Introduction to WebGL

COMP 531 presentation

Jing Guo

11/29/2016

WebGL = Web Graphics Library

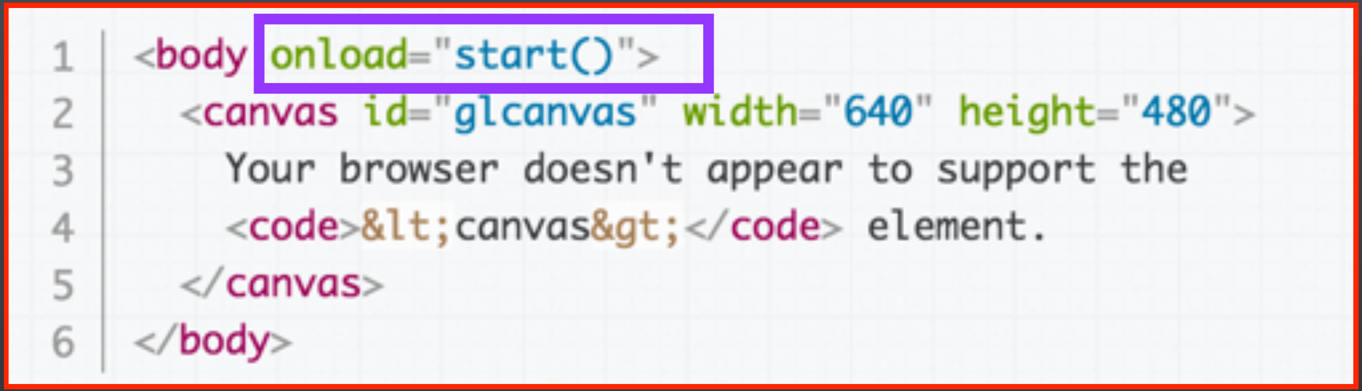
- Render 2D/3D graphics in the browser
  - ✦ Provide Javascript API based on OpenGL ES 2.0
  - ✦ An example
- No plug-in required, access to GPU
  - ✦ Powerful, scalable, flexible
  - ✦ Render to `<canvas>` element in HTML
- Support by many browsers

