WebGL によるデータ可視化入門[*1]

開発ツールと DOM API

陰山 聡

神戸大学 システム情報学研究科 計算科学専攻

2013.05.21

[*1] 情報可視化論 X021（2013 年前期） LR301 →演習室

WebGL のコーディングスタイル

（この講義での）命名規則

- canvas.getContext() で取り出した WebGLRenderingContext は常に gl という変数に保存。
- gl はグローバル変数とする
- JavaScript もシェーダも camelCase
- シェーダの変数では、属性（attribute）には a のプリフィックス。 e.g., aVertexPosition
- シェーダの変数では、varying 変数には v のプリフィックス。 e.g., vColor
- シェーダの変数では、uniform には u のプリフィックス。 e.g., uMVMatrix

WebGL コードのデバッグ

JavaScript のコンソール

- さまざまなブラウザで「コンソール表示」
- JavaScript の console.log() でプリントされたテキスト
- エラーメッセージを見るのに便利

様々なデバッグツール

- Chrome デベロッパーツール（組み込み）
- Firebug （Firefox のエクステンション）
- Web Inspector

Chrome デベロッパーツール

ウェブアプリ開発用（WebGL 専用ではない）

【演習室の Chrome では未確認】

- ブラウザウィンドウの右上隅の設定アイコン
- ツール → デベロッパーツール
- 後述する Web Inspector と似ているので説明省略

WebGL Inspector

- WebGL 専用
- Chrome エクステンション
- インストールすると WebGL ページではアドレスバーの右に GL という赤文字が出る

WebGL Inspector: メニュー

- Trace
- Timeline
- State
- Textures
- Buffers
- Programs
- 右の方にある Frame controle ボタンで一時停止可能

WebGL Inspector: Trace

- （例題：林君のプログラム）
- ブラウザの右上、赤文字の GL の下にある Capture をクリック
- Redundant calls ボタンで無駄な重複呼び出しが分かる ←便利
- 矢印ボタンでステップ実行
- ソースコードをマウスでクリックするとその行の実行

WebGL Inspector: Timeline

- （例題：webgl_sample_spiral_05.html）
- Enable にすること
- 負荷の種類をグラフで表示

WebGL Inspector: State

- 様々な状態表示

WebGL Inspector: Textures

- テクスチャに関するデータ
- 画像そのものを見ることもできる
- テクスチャマップがあるコードの開発・デバッグに便利

WebGL Inspector: Buffers

- 左の Buffer n をクリック
- バッファに関する情報
- 数値そのものを確認することができる
- データ渡しのバグを見つけるのに便利

WebGL Inspector: Programs

- 左の Program n をクリック
- シェーダプログラムに関する情報
- ソースコード（頂点とフラグメント）
- リンク情報
- attribute と uniform の情報

