

FBO Readback Using PBO

BY ASHWIN NANJAPPA

Web: <http://www.comp.nus.edu.sg/~ashwinna/>

Recently I learnt about *Pixel Buffer Objects (PBO)* and have been trying to use them to speed up readback of *FrameBuffer Objects (FBO)*. PBO can be used to speedup transfer of data from CPU to GPU too. But, that is not what this writeup is about.

1. First we create a PBO.

```
GLuint pbo;  
glGenBuffers(1, &pbo);
```

2. Inform the driver about the kind of data transfer the PBO will be used for. (I'm assuming the FBO is of type GLfloat here.)

```
glBindBuffer(GL_PIXEL_PACK_BUFFER_ARB, pbo);  
glBufferDataARB(GL_PIXEL_PACK_BUFFER_ARB, width * height * sizeof(GLfloat),  
NULL, GL_STREAM_READ_ARB);
```

3. Unbind the PBO so you can do other stuff. The PBO can be called into action later once the FBO is ready with the appropriate data.

```
glBindBufferARB(GL_PIXEL_PACK_BUFFER_ARB, 0);  
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, 0);
```

4. Assume you've finished all the operations on the FBO and are ready to readback the FBO to CPU. Now, bind the FBO's texture object as the source for the PBO readback. The `glReadPixels` call (with NULL parameter) doesn't block and just *initiates* a DMA transfer of the buffer from GPU to CPU (driver memory).

```
glReadBuffer(fboTex);  
glBindBuffer(GL_PIXEL_PACK_BUFFER_ARB, pbo);  
glReadPixels(0, 0, width, height, format, type, 0);
```

5. After the above call, you can go ahead and do some CPU operations while the GPU-to-CPU DMA transfer takes place in the background. If you can put your CPU to good use at this step, it essentially means that later you're getting the FBO readback done for almost zero cost.

6. Get the pointer to the PBO buffer memory (which is now on the CPU) and use it. This call blocks.

```
GLfloat * buf = NULL;  
glBindBuffer(GL_PIXEL_PACK_BUFFER_ARB, pbo);  
buf = glMapBuffer(GL_PIXEL_PACK_BUFFER_ARB, GL_READ_ONLY);
```

7. Either use the memory pointer thus obtained directly for read-only operations or copy its contents from driver space to application space (using say `memcpy`).

8. Once you're done with using the PBO memory, unbind it so that the driver can release the associated memory.

```
glBindBuffer(GL_PIXEL_PACK_BUFFER_ARB, pbo);  
glUnmapBuffer(GL_PIXEL_PACK_BUFFER_ARB);  
glBindBuffer(GL_PIXEL_PACK_BUFFER_ARB, 0);
```

9. Steps 4 to 8 can be repeated as many times as required.

10. At the end of your application, delete the PBO.

```
glDeleteBuffers(1, &pbo);
```

That's about it. Even though in my application I didn't put the CPU to any use at step 5, I saw my FBO readback speed double by using PBOs!

(The only drawback I can see is the stability of the application. I run into memory leaks if I leave my application running for a long time. I'm still not able to figure out if its a bug in my PBO usage or the NVIDIA driver.)

References

1. *GPGPU::Fast Transfer Tutorial* by Dominik Göddeke

<http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/tutorial3.html>

□