Tugas IF3260 Grafika Komputer
Web GL Fundamentals - Manipulation and Image Processing

R. B. Wishnumurti - 13519203

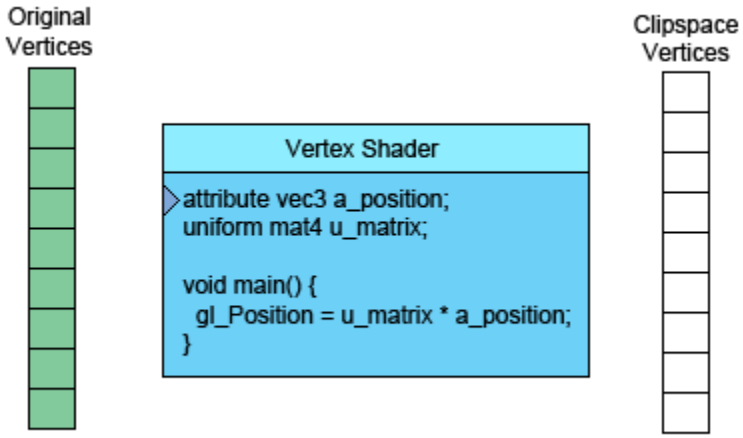
link repository github | link drive video

How it Works

Pada modul ini kita belajar bagaimana WebGL bekerja secara mendasar dan apa yang dilakukan oleh unit pemrosesan grafis. Proses pertama adalah memroses vertices menjadi clip space vertices dan proses kedua adalah menggambar piksel berdasarkan bagian pertamanya.

```
var primitiveType = gl.TRIANGLES;
var offset = 0;
var count = 9;
gl.drawArrays(primitiveType, offset, count);
```

Vertex sahader adalah sebuah fungsi yang ditulis dalam GLSL yang dipanggil untuk setiap vertex. Variabel gl_Position akan ditentukan dari hasil kalkulasi dengan nilai clipspace untuk verteks yang sekarang dan unit pemrosesan grafis akan menggambil nilai tersebut dan menyimpannya.



Kita ingin menggambar sebuah segitiga, proses dimulai dengan menentukan tiga titik yang nantinya akan di rasterisasi atau digamabar dengan piksel. Pada setiap piksel akan memanggil fragmen dari shader untuk menentukan warna dengan variable gl_FragColor pada piksel tersebut.

```
varying vec4 v_color;
...
void main() {
 // Multiply the position by the matrix.
 gl_Position = vec4((u_matrix * vec3(a_position, 1)).xy, 0, 1);

 // Convert from clip space to color space.
 // Clip space goes -1.0 to +1.0
 // Color space goes from 0.0 to 1.0
 v_color = gl_Position * 0.5 + 0.5;
}
```

```
precision mediump float;

varying vec4 v_color;

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

Misal kita memiliki 3 ujung/vertices dengan nilainya, WebGL akan menghubungkan variasi di vertex shader ke berbagai nama dan jenis yang sama di shader fragmen. Shader vertex kami menerapkan matriks untuk menerjemahkan, memutar, menskala, dan mengonversi ke clip space dan juga mengubahnya menjadi ruang warna dan menulisnya ke berbagai v_color yang ditentukan. Nilai yang tadi ditulis ke v_color akan diinterpolasi dan diterukan ke fragment shader unutk setiap piksel.

Kita juga dapat meneruskan lebih banyak data ke vertex shader yang kemudian dapat kita teruskan ke fragment shader. Saat kita menggambar persegi panjang, yang terdiri dari 2 segitiga, dalam 2 warna, kita akan menambahkan atribut lain ke vertex shader sehingga dapat meneruskan lebih banyak data ke fragmen shader.

Image Processing

Untuk menggambar gambar di WebGL kita perlu menggunakan tekstur mirip dengan cara WebGL mengekspetasikan koordinat clip space saat merender, bukan piksel, WebGL membutuhkan koordinat tekstur saat membaca tekstur yang berubah dari 0,0 hingga 1,0 terlepas dari dimensi teksturnya. Saat menggambar sebuah bentuk, kita perlu memberi tahu WebGL tempat mana dalam tekstur yang sesuai dengan setiap titik dalam persegi panjang. Kita akan meneruskan informasi ini dari vertex shader ke fragment shader menggunakan jenis khusus WebGL akan menginterpolasi nilai yang diberikan di vertex shader saat menggambar setiap piksel menggunakan fragment shader.

Saat menggunakan vertex shader, kita menambahkan atribut untuk meneruskan koordinat tekstur dan kemudian meneruskannya ke fragmen shader.

