Пустой слайд для интриги

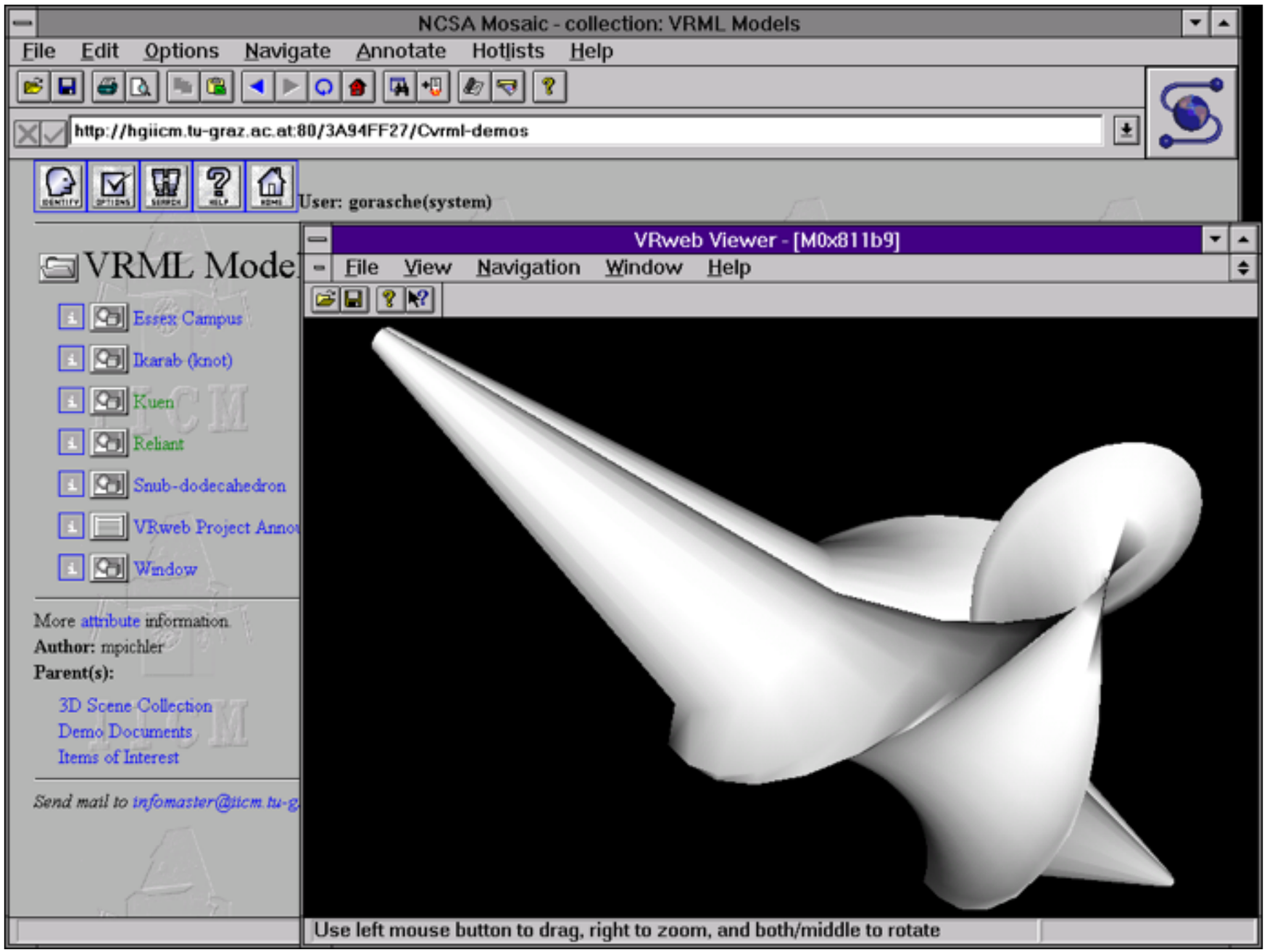
WebGL

Кирилл Дмитренко

- Минутка компьютерной археологии.
- Что такое WebGL?
- Hello World!
- Изучение.
- WebGL и шрикатон.

