HO CHI MINH CITY UNIVERSITY OF SCIENCE

FACULTY OF INFORMATION TECHNOLOGY

COMPUTER GRAPHICS

Week 01: WebGL Introduction

Associate Professors:

Lý Quốc Ngọc
Phạm Thanh Tùng
Võ Thế Hào

Student:

Võ Nguyễn Hoàng Kim – 21127090 – 21TGMT

Contents

A. Exercise

1. The goal of the " createShader " function? Why do we have to pass code as a string to create shaders?

createShader: This function serves the purpose of generating and compiling shaders in the context of WebGL. When it is successful, the function provides the compiled shader object as its output. In the event of a compilation error, it records the error and deallocates resources.

We must pass code as a string to create shaders due to the nature of these languages and how they are processed by the GPU. There are some reasons: [1]

- GPUs compile shaders during runtime, and they cannot directly execute high-level programming languages like JavaScript or Java. Offering the shader code as a string aids the GPU in parsing and compiling it quickly.
- By enabling developers to provide shader code as a string, they possess the flexibility to generate and adjust shaders as required.

2. Explain the meaning & parameters (including value type & data range) of the following functions.

I have based on the sample program provided to analyze these function (with meaning and their parameters).

a. gl.bufferData [2]

Meaning:

It is used to initialize a buffer object in WebGL with vertex data. It sets up specified data and usage pattern.

Parameters:

- **gl.ARRAY_BUFFER:** (GLenum) represents the specific buffer object target to which we aim to bind the data. It is used for the vertex attributes (like vertex coordinates, texture coordinates data, or vertex color data).
- **new Float32Array(vertices):** (GLintptr) creates a new Float32Array from the vertices array. Float32Array is a typed array that is often used to store 32-bit floating-point data,
- **gl.STATIC_DRAW:** (GLenum) specifies the expected usage pattern for the data. This data is expected to be static and used for many times.

b. gl.vertexAttribPointer [3]

Meaning:

It specifies how the vertex attributes (**coord**) are stored in a buffer and should be used during rendering.

Parameters

- **coord:** (GLuint) represents the location of the attribute within the vertex shader program.
- **3:** (GLint) This parameter specifies that each vertex attribute is composed of three values. This often represents 3D coordinates (x, y, z).
- **gl.FLOAT:** (GLenum) specifying the data type of each component in the array. It uses 32-bit IEEE floating point number.
- **false:** (GLboolean) specifying whether integer data values should be normalized into a certain range when being cast to a float. It is set to **false** for non-normalized data.
- **0:** (GLsizei) specifying the offset in bytes between the beginning of consecutive vertex attributes. Cannot be negative or larger than 255. it's 0, the attributes are tightly packed.
- **0:** (GLintptr) This parameter defines an offset in bytes to start reading data from the buffer. In this case, it starts at the beginning.

c. gl.viewport [4]

Meaning:

It is used to set the viewport for rendering, which specifies the affine transformation of **top** and **left** from normalized device coordinates to window coordinates.

Parameters

- **top** (GLint) specifies the starting position of the viewport, usually in pixels. Default value: 0.
- **left** (GLint) specifies the starting position of the viewport, usually in pixels. Default value: 0.
- **width:** (GLsizei) the non-negative value specifying the width of the view port. Default value: width of the canvas.
- **height:** (GLsizei) the non-negative value specifying the height of the view port. Default value: height of the canvas.

3. In what form are colors in WebGL transmitted/used? What is the range of values of color parameters / color channels?

[5] In WebGL, colors are transmitted/used in the form of RGBA values, where RGBA stands for Red, Green, Blue and Alpha.

These values typically represented as floating-point numbers or integers. To be more specified:

Red: controls the intensity of the red component. This value is in the range of 0.0 (no red) to 1.0 (full red).

Green: controls the intensity of the green component. This value is in the range of 0.0 (no green) to 1.0 (full green).

Blue: controls the intensity of the blue component. This value is in the range of 0.0 (no blue) to 1.0 (full blue).
Alpha: controls the transparency or opacity of the color. This value is in the range of 0.0 (full transparency) to 1.0 (full opacity).

4. **In the function "clearGL" what will happen when the line gl.enable(gl.DEPTH_TEST)" is uncommented.**

   The line **gl.enable(gl.DEPTH_TEST)** activates depth comparisons and updates to the depth buffer [6]. When this line in the **clearGL** function is uncommented, it enables depth testing in the WebGL rendering context. It is used to determine the visibility of objects based on their depth in the 3D scene, ensuring that objects in the foreground are rendered on top of objects in the background.

5. **When using Jsfilddle, if the program has an error (the code cannot run), how can we find out which line has the error?**

   To identify the erroneous line, we can inspect the Console. However, the Console within Jsfilddle might not pinpoint the exact problematic line in the JavaScript field, but instead, it hints at a line in the source file, which can be accessed through the browser's developer tools.
   Thus, if we want to know exactly what erroneous line is, we could open developer tools to find out it (through the error is displayed on the Console in this tool). When you access the source file, you'll notice that the line with an error is indicated by a red error symbol displayed at its end.

B. References

[1] [Online]. Available: https://stackoverflow.com/questions/57930141/why-does-webgl-use-string-representations-of-glsl-code-blocks.

[2] "WebGLRenderingContext: bufferData() method," Mozilla, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/bufferData.

[3] "WebGLRenderingContext: vertexAttribPointer() method," Mozillia, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/vertexAttribPointer.

[4] "WebGLRenderingContext: viewport() method," Mozillia, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/vertexAttribPointer.

[5] "Using shaders to apply color in WebGL," Mozillia, [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Tutorial/Using_shaders_to_apply_color_in_WebGL.

[6] "WebGLRenderingContext: enable() method," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/WebGLRenderingContext/enable.