```
attribute vec2 a_texCoord;
...
varying vec2 v_texCoord;

void main() {
  ...
  // pass the texCoord to the fragment shader
  // The GPU will interpolate this value between points
  v_texCoord = a_texCoord;
}
```

Lalu kita supply fragment shader untuk mencari warna dari tekstur yang tersedia.

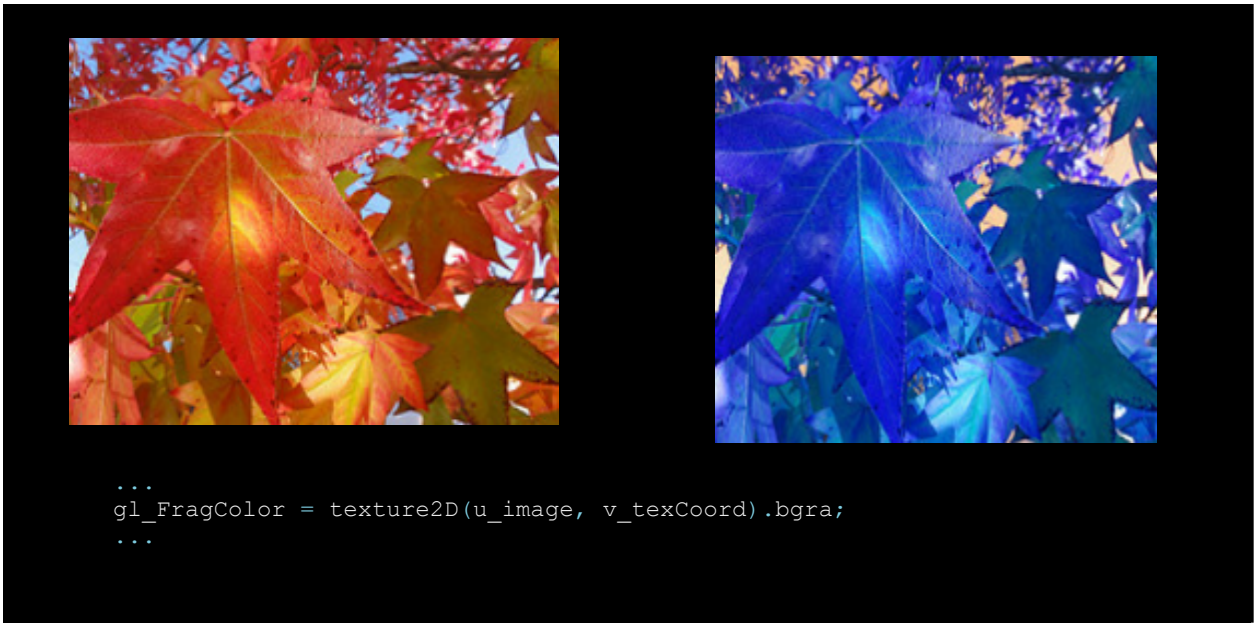
```
<script id="fragment-shader-2d" type="x-shader/x-fragment">
precision mediump float;

// our texture
uniform sampler2D u_image;

// the texCoords passed in from the vertex shader.
varying vec2 v_texCoord;

void main() {
  // Look up a color from the texture.
  gl_FragColor = texture2D(u_image, v_texCoord);
}
</script>
```

Lalu kita supply fragment shader untuk mencari warna dari tekstur yang tersedia. Lalu kita memuat sebuah gambar, membaut teksturm dan menyalin gambar ke tekstur dan gambar pun berhasil muncul. Kita dapt memanipulasi warna gambar misal dari warna merah ke biru dengan mengganti nilai dr gl_Fragcolor.



Kita juga bisa melakukan pemrosesan gambar yang mereferensikan piksel lain karena WebGL mereferensikan tekstur dalam koordinat tekstur yang berkisar dari 0,0 hingga 1,0 maka kita dapat menghitung berapa banyak yang harus dipindahkan untuk 1 piksel onePixel = 1.0 / textureSize. Kernel konvolusi hanyalah matriks 3x3 di mana setiap entri dalam matriks mewakili berapa banyak untuk mengalikan 8 piksel di sekitar piksel yang akan dirender. Hasilnya akan dibagi dengan bobot kernel (jumlah semua nilai dalam kernel) atau 1.0, manapun yang lebih besar.