VSYNC to block the
wglPresentTextureToVido until the previous
// texture was displayed.
const GLuint attrib[] = { GL_WAIT_FOR_PREVIOUS_VSYNC,
0 };

// Set the new desktop texture
wglPresentTextureToVideoAMD(m_uiPresentTexture, attrib);

if (m_bStartPostProcessing)
{
   m_bStartPostProcessing = false;
   wglEnablePostProcessAMD(true);
   glFinish();
}
```

`bool wglDesktopTargetAMD(GLuint uiDesktopIndex)`

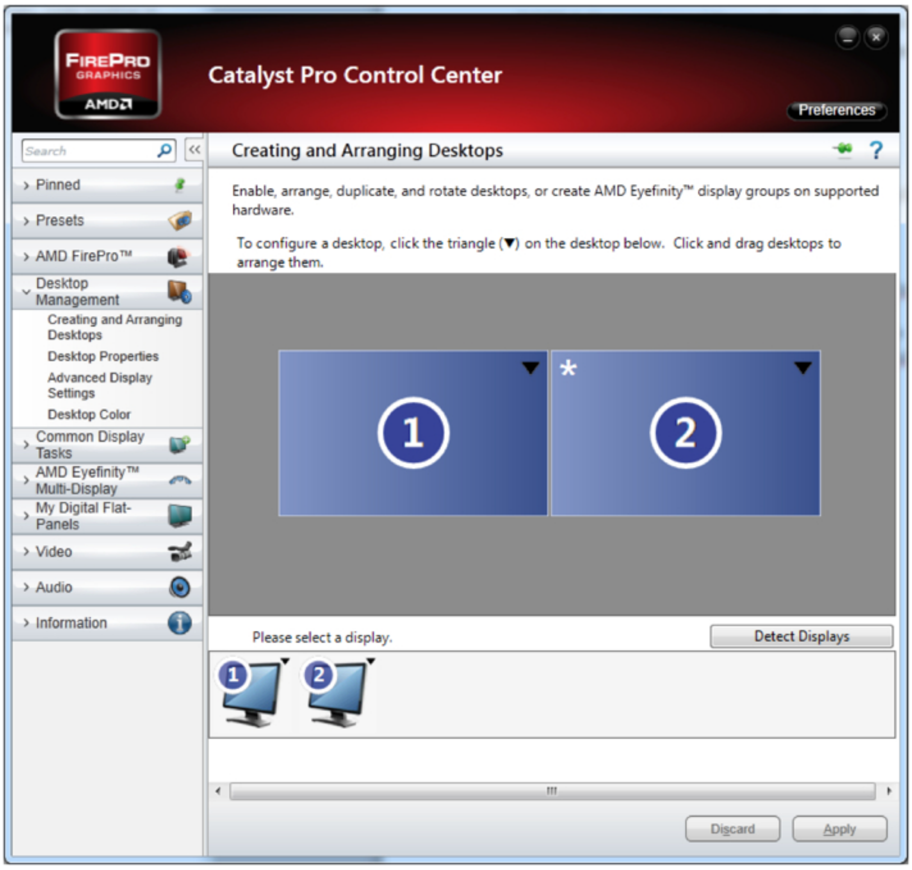
Using this function the application can define which desktop texture will be returned when calling `wglGetDesktopTextureAMD`.

`uiDesktopIndex` specifies the desktop identity as seen in CCC (Catalyst Control Center). Figure 6 shows an extended desktop configuration.

Calling `wglDesktopTargetAMD` with `uiDesktopIndex = 1` would specify the left desktop as input.

Figure 6

Extended Desktop



Enable, arrange, duplicate, and rotate desktops, or create AMD Eyefinity™ display groups on supported hardware.

To configure a desktop, click the triangle (▼) on the desktop below. Click and drag desktops to arrange them.

Please select a display.

`GLint wglGetDisplayOverlapsAMD(GLuint uiDisplay1, GLuint uiDisplay2)`

This function indicates the number of pixels by which two displays in one Eyefinity DisplayGroup overlap.