演習

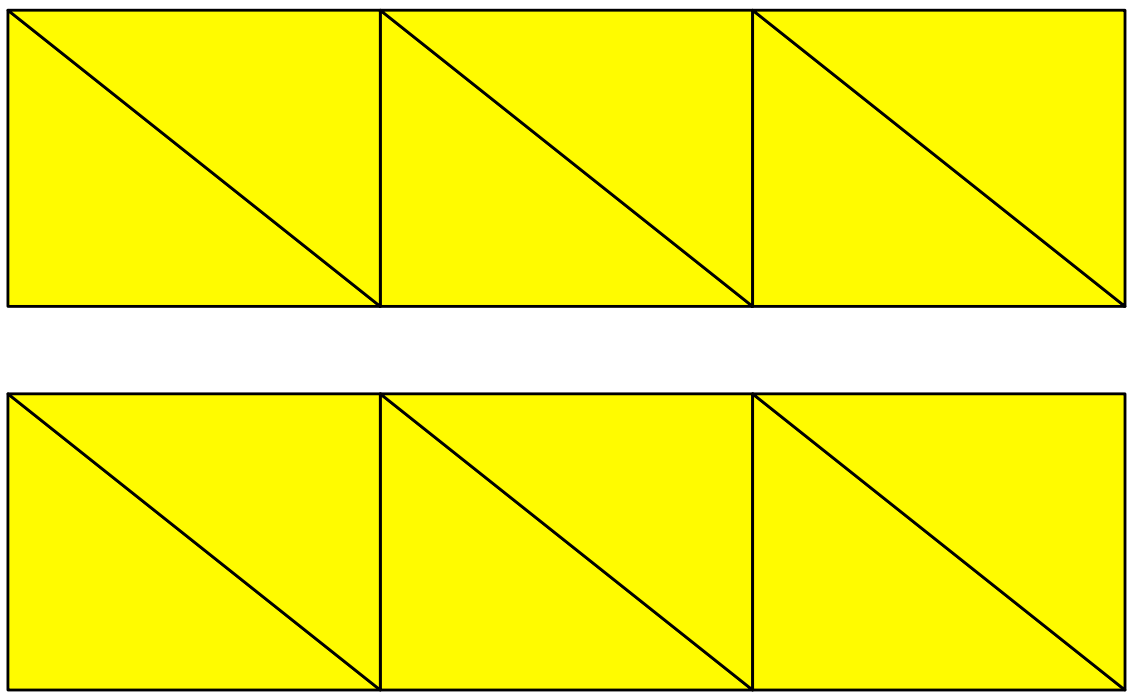
演習 01

- 先週の演習課題の続き。同じ図を描け。
- ただし、今日は WebGL Inspector 等のツールも使ってみること。

演習 02

（時間があれば）次の図を描け。

- 二つの長方形。
- 面は黄色。
- 三角形の黒線も描くこと（線と面を描く必要がある）。



DOM

document object

- オブジェクト指向については特に説明不要であろう。
- document object = web ページ
- オブジェクト名. メソッド名（引数...）
- サンプルコード： document_object.html

DOM とは

Document Object Model (DOM)

- HTML や XML 文書のためのプログラミング・インターフェース
  - プログラム（スクリプト言語）から（構造をもつ）文書にアクセスする
  - プログラム（スクリプト言語）から文書の構造、スタイル、内容を変更する
- 文書をオブジェクト指向的に扱う
- 文書＝ノード ＋ オブジェクト
- オブジェクト＝メソッド＋プロパティ
- スクリプト言語 $\Longleftrightarrow$ DOM $\Longleftrightarrow$ HTML 文書

