



Webブラウザ上の3Dグラフィック技術
「WebGL」の現状と未来

伊藤 千光

自己紹介



伊藤千光 (いとう ちひろ)

<http://webos-goodies.jp/>

@webos_goodies

フリーランスのWebエンジニアです。

Agenda

1. WebGLの概要
2. 利用例
3. 開発環境
4. WebGLの将来
5. WebGLの課題



WebGLの概要



WebGLとは

- HTMLページの一部として3Dグラフィックを表示
 - canvasタグの内部に描画する
- OpenGL ES 2.0のJavaScript実装
 - プログラマブルシェーダーによる高度な表現
 - 言語仕様の違いやセキュリティへの配慮から細かい挙動が異なる
- Khronosグループが策定・管理
 - 議論は公開ML上で行われている



WebGLの現状

- 2011年3月にWebGL 1.0が正式リリース
 - セキュリティ問題が指摘され、修正（予定）
- デスクトップブラウザの対応状況
 - Google Chrome、Firefoxは標準で有効
 - Opera、Safariは設定変更が必要
 - 環境によっては動作しないこともある
- モバイルブラウザの対応状況
 - AndroidではFirefox, Operaが対応
 - iOSのSafariはiAdのみ対応



利用例

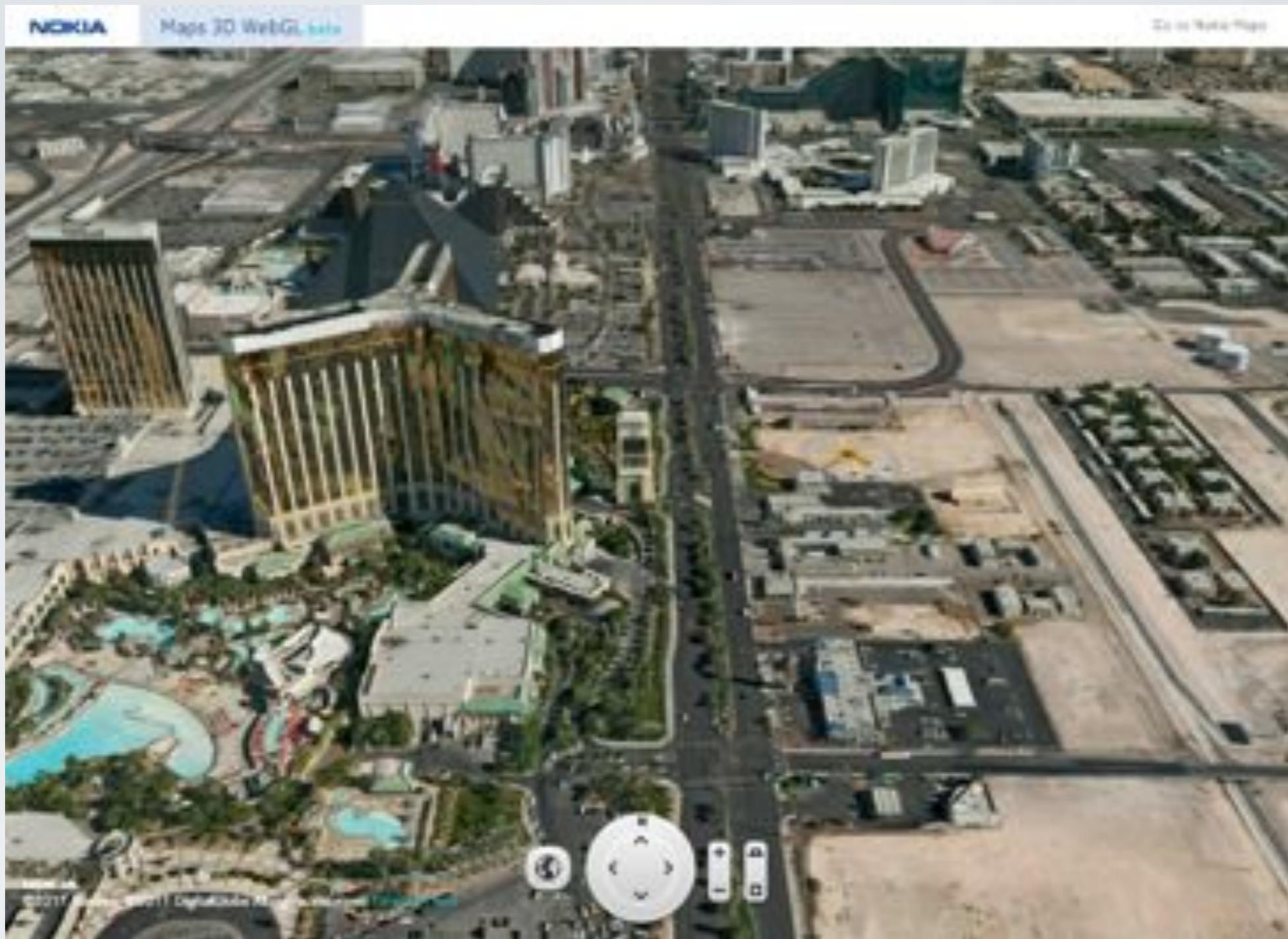


Demo : SKID Racer



<https://skid.gamagio.com>

Nokia Maps 3D



<http://maps3d.svc.nokia.com/webgl/>

3DTin



<http://www.3dtin.com/>

開發環境



WebGL開発の特徴

- 最低限Webブラウザとエディタがあれば開発できる
 - JavaScriptなのでコンパイル不要
 - WebブラウザにJSデバッガも用意されている
- 実行効率より開発効率が重視される傾向
 - 一般的にWeb開発のスパンは短い
 - フレームワークで開発負荷を軽減
- すべてのデータをネット経由で読む
 - ローカルディスクより桁違いに遅い



フレームワーク

- 多数のフレームワークが開発されている
 - CopperLicht
 - GLGE
 - PhiloGL
 - SceneJS
 - Three.js
 - etc...
- 最近ではThree.jsの採用が増えつつある



Three.jsの使用例

// Initialize

```
var scene = new THREE.Scene();
var renderer = new THREE.WebGLRenderer();
renderer.setSize(500, 500);
renderer.setClearColorHex(0x000000, 1);
document.body.appendChild(
  renderer.domElement);
```

// Add a camera

```
var camera = new
THREE.PerspectiveCamera(15, 500 / 500);
camera.position =
  new THREE.Vector3(0, 0, 8);
camera.lookAt(new THREE.Vector3(0, 0, 0));
scene.add(camera);
```

// Add a light

```
var light = new
THREE.DirectionalLight(0xcccccc);
light.position =
  new THREE.Vector3(0.577, 0.577, 0.577);
var ambient = new
THREE.AmbientLight(0x333333);
scene.add(light);
scene.add(ambient);
```

// Add an object

```
var geometry =
  new THREE.SphereGeometry(1, 32, 16);
var texture =
  THREE.ImageUtils.loadTexture(