chrome Opera Internet Explorer 11

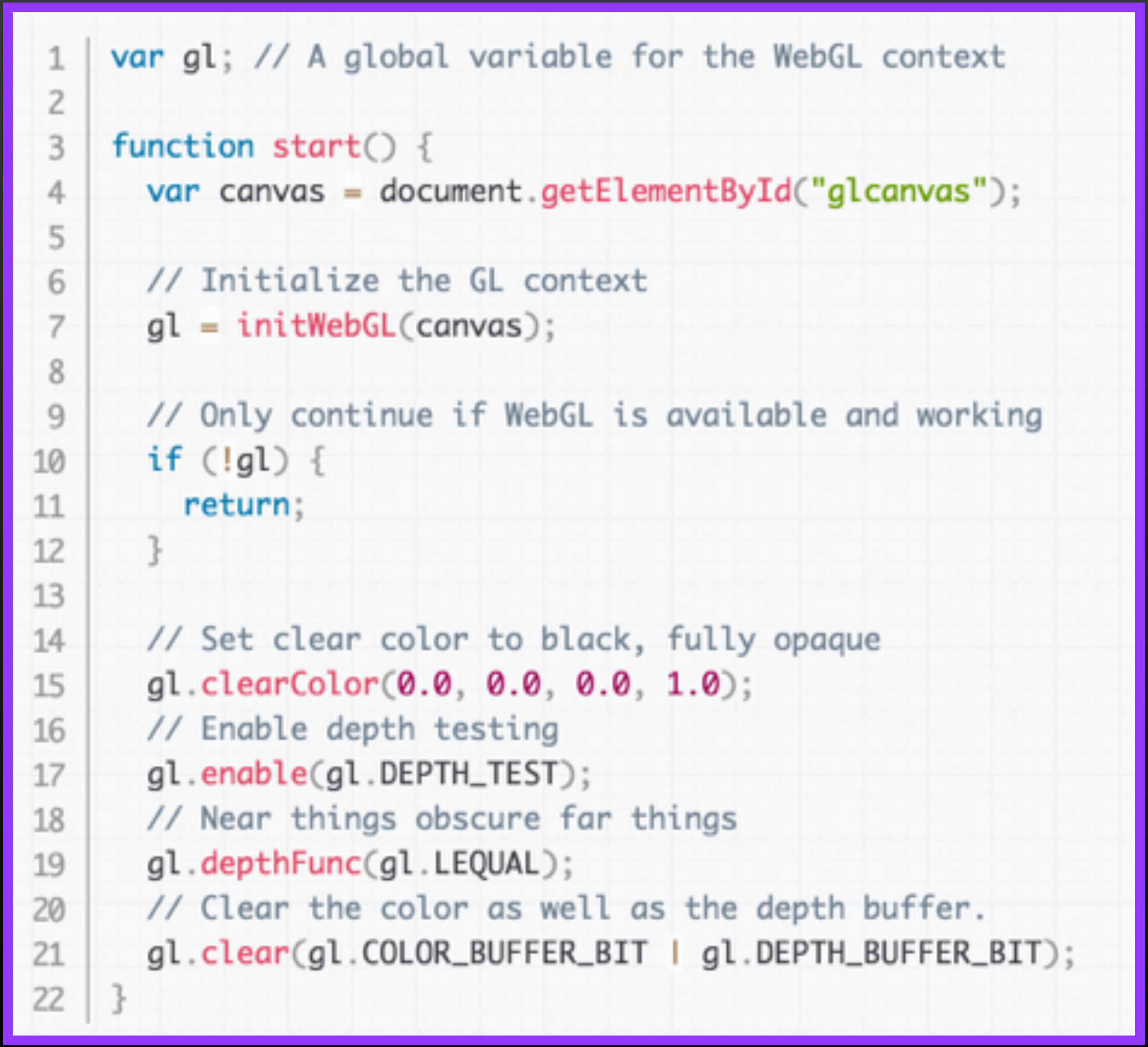
Safari Firefox

Implementation of WebGL

- Initialization
  - Create <canvas> tag
  - Establish reference



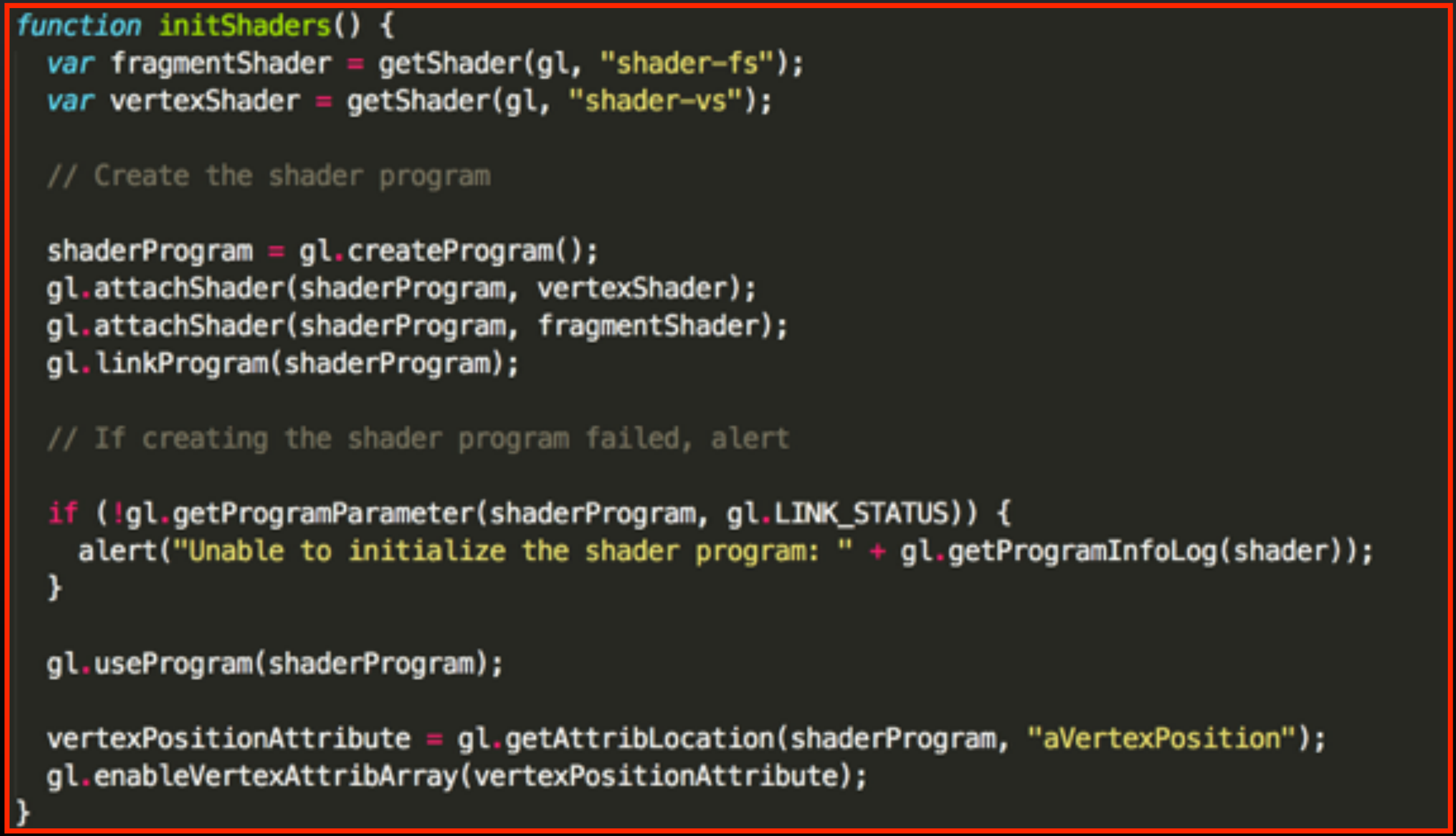
```html
<body onload="start()">
  <canvas id="glcanvas" width="640" height="480">
    Your browser doesn't appear to support the
    <code>&lt;canvas&gt;</code> element.
  </canvas>
</body>
```



```javascript
var gl; // A global variable for the WebGL context

function start() {
  var canvas = document.getElementById("glcanvas");

  // Initialize the GL context
  gl = initWebGL(canvas);

  // Only continue if WebGL is available and working
  if (!gl) {
    return;
  }

  // Set clear color to black, fully opaque
  gl.clearColor(0.0, 0.0, 0.0, 1.0);
  // Enable depth testing
  gl.enable(gl.DEPTH_TEST);
  // Near things obscure far things
  gl.depthFunc(gl.LEQUAL);
  // Clear the color as well as the depth buffer.
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
}
```