URL: `http://www.w3.org/DOM/DOMTR`

DOMを使うための準備

特になし

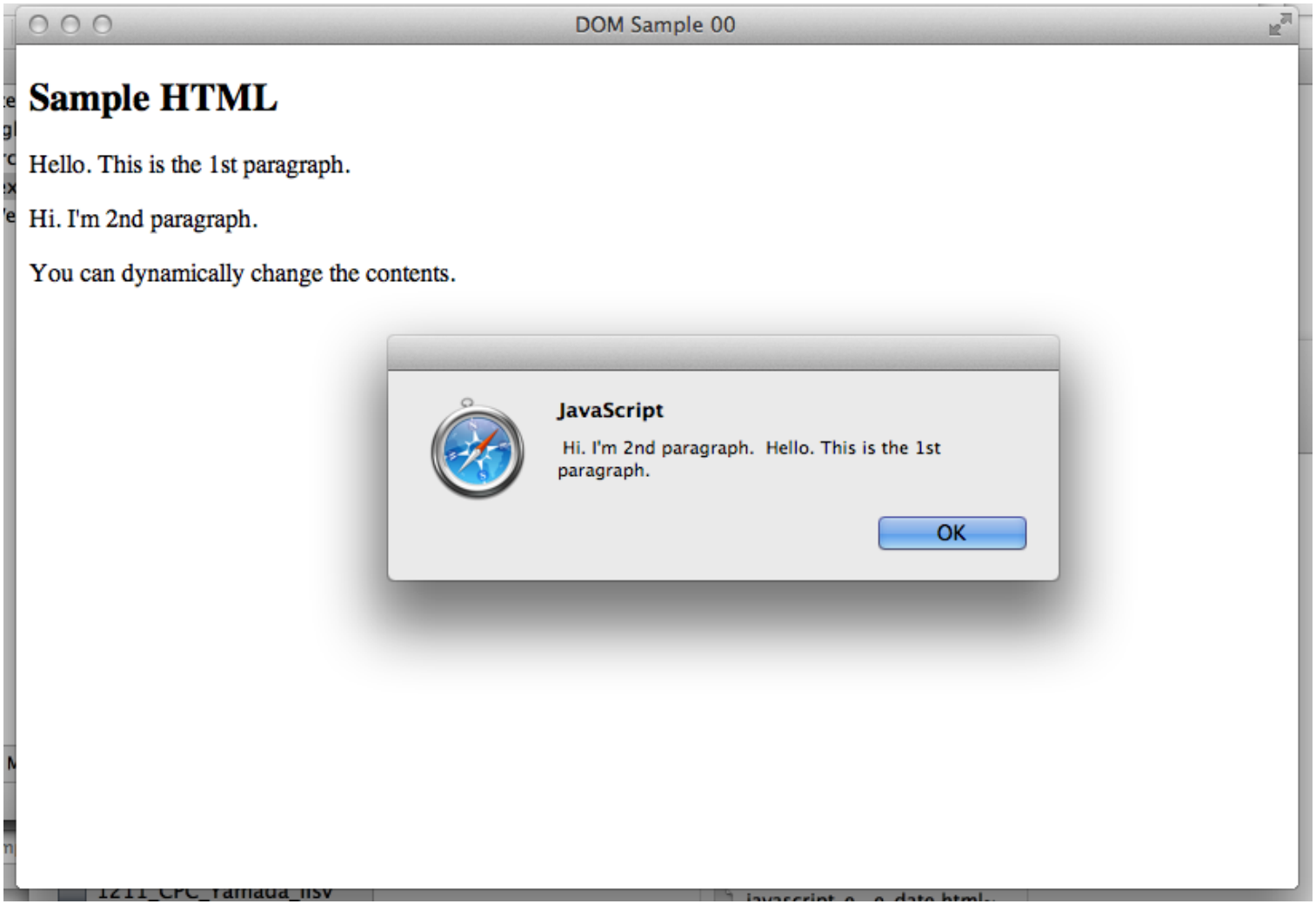
ブラウザに組み込まれている

DOMを使ったサンプルプログラム：dom_sample_00.html

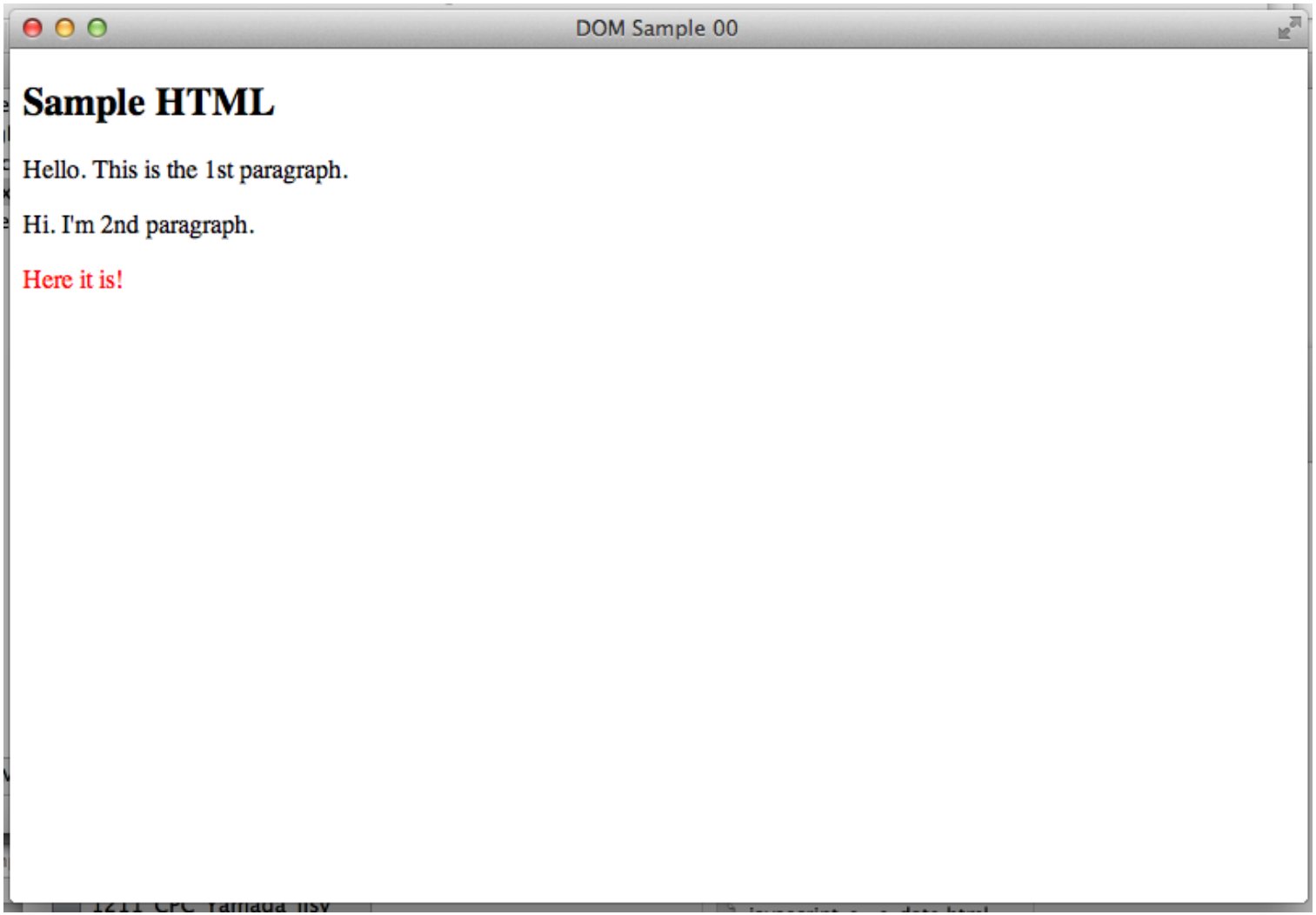
```
<!DOCTYPE HTML>
<html lang="en">
<head>
<title>DOM Sample 00</title>
<meta charset="utf-8">
<script type="text/javascript">
window.onload = function() {
   var paragraphs = document.getElementsByTagName("p");
   var p00 = paragraphs[0].textContent; // 1st paragraph
   var p01 = paragraphs[1].textContent; // 2nd paragraph
   alert(p01 + p00);
   document.getElementById("id000").style.color = "red";
   document.getElementById("id000").textContent = "Here it is!
       " ;
}
```

```
</head>

<body>

<h2> Sample HTML </h2>
<p> Hello. This is the 1st paragraph. </p>

<p> Hi. I'm 2nd paragraph. </p>

<div id="id000"> You can dynamically change the contents. </
    div>
</body>
</html>
```