    'images/earth.jpg')
var material =
  new THREE.MeshPhongMaterial({
    color: 0xffffffff, specular: 0xcccccc,
    shininess:50, ambient: 0xffffffff,
    map: texture,
    bumpMap:texture, bumpScale: 0.05 });
var mesh = new THREE.Mesh(
  geometry, material);
scene.add(mesh);
```

// Render the scene

```
var baseTime = +new Date;
function render() {
  requestAnimationFrame(render);
  mesh.rotation.y =
    0.3 * (+new Date - baseTime) / 1000;
  renderer.render(scene, camera);
};
render();
```

実行結果



Chrome Developer Tools

The image shows a Chrome browser window displaying a WebGL application titled "100,000 Stars" at the URL `workshop.chromeexperiments.com/stars/`. The application features a 3D starfield with a grid floor and a play button labeled "Take a tour." The Chrome Developer Tools interface is open, showing the following components:

- Top Panel:** Elements, Resources, Network, Sources, Timeline, Profiles, Audits, Console, PageSpeed.
- Left Panel:** Events, Frames (selected), Memory.
- Right Panel:** Performance timeline showing a 60 FPS frame rate. Two records are highlighted with durations of 16.289 ms and 16.308 ms.
- Records List:**
 - Recalculate Style
 - Animation Frame Fired (animation)
 - Recalculate Style (program).2
 - Composite Layers
 - Recalculate Style
 - Recalculate Style
 - Animation Frame Fired (animation)
 - Recalculate Style (program).2
 - Composite Layers
- Bottom Panel:** Legend for Loading, Scripting, Rendering, and Painting. Status: "2 of 65 frames shown (avg: 16.298 ms, σ: 0.009 ms)".

WebGL Inspector

The screenshot displays the WebGL Inspector interface. At the top, a browser window shows the URL `workshop.chromeexperiments.com/stars/`. The main view is a 3D scene of a star field with a grid overlay. A 'Take a tour.' button is visible in the top left, and 'Capture' and 'UI' buttons are in the top right.

