

# Мультимедиа технология

Москин Николай Дмитриевич  
доцент, к.т.н., институт математики и  
информационных технологий  
Петрозаводский государственный университет

# § 5 Введение в WebGL

---

WebGL (Web-based Graphics Library) – это технология, позволяющая рисовать, отображать и взаимодействовать со сложной, интерактивной трехмерной компьютерной графикой в веб-браузерах.

WebGL, в сочетании с HTML5 и JavaScript, делает трехмерную графику доступной для веб-разработчиков и открывает возможность создания веб-приложений следующего поколения, с простыми и понятными пользовательскими интерфейсами и веб-контентом.

# История WebGL

---

Из технологий отображения трехмерной графики на персональных компьютерах наибольшее распространение получили Direct3D и OpenGL. Direct3D – составная часть пакета технологий Microsoft DirectX – это технология отображения трехмерной графики, предназначенная для использования на платформе Windows. Она является лицензионным программным интерфейсом (API) и контролируется корпорацией Microsoft. Альтернативная ей технология OpenGL получила широкое распространение, благодаря ее открытости.

# История WebGL

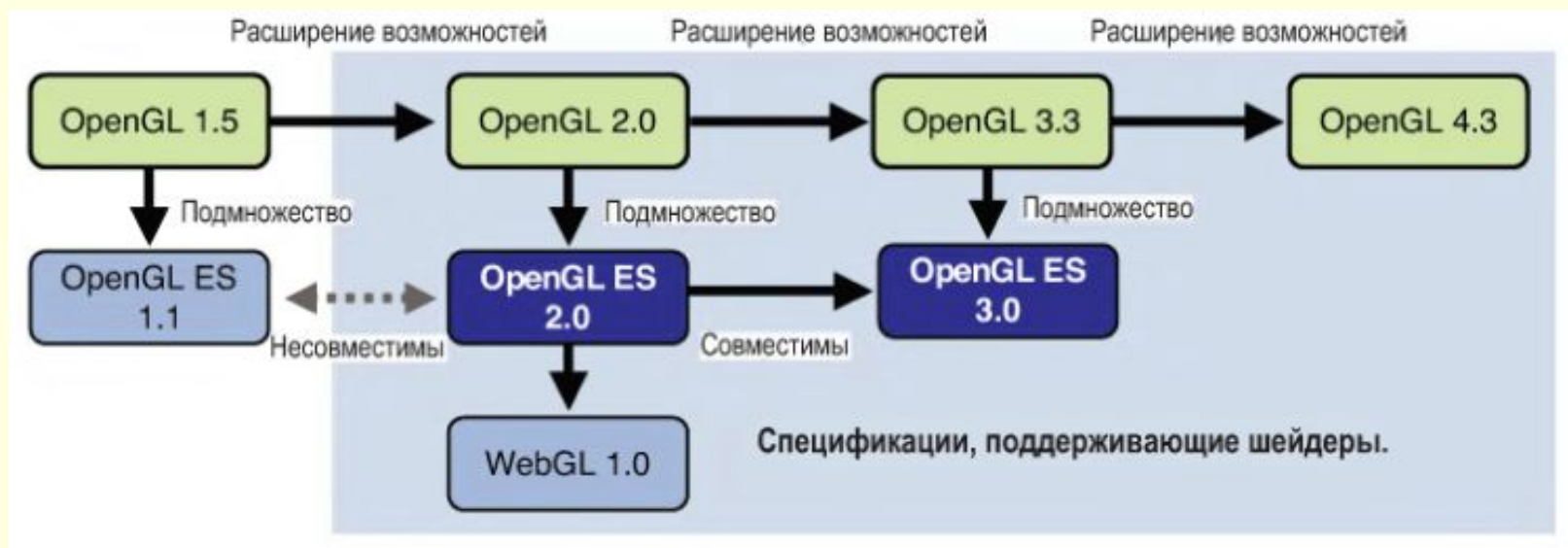
---

Несмотря на то, что технология WebGL корнями уходит в OpenGL, в действительности она является дальнейшим развитием версии OpenGL для встраиваемых систем, таких как смартфоны и игровые консоли.

Эта версия, известная как OpenGL ES (for Embedded Systems), создана в 2003-2004 годах, а затем обновлена в 2007 году (ES 2.0) и в 2012 (ES 3.0). WebGL основана на версии ES 2.0. Последняя версия WebGL 2.0 вышла 5 июля 2017 года.