読み込み結果


DOM Sample 00
Sample HTML
Hello. This is the 1st paragraph.
Hi. I'm 2nd paragraph.
You can dynamically change the contents.
JavaScript
Hi. I'm 2nd paragraph. Hello. This is the 1st paragraph.
OK

読み込み結果


DOM Sample 00
Sample HTML
Hello. This is the 1st paragraph.
Hi. I'm 2nd paragraph.
Here it is!

DOM を使ってシェーダソースコードをロードする

- これまでのサンプルプログラムでは、シェーダソースコードは JavaScript の文字列変数として直接書いていた。
- DOM API を使えばもっと読みやすくできる
- HTML の script タグとして書く：
  - 頂点シェーダ：
    ```
    <script id= ”shader − vs ” type= ”x − shader/x − vertex ”>
    ```
  - フラグメントシェーダ：
    ```
    <script id="shader-fs" type="x-shader/x-fragment">
    ```
- この二つの script type をブラウザは知らないので無視する。
- HTML のヘッダに書かずにファイルから読み出す方法もある。

webgl_sample_triangle_02_shader_from_DOM.html

```
<script id="shader-vs" type="x-shader/x-vertex">
  attribute vec3 aVertexPosition;

  void main() {
    gl_Position = vec4(aVertexPosition, 1.0);
  }
</script>

<script id="shader-fs" type="x-shader/x-fragment">
  precision mediump float;

  void main() {
    gl_FragColor = vec4(0.2, 0.4, 0.6, 1.0);
  }
</script>
```

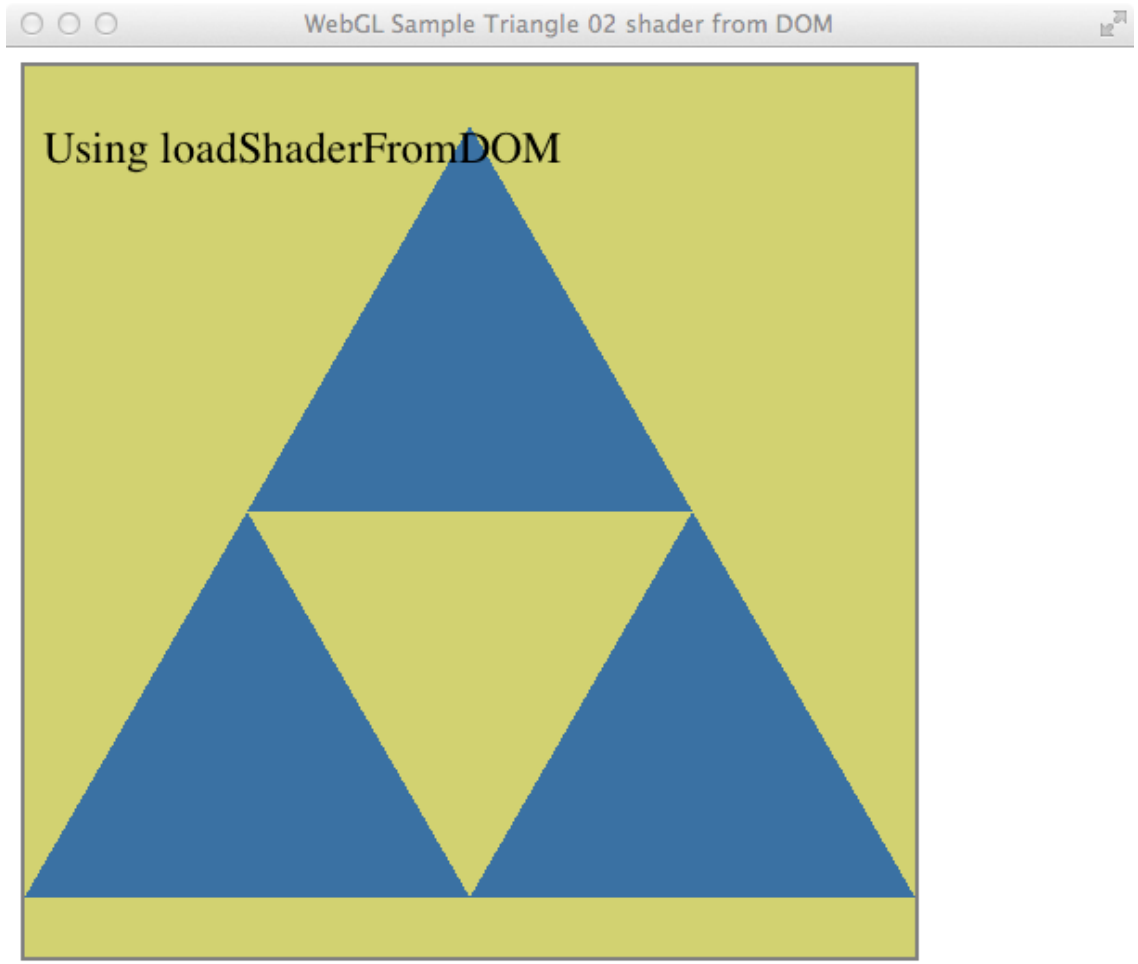
あとはDOMを使って読み込めばいい

```
function loadShaderFromDOM(id) {
  var shaderScript = document.getElementById(id);

  if (!shaderScript) {
    return null;
  }

  var shaderSource = "";
  var currentChild = shaderScript.firstChild;
  while (currentChild) {
    if (currentChild.nodeType == 3) { // 3 <= TEXT_NODE
      shaderSource += currentChild.textContent;
    }
    currentChild = currentChild.nextSibling;
  }
```

```
var shader;
if (shaderScript.type == "x-shader/x-fragment") {
  shader = gl.createShader(gl.FRAGMENT_SHADER);
} else if (shaderScript.type == "x-shader/x-vertex") {
  shader = gl.createShader(gl.VERTEX_SHADER);
} else {
  return null;
}

gl.shaderSource(shader, shaderSource);
gl.compileShader(shader);

if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
  alert(gl.getShaderInfoLog(shader));
```

```
    return null;
  }
  return shader;
}

function setupShaders() {

  var vertexShader = loadShaderFromDOM("shader-vs");
  var fragmentShader = loadShaderFromDOM("shader-fs");
```

実行結果


WebGL Sample Triangle 02 shader from DOM
Using loadShaderFromDOM

レポート課題

- 先週の演習課題（長方形）またはその（好きな）応用図を描け。
- ただしシェーダプログラムは DOM API を使ってロードすること。
- 提出はメールで。添付ファイルは２つ[*2]。
  1. レポートの PDF ファイル
  2. 作成した HTML ファイル
     - ファイル拡張子は html
     - このファイル名は、そのままウェブ公開するかもしれないので注意
- gmail アドレス：kageyama.lecture@...
- メールのタイトル： 学籍番号_氏名
- レポート（PDF ファイル形式に限る）には以下を記述すること
  - 学籍番号と氏名
  - どのような図形を描いたか
  - 描いた図形のキャプチャ図
  - 自分の HTML ファイルをウェブで公開された場合の希望名（フルネーム／名字のみ／イニシャル）
- 締め切り： 5/27（月）15:00

[*2] アーカイブはしないでください。

索引

References