The bottom panel shows the 'Textures' tab with the following data:

Texture Name	Dimensions	Texture Parameters
lensflare2	512 x 512	TEXTURE_WRAP_S: REPEAT, TEXTURE_WRAP_T: REPEAT
a_px	1 x 1	TEXTURE_MIN_FILTER: LINEAR_MIPMAP_LINEAR, TEXTURE_MAG_FILTER: LINEAR
sun_surface	1024 x 1024	
star_colorshift	512 x 1	
solarflare	512 x 256	
sun_halo		

The 'History' section shows the following log entry:

```
texImage2D(TEXTURE_2D, 0, RGBA, RGBA, UNSIGNED_BYTE, [object HTMLImageElement])
```

Source: http://workshop.chromeexperiments.com/stars/images/sun_surface.png

A small thumbnail of the sun surface texture is shown below the log entry. On the right side of the bottom panel, there is a 'Frame Control' section with 'Normal', 'Stepped', and 'Paused' options, and a '100% | FB' zoom indicator next to a larger view of the sun surface texture.

WebGLのこれから

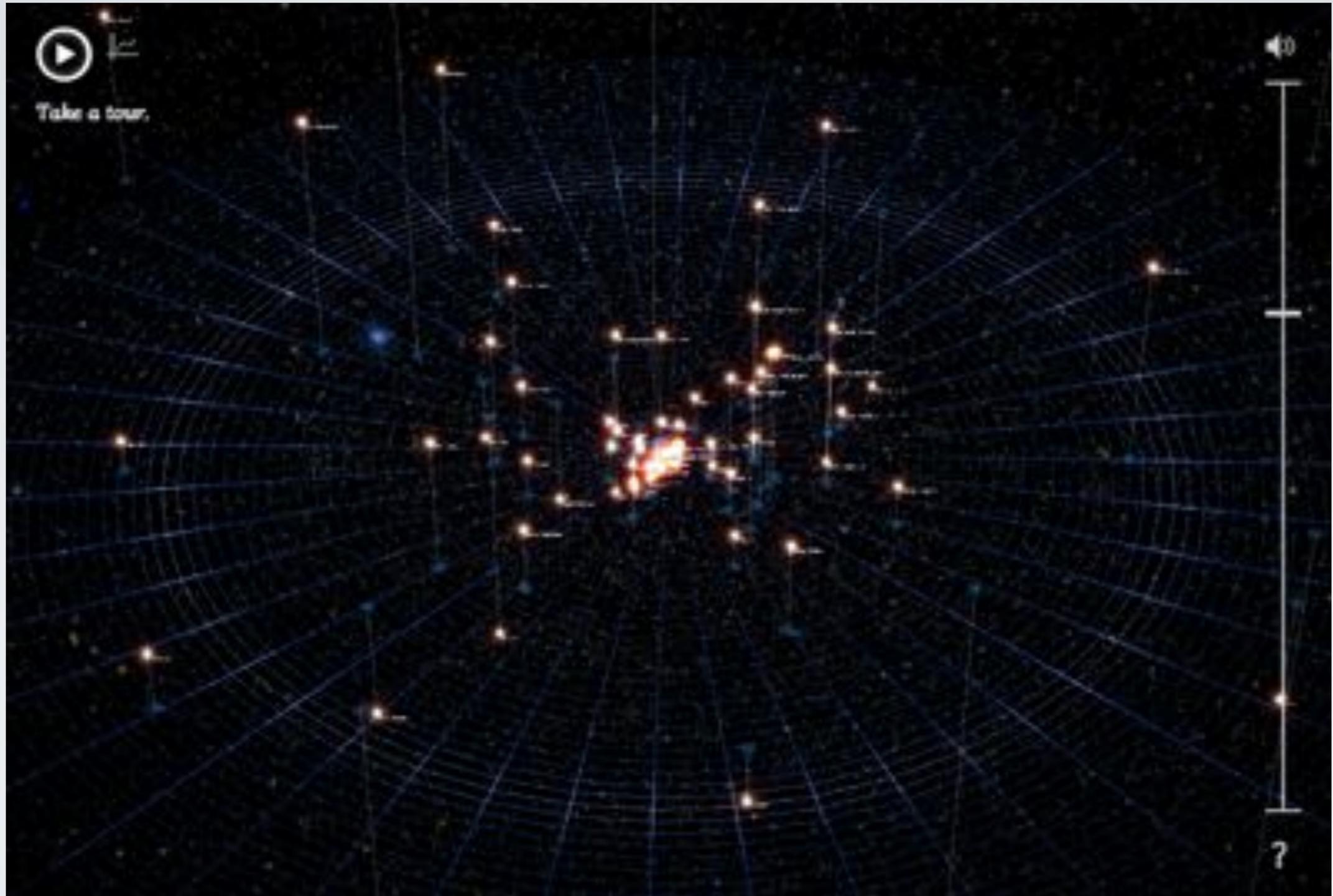


WebGLの使い方の変化

- デスクトップの機能をWebにもたらす
 - ゲーム
 - ビジュアルデモ
 - 各種ツール などなど
- 徐々にWebを活用したものが増えてきている
 - HTMLコンテンツとの融合
 - HTML5の機能の活用
 - Webコンテンツの取り込み