When configuring a DisplayGroup it is possible to define an overlap of displays. If an overlap is defined, the desktop texture is smaller than the present texture and it is up to the DOPP application to generate the correct present texture.

This can be useful if an application wants to implement edge blending using DOPP. In this case it can copy the overlap region of the desktop texture to the corresponding region of the present texture and blend the region.

The example below in figure 7 illustrates how the different textures look. The 2×1 DisplayGroup has an original resolution of 3200×1200. An overlap of 200 pixels was defined, hence the desktop texture is only 3000×1200, the present texture is still 3200×1200 and each output of the GPU shows a 1600×1200 image.

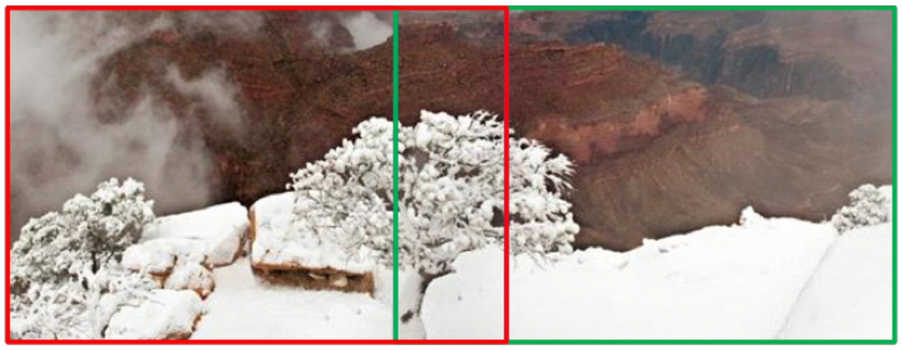


Figure 7

2×1 SLS Desktop

eg 3000×1200 with 200 pixel overlap.

The application will compose the present texture as shown in figure 8 and can treat the blending zone with a different shader for each output.

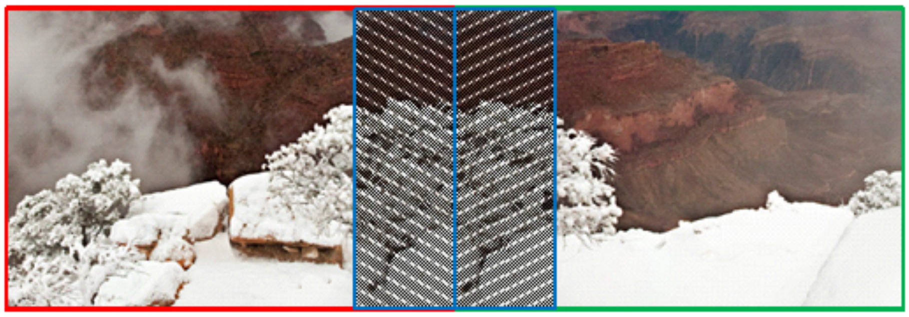


Figure 8

Present Texture

eg 3000×1200.

The images sent to the projectors look like these shown in figure 9 below.

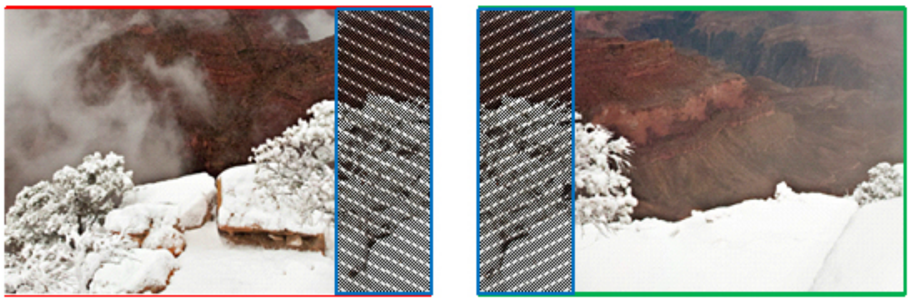


Figure 9

Actual Projected Images

Left projector (eg 1600×1200) and right projector (eg 1600×1200).

To query the layout of an Eyefinity Group, DOPP defines two new parameters that are accepted by `glGetIntegerv`.

`GL_DOPP_GRID_ROW` can be used to query the number of rows in a DisplayGroup and `GL_DOPP_GRID_COLUMN` to query the number of columns.

Sample Transformations

Figure 10

Original Image

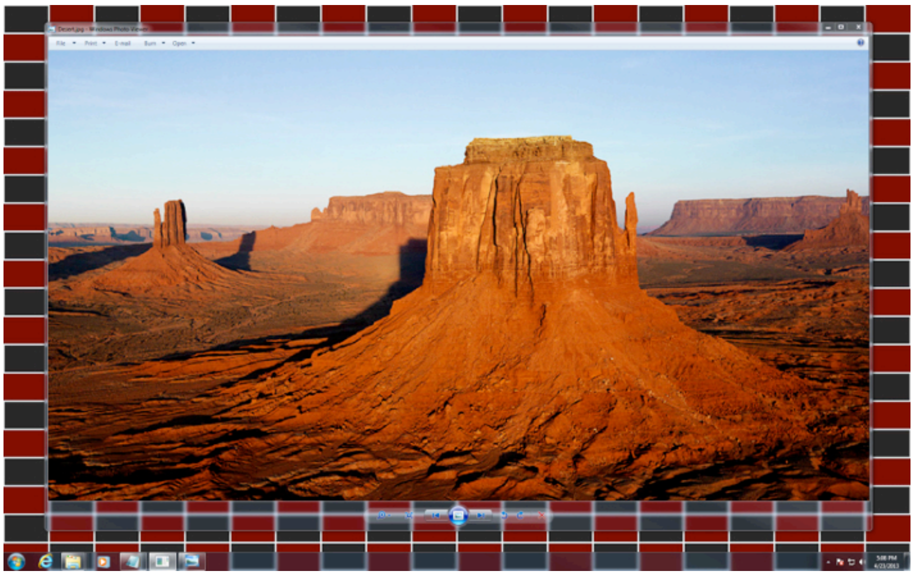


Figure 11

Barrel Distortion

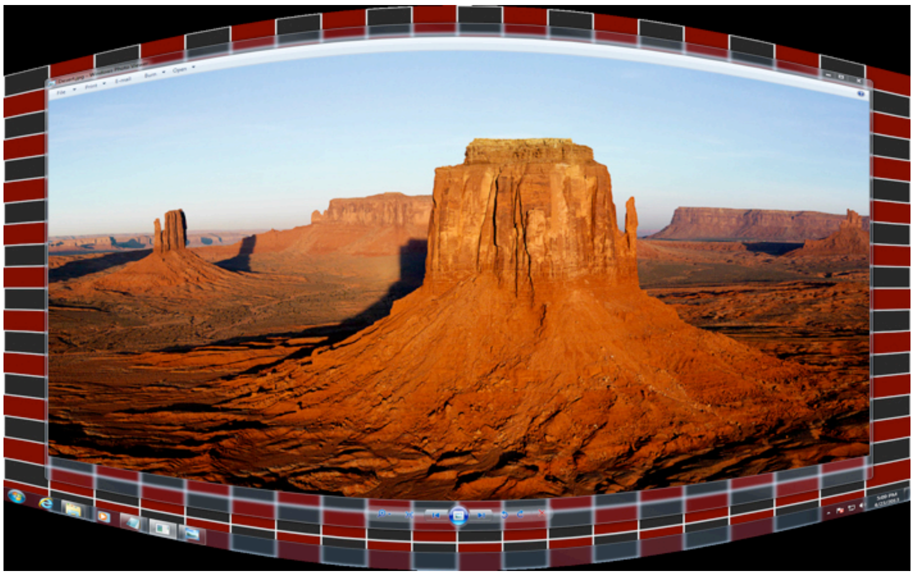


Figure 12

Curved Screen Y Compression