3D в вебе: VRML

1994 год



С тех пор

- Flash;
- Java;
- Unity3D;
- …

Canvas3D

- Mozilla: биндинги к OpenGL;

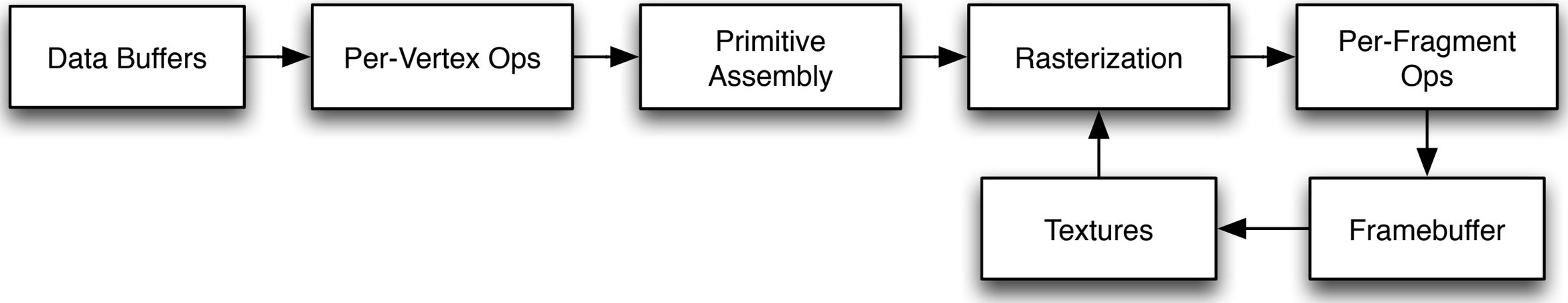
- Opera: какой-то ужас.

WebGL

- 2011 г. – стандарт;
- основан на OpenGL ES 2.0.