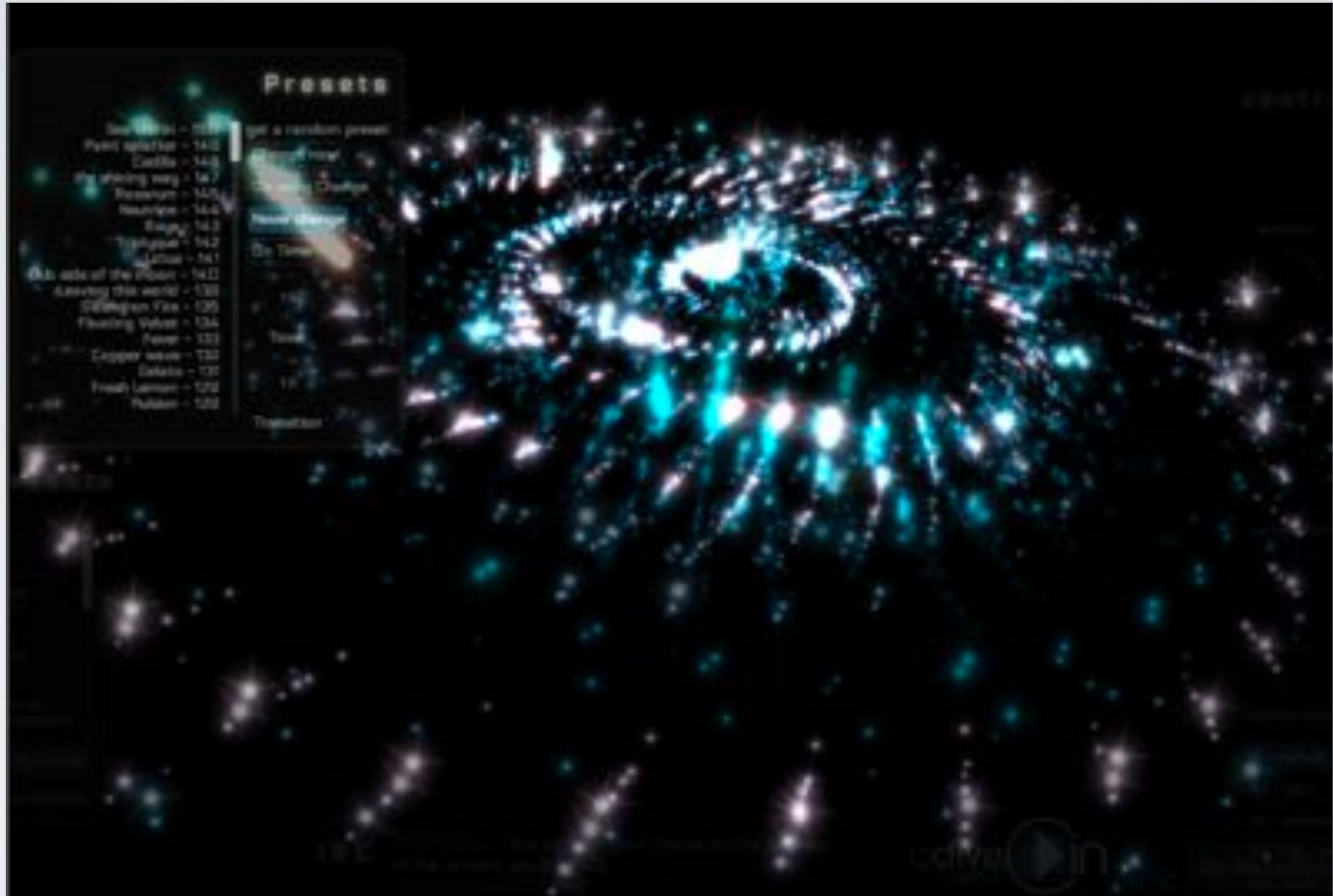


WebGL + CSS 3D Transform



<http://workshop.chromeexperiments.com/stars/>

WebGL + Audio API



<http://do.a dive.in/music/>

WebGL + getUserMedia



<http://mrdoob.com/lab/javascript/webcam/displace/>

WebGL + Geolocation API



<http://www.clicktorelease.com/code/streetViewReflectionMapping/>

まだまだ多くの可能性

- CSS Shaders
- Web Socket
- WebRTC
- DeviceOrientation Event
- Web上のあらゆるリソースを活用



WebGLの課題



セキュリティ問題

- DoS攻撃の可能性
 - GL_ARB(EXT)_robustnessで対処
- クロスオリジンリソースアクセス
 - タイミング攻撃による画像の同定
 - CORSによるアクセス制御で対処
- 詳細はWhite paper参照
<http://www.khronos.org/webgl/security/>



ブラウザサポート

- Internet Explorerはサポート予定なし
 - 「WebGLは有害な仕様」という見解
 - 無数のデバイスドライバが存在するWindowsではたしかに難しい面がある
- モバイルでの普及
 - 多くのデバイスがGL_EXT_robustness未サポート
 - 消費電力も問題



ワークフローの確立

- ツールセット
 - デスクトップ向けのを流用している状態
 - Webコンテンツとの連携は考慮されていない
- アセットの配信は手探りの状態
 - 標準フォーマットの確立が望まれる
 - WebGLTF が有望？
- 人材の教育も必要
 - Webと3DCGの双方の知識を持つ
人材の育成



参考資料

- WebGL
 - <http://www.khronos.org/webgl/>
 - <http://learningwebgl.com/blog/>
 - <https://sites.google.com/site/hackthewebgl/learning-webglhon-yaku/the-lessons>
- Three.js
 - <https://github.com/mrdoob/three.js/>
 - <http://www.atmarkit.co.jp/fwcr/design/benkyo/webgraphics05/01.html>
- Demos & Samples
