midterm_legacy.js

```js
import {readShaderFile} from './examples/shader.js';

/* Pipeline for shader ~ program w/o shader.js */
// 1. shader source: readShaderFile 사용
const vertexShaderSource = readShaderFile('shVert.glsl');
const fragmentShaderSource = readShaderFile('shFrag.glsl');

// 2. create shader
vertexShader = gl.createShader(gl.VERTEX_SHADER);
fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);

// 3. shader source 붙이기
gl.shaderSource(vertexShader, vertexShaderSource);
gl.shaderSource(fragmentShader, fragmentShaderSource);

// 4. compile
gl.compileShader(vertexShader);
gl.compileShader(fragmentShader);

// 5. create Program
const program = gl.createProgram();

// 6. attatch shader to program
gl.attatchShader(program, vertexShader);
gl.attatchShader(program, fragmentShader);

// 7. link program
gl.linkProgram(program);

// 8. use 선언
gl.useProgram(program);

/* Pipeline for vao ~ draw call w/o utils */
// vao, vbo
const vertices = Float32Array([

])
const indices = Uint16Array([

])

const vao = gl.createVertexArray();
gl.bindVertexArray(vao);

const vertexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
// 만들어 둔 array의 data를 버퍼로 옮기기기
gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW)
// shader의 attribute location 활성화화
gl.enableVertexAttribArray(0) // (location)
// vertexAttribArray(attrib location, num of data per vertex, type of data, normalize,
stride, offset)
gl.vertexAttribArray(0, 2, gl_FLOAT, false, 0, 0);

const indexBuffer = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER. indexBuffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, indices, gl.STATIC_DRAW);

// actual draw call (in render())
// gl.drawElements(mode, index_count, type, byte_offset)
// gl.UNSIGNED_SHORT = Uint16Array
gl.drawElements(gl.TRIANGLES, 6, gl.UNSIGNED_SHORT, 0);
// gl.drawArrays(mode, first, count)
gl.drawArrays(gl.TRIANGLES, 0, 6);
```