# История WebGL



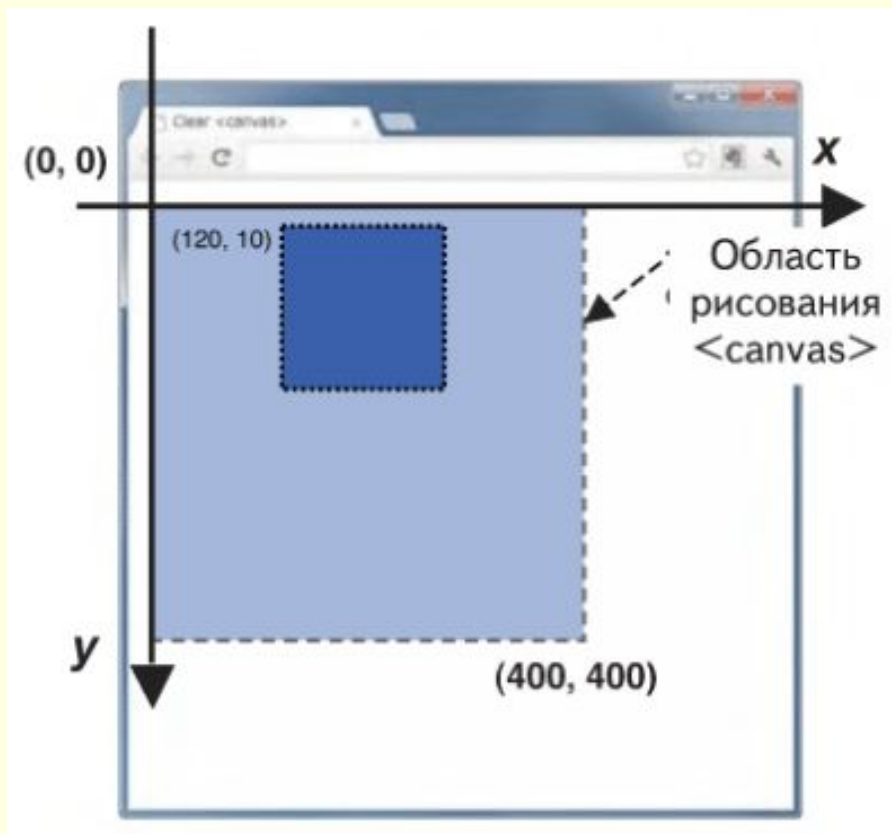
Шейдеры - это компьютерные программы, позволяющие создавать сложные визуальные эффекты с использованием специализированного языка программирования, похожего на язык С.

# Элемент `<canvas>`

---

Система WebGL использует для рисования элемент `<canvas>`, появившийся в HTML5 и определяющий область веб-страницы, предназначенную для рисования. Без использования технологии WebGL, элемент `<canvas>` позволяет рисовать только 2-мерную графику с применением функций JavaScript.

# Система координат элемента <canvas>



Начало системы координат элемента <canvas> находится в левом верхнем углу, ось X определяет координату по горизонтали (слева направо), ось Y – по вертикали (сверху вниз).

