POLITECNICO
MILANO 1863

Exercises 02

Computer Graphics 2021

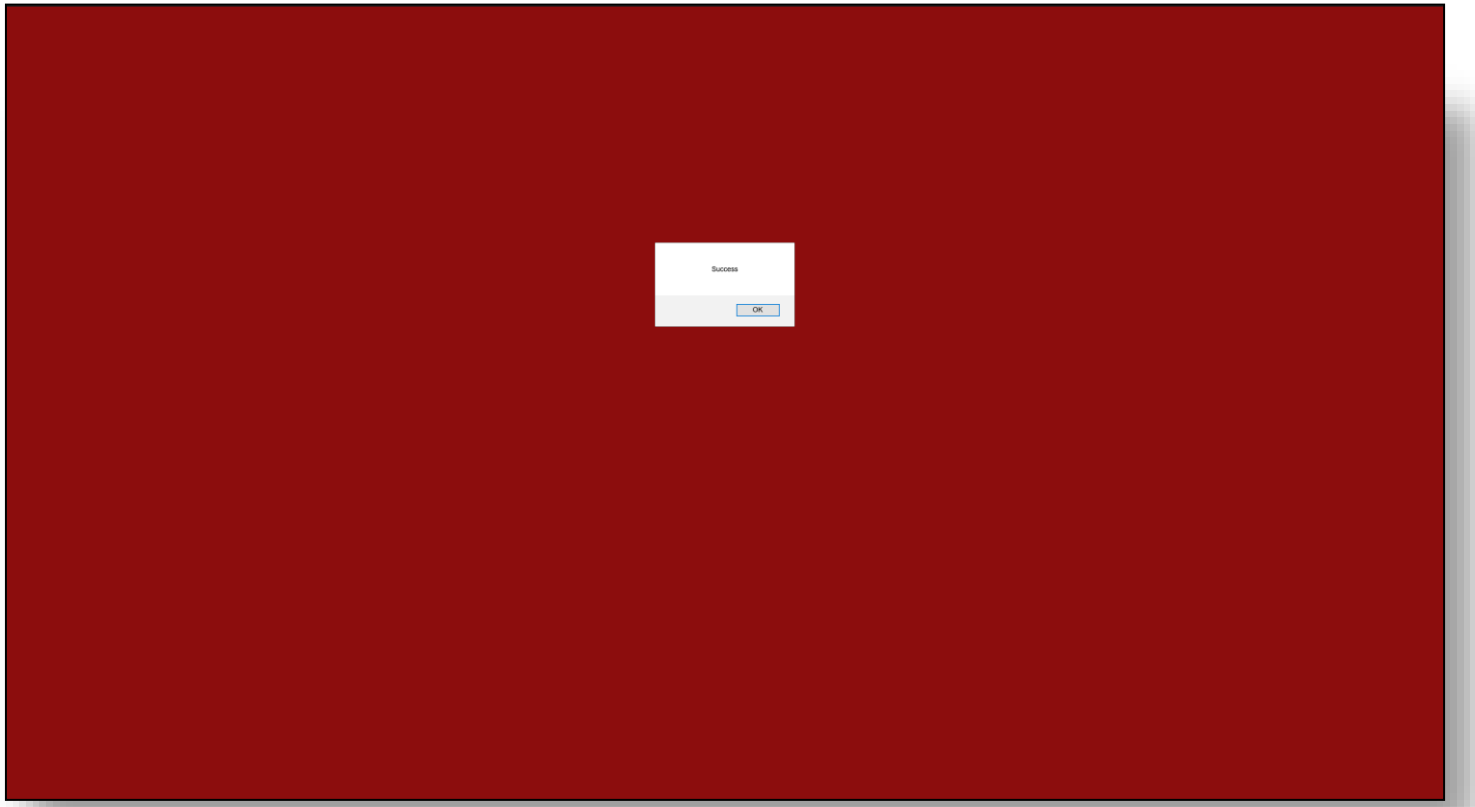
Erica Stella (erica.stella@polimi.it)

DISCLAIMER

- These exercises are purely for learning purposes, they will NOT be asked during the exams
- Don't worry if you don't finish them today, solutions will be posted, but **please review them before the next lesson** because we will build upon them
- If you have questions, you can use the forum or ask me next time ☺
- Before plunging into the code, please open the file for a quick look ☺

Ex 1

- Get a WebGL2 context from the canvas, resize the canvas to full screen, and use the **`alert`** function to notify whether the context was retrieved successfully
- Hint:
  - Use the **`autoResizeCanvas`** function in script.js
- Final look:



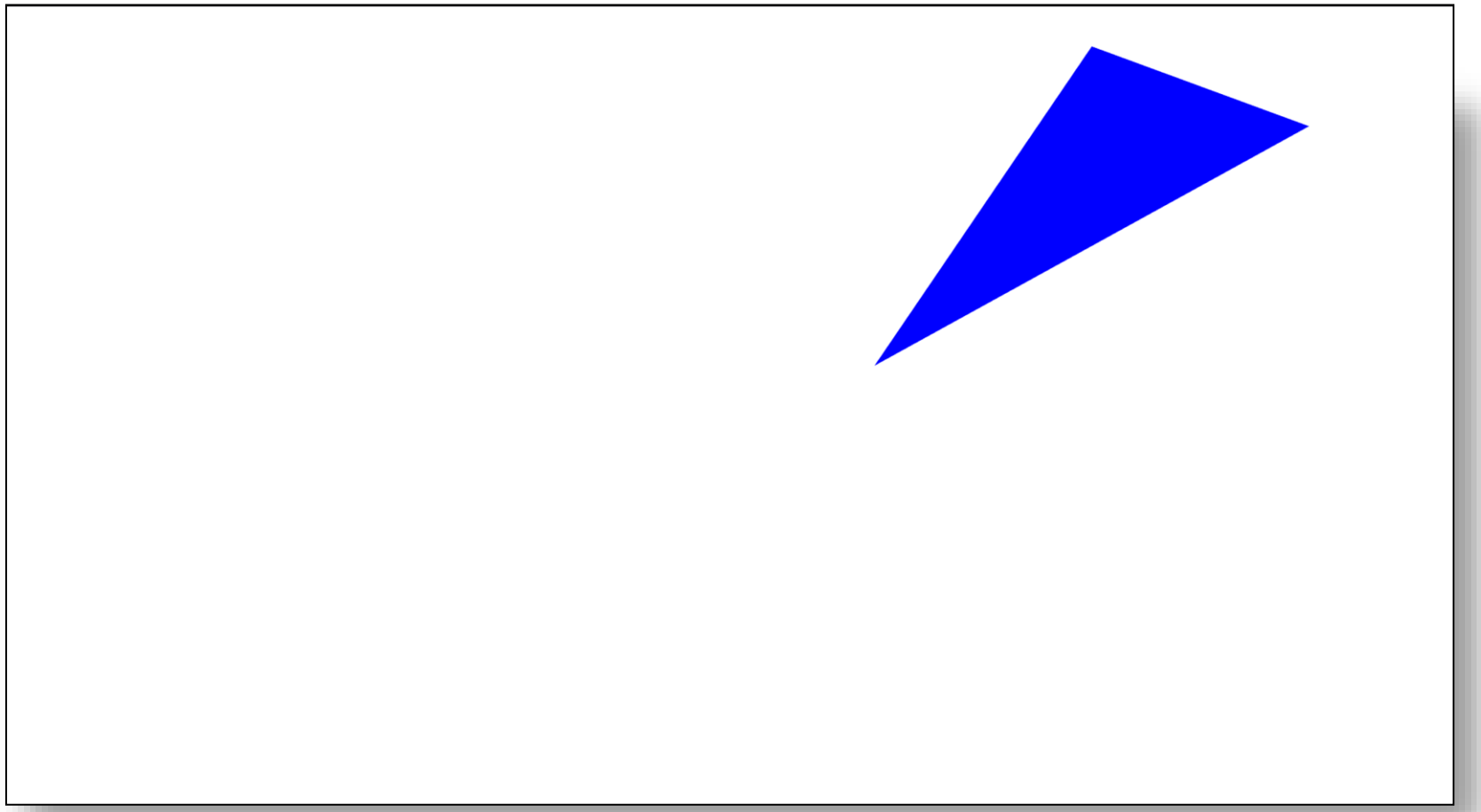
Solution Ex 1

script.js

```
function main() {
  var canvas = document.getElementById("my-canvas");
  autoResizeCanvas(canvas);
  var gl = canvas.getContext("webgl2");
  if (!gl) {
    alert("GL context not opened");
    return;
  }else{
    alert("Success")
  }
}
```

Ex 2

- Set up the VBO to pass the x,y coordinates in the **`positions`** array to the **`a_position`** attribute of the GLSL ES program
- Final look:



Solution Ex 2

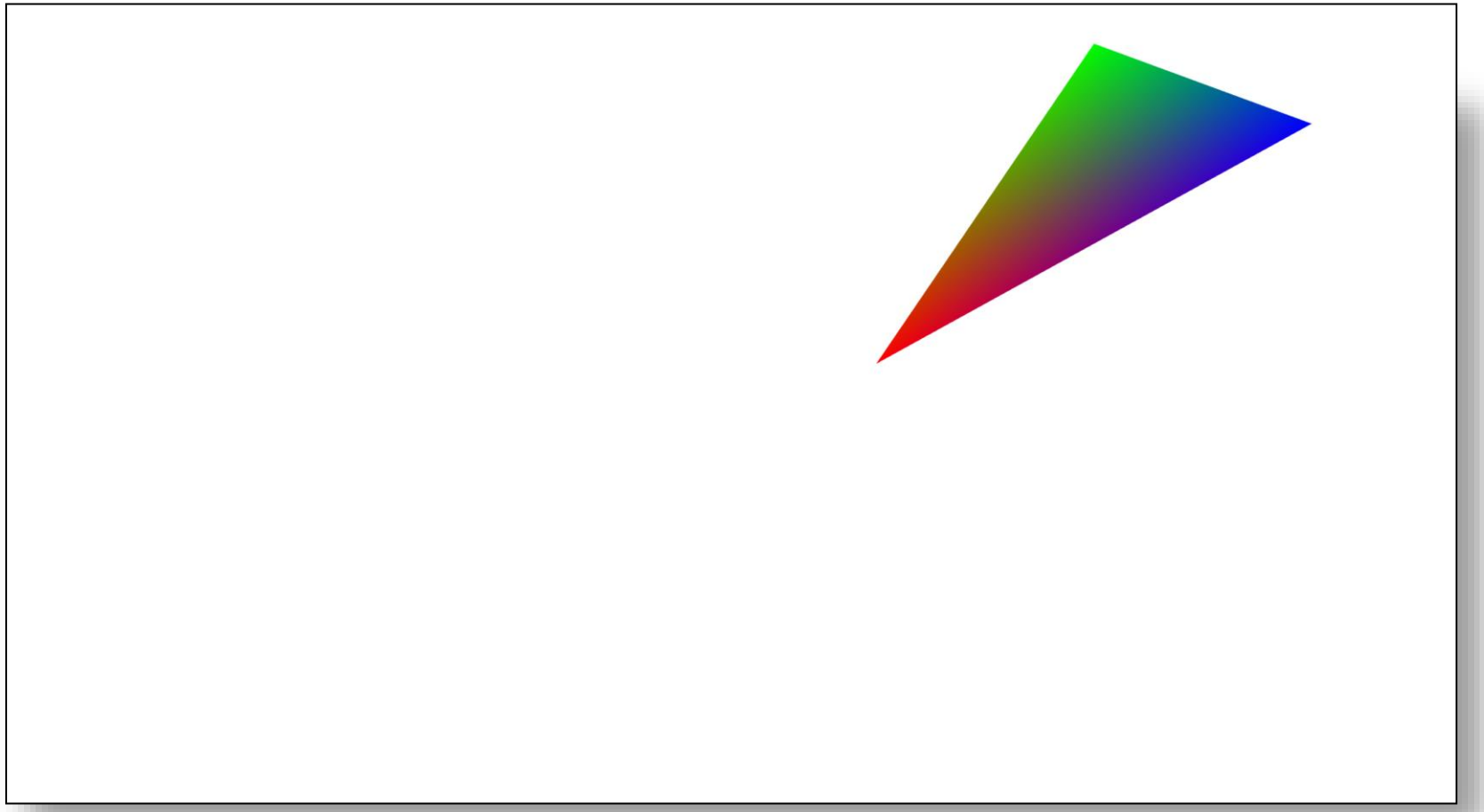
- Same as the one in the example folder ☺

script.js

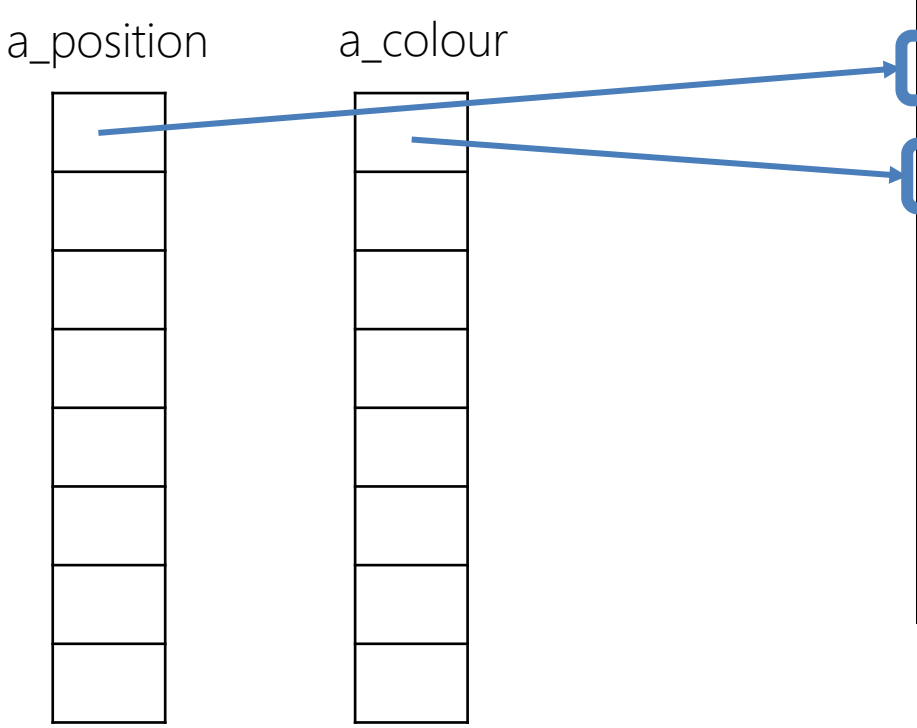
```
  // Create a buffer and put three 2d clip space points in it
  var positionBuffer = gl.createBuffer();
  // Bind it to ARRAY_BUFFER because it's going to contain attribute data
  gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
  //Send data in positions to the currently-bound buffer in the gpu
  gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(positions), gl.STATIC_DRAW);
  //Look up where the vertex data needs to go.
  var positionAttributeLocation = gl.getAttribLocation(program, "a_position");
  //Turn on the attribute
  gl.enableVertexAttribArray(positionAttributeLocation);
  // Tell the attribute how to get data out of positionBuffer (ARRAY_BUFFER)
  var size = 2;          // 2 components per iteration
  var normalize = false; // don't normalize the data
  var stride = 0;        // 0 = move forward stride * sizeof(type) each iteration to get the ne
xt position
  var offset = 0;        // start at the beginning of the buffer
  gl.vertexAttribPointer(positionAttributeLocation, size, gl.FLOAT, normalize, stride, offset);
```

Ex 3

- Set up the VBO to pass the RGB colours in the **`colours`** array to the **`a_colour`** attribute of the GLSL ES program
- Final look:



Ex 3 – Behind the scenes



Vertex Shader

```
#version 300 es

in vec4 a_position;

in vec3 a_colour;

out vec3 fs_col;

void main() {
  fs_col = a_colour;
  gl_Position = a_position;
}
```

Solution Ex 3

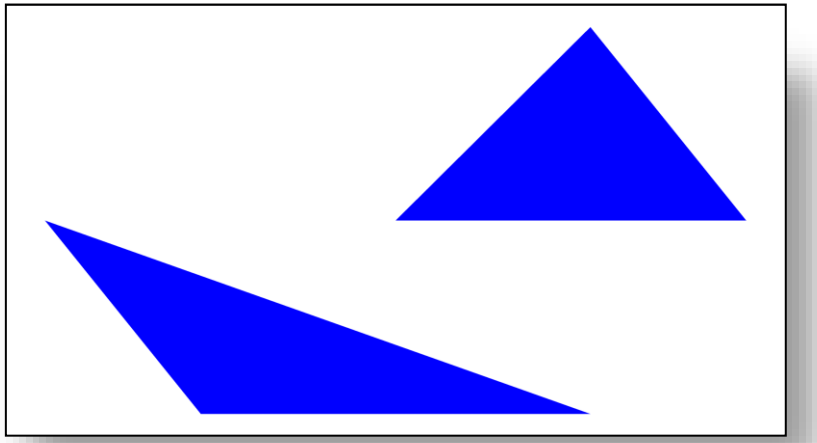
script.js

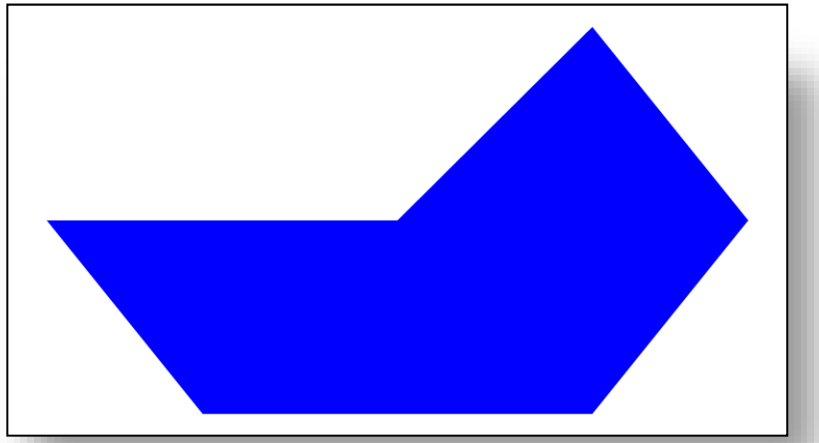
```
//Create a buffer to hold the rgb colours
var colourBuffer = gl.createBuffer();
// Bind it to ARRAY_BUFFER because it's going to contain attribute data
gl.bindBuffer(gl.ARRAY_BUFFER, colourBuffer);
//Send data in colours to the currently-bound buffer in the gpu
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(colours), gl.STATIC_DRAW);
//Look up where the colour data needs to go
var colourAttributeLocation = gl.getAttribLocation(program, "a_colour");
//Enable the attribute
gl.enableVertexAttribArray(colourAttributeLocation);
//We are passing 3 values (rgb) at a time
size = 3;
normalize = false;
stride = 0;
offset = 0;
gl.vertexAttribPointer(colourAttributeLocation, size, gl.FLOAT, normalize, stride, offset);
```

