***WebGL Fundamentals***

1. WebGL diturunkan dari OpenGL yang lebih dahulu masuk ke dalam dunia grafika komputer. Oleh karena itu, WebGL memiliki banyak kemiripan cara kerja dengan OpenGL.
2. Keberadaan WebGL sangat didukung oleh pengembangan HTML5 yang saat itu juga memperkenalkan komponen baru, yaitu *canvas*.
3. Keuntungan WebGL adalah WebGL memiliki manajemen memori yang otomatis (tidak seperti OpenGL), bekerja dengan cepat karena memiliki akses terhadap perangkat grafis, portabel dan memberikan publisitas karya yang lebih baik, tidak membutuhkan kompilasi, serta mudah diintegrasikan ke dalam *website* karena dapat dikembangkan dengan menggunakan JavaScript. Alhasil, banyak sekali pustaka dan kakas yang dimiliki oleh WebGL.
4. WebGL juga menggunakan GLSL (*Graphics Library Shading Language*) yang menggunakan versi standar OpenGL ES 2.0.
5. Biasanya, pada berbagai macam media grafika komputer tersedia tiga primitif atau bentuk dasar, yaitu titik, garis, dan segitiga. Hal ini dikarenakan ketiga bentuk ini lebih mudah dirasterisasi oleh GPU dibandingkan bentuk lainnya.
6. Dalam menggambarkan objek, WebGL menggunakan koordinat dengan prinsip *right-hand*. Selain itu, pada WebGL juga terdefinisi beberapa tipe data vector dan matriks seperti *vec2, vec3, vec4*, dan *mat4*.
7. WebGL hanyalah berupa mesin rasterisasi yang digunakan untuk menggambarkan titik, garis, dan segitiga. WebGL berjalan di atas GPU.
8. Untuk melakukan *rendering* gambar tertentu dengan WebGL, maka diperlukan dua fungsi utama. Fungsi tersebut adakah *vertex shader* yang akan digunakan WebGL untuk menghitung letak sebuah titik, sedangkan *fragment shader* digunakan untuk melakukan pemberian warna terhadap objek yang dibangun. Keduanya dapat ditulis dalam GLSL (GL *Shader Language*).
9. Untuk memberikan data terkait informasi titik dan warna tersebut, maka ada 4 cara, yaitu atribut dan *buffer*, *uniforms*, *varying*, dan *textures*. *Buffer* adalah larik data biner yang dikirimkan ke GPU dan dapat berisi apapun seperti koordinat tekstur serta posisi dan warna simpul. Atribut adalah cara bagi GPU untuk mengambil data yang ada di dalam *buffer. Uniforms* adalah variabel global yang ditetapkan sebelum mengeksekusi *shader*. *Varyings* adalah cara bagi *vertex shader* untuk memberikan data yang akan diinterpolasi oleh *fragment shader*. Adapun *textures* adalah larik data yang dapat diakses secara acak di dalam suatu program.
10. *Vertex Shader* pada WebGL bekerja terlebih dahulu sebelum dilanjutkan oleh rasterisasi yang dilakukan oleh *fragment shader*. Warna dan posisi dari suatu simpul dapat diberikan kepada *shader* melalui *buffer* secara bersamaan dengan memerhatikan urutan eksekusi tersebut.
11. Pada WebGL, terdapat konsep *clip space* yang merupakan wilayah tempat objek akan digambarkan. WebGL menggunakan koordinat *clip space* yang diberikan oleh pemrogram untuk menentukan bagaimana objek akan digambarkan dengan skala dan ukuran kanvas yang sesuai. *Clip Space* memiliki koordinat yang merentang dari 1 sampai -1, baik untuk sumbu X maupun sumbu Y.

***WebGL: How It Works***

1. Pada tingkat dasar, terdapat dua fungsi dari GPU dalam program WebGL. Fungsi pertama adalah melakukan pemrosesan *vertices* atau *data stream* dan melakukan transformasi yang sesuai ke dalam *clip space vertices*. Fungsi keduanya tentunya adalah menggambarkan *pixel* hasil pemrosesan tersebut.
2. Jumlah simpul yang ingin WebGL gambarkan diberikan pada metode *drawArrays*, lengkap dengan tipe primitif (titik, garis, segitiga) yang menjadi dasar rasterisasi dan *buffer offset* yang digunakan untuk memulai *shading*.
3. Pada *vertex shader*, ada atribut spesial yang digunakan untuk menentukan posisi dari suatu *vertex*, yaitu *gl\_Position*. Misalkan tipe primitif yang digunakan adalah segitiga, maka ketika sudah terdapat tiga buah simpul, WebGL akan melakukan rasterisasi yang sesuai untuk ketiga simpul tersebut sehingga membentuk segitiga dengan menggunakan warna yang ditetapkan melalui atribut spesial pada *fragment shader*, yaitu gl\_*FragColor*.