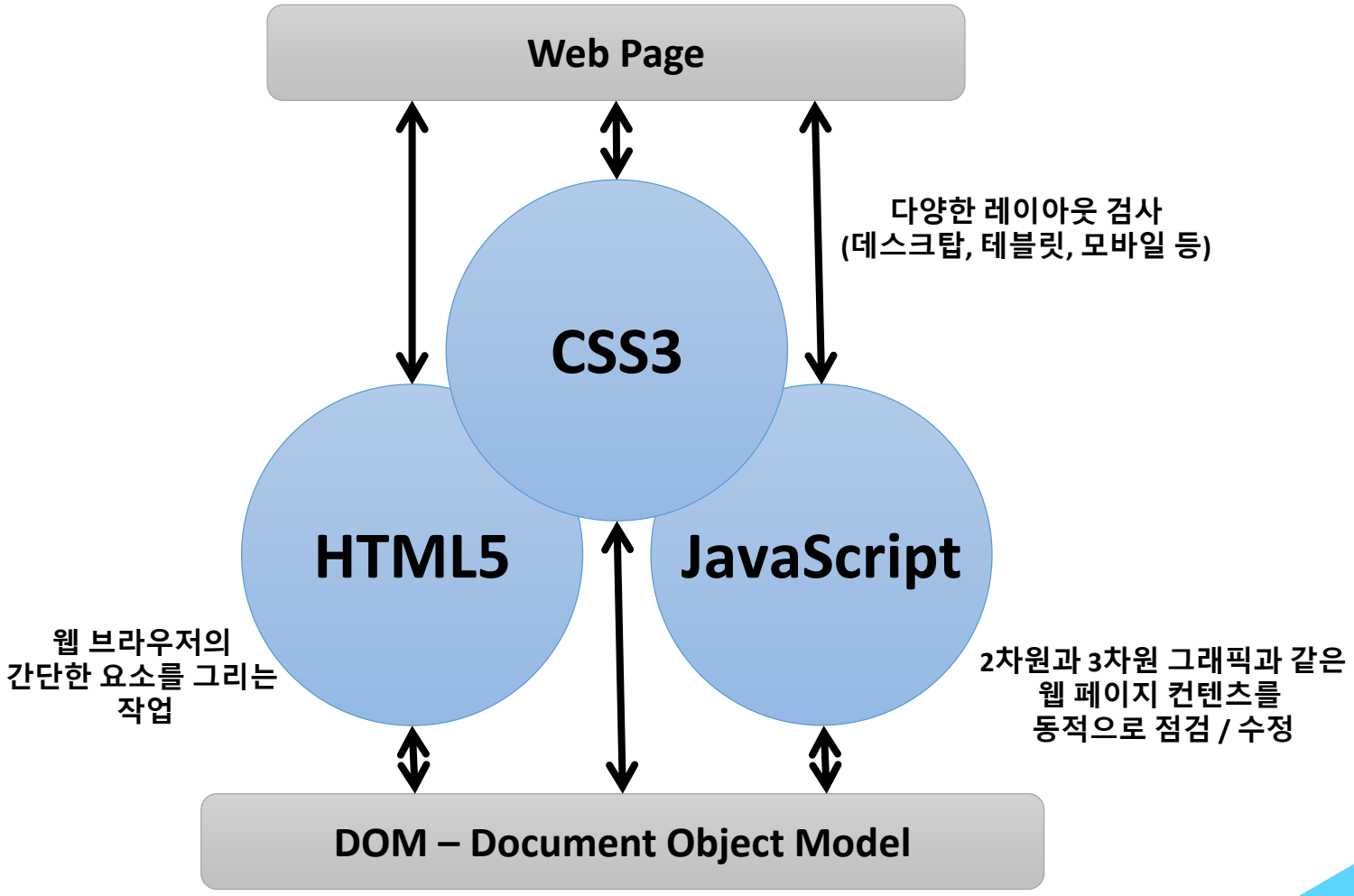
WebGL I (All-round)

