WebGL: How to bind values to a mat4 attribute?

Asked 4 years, 5 months ago Active 2 months ago Viewed 2k times

3

3

In some WebGL application, let's assume that we have a GLSL vertex shader which starts like this:

```
attribute vec4 foo1;
attribute vec4 foo2;
attribute vec4 foo3;
attribute vec4 foo4;
```

and some corresponding Javascript code for binding a data structure for those attributes:

```
var buf = gl.createBuffer(), loc;
gl.bindBuffer(gl.ARRAY_BUFFER, buf);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array([…]));

loc = gl.getAttribLocation(program, 'foo1');
gl.enableVertexArray(loc);
gl.vertexAttribPointer(loc, 4, gl.FLOAT, false, 16, 0);

loc = gl.getAttribLocation(program, 'foo2');
gl.enableVertexArray(loc);
gl.vertexAttribPointer(loc, 4, gl.FLOAT, false, 16, 4);

loc = gl.getAttribLocation(program, 'foo3');
gl.enableVertexArray(loc);
gl.vertexAttribPointer(loc, 4, gl.FLOAT, false, 16, 8);

loc = gl.getAttribLocation(program, 'foo4');
gl.enableVertexArray(loc);
gl.vertexAttribPointer(loc, 4, gl.FLOAT, false, 16, 12);
```

Now, according to the GL ES 2.0 specs, a vertex shader attribute can be defined as either a `float`, `vec2`, `vec3`, `vec4`, `mat2`, `mat3` or `mat4`.

So if I change the vertex shader code to define just one `mat4` attribute, like so...

```
attribute mat4 foo;
```

... the question is **what is the corresponding JS code to bind some pointers to a `mat4` attribute?**

I have found the question mat3 attribute in WebGL, but the answer is not explicit enough. Reading the answers and some other documentation, it *seems* that the correct solution is along the lines of:

```
loc = gl.getAttribLocation(program, 'foo');
gl.enableVertexArray(loc);
gl.vertexAttribPointer(loc  , 4, gl.FLOAT, false, 16, 0);
gl.vertexAttribPointer(loc+1, 4, gl.FLOAT, false, 16, 4);
gl.vertexAttribPointer(loc+2, 4, gl.FLOAT, false, 16, 8);
```

Share Follow

asked Aug 9 '16 at 14:07

IvanSanchez

14.9k 3 20 33

1 Answer

Active Oldest Votes

7

You're correct. From the spec section 2.10.4

> When an attribute variable is declared as a `mat2`, its matrix columns are taken from the `(x, y)` components of generic attributes `i` and `i + 1`. When an attribute variable is declared as a `mat3`, its matrix columns are taken from the `(x, y, z)` components of generic attributes `i` through `i + 2`. When an attribute variable is declared as a `mat4`, its matrix columns are taken from the `(x, y, z, w)` components of generic attributes `i` through `i + 3`.

stride and offsets in WebGL are in bytes so I suspect you wanted

```
gl.vertexAttribPointer(loc  , 4, gl.FLOAT, false, 64, 0);
gl.vertexAttribPointer(loc+1, 4, gl.FLOAT, false, 64, 16);
gl.vertexAttribPointer(loc+2, 4, gl.FLOAT, false, 64, 32);
gl.vertexAttribPointer(loc+3, 4, gl.FLOAT, false, 64, 48);
```

Let's check

```
var vs = `
attribute mat4 matrix;
attribute vec4 color;

varying vec4 v_color;

void main() {
  gl_PointSize = 10.0;
  gl_Position = matrix * vec4(0, 0, 0, 1);
  v_color = color;
}
`;
var fs = `
precision mediump float;

varying vec4 v_color;

void main() {
  gl_FragColor = v_color;
}
`;

var m4 = twgl.m4;
var gl = document.querySelector("canvas").getContext("webgl");
var program = twgl.createProgramFromSources(gl, [vs, fs]);

var matrixLoc = gl.getAttribLocation(program, "matrix");
var colorLoc = gl.getAttribLocation(program, "color");

function r(min, max) {
  if (max === undefined) {
    max = min;
```

```
function makeBuffer(gl, array) {
  const buf = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, buf);
  gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(array), gl.STATIC_DRAW);
  return buf;
}

var buffers = {
  matrices: makeBuffer(gl, matrices),
  colors: makeBuffer(gl, colors),
};

gl.useProgram(program);

gl.bindBuffer(gl.ARRAY_BUFFER, buffers.matrices);
for (var ii = 0; ii < 4; ++ii) {
  gl.enableVertexAttribArray(matrixLoc + ii);
  gl.vertexAttribPointer(matrixLoc + ii, 4, gl.FLOAT, 0, 64, ii * 16);
}

gl.bindBuffer(gl.ARRAY_BUFFER, buffers.colors);
gl.enableVertexAttribArray(colorLoc);
gl.vertexAttribPointer(colorLoc, 4, gl.FLOAT, 0, 0, 0);

gl.drawArrays(gl.POINTS, 0, numPoints);
```

```
canvas { border: 1px solid black; }
```

```
<script src="https://twgljs.org/dist/4.x/twgl-full.js" crossorigin></script>
<canvas></canvas>
```

Run code snippet   Expand snippet

Share Follow   edited Nov 9 '20 at 14:07   answered Aug 9 '16 at 14:31
gman
79.5k 24 183 280

Thanks - that particular bit of the spec eluded me! (also, I copy-pasted code too quickly to tell bytes from 4-byte-groups apart) – IvanSanchez Aug 9 '16 at 15:01