Ex 4

- Use the positions array to draw the following primitive types: **`gl.TRIANGLES`**, **`gl.TRIANGLE_FAN`**, **`gl.LINES`**, **`gl.LINE_LOOP`**
- Final look:

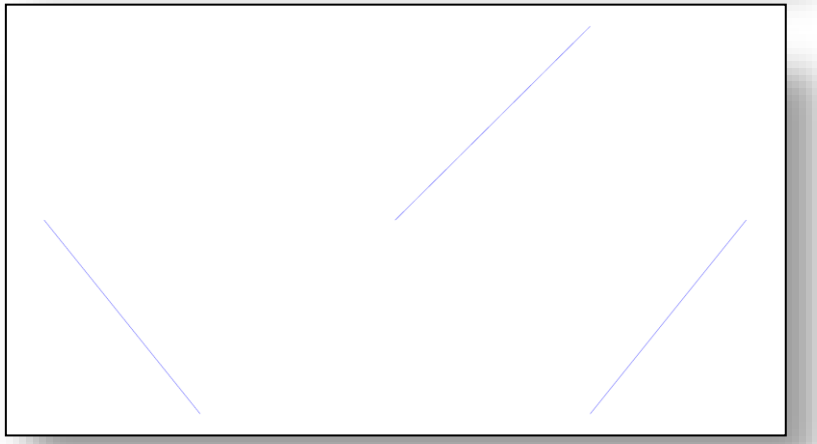
gl.TRIANGLES

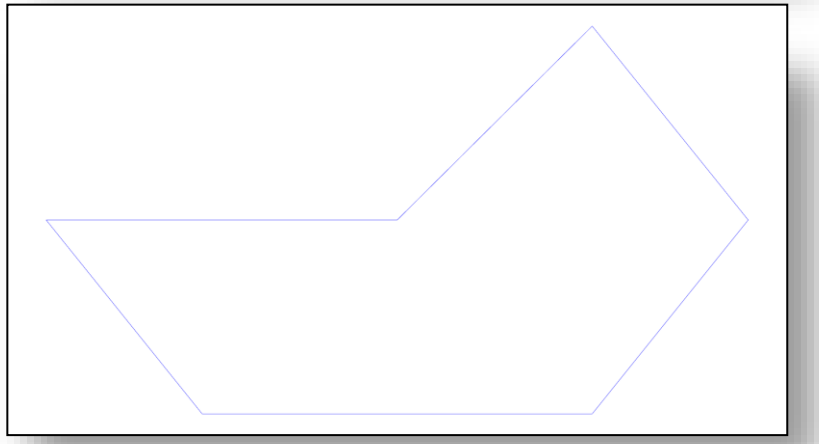




gl.TRIANGLE_FAN

gl.LINES





gl.LINE_LOOP

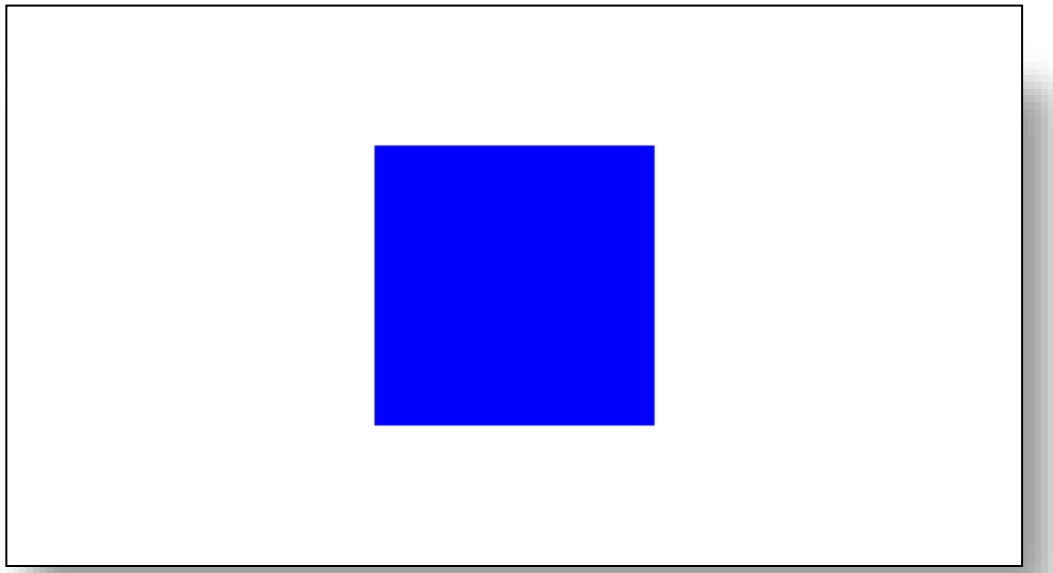
Solution Ex 4

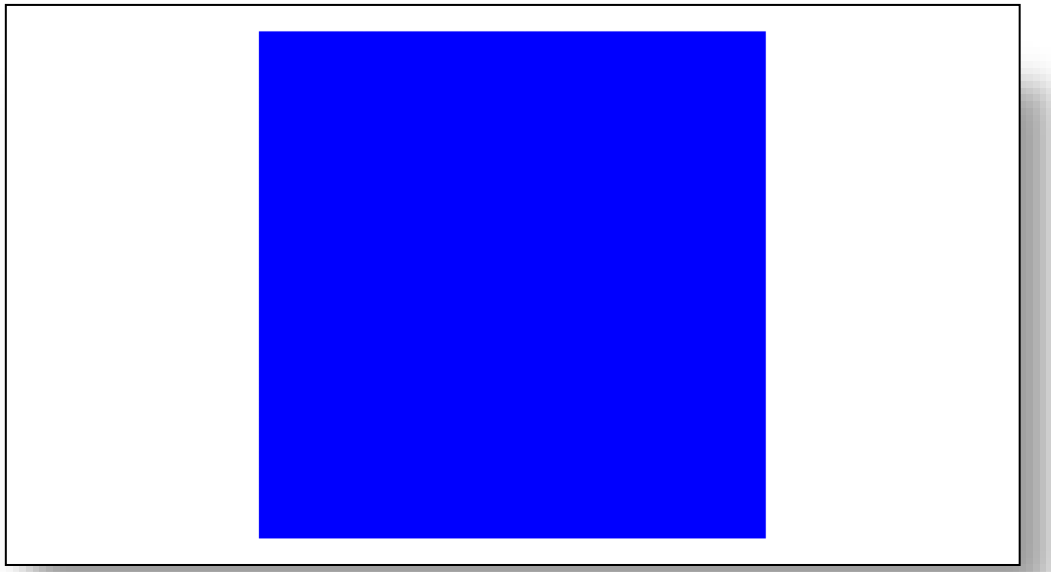
script.js

```
function main() {
  […]
  //Draw call
  //var primitiveType = gl.TRIANGLES;
  //var primitiveType = gl.TRIANGLE_FAN;
  //var primitiveType = gl.LINES;
  var primitiveType = gl.LINE_LOOP;
  var offset = 0;
  var count = 6; //Pay attention to the number of vertices you have to draw
  gl.drawArrays(primitiveType, offset, count);
}
```

Ex 5

- Apply aspect ratio correction to the vertices of the rectangle in the **`positions`** array (which should actually be a quad if you look at its vertices…)
- Final look:





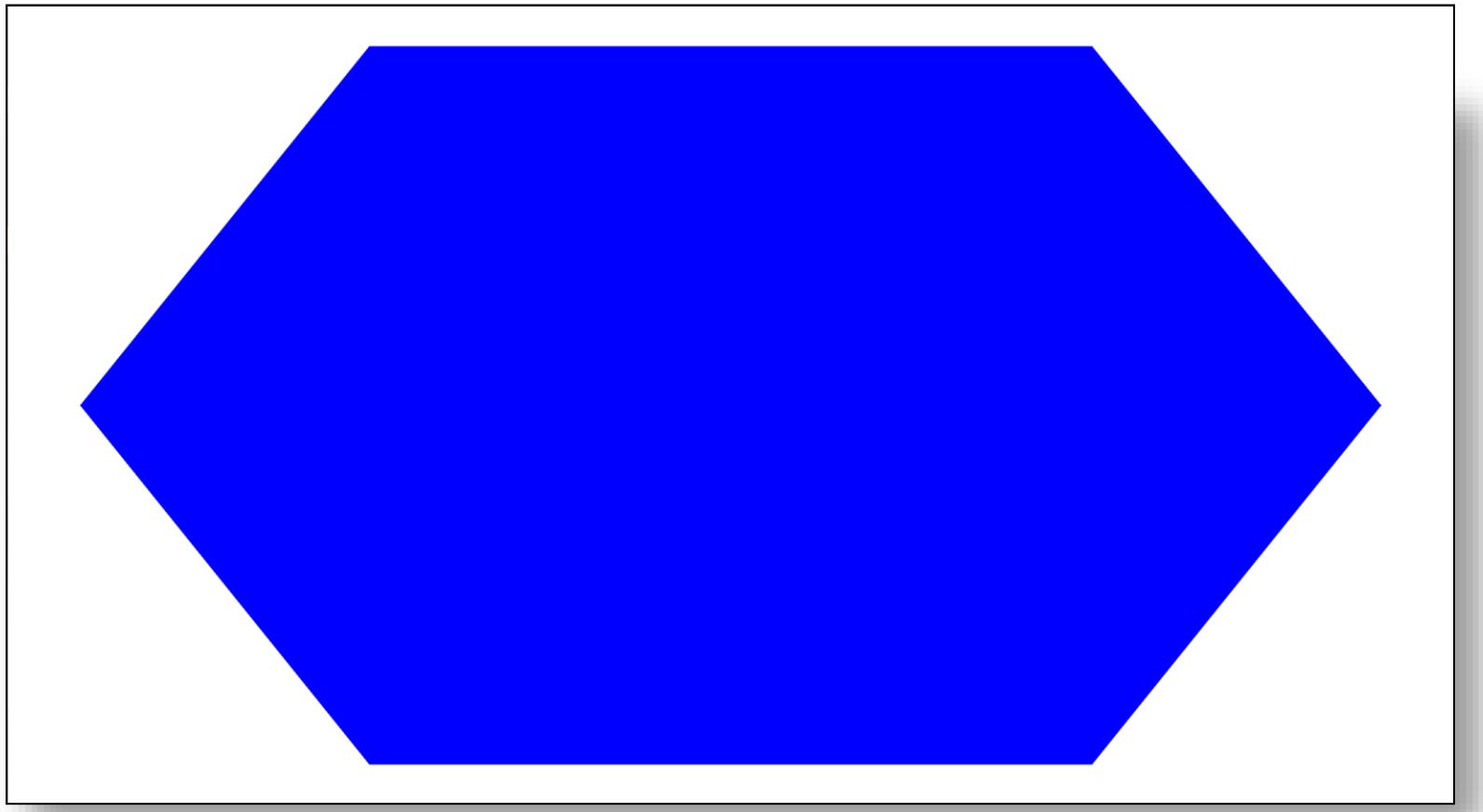
Solution Ex 5

script.js

```
var aspectRatio = gl.canvas.width/gl.canvas.height;

var positions = [
  -0.5,-0.5 * aspectRatio,
  -0.5,0.5 * aspectRatio,
  0.5,0.5 * aspectRatio,
  0.5,-0.5 * aspectRatio
];

/*This is also correct
var positions = [
  -0.5/aspectRatio,-0.5,
  -0.5/aspectRatio,0.5,
  0.5/aspectRatio,0.5,
  0.5/aspectRatio,-0.5
];*/
```

Ex 6

- Draw the shape represented by the **`positions`** array and the **`indices`** array with **`gl.TRIANGLES`** as primitive type.
- Final look:



Solution Ex 6

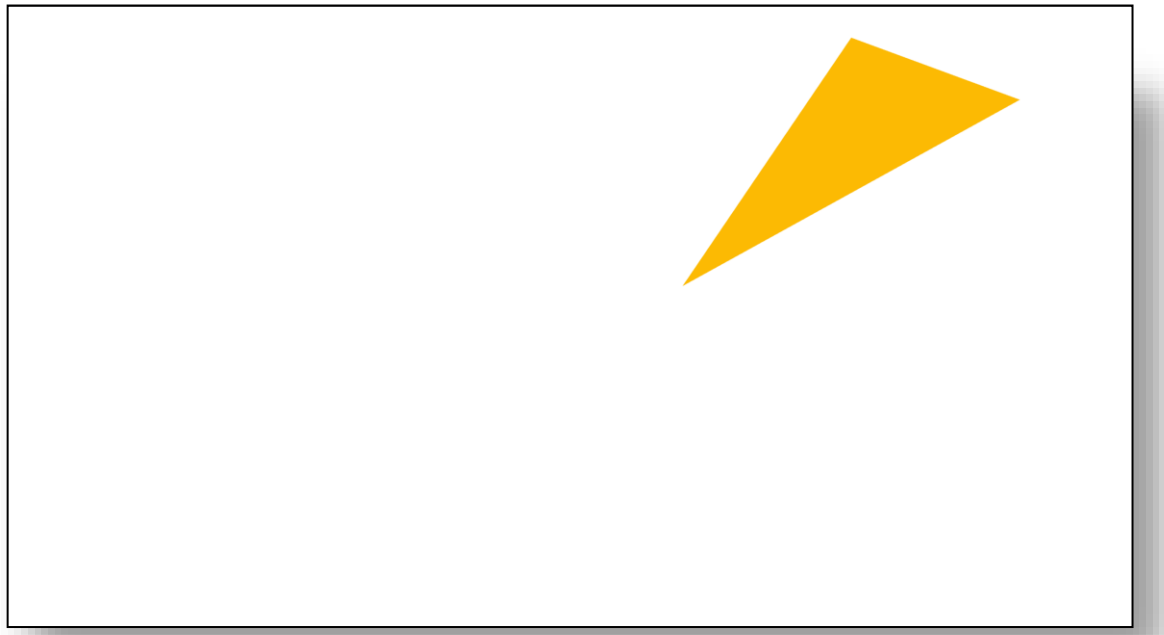
script.js

```
//Create the buffer that will hold the indices and send the data
var indices = [0,1,2,0,2,3,0,3,4,0,4,5,0,5,6,0,6,1];
var indexBuffer = gl.createBuffer();
//Here the buffer must be gl.ELEMENT_ARRAY_BUFFER to specify it contains indices
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, indexBuffer);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, new Uint16Array(indices), gl.STATIC_DRAW);

[..]

//bind index buffer to be sure that is the current active
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, indexBuffer);
//drawElements uses the indices to draw the primitives
gl.drawElements(gl.TRIANGLES, indices.length, gl.UNSIGNED_SHORT, 0 );
```