# DrawRectangle.htm

---

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Draw a blue rectangle (canvas version)</title>
  </head>
  <body onload="main()">
    <canvas id="example" width="400" height="400">
      Please use a browser that supports "canvas"
    </canvas>
    <script src="DrawRectangle.js"></script>
  </body>
</html>
```

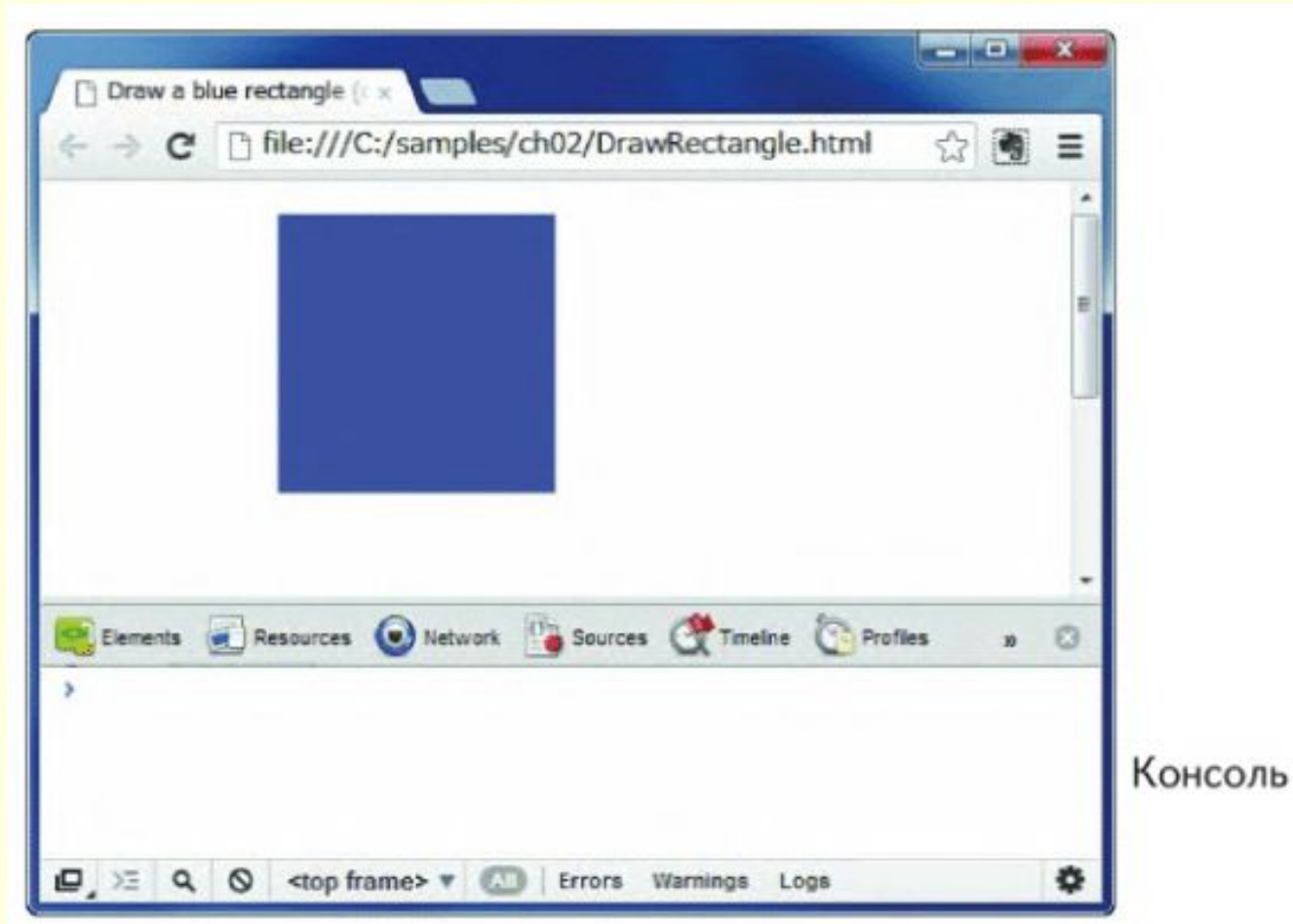


# DrawRectangle.js

---

```
function main() {  
  // получить ссылку на элемент <canvas>  
  var canvas = document.getElementById('example');  
  
  if(!canvas) {  
    console.log('Failed to retrieve the <canvas> element');  
    return; }  
  
  // получить двухмерный контекст отображения  
  var ctx = canvas.getContext('2d');  
  