3D 웹 그래픽 기술

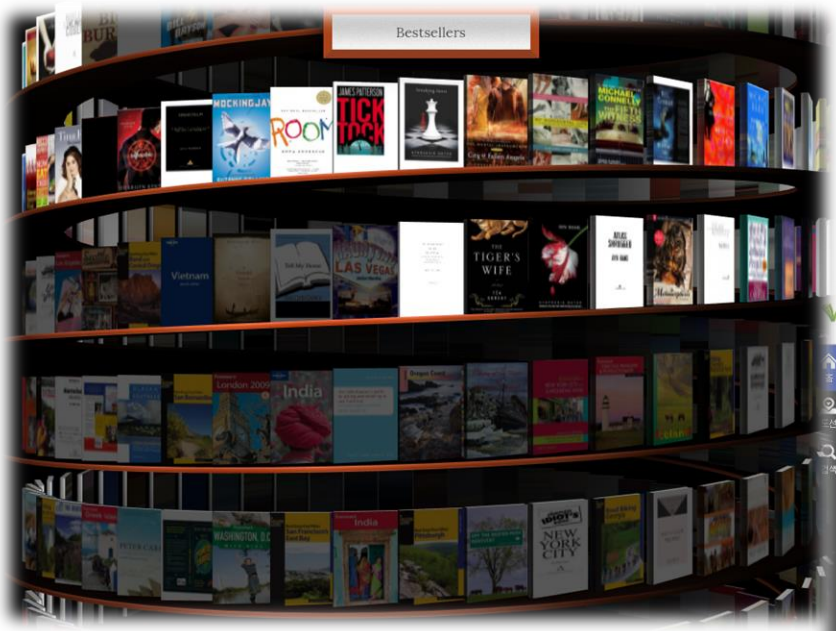
WebGL

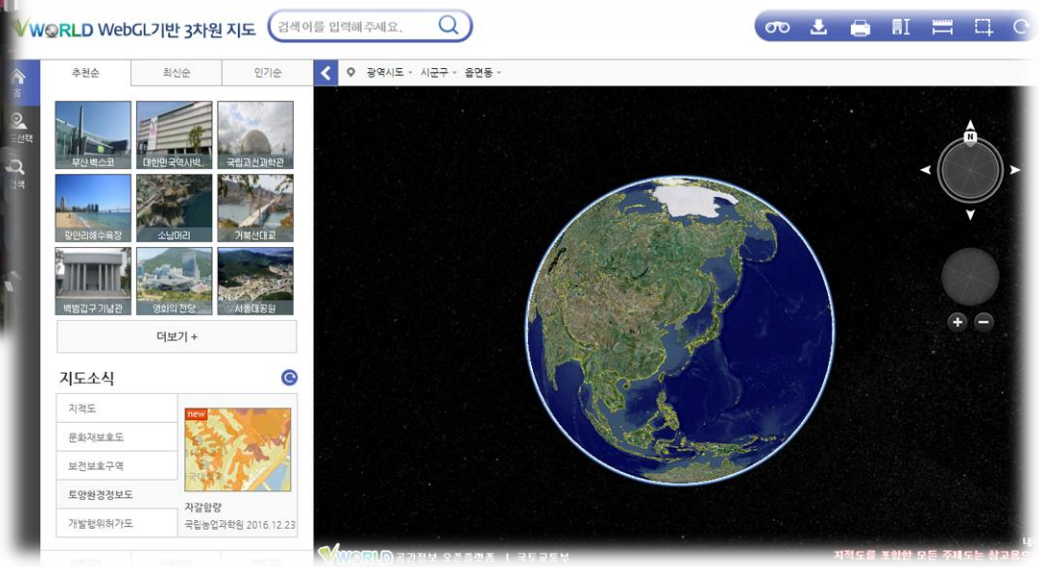


http://bookcase.chromeexperiments.com/

3D 웹 그래픽 기술

WebGL





http://map.vworld.kr/map/wcmaps.do

http://bookcase.chromeexperiments.com/

3D 웹 그래픽 기술

WebGL

- HTML5를 활용한 3D 그래픽
- 웹 브라우저에서 3D 그래픽에 대한 그리기(drawing), 시각화(displaying), 상호 작용(interacting) 을 적용할 수 있는 기술
- 명확히 웹 표준은 아니지만 거의 표준처럼 인식
- 전통적으로 3D 그래픽은 하이엔드 급 컴퓨터/게임 콘솔로 제한되고 복잡한 프로그래밍 필요