 - <http://www.webgl.com/>
 - <http://playwebgl.com/>
 - <http://www.chromeexperiments.com/>
 - <https://developer.mozilla.org/ja/demos/>



ご清聴ありがとうございました！



Webにおけるグラフィックス表現の変遷



HTML4時代

- 基本は静止画像のみ
- サーバーサイドでグラフ画像などを生成していた
- 標準化されたグラフィックAPIの不在(*)
 - DOM APIを使用したハック的な手法
 - 非標準なプラグイン (Flashなど)
- Ajax技術の発展により、限界が露呈
 - HTML5の策定へ



* SVGは存在していましたが、普及していませんでした

HTML5時代

- 標準化された2DグラフィックAPIの普及
 - SVG (DOMベースのベクトルグラフィックAPI)
 - Canvas (JSベースのピクセルグラフィックAPI)
- すべてを標準技術で賄う (プラグイン不要)
- 動画再生の標準サポート
- webcamへのアクセス
- JavaScriptの実行速度も大幅に向上



HTML5 + WebGL

- デスクトップ並みの高度な3Dグラフィック
- GPUの性能を100%引き出す
- プログラマブルシェーダーが利用可能



Three.jsとは

- WebGLフレームワークのひとつ
- 使いやすいAPIを提供
- 多くの採用事例
- WebGL開発のスタンダードになりつつある
- オープンソースソフトウェア（MITライセンス）



WebGLの利用状況

- 現状では、一般的なWebページでの利用は難しい
- 技術デモは多数公開されている
- Chrome Web Storeやブラウザエクステンションなど
- メジャーな企業による実験的な利用も増えてきた
 - Google Maps GLおよびフォトツアー
 - Nokia Maps 3D
 - CNN ECOSPHERE



WebGLの利点

- リアルタイム3DグラフィックをWebにもたらし
- 広範囲な配布が容易
- インストール不要
- クロスプラットフォーム
- 開発が容易、開発環境もほぼ不要
- HTML・CSSとの組み合わせ
- 他のHTML5系APIとの組み合わせ



WebGL + Video

Real time color 3D histogram analysis

Visualize frame by frame color decomposition. Choose from different [color schemes](#) below. Feel free to pause the video and to zoom/pan the histogram below.