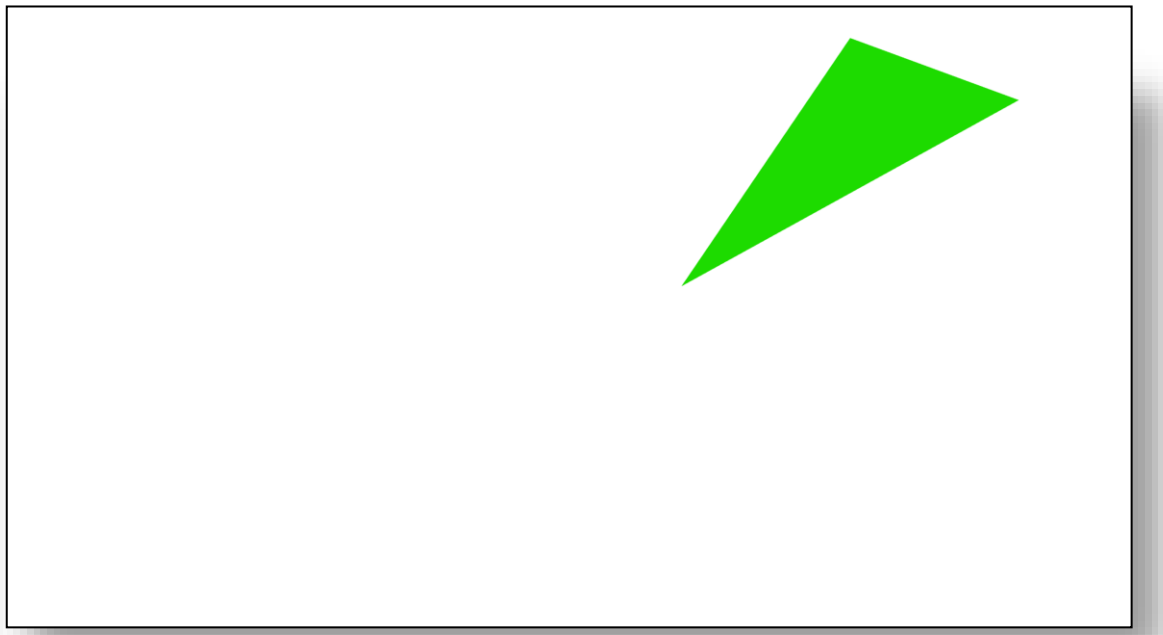
Ex 7

```
uniform vec3 u_colour;
uniform float colour_choice;
uniform vec2 u_second_colour;
```

- In the GLSL program, these 3 uniforms are specified:
- You need to pass these 3 uniforms to the GLSL program from WebGL. By changing colour_choice between 1.0 and 0.0 you should see the change in colour pictured in the images below
- Final look:



colour_choice = 1.0



colour_choice = 0.0

Solution Ex 7

script.js

```
//******Initialisation of variables******/
var colour = [252.0/255.0, 186.0/255.0, 3.0/255.0];
var colour_choice = 0.0;
var second_colour = [29.0/255.0, 219.0/255.0];
//******Retrieving the uniforms locations******/
var colourLocation = gl.getUniformLocation(program, "u_colour");
var colourChoiceLocation = gl.getUniformLocation(program, "colour_choice");
var secondColourLocation = gl.getUniformLocation(program, "u_second_colour");
[..]
gl.useProgram(program);
//Three values passed as an array
gl.uniform3fv(colourLocation, colour);
//One value passed singularly
gl.uniform1f(colourChoiceLocation, colour_choice);
//Two values passed as an array
gl.uniform2fv(secondColourLocation, second_colour);
```