Introduction

After the release of Beginning OpenGL Game Programming 2nd Edition, certain OpenGL 3.x drivers became stricter in their compliance with the specification. One particular area which drivers started respecting was that of Precision Qualifiers in GLSL. This change caused many of the GLSL shaders from the book to stop working. The accompanying files fix this this issue. This document will explain what Precision Qualifiers are.

Precision Qualifiers

Precision qualifiers have no real meaning in GLSL and mainly exist for compatibility with OpenGL ES. Their existence in a GLSL 1.30 shader does not change the type of the variable they are associated with. However, the specification requires that these qualifiers are specified, even though though they have no effect. It could be argued that omission of a precision qualifier should not cause an error but, unfortunately, the behaviour in this situation depends on the driver.

Some examples of precision qualifiers in a GLSL shader follow:

```
in highp float my_var;
out lowp float my_var;
out mediump float my_var;
```

As I mentioned, the addition of highp, mediump and lowp has no effect in GLSL 1.30 but they do have an effect if the shader was used under OpenGL ES. Under OpenGL ES, the qualifiers indicate the minimum precision level that each variable should use (low, medium or high). More details can be found in the OpenGL ES Shading Language specification.

You can set a precision qualifier for float and integer declarations, this also affects types made up of several floating or integer types (e.g. vectors and matrices).

Default Precision Qualifiers

Although you can specify a precision qualifier for each variable individually, you can also specify a default precision qualifier for a shader stage (e.g. fragment). A default is set using the precision statement:

```
precision highp float;
```

This will declare the default precision qualifier for all float variables used in the shader. You can override this using per-variable qualifiers. The GLSL specification implicitly defines the following default precision qualifiers for vertex shaders:

```
precision highp float;
precision highp int;
```

It also defines the following precision qualifier for fragment shaders:

```
precision mediump int;
```

You will notice that there is no default precision qualifier specified for the float type in the fragment shader. This explains why not specifying a precision qualifier for floats in the fragment shader can cause an error on certain drivers. The solution is to add the following line to the fragment shader:

```
precision highp float;
```

Setting the float to high precision in the fragment shader matches the setting in the vertex shader. Setting it to medium or low precision can cause a type mismatch error with certain drivers. Although the specification is a little ambiguous about whether or not this is actually an error as the qualifiers have no semantics.