Introduction to Graphics Programming and its Applications

繪圖程式設計與應用

Appendix: OpenGL Shading Language

Instructor: Hung-Kuo Chu

Department of Computer Science

National Tsing Hua University

CS4505

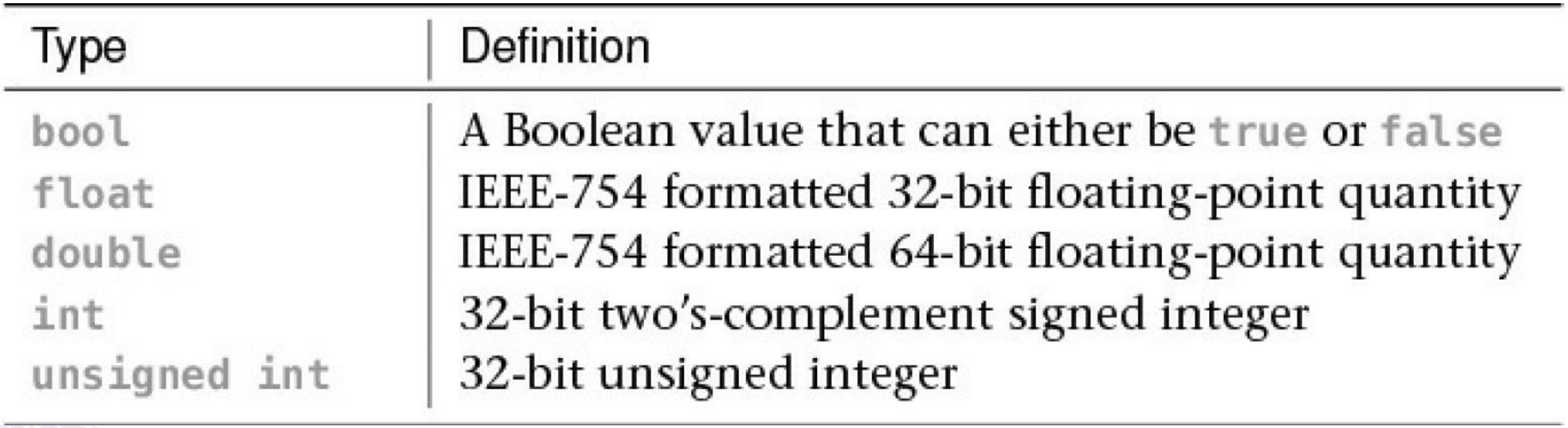
OpenGL Shading Language

- OpenGL Shading Language (**GLSL**), is a high-level shading language based on the syntax of the **C programming language**.
- With advances in graphics cards, new features have been added to allow for increased **flexibility** in the rendering pipeline at the **vertex** and **fragment** level.

