Richard Rivaldo
13519185 / K04

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* untuk setiap *pixel* yang digambarkan. 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.

Richard Rivaldo
13519185 / K04

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*.
4. *Buffer* adalah cara untuk memberikan simpul dan data lain mengenai simpul tersebut kepada GPU. Perintah *gl_createBuffer* melakukan pembuatan *buffer* baru, sedangkan *gl_bindBuffer* menjadikan *buffer* tersebut sebagai *buffer* yang akan dikerjakan. *gl_bufferData* menyalin data ke dalam *buffer* yang telah ditetapkan tadi.
5. *Buffer* dan isinya diinisialisasi di awal program. Agar WebGL dapat mengambil data yang ada di dalam *buffer* tersebut, maka WebGL perlu tahu lokasi atribut tempat data tersebut akan ditampung dengan perintah *gl_getAttribLocation*. Setelah itu, WebGL juga perlu diberitahu bahwa data akan diberikan melalui *buffer* dengan perintah *gl_enableVertexAttribArray*.
6. Setelah itu, maka dapat digunakan perintah *gl_vertexAttribPointer* yang menerima beberapa argumen. Selain lokasi atribut yang telah didapat tadi, perintah ini juga menerima jumlah komponen yang akan diambil dari *buffer* yang telah di-*bind* tadi (bernilai 1-4, sesuai dengan tipe data yang ada pada GLSL) serta tipe data yang akan diambil.
7. Perintah ini juga menerima argumen *stride* dan *offset* yang masing-masing menentukan *step* dari sebuah data dan data berikutnya yang akan diambil serta seberapa jauh data yang ingin diambil pertama kali dari *buffer* yang diberikan. *Stride* yang bernilai 0 berarti pergeseran setiap pengambilan data disesuaikan dengan ukuran tipe data yang dipilih, sedangkan *offset* yang bernilai 0 berarti data diambil dari awal *buffer*.
8. Selain argumen di atas, atribut lain yang juga diterima oleh perintah tersebut adalah *normalizeFlags*, yaitu atribut yang akan melakukan normalisasi, terutama terhadap *range* dari warna yang ada pada *buffer*, sesuai dengan *range* dari tipe data yang akan diproses oleh WebGL sesuai dengan argumen yang diberikan tadi.
9. Pada implementasinya, terdapat banyak sekali *boilerplate code* atau kode yang seringkali diulang-ulang dalam pembuatan suatu program WebGL, misalnya dalam melakukan *setup* untuk *shaders* dan program WebGL dengan menggunakan kode GLSL. Oleh karena itu, pustaka *WebGL Utils* menyediakan fungsi untuk mengurangi *boilerplate* tersebut.
10. Selain pustaka tersebut, juga terdapat banyak pustaka lain untuk pengembangan program WebGL, misalnya pustaka *m3.js* yang dapat digunakan untuk melakukan kalkulasi transformasi matriks objek yang ada pada WebGL dengan mudah.

Richard Rivaldo
13519185 / K04

WebGL: Image Processing

1. Untuk me-*render* suatu gambar, maka dapat digunakan tekstur pada WebGL. Sama seperti sebelumnya, WebGL tetap akan menggunakan koordinat *clip space* untuk menggambarkan tekstur tersebut, tetapi dengan rentang 0.0 sampai dengan 1.0.
2. Tekstur dapat diberikan dari *vertex shader* kepada *fragment shader* dengan menggunakan variabel yang bertipe *varying* (lagi-lagi, artinya adalah akan dilakukan interpolasi untuk setiap *pixel* yang akan digambarkan).
3. *Fragment shader* akan menggunakan informasi warna yang ada pada gambar yang menjadi sumber tekstur dan WebGL akan memetakan warna tersebut sesuai dengan koordinat yang telah diberikan melalui *vertex shader* tadi.
4. WebGL memungkinkan adanya manipulasi gambar yang dihasilkan melalui modifikasi, baik pada sistem warna yang digunakan, serta penggunaan *kernel* dan perubahan pada setiap *pixel* yang akan digambarkan berdasarkan perhitungan setiap *pixel*.
5. Pada WebGL, atribut *uniform* memiliki *default* 0 sehingga secara *default* pula akan menggunakan unit tekstur 0 yang juga merupakan unit tekstur aktif secara *default*.
6. Perintah *bindTexture* menggunakan unit tekstur tersebut, tetapi unit tekstur yang aktif dapat diubah dengan perintah *gl.activeTexture*. Setiap mesin memiliki jumlah maksimum unit tekstur yang dapat digunakan untuk masing-masing *vertex* maupun *fragment shader*.
7. Pada WebGL, terdapat suatu konvensi penulisan variabel pada kode GLSL. a_ digunakan sebagai *prefix* untuk atribut yang disuplai melalui *buffer*. Masukan *uniform* yang diberikan kepada *shader* diawali dengan *prefix* u_, sedangkan atribut *varying* menggunakan *prefix* v_.

Richard Rivaldo
13519185 / K04

Proyek Tutorial WebGL

1. Berikut merupakan tautan dari *repository* yang digunakan untuk menampung hal-hal yang berkaitan dengan proyek WebGL ini.
2. Video singkat mengenai penjelasan dari proyek ini dapat diakses melalui tautan YouTube berikut.
3. *Repository* tersebut berisikan beberapa proyek yang dilakukan berdasarkan *tutorial* WebGL, meliputi *WebGL Fundamentals*, *WebGL How It Works*, dan *WebGL Image Processing*. Di dalam *repository* ini juga terdapat rujukan kepada *repository* lain yang berisikan proyek WebGL lalu yang pernah dikerjakan.
4. Teknik-teknik serta konsep yang digunakan dalam pembuatan proyek-proyek tersebut telah dijelaskan pada bagian-bagian di atas.