WebGL — API к GPU

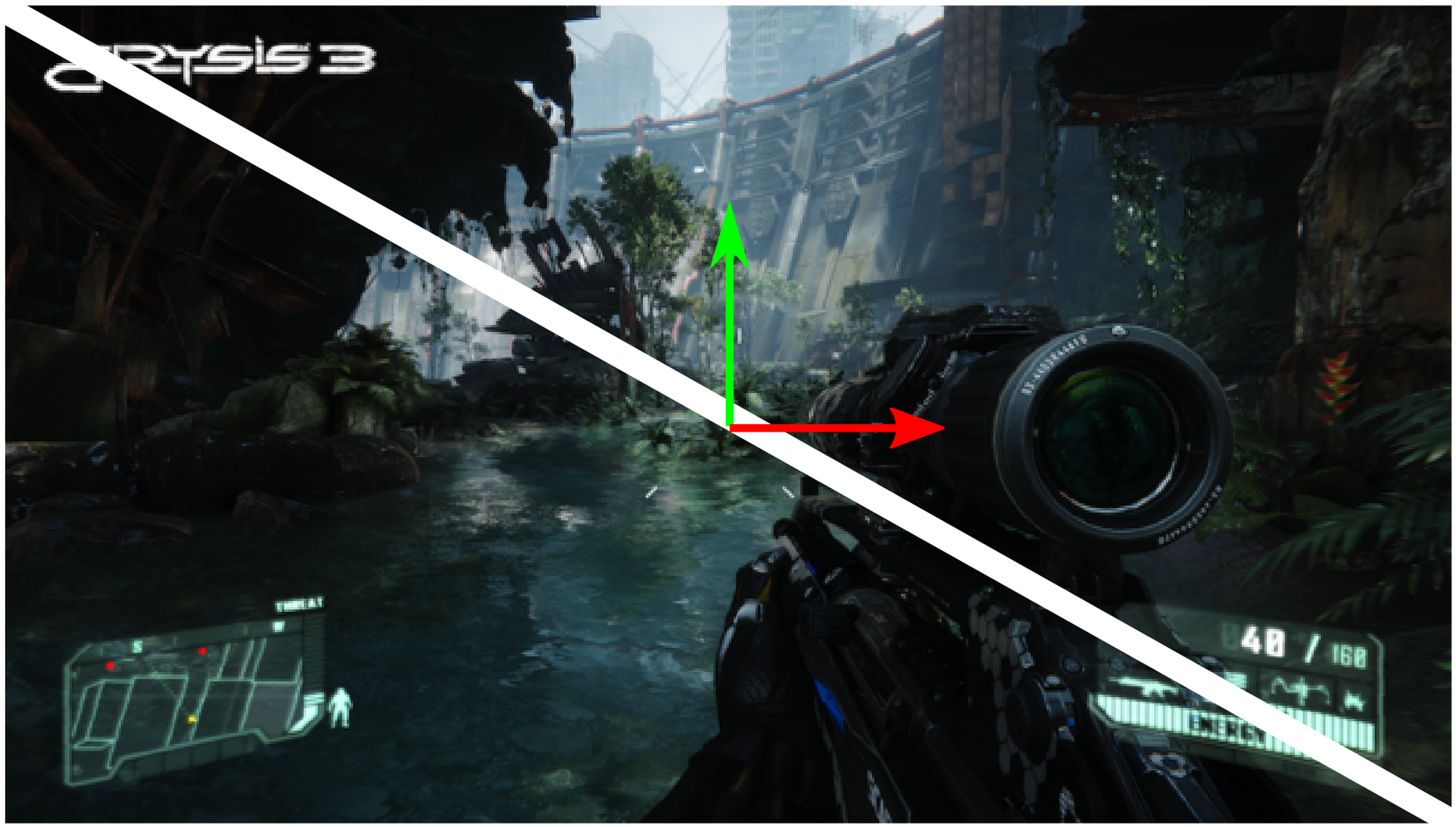
Pipeline


Data Buffers
Per-Vertex Ops
Primitive Assembly
Rasterization
Per-Fragment Ops
Textures
Framebuffer

Давайте порисуем

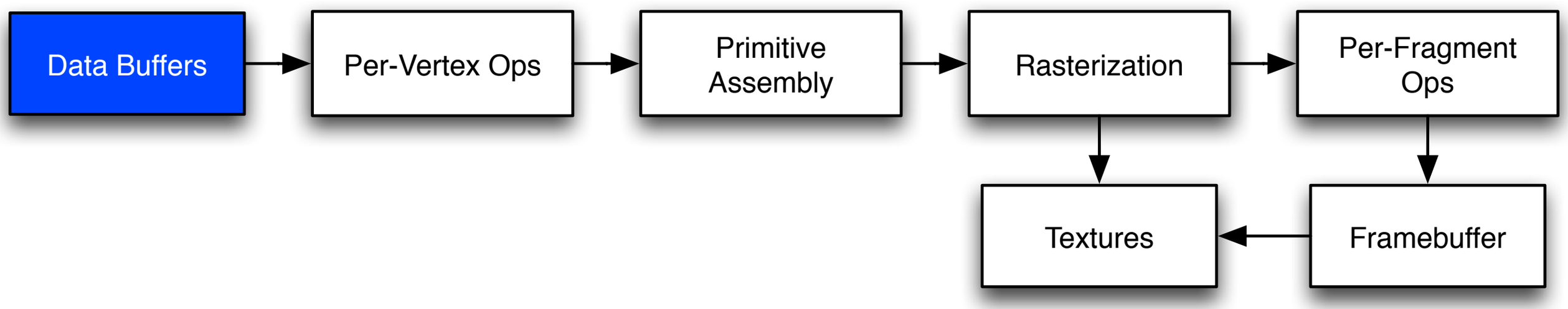
Давайте порисуем

(-1, 1) (1, 1)



(-1, -1) (1, -1)

Data Buffers



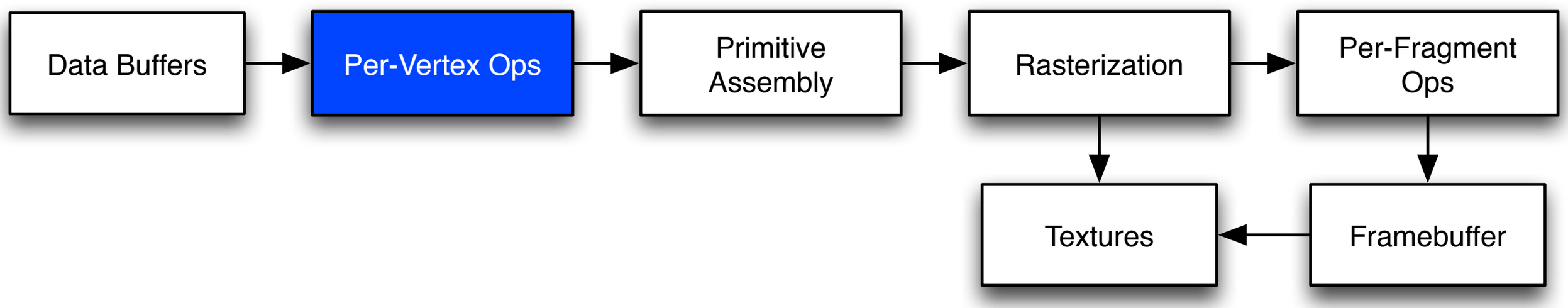
Data Buffers

- Данные (`ARRAY_BUFFER`);
- индексы (`ELEMENT_ARRAY_BUFFER`).

Data Buffers

```
// создаем буфер
var buffer = gl.createBuffer();
// биндим его к “цели”
gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
// кладем в него данные
gl.bufferData(
    gl.ARRAY_BUFFER,
    // typed array с данными
    new Float32Array([-1, -1, ..]),
    gl.STREAM_DRAW
);
```

Per-Vertex Ops



Vertex Shader

- Пишутся на C-подобном ЯП GLSL;
- входные данные - атрибуты вершин и юниформы;
- выходные данные:
  - координата вершины (`gl_Position`);
  - интерполируемые переменные (`varying`).

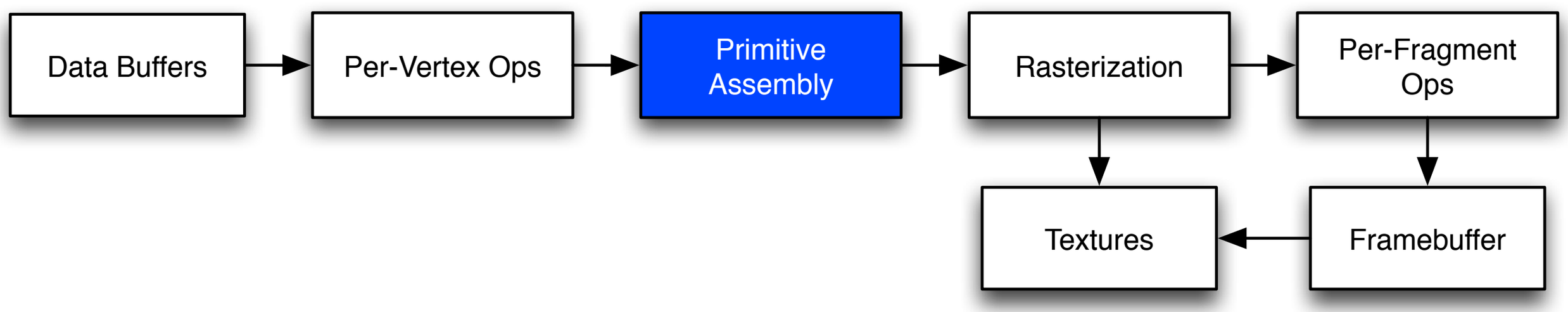
Vertex Shader

```
attribute vec2 vertexPosition;
attribute vec2 vertexTexCoord;

varying vec2 texCoord;

void main(void) {
    gl_Position =
        vec4(vertexPosition, 0, 1);
    texCoord = vertexTexCoord;
}
```

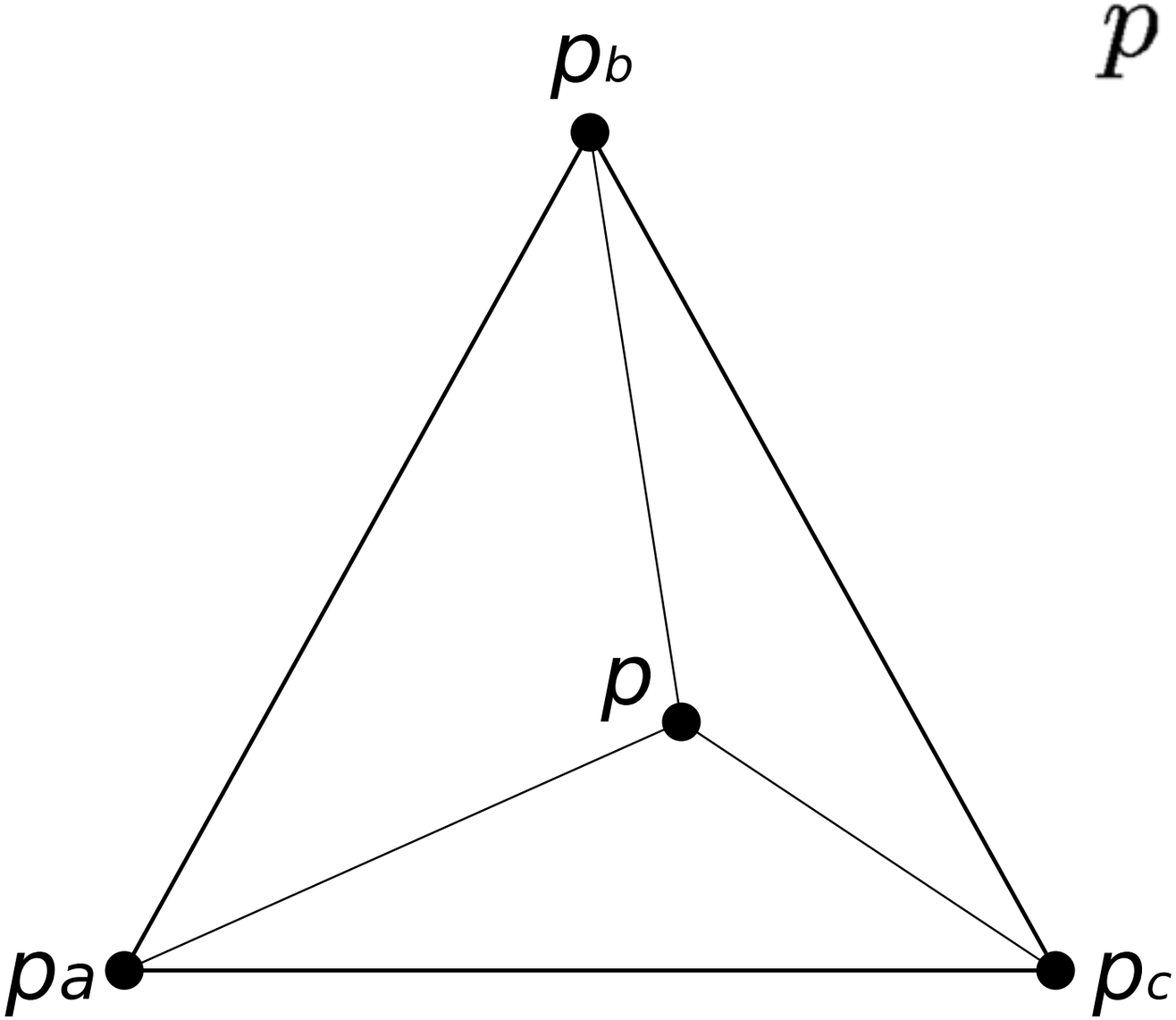
Vertex Shader

```
var vs =
    gl.createShader(gl.VERTEX_SHADER);
gl.shaderSource(
    vs,
    // строка к кодом шейдера
    VS_SOURCE
);
gl.compileShader(vs);
```