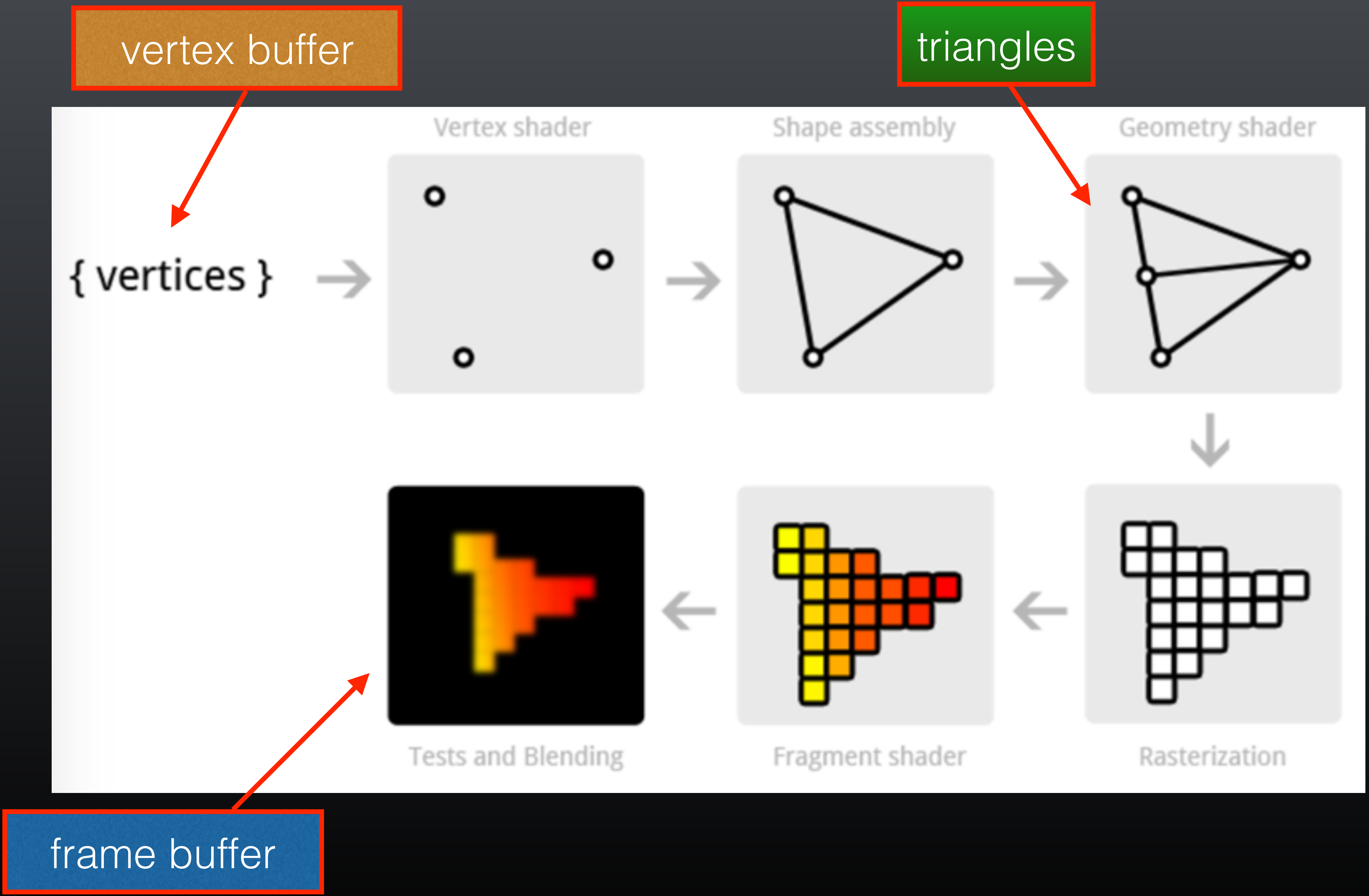
Initialization of Shaders

- Vertex shader
  - ✦ compute attributes of vertices: projected position, color, texture …
- fragment shader
  - ✦ fragment: individual pixel
  - ✦ compute attributes of fragment



```
function initShaders() {
  var fragmentShader = getShader(gl, "shader-fs");
  var vertexShader = getShader(gl, "shader-vs");

  // Create the shader program

  shaderProgram = gl.createProgram();
  gl.attachShader(shaderProgram, vertexShader);
  gl.attachShader(shaderProgram, fragmentShader);
  gl.linkProgram(shaderProgram);

  // If creating the shader program failed, alert

  if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
    alert("Unable to initialize the shader program: " + gl.getProgramInfoLog(shader));
  }

  gl.useProgram(shaderProgram);

  vertexPositionAttribute = gl.getAttribLocation(shaderProgram, "aVertexPosition");
  gl.enableVertexAttribArray(vertexPositionAttribute);
}
```

OpenGL rendering pipeline



Pros and Cons

- Pros
  - No plug-in
    - Compare to Unity3D, Silverlight
  - Flexible and Scalable
  - Easy to integrate
    - Access to full DOM element

- Cons
  - Largely dependent on graphics unit

Thanks !