Appendix

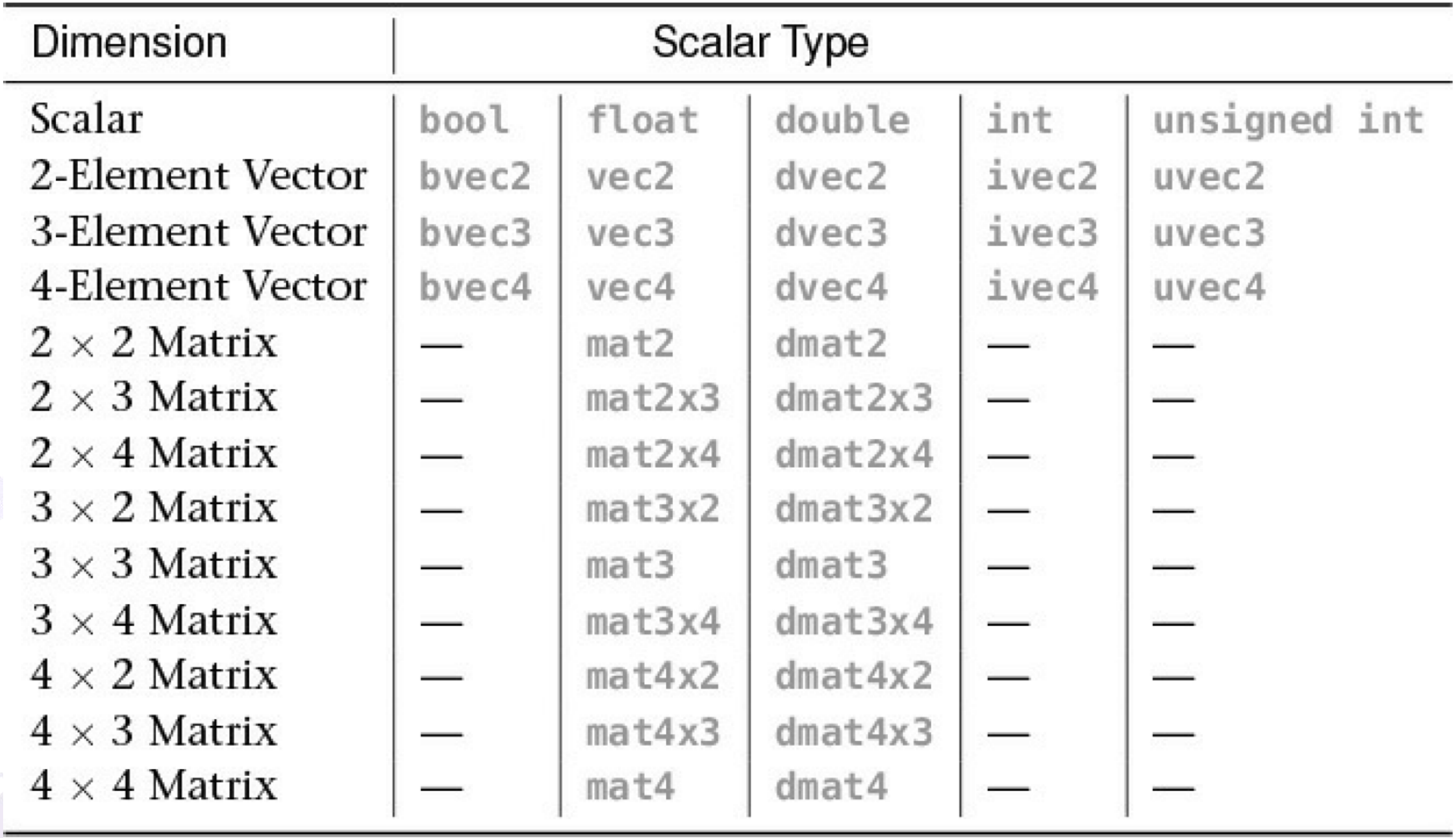
GLSL SYNTAX AND DATA TYPES

Scalar Types



Type	Definition
bool	A Boolean value that can either be true or false
float	IEEE-754 formatted 32-bit floating-point quantity
double	IEEE-754 formatted 64-bit floating-point quantity
int	32-bit two's-complement signed integer
unsigned int	32-bit unsigned integer

Vector and Matrix Types



Dimension	Scalar Type				
Scalar	bool	float	double	int	unsigned int
2-Element Vector	bvec2	vec2	dvec2	ivec2	uvec2
3-Element Vector	bvec3	vec3	dvec3	ivec3	uvec3
4-Element Vector	bvec4	vec4	dvec4	ivec4	uvec4
2 × 2 Matrix	—	mat2	dmat2	—	—
2 × 3 Matrix	—	mat2x3	dmat2x3	—	—
2 × 4 Matrix	—	mat2x4	dmat2x4	—	—
3 × 2 Matrix	—	mat3x2	dmat3x2	—	—
3 × 3 Matrix	—	mat3	dmat3	—	—
3 × 4 Matrix	—	mat3x4	dmat3x4	—	—
4 × 2 Matrix	—	mat4x2	dmat4x2	—	—
4 × 3 Matrix	—	mat4x3	dmat4x3	—	—
4 × 4 Matrix	—	mat4	dmat4	—	—

Creating and Accessing Vectors

- Constructors

```
vec3 foo = vec3(1.0);
vec3 bar = vec3(foo);
vec4 baz = vec4(1.0, 2.0, 3.0, 4.0);
vec4 bat = vec4(1.0, foo);
```

- Accessing

```
vec4 foo(1.0,2.0,3.0,4.0);
foo.xyz==foo.rgb==foo.stp;
foo.rrrr => a vec4 type vector (1.0,1.0,1.0,1.0);
foo.zyx => a vec3 typ vector (3.0,2.0,1.0);
foo.xyba => you can't mix the fields due to the union.
```

```
typedef union vec4_t
{
struct
{
                float x;
                float y;
                float z;
                float w;
};
struct
{
                float s;
                float t;
                float p;
                float q;
};
struct
{
                float r;
                float g;
                float b;
                float a;
};
} vec4;
```

Arrays

- Declaring array types is just like C or C++:

```
float var[6] = { 1.0, 2.0, 3.0, 4.0, 5.0, 6.0 };
```

- or we can also implicitly defines the constructor for the array.

```
float[6] var = float[6](1.0, 2.0, 3.0, 4.0, 5.0, 6.0);
```

Arrays of Arrays

- Although GLSL doesn’t officially support multidimensional arrays, it does support arrays of arrays. This means that you can put array types into arrays.

```
float a[10]; // 'a' is an array of 10 floats.
float b[10][2]; // 'b' is an array of 2 arrays of 10 floats.
float c[10][2][5]; // 'c' is an array of 5 arrays of 2
arrays of 10 floats.
```

Structures

- You can also build arrays of structure types like C or C++ in GLSL

```
struct foo
{
      int a;
      vec2 b;
      mat4 c;
};
struct bar
{
      vec3 a;
      foo[7] b;
};
bar[29] baz;
```

Built-in Matrix and Vector Functions

- Arithmetic/logical/boolean operators (e.g., such as +, -, *,etc).

```
vec4 mix(vec4 x, vec4 y, float a)
{
      return x + a * (y - x);
}
```

- Matrix functions
  - transpose(), inverse(), determinant(), outerProduct(), dot(), cross(), normalize(), etc.

https://en.wikibooks.org/wiki/GLSL_Programming/Vector_and_Matrix_Operations