Primitive Assembly

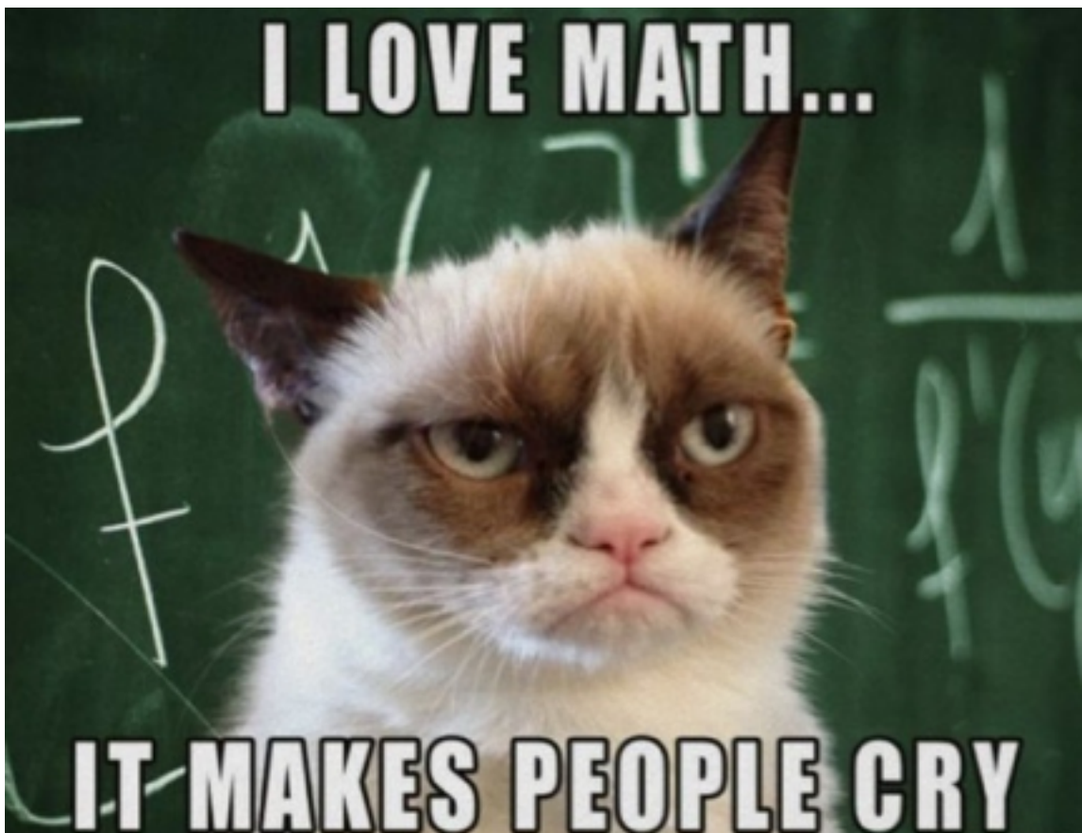


Интерполируемые переменные

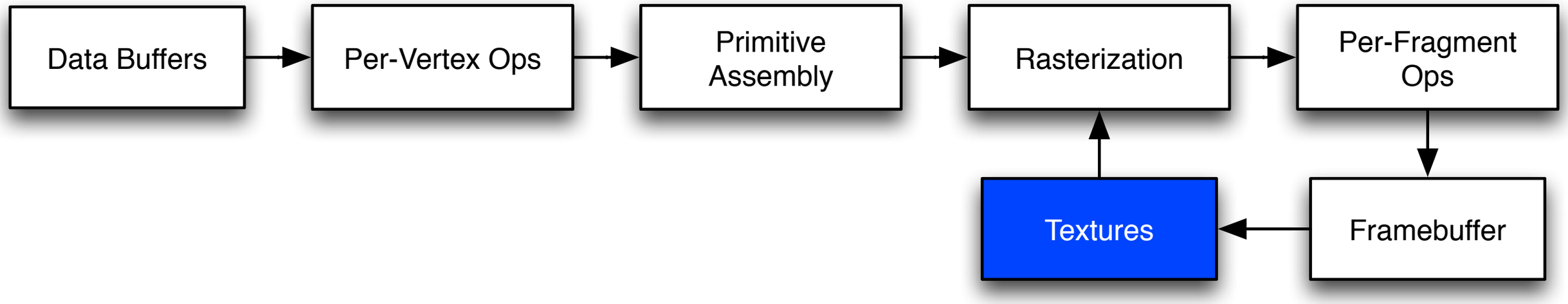


$$p = a p_a + b p_b + c p_c$$

$$a + b + c = 1$$



Textures


Data Buffers
Per-Vertex Ops
Primitive Assembly
Rasterization
Per-Fragment Ops
Textures
Framebuffer

Textures

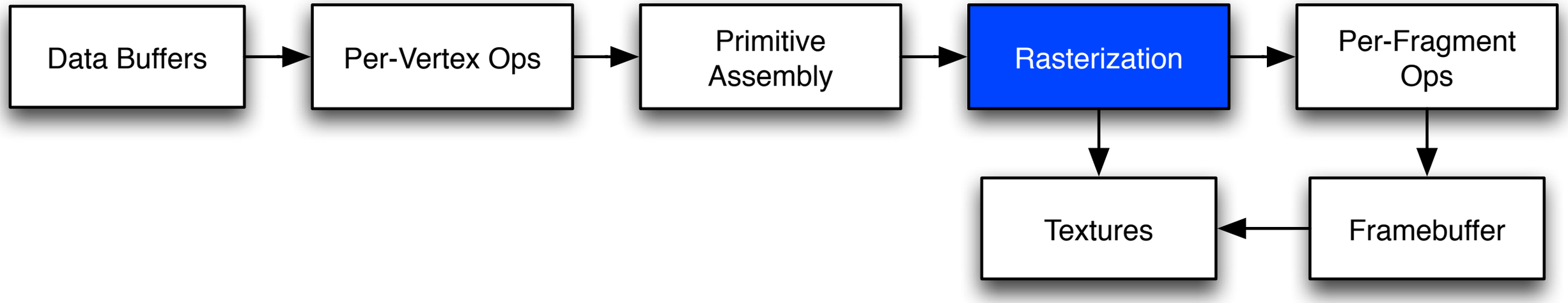
Textures

```
gl.activeTexture(gl.TEXTURE0);
var texture = gl.createTexture();
gl.bindTexture(gl.TEXTURE_2D, texture);
gl.texImage2D(
    gl.TEXTURE_2D,
    0,
    gl.RGBA,
    gl.RGBA,
    gl.UNSIGNED_BYTE,
    image
);
```

Textures

```
gl.texParameteri(gl.TEXTURE_2D,
    gl.TEXTURE_MIN_FILTER, gl.LINEAR);
gl.texParameteri(gl.TEXTURE_2D,
    gl.TEXTURE_MAG_FILTER, gl.LINEAR);

gl.texParameteri(gl.TEXTURE_2D,
    gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
gl.texParameteri(gl.TEXTURE_2D,
    gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
```

Rasterization



Fragment Shader

- Тоже GLSL;
- входные данные - интерполируемые переменные и юниформы;
- выходные данные - цвет.

Fragment Shader

```
precision mediump float;

varying vec2 texCoord;

uniform sampler2D texture;

void main(void) {
    gl_FragColor = texture2D(
        texture,
        texCoord
    );
}
```

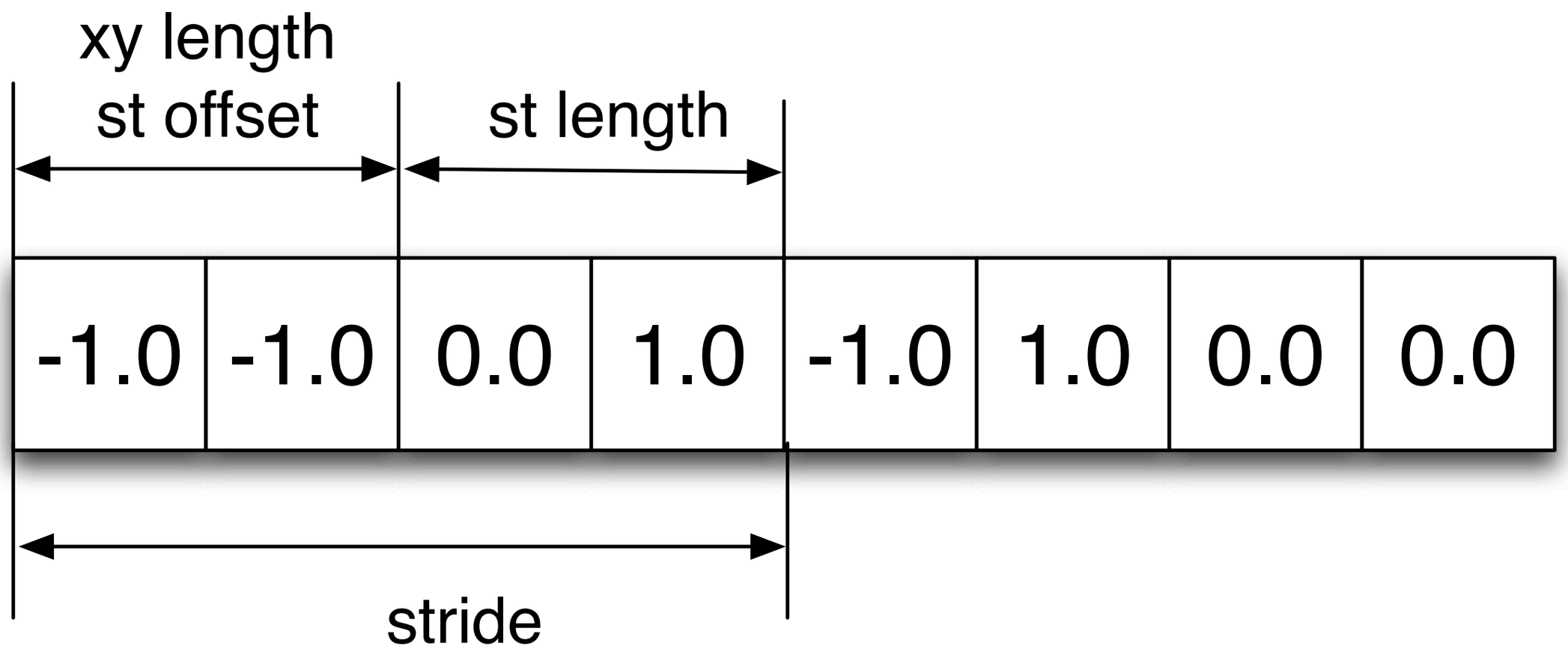
Fragment Shader

```
var fs =
    gl.createShader(gl.FRAGMENT_SHADER);

gl.shaderSource(fs, FS_SOURCE);
gl.compileShader(fs);
```

Shader Program

```
// создаем программу
var program = gl.createProgram();
// подключаем вершинный…
gl.attachShader(program, vs);
// … и фрагментный шейдеры
gl.attachShader(program, fs);
// “собираем”
gl.linkProgram(program);
// переключаемся на нее
gl.useProgram(program);
```

Данные в буфере



```
var vpAttr = gl.getAttribLocation(
    program,
    ‘vertexPosition’
);
gl.enableVertexAttribArray(vpAttr);
gl.vertexAttribPointer(
    vpAttr,
    2, // “размерность” атрибута
    gl.FLOAT, // тип
    false, // бесполезный параметр
    16, // stride
    0 // offset
);
```