Choose a Color Scheme:

RGB Cube Chroma Luma Hue Chroma Value Hue

Other Options:

Camera:

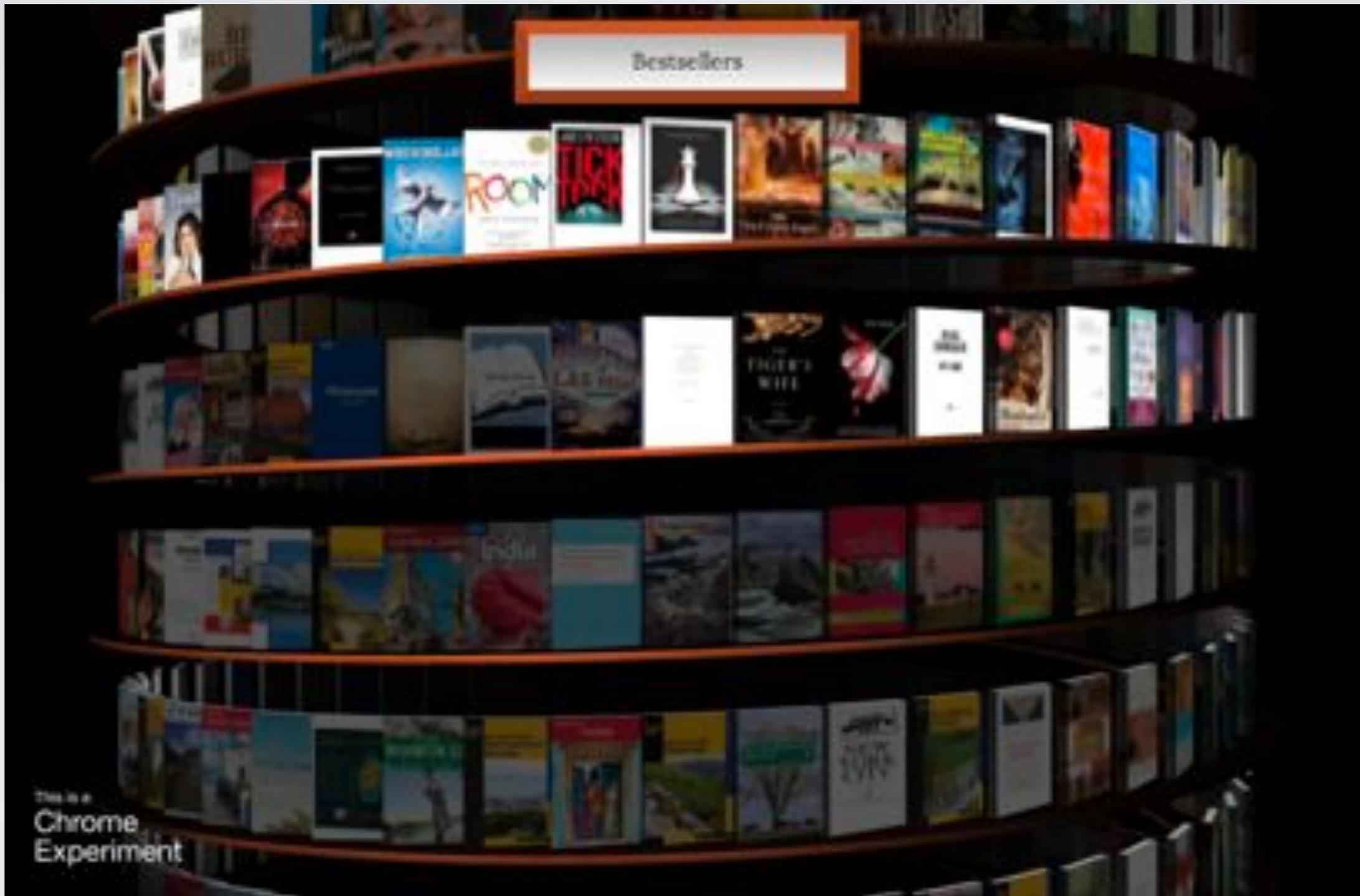
Grain: None Small Large

Background:

Copyright © [Mara-Luz Caballero](#) and [Vicente Garcia-Bermejo](#)

<http://www.senchalabs.org/philogl/PhiloGL/examples/histogram/>

WebGL + Google Books API



<http://workshop.chromeexperiments.com/bookcase/>