Computer Graphics (COMP 4071) Midterm October 24, 2022

**Rules** This is an Open Book midterm. You may access our textbook, and any of the files on our class Moodle pages, including PowerPoint slides of the lectures, code in the book’s example files, or code I have uploaded to help you in your homework. You may access your own homework submission files. You may also access any handwritten notes you have prepared in advance, plus up to 5 pages (double-sided pages count as 2 pages) of photocopied or downloaded and printed information that you have prepared in advance. You may not access any other books or reference files either online or pre-downloaded onto your laptop. (Instead summarize the content you want on your handwritten notes or your 5 printed pages.) You may not access other websites or online information, nor may you communicate with other people or bots via email, chat, or any other communication method. You may use a calculator on your laptop or cellphone for arithmetic.

1. (5 points, read the above rules before starting, and answer after you complete the other questions) Copy and acknowledge with your typed or handwritten signature the words: “I have read and followed the rules above about this Open Book exam.”

Ans: “I have read and followed the rules above about this Open Book exam.”

1. (15 points) List all the GLSL ES shader program statements necessary to pass an interpolated color to a Fragment Shader and use it as its output to the frame buffer. You do not need to include the JavaScript statements to specify the vertex locations and colors to the Vertex Shader. You may assume that the color of a vertex has already been passed to the attribute variable a\_Color in the Vertex Shader. The question concerns only how the colors get interpolated and passed to the Fragment shader, and sent to its output color. Identify which statements belong to which shader, and whether they are inside or above that shader program main () block. You do not need to include the red characters ‘ … \n ‘ + .

Ans:

when you specify a different color for each the triangle’s vertex. Previously, you saw that the value assigned to the varying variable inside the vertex shader is passed as a varying variable with identical attributes (same name, same type) to the fragment shader (see Figure 5.14). However, to be more precise, the value assigned to the varying variable in the vertex shader is interpolated at the rasterization stage. Consequently, the value passed to the fragment shader actually differs for each fragment based on that interpolation (see Figure 5.15). This is the reason the varying variable has the name “varying.”

More specifically, in ColoredTriangle, because we only assign a different value to the  
varying variable for each of the three vertices, each fragment located between vertices  
must have its own color interpolated by the WebGL system.  
For example, let’s consider the case in which the two end points of a line are specified  
with different colors. One of the vertices is red (1.0, 0.0 0.0), whereas the other one is blue (0.0, 0.0, 1.0). After the colors (red and blue) are assigned to the vertex shader’s v\_Color, the RGB values for each of the fragments located between those two vertices are calculated and passed to the fragment shader’s v\_Color.

var VSHADER\_SOURCE =

…........

'attribute vec4 a\_Color;\n' +

'varying vec4 v\_Color;\n' + // variable changes

'void main() {\n' +

…........

' v\_Color = a\_Color;\n' +

'}\n';

var FSHADER\_SOURCE =

…..........

'varying vec4 v\_Color;\n' +

'void main() {\n' +

' gl\_FragColor = v\_Color;\n' + // get data from vertex shader

'}\n';

1. (20 points) Let A be a 3D point with viewing coordinates (-1.5, 1.2, -1.5). What are the perspective projection clipping coordinates in the [-1, 1] cube of this point if the near clipping plane is at z = -1, (near distance 1), the far clipping plane is at z = -2 (far distance 2), the field of view angle is 90 degrees, and the aspect ratio is 1? Note that tan (45 degrees) = 1.

Ans:

1. (10 points) The yellow RGBA color (1.0, 1.0, 0.0, 0.4) is to be composited over the cyan RGB color (0., 1., 1.) in the frame buffer. What is the resulting RGB color in the frame buffer?

Ans: The resulting color will be (0.4, 1.0, 0.6).

The questions continue on the next page.

1. Shape

   Description automatically generatedA picture containing text, monitor

   Description automatically generated(10 points) In the left image above, A is at (-.5, -.5) with RGB red color (1., 0., 0.), B is at (-.5, .5) with green RGB color (0., 1., 0.), C is at (.5, .5) with blue RGB color (0., 0., 1.) and D is at (.5, -.5) with white RGB color (1., 1., 1.). The colors are linearly interpolated across triangles ABC and ACD. What is the RGB color of the point E = (-.25, -.25), which lies ¼ of the way from A to C along the diagonal line AC?

D

CVCV

A

D

CVCV

C

B

B

C

A

Ans: Resulting color of E will be (0.75, 0, 0.25).

1. (15 points) In the right image above, the positions and colors of the four vertices A, B, C, and D are the same as in question 5 above, and so is the position of the point E, but now the colors are bilinearly interpolated between the four vertices. What is the RGB color of point E?

Ans: Resulting color of E will be (0.75, 0.375, 0.25).

7) (25 points) Specify all the JavaScript and GLSL ES shader program statement additions or revisions necessary to change examples/ch02/HelloPoint2.js to pass a floating-point number specified in the JavaScript code to the Vertex Shader to use for the point size, rather than hard coding it in the Vertex Shader. You are permitted, but not required, to test these changes by running the code on your laptop. You answer may just list the additions and revisions, or may include the whole changed code.

Ans: we will put this statement ('attribute float a\_PointSize; \n' +) inside vertex shader program and we will also add (' gl\_PointSize = a\_PointSize;\n') inside main function of vertex shader program. Furthermore, we will add (var a\_PointSize = gl.getAttribLocation(gl.program, 'a\_PointSize');) in line 39 of examples/ch02/HelloPoint2.js program and finally we will add (gl.vertexAttrib1f(a\_PointSize, 5.0);) in line 42 of the respective program.

Here is the full modified code:

// HelloPint2.js (c) 2012 matsuda

// Vertex shader program

var VSHADER\_SOURCE =

  'attribute vec4 a\_Position;\n' + // attribute variable

  'attribute float a\_PointSize; \n' +

  'void main() {\n' +

  '  gl\_Position = a\_Position;\n' +

  '  gl\_PointSize = a\_PointSize;\n' +

  '}\n';

// Fragment shader program

var FSHADER\_SOURCE =

  'void main() {\n' +

  '  gl\_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +

  '}\n';

function main() {

  // Retrieve <canvas> element

  var canvas = document.getElementById('webgl');

  // Get the rendering context for WebGL

  var gl = getWebGLContext(canvas);

  if (!gl) {

    console.log('Failed to get the rendering context for WebGL');

    return;

  }

  // Initialize shaders

  if (!initShaders(gl, VSHADER\_SOURCE, FSHADER\_SOURCE)) {

    console.log('Failed to intialize shaders.');

    return;

  }

  // Get the storage location of a\_Position

  var a\_Position = gl.getAttribLocation(gl.program, 'a\_Position');

  if (a\_Position < 0) {

    console.log('Failed to get the storage location of a\_Position');

    return;

  }

  var a\_PointSize = gl.getAttribLocation(gl.program, 'a\_PointSize');

  // Pass vertex position to attribute variable

  gl.vertexAttrib3f(a\_Position, 0.0, 0.0, 0.0);

  gl.vertexAttrib1f(a\_PointSize, 5.0);

  // Specify the color for clearing <canvas>

  gl.clearColor(0.0, 0.0, 0.0, 1.0);

  // Clear <canvas>

  gl.clear(gl.COLOR\_BUFFER\_BIT);

  // Draw

  gl.drawArrays(gl.POINTS, 0, 1);

}

Remember to answer question 1 when you are finished with the exam.