*Appendix L*

# `std140` Uniform Buffer Layout

This appendix describes the layout of the members of a uniform buffer object when the layout is specified with the layout modifier `std140`. It has the following major sections:

- "Using the `std140` Layout Qualifier"
- "`std140` Layout Rules"

## Using the `std140` Layout Qualifier

Generally, when you group a number of uniform variables in a uniform buffer object, you need to query the offset (and format and type, if you don't know those values from the shader source). For large collections of uniforms, this process can result in needed to make many queries. The `std140` layout qualifier requests that the GLSL shader compiler organize the variables in uniform block according to a set of rules, where you can predictably compute the offset of any member in the block, knowing its type and offset. This feature is only available with GLSL Version 1.40 at the time of this writing (but will presumably be backward compatible with future versions of GLSL).

In order to qualify a block to use the std140 layout, you need to add layout directive to its declaration, as demonstrated below:

```
layout (std140) uniform UniformBlock {
    // declared variables
};
```

## `std140` Layout Rules

The set of rules shown in Table L-1 are used by the GLSL compiler to layout members in a `std140`-qualified uniform block. The offsets of members in the block are accumulated based on the sizes of the previous members in the block (those declared before the variable in question), and the starting offset. The starting offset of the first member is always zero.

| Variable Type | Variable Size/Offset |
|---|---|
| Scalar variable type (bool, int, uint, float) | Size of the scalar in basic machine types (e.g., sizeof(GLfloat)) |
| Two-element vector (bvec2, ivec2, uvec2, vec2) | Twice the size of the underlying scalar type |
| Three-element vector (bvec3, ivec3, uvec3, vec3)<br>Four-element vector (bvec4, ivec4, uvec4 vec2) | Four times the size of the underlying scalar type |

**Table L-1** `std140` Layout Rules

| Variable Type | Variable Size/Offset |
|---|---|
| An array of scalars or vectors | Each element in the array is the size of the underlying type, and the offset of any element is its index (using zero-based indexing) times the elements size. The entire array is padded to be a multiple of the size of a vec4. |
| Single, or arrays of column-major matrices of size *C* columns and *R* rows | Stored as an array of *C* vectors each with *R* components, and potentially padded like other arrays.<br>If the variable is an array of *M* column-major matrices, it is stored as an array of $M \times C$ vectors each with *R* components (and potentially padded). |
| Single, or arrays of row-major matrices with R rows and C columns | Stored as an array of *R* vectors each with *C* components, and potentially padded like other arrays.<br>If the variable is an array of *M* column-major matrices, it is stored as an array of $M \times R$ vectors each with *C* components (and potentially padded). |
| A single structure definition, or an array of structures | Offsets and sizes of the structure's members are computed using the preceding rules. The structure's size will be padded out to a multiple of the size of a vec4.<br>An array of structures' offsets are computed considering the alignment and padding of the individual structures, with structure member's offsets computed using the preceding rules. |

**Table L-1 (continued)** `std140` Layout Rules