3D 웹 그래픽 기술

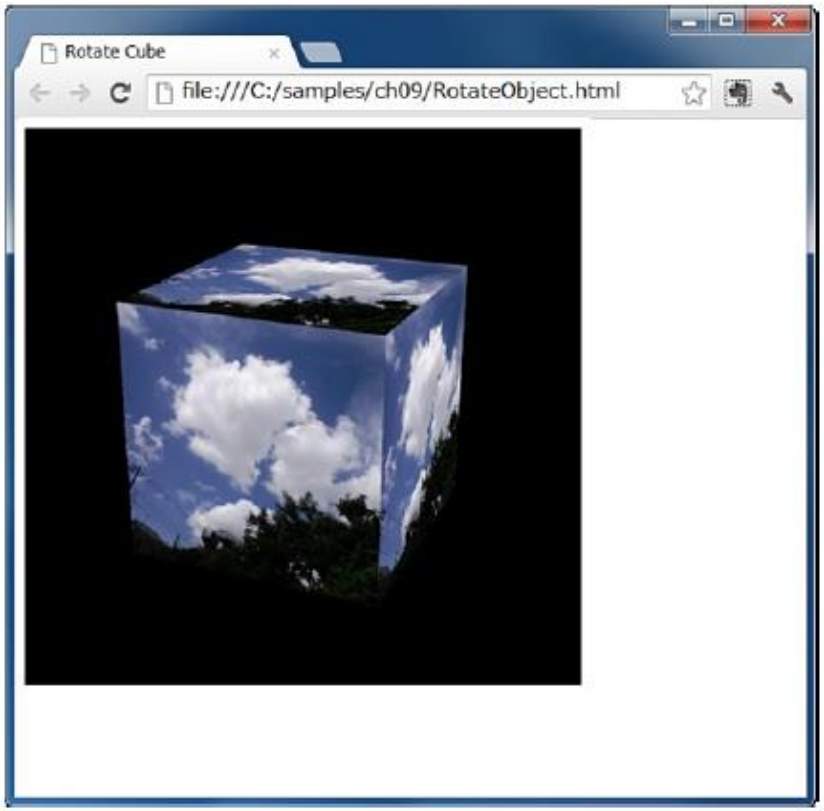
WebGL

- 2011년 03월 WebGL 1.0 발표 (OpenGL ES 2.0 기반)
- 2017년 02월 WebGL 2.0 발표 (OpenGL ES 3.0 기반)
- 비영리 단체인 크로노스(KHRONOS) 그룹에 의해 관리

KHRONOS GROUP

Advantages of WebGL

- Text Editor 만을 이용하여 3D 그래픽 어플리케이션을 개발할 수 있음
  > 3D 프로그램을 만들 때 컴파일러 / 링커와 같은 특수 응용 프로그램 개발 도구가 필요하지 않음 (브라우저 내장되어 있기 때문)



Browser (Chrome)

Notepad

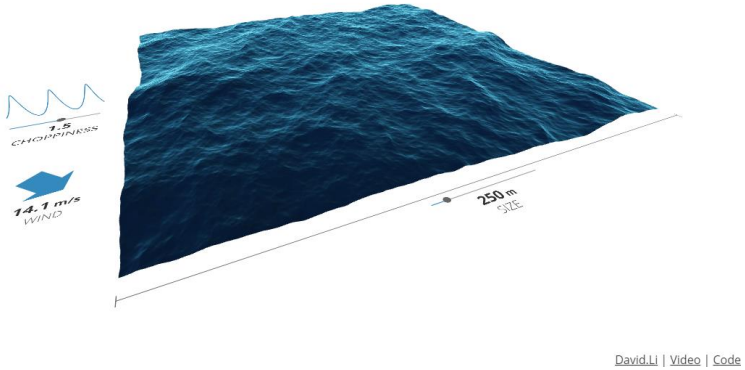
3D 웹 그래픽 기술

Advantages of WebGL

- 표준 웹 기술을 사용하여 3D 그래픽 프로그램 쉽게 배포하여 다른 웹 사용자 바로 사용 가능

- 브라우저의 모든 기능 활용 가능

http://carvisualizer.plus360degrees.com/threejs/



http://david.li/waves/

http://webglsamples.org/

https://experiments.withgoogle.com/chrome?tag=WebGL

3D 웹 그래픽 기술

Origins of WebGL

- 가장 널리 사용되는 기술 : Direct3D / OpenGL

- Direct3D
  > Microsoft DirectX 기술 일부
  > Windows 플랫폼에서 사용되는 3D 그래픽 기술
  > Microsoft가 제어하는 독점적인 API

