Using the Z Buffer for Visual and Special Effects

Introduction

The PlayStation 2 graphics chip is very powerful, flexible and has an incredible pixel fill rate for both un-textured and textured primitives. The Graphics Synthesizer is capable of processing 16 pixels per cycle when not performing texture mapping and 8 pixels per cycle when performing texture mapping. The GS runs at 150 Mhz allowing an achievable maximum pixel fill rate of 2.4 Giga pixels without texture mapping turned on and 1.2 Giga pixels with texture mapping. These performance figures do not reduce even when using 32 bit pixels with bilinear filtering, alpha blending, and Z buffer testing turned on. Also, the draw, display, z and texture buffers can be located at any location within the 4 Mega Bytes of Embedded DRAM on the PlayStation 2 graphics chip. All these buffers are not fixed to their pixel formats, a 24-Bit Z Buffer can actually be used as a 24-Bit Texture. With these flexible options, developers can take advantage of the massive pixel fill rate and perform some interesting visual and special effects. I'll describe two techniques that developers can easily incorporate into their game titles with very little overhead.

Using Z Buffer Channel for Fogging

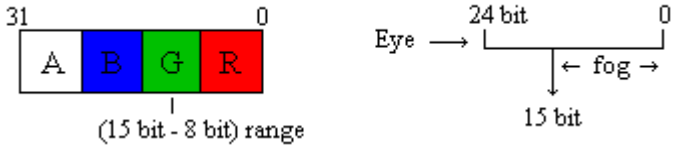
This technique demonstrates how to use the 'fog' feature of the GS, where objects appear to fade into the background as they move into the distance. This technique helps remove the ugly effect of 'pop-up', where large objects suddenly appear onscreen as they move into the camera's drawing range.

The GS allows fogging for all primitive types by setting the FOG attribute in the primitive. However, this per-vertex method actually reduces the fill rate and isn't perspective correct. The reason why it reduces the fill rate is because it's effectively a second alpha blend pass when applying the fog colour to the final primitive. The post process pixel-level fogging technique is better because it's perspective correct and is very fast to apply as a final pass over the current Draw Buffer. I'll now go into great detail on how to apply this technique into your game title.

Finding a good Z range

The GS channel copying trick can quickly and easily grab the bit range between 15-8 when using a 24/32-Bit Z Buffer, which maps out reasonably well onto your game world. These middle bits will start the fog in the middle of the scene and will extend to the far clipping plane. The Red and Blue channel bit ranges can be extracted but requires another technique that isn't as fast as grabbing the Green channel and they both don't very map well to the scene. Please see the diagram below

32-Bit Pixel Format

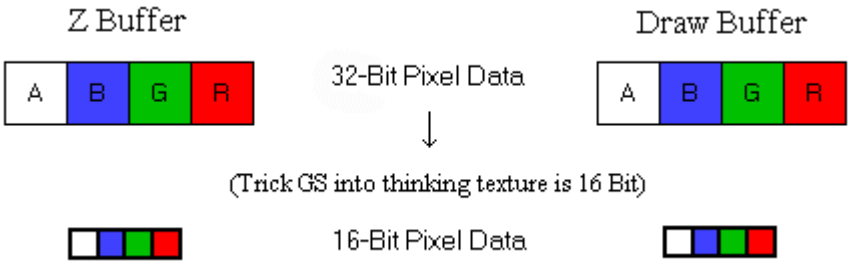


Moving the Green Channel of the Z Buffer

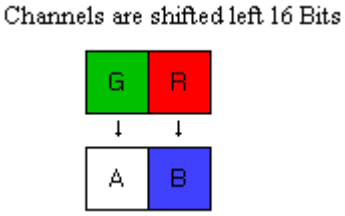
We therefore have to somehow find a fast and easy method to move the Green channel of the Z Buffer into the Alpha Channel of the Draw Buffer. There is basically one function that is used to convert the Z Buffer into a texture, move Red and Green channels into Blue and Alpha channels, and finally move the Alpha channel of the converted texture into the Alpha channel of the Draw Buffer. The first three steps can be done in one quick single pass:

1) Represent Z Buffer storage format as 32 Bit texture storage format
2) Move Red and Green channels of Z Buffer texture into Blue and Alpha channels of Z Buffer texture
3) Move Alpha channel of Z Buffer texture into Draw Buffer Alpha channel
4) Draw non-textured sprite as fog pass onto Draw Buffer using Destination Alpha

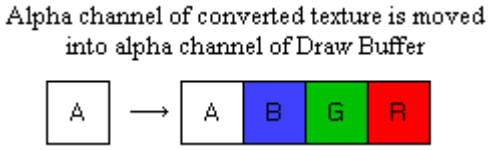
1)



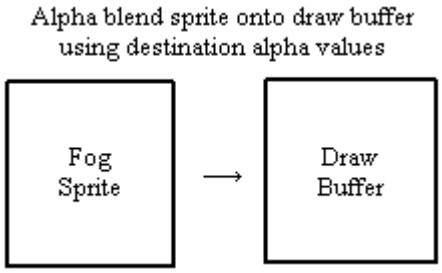
2)



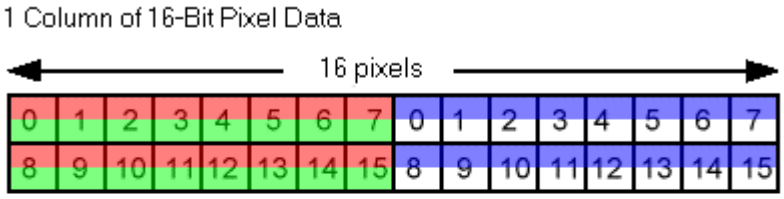
3)

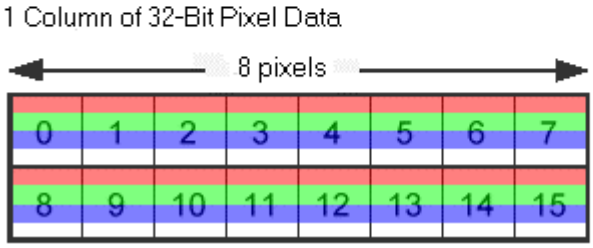


4)

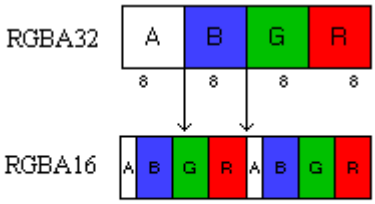


The secret to this function is to trick the GS into thinking the texture is 16-bit when it is actually 32-bit. The next two diagrams show the relationship between the two pixel formats.





The destination buffer for this function will be the current Draw Buffer. The pixel mode for this buffer is 16-Bit and the lower 8 Bits are masked out because copying is done in-place and only the alpha channel should be written to. The diagram below shows how the masking is achieved.



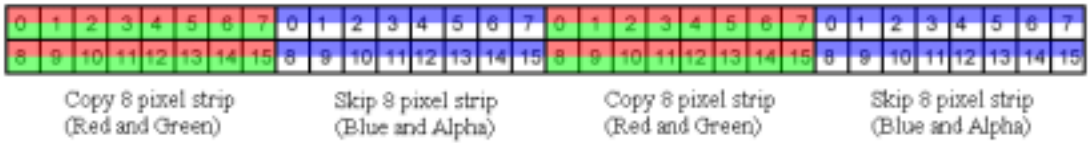
* Masking off 5bits of RED and 3bits of GREEN
* Remaining 8bits written to alpha channel

The Z Buffer is not used in this operation so the Z Writes are turned off and set the Z Testing is set to Z Always. It is also very important to change the scissoring region of the copying operation as because the GS is tricked into thinking the Z Buffer is 16-bit, the scissor area height must doubled before performing any buffer copying.

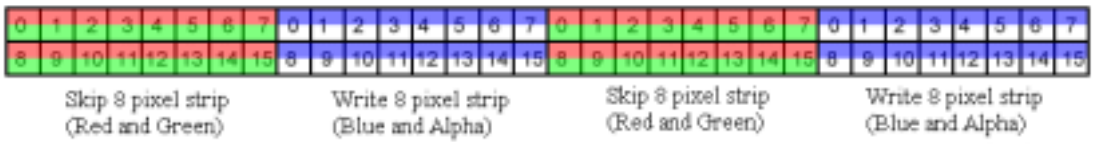
Why use 8 pixel wide strips?

The best way to copy between the source and destination buffers is to use eight pixel wide strips since only two channels of the Z Buffer will be copied. For the source buffer, eight pixel wide strips are copied and then eight pixels are skipped. The destination starting offset is eight pixels because only the Blue and Alpha channels are written to the buffer.

Z16-Bit Source Data



16-Bit Destination Data

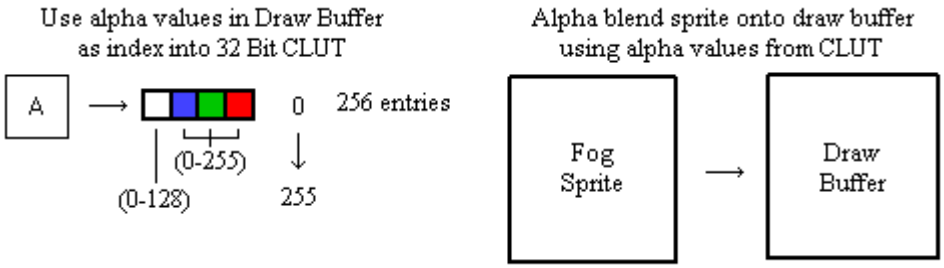


Creating the Fog Visual Appearance

Another function is used to draw one fog pass over the draw buffer using destination alpha values. There is no need to update the Z Buffer but the Z Test still must be performed. The alpha input value comes from the Alpha channel in the Draw Buffer, which happens to be the recently copied Green channel of the Z Buffer. A non-textured sprite is finally alpha blended over the Draw Buffer at a fixed Z range in 32 pixel wide strips.

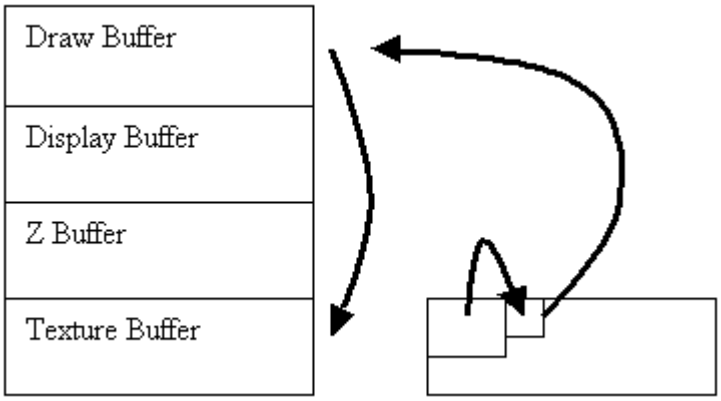
For speed the GS fetches both frame buffer and Z Buffer at the same time. The block layout for frame-buffer and Z Buffer is laid out in a format, which effectively divides the page into two parts. While the frame-buffer accesses the left side of a page, the Z Buffer will be accessing the right side (and vice-versa), allowing both fetches to happen from different physical memory blocks internally and thus occur in parallel. This means that the page buffer will only cache half a page of screen and half a page of Z Buffer at any given time. For this reason you should consider the page cache width for frame-buffer or Z Buffer to be effectively 32 pixels wide, rather than 64.

It is also possible to get a more linear fog curve by using an 8-Bit texture and CLUT to represent the fog density and the fog colour. The GS alpha value range is between 0-255 with 1.0 represented by 128 so that the alpha channel can be used to over saturate the scene with values greater than 128. By using the Alpha channel as an 8-Bit CLUT, the fogging range 0-255 can be mapped to 0.0-1.0.



Using Z Buffer Channel for Depth of Field

Another nice technique that can be achieved with the alpha values in the Draw Buffer is a depth of field effect. The lens flare effect used to be 'the' cool graphical effect but now depth of field seems to be the next big graphical effect that developers will incorporate into their game titles. This technique gives a convincing illusion that objects in the distance appear out of focus. After performing all drawing operations, bilinear shrink the Draw Buffer into the texture buffer. Depending on how much 'out of focus' you want to apply to your scene, another down sample might be necessary or you could try alpha blending the current image in-place to get a blurrier image. After down sampling the Draw Buffer, simply bilinear super sample the off screen buffer onto the current Draw Buffer using destination alpha values using 32 pixel wide sprites to reduce page breaks.



Conclusion

In conclusion, the GS is very flexible and powerful! As described in this article, developers can easily use the Z Buffer to create visual and graphical effects for their game titles. If developers have any further questions regarding these techniques, please feel free to contact me via the SCEE Developer Support email address.