  // нарисовать синий квадрат  
  ctx.fillStyle = 'rgba(0, 0, 255, 1.0)'; // выбрать синий цвет  
  ctx.fillRect(120, 10, 150, 150); // заполнить заливку квадрата  
}
```

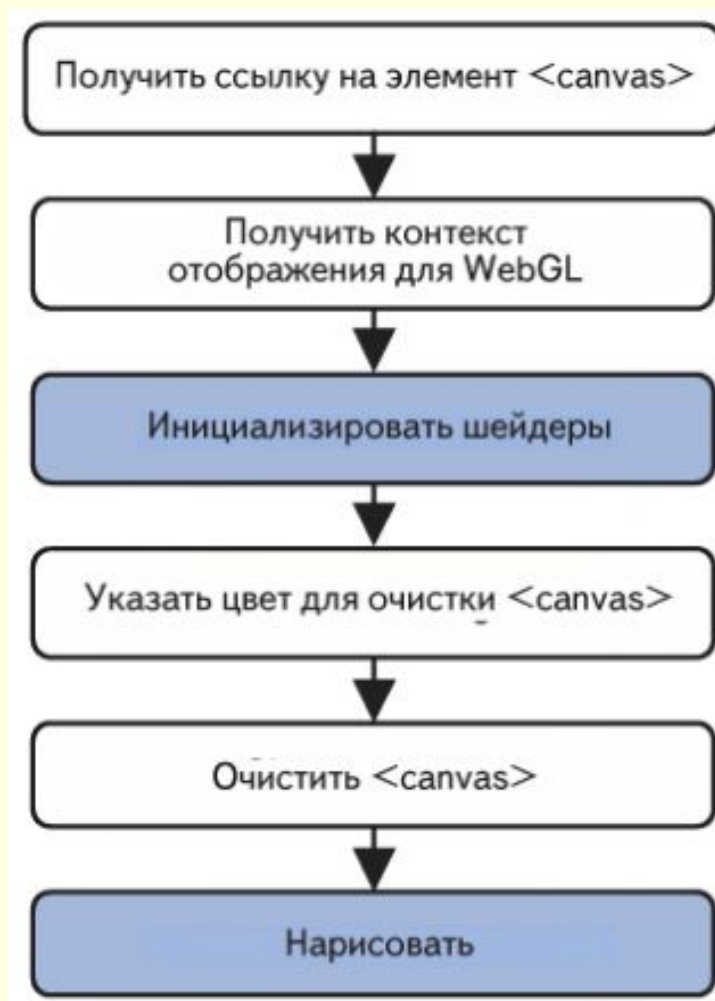
# Пример (1)



# Шейдеры

WebGL опирается на механизм рисования нового типа, который называется *шейдером* (shader), обладающий большей гибкостью и широтой возможностей при рисовании двух- и трехмерных объектов, и который должен использоваться всеми WebGL-приложениями.

На рисунке показана последовательность операций, выполняемых WebGL-программой.



# Вершинные и фрагментные шейдеры

---

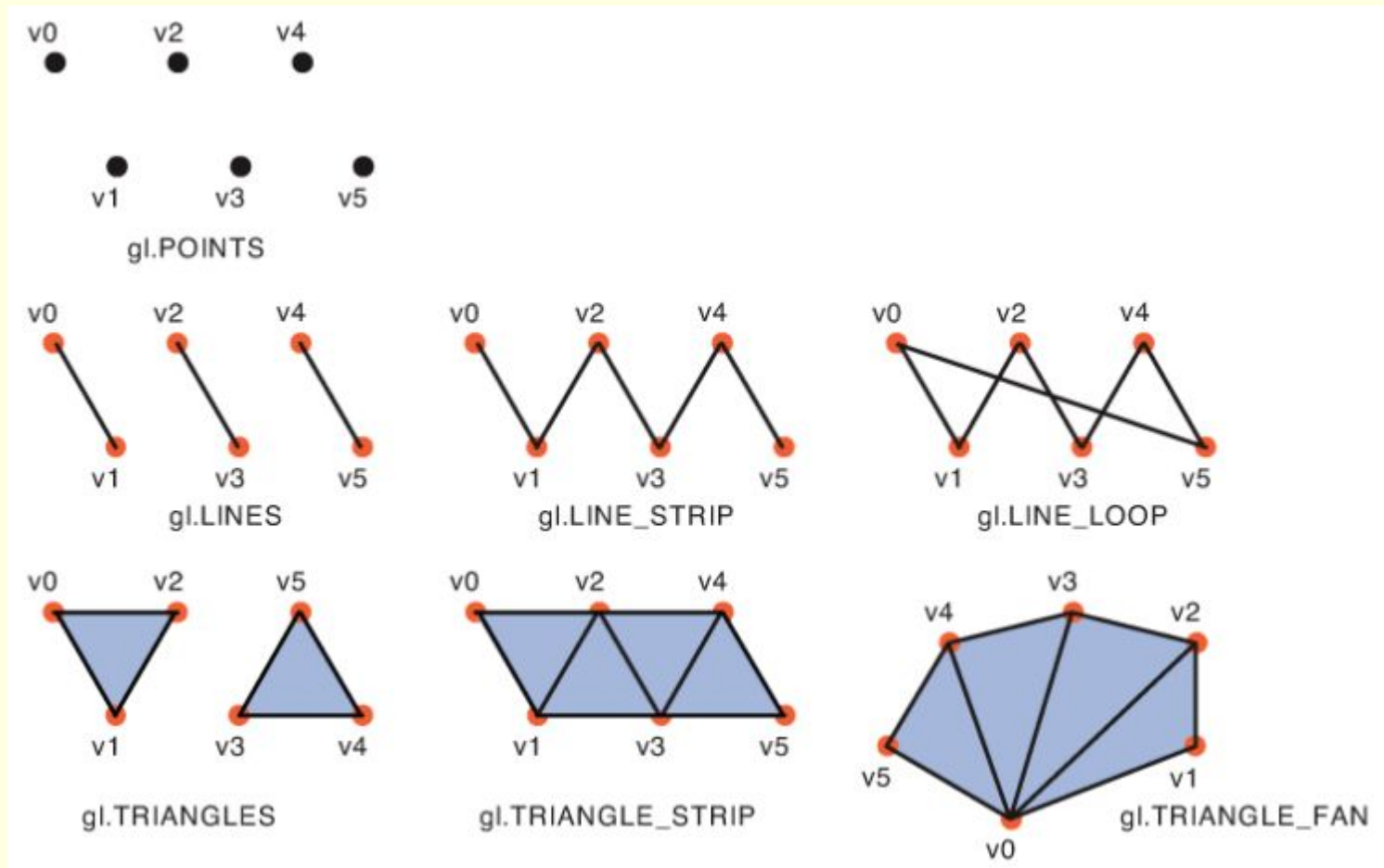
- *Вершинный шейдер* (vertex shader) – это программа, описывающая характеристики вершин (координаты, цвет и др.), а вершина – это точка в двух- или трехмерном пространстве, например, угол или вершина двух- или трехмерной фигуры;
- *Фрагментный шейдер* (fragment shader) – это программа, реализующая обработку фрагментов изображений, например, определение освещенности, где под фрагментом подразумевается простейший элемент изображения.

# Простые фигуры WebGL

---

- Точка (`gl.Points`) – группа точек. Точки рисуются в координатах вершин `v0`, `v1`, `v2...`
- Отрезок (`gl.LINES`) – группа отдельных отрезков, которые рисуются между парами вершин. Если число вершин нечетное, последняя вершина игнорируется.
- Ломаная (`gl.LINE_STRIP`) – группа связанных между собой отрезков между парами вершин `(v0, v1)`, `(v1, v2)`, `(v2, v3),...`
- Замкнутая ломаная (`gl.LINE_LOOP`) – группа связанных между собой отрезков. В отличие от `gl.LINE_STRIP` рисует отрезок, соединяющий последнюю и первую вершины.

# Простые фигуры WebGL



# Простые фигуры WebGL

---

- Треугольник (`gl.TRIANGLES`) – группа отдельных треугольников. Треугольники задаются триадами (`v0`, `v1`, `v2`), (`v3`, `v4`, `v5`),... Если числа вершин не кратно 3, лишние вершины игнорируются.
- Треугольники с общими сторонами (`gl.TRIANGLE_STRIP`). Первые три вершины образуют первый треугольник, а второй треугольник образуется из следующей вершины и двух предшествующих, входящих в состав первого треугольника.
- Треугольники с общей вершиной (`gl.TRIANGLE_FAN`). Первые три вершины образуют первый треугольник, а второй треугольник образуется из следующей вершины, одной стороны предыдущего треугольника и первой вершины.

# HelloTriangle.htm

---

```
<!DOCTYPE html>
<html lang="en">
  <head><meta charset="utf-8" />
    <title>Hello Triangle</title></head>
    <body onload="main()">
      <canvas id="webgl" width="400" height="400">
        Please use a browser that supports "canvas"</canvas>
      <script src="lib/webgl-utils.js"></script>
      <script src="lib/webgl-debug.js"></script>
      <script src="lib/cuon-utils.js"></script>
      <script src="HelloTriangle.js"></script>
    </body>
  </html>
```



# HelloTriangle.js (начало)

---

```
// Вершинный шейдер
var VSHADER_SOURCE =
    'attribute vec4 a_Position;\n' +
    'void main() {\n' +
    '    gl_Position = a_Position;\n' +
    '}\n';
```

```
// Фрагментный шейдер
var FSHADER_SOURCE =
    'void main() {\n' +
    '    gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);\n' +
    '}\n';
```

```
function main() {
    // Получить ссылку на элемент <canvas>
    var canvas = document.getElementById('webgl');
```

# HelloTriangle.js (продолжение)

---

```
// Получить контекст отображения для WebGL
var gl = getWebGLContext(canvas);
if (!gl) { console.log('Failed to get the rendering context for WebGL');
    return; }

// Инициализировать шейдеры
if (!initShaders(gl, VSHADER_SOURCE, FSHADER_SOURCE)) {
    console.log('Failed to initialize shaders.');
```

```
return; }

// Определить координаты вершин
var n = initVertexBuffers(gl);
if (n < 0) {
    console.log('Failed to set the positions of the vertices');
```

```
return; }

// Указать цвет для очистки области рисования <canvas>
gl.clearColor(0, 0, 0, 1);
```

# HelloTriangle.js (продолжение)

---

```
// Очистить <canvas>
```

```
gl.clear(gl.COLOR_BUFFER_BIT);
```

```
// Нарисовать треугольник
```

```
gl.drawArrays(gl.TRIANGLES, 0, n);
```

```
function initVertexBuffers(gl) {
```

```
    var vertices = new Float32Array([0, 0.5, -0.5, -0.5, 0.5, -0.5]);
```

```
    var n = 3; // число вершин
```

```
    // Создать буферный объект
```

```
    var vertexBuffer = gl.createBuffer();
```

```
    if (!vertexBuffer) {
```

```
        console.log('Failed to create the buffer object');
```

```
        return -1; }
```

# HelloTriangle.js (окончание)

---

```
// Определить тип буферного объекта
gl.bindBuffer(gl.ARRAY_BUFFER, vertexBuffer);
// Записать данные в буферный объект
gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

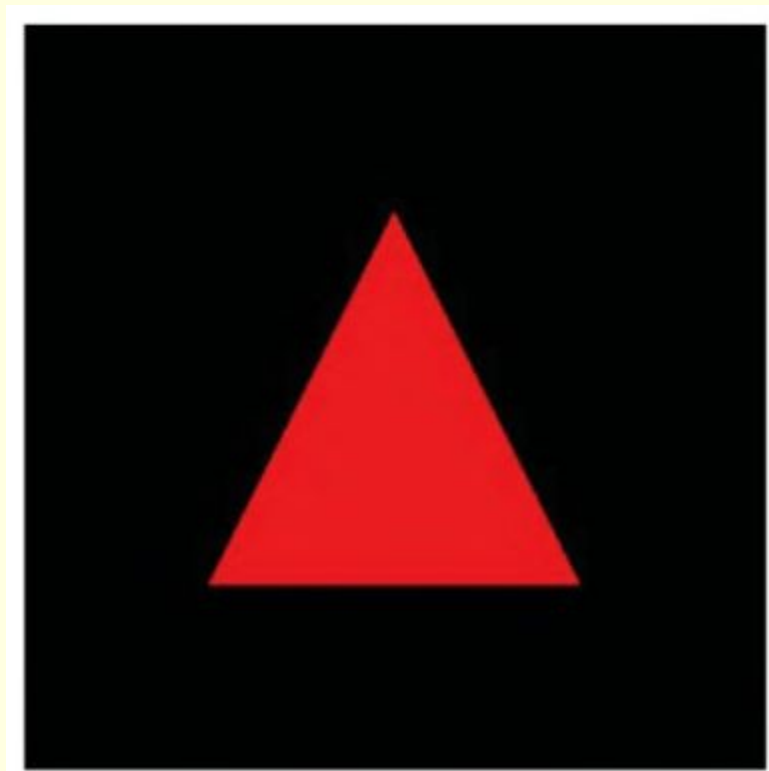
var a_Position = gl.getAttributeLocation(gl.program, 'a_Position');
if (a_Position < 0) {
    console.log('Failed to get the storage location of a_Position'); return -1; }
// Сохранить ссылку на буферный объект в переменной a_Position
gl.vertexAttribPointer(a_Position, 2, gl.FLOAT, false, 0, 0);

// Разрешить присваивание переменной a_Position
gl.enableVertexAttribArray(a_Position);

return n; }
```

# Пример (2)

---



# Литература

---

В презентации использовались примеры из книги:

- Коичи Мацуда, Роджер Ли WebGL: программирование трехмерной графики (Kouichi Matsuda, Rodger Lea WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL) / пер. с англ. Киселев А. Н. - М.: ДМК Пресс, 2015.