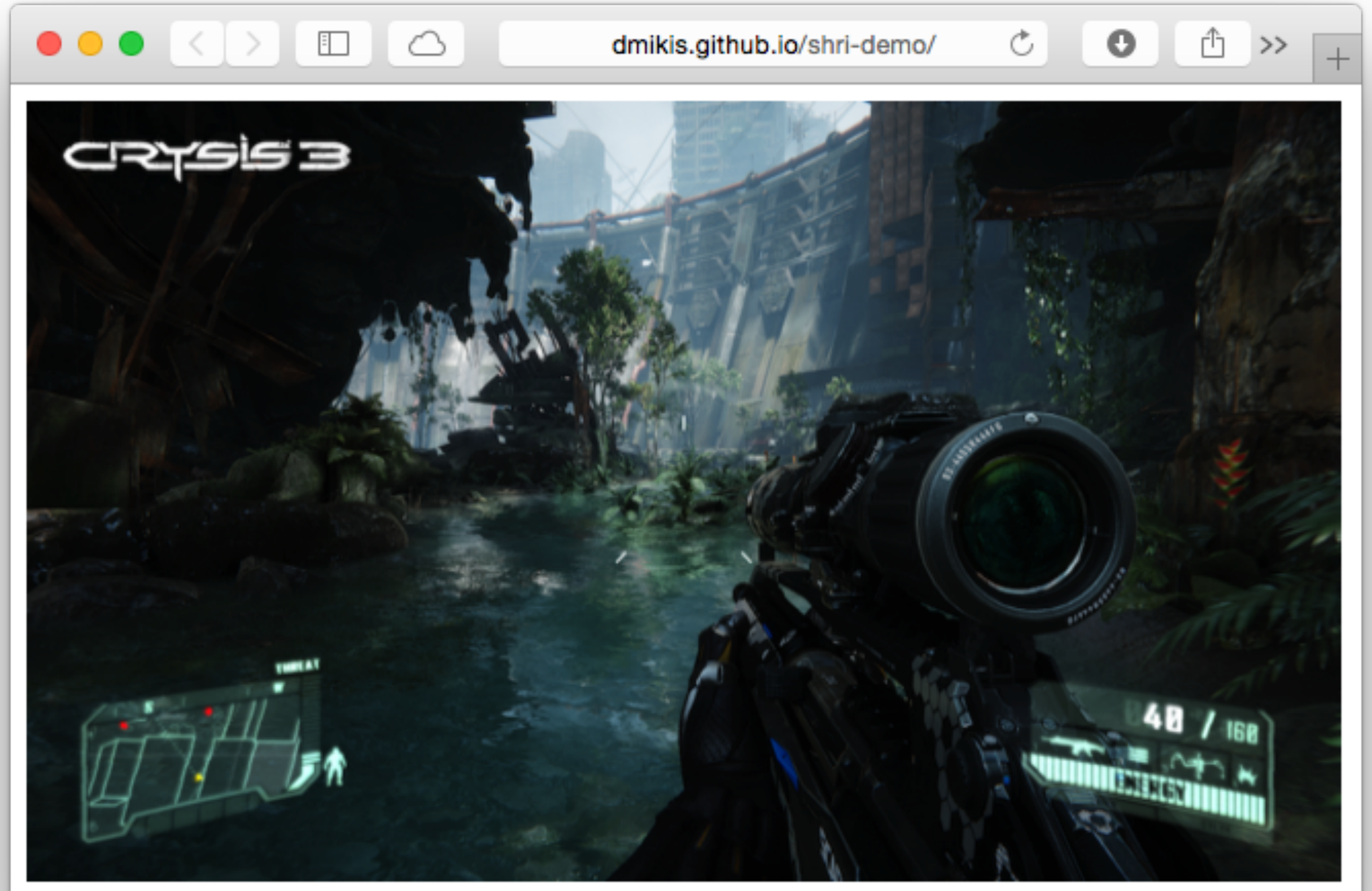
```
var vtAttr = gl.getAttribLocation(
    program,
    ‘vertexTexCoord’
);
gl.enableVertexAttribArray(vtAttr);
gl.vertexAttribPointer(
    vtAttr,
    2, // “размерность” атрибута
    gl.FLOAT, // тип
    false, // бесполезный параметр
    16, // stride
    8 // offset
);
```

```
var textureUniform =
    gl.getUniformLocation(
        program,
        'texture'
    );
gl.uniform1i(textureUniform, 0);
```

Я все это написал и ничего не произошло!!!

Draw call

```
gl.drawArrays(gl.TRIANGLES, 0, 6);
```


dmikis.github.io/shri-demo/
CRYSIS 3
40 / 160

Как-то все это низко

- WebGL оперирует буферами, текстурами, шейдерами и состоянием видеокарты.
- Мы же хотим объекты, материалы, источники света и камеры.

Библиотеки

- Графы сцены: Three.js, Babylon.js, etc;
- 2D и фильтры: Pixi.js, etc;
- Игровые движки: PlayCanvas.

Изучение

- Спецификации WebGL, GL ES 2.0 и GLSL :)
- WebGL Workshop
- Learning WebGL Lessons
- Книга Interactive Computer Graphics
- Coursera: WebGL
- http://bit.ly/1ntWqIA

WebGL для шрикатона

Фильтры и искажения для фоточек.

Контакты

Кирилл Дмитренко

Разработчик интерфейсов

dmikis@yandex-team.ru

@dmikis

dmikis