- OpenGL
  > Open Source (royalty-free)
  > 다양한 플랫폼에 널리 사용 (Macintosh, Linux, Smart Phone, Tablet, Game Console)
  > Windows에서도 잘 지원

Origins of WebGL

위키백과
우리 모두의 백과사전

대문
사용자 모임
요즘 화제
최근 바뀜
모든 문서 보기
임의 문서로
도움말
기부

도구
여기를 가리키는 문서
가리키는 글의 바뀜
파일 올리기
특수 문서 목록
고유 링크
문서 정보
위키데이터 항목
이 문서 인용하기

인쇄/내보내기
책 만들기
PDF로 다운로드
인쇄용 판

다른 프로젝트
위키미디어 공용

다른 언어
العربية
English
Español
हिन्दी
Bahasa Indonesia
Bahasa Melayu
Português

위키백과는 누구나 고칠 수 있는 백과사전입니다. 편집 버튼을 눌러보세요. (자세한 정보) [숨기기]

DirectX

위키백과, 우리 모두의 백과사전.

Microsoft DirectX(마이크로소프트 다이렉트엑스)는 멀티미디어, 특히 게임 프로그래밍에서 마이크로소프트 플랫폼에서 작업을 위한 API의 집합이다. 다이렉트엑스는 마이크로소프트 윈도우, 세가, 드림캐스트, 마이크로소프트 엑스박스 및 엑스박스 360을 위한 비디오 게임 개발에 널리 쓰인다.

다이렉트엑스는 또한 게임뿐 아니라 최근에 나온 3차원 그래픽 하드웨어를 사용하여 높은 품질의 3차원 그래픽을 빠르게 렌더링할 수 있기 때문에 소프트웨어 업계 전반에서 사용되기도 한다.

다이렉트엑스 런타임과 소프트웨어 개발킷은 무료이지만 개조는 할 수 없는 클로즈드 소스(오픈 소스의 반대 개념) 소프트웨어이다. 다이렉트엑스 런타임은 원래 컴퓨터 게임 개발자들에게만 공개되었으나, 최근에는 기본적으로 윈도우에 포함되어 있다. 상위 버전으로 업데이트하고 싶을 경우 마이크로소프트의 공식 홈페이지를 통해 설치할 수 있다.

Direct3D 9Ex, Direct3D 10은 윈도우 비스타 이상, Direct3D 11은 윈도우 비스타 SP2 또는 윈도우 7 이상, Direct3D 12는 윈도우 10에서만 사용할 수 있다. 그 까닭은 이러한 새로운 버전들은 윈도우 비스타에 도입되었던 새로운 윈도우 디스플레이 드라이버 모델이 있어야 동작하기 때문이다. 그래서 DirectX 조건에 맞더라도 드라이버가 윈도우 디스플레이 드라이버 모델을 지원하지 않으면 사용할 수 없다. 새로운 비스타/WDDM 그래픽스 구조에는 데스크톱 창 관리자와 같이, 그래픽 하드웨어를 여러 개의 응용 프로그램과 서비스에 가상화할 수 있게 도와 주는 새로운 비디오 메모리 관리자를 포함하고 있다.

목차 [숨기기]
1 구성 요소
2 역사
3 대안
4 같이 보기

DirectX

마이크로소프트 윈도우의 구성 요소

Microsoft® DirectX®

자세한 정보

종류	API
기본 포함 운영 체제	윈도우 95 OSR2 윈도우 NT 4.0 및 이후 출시 운영 체제

관련 구성 요소

Direct2D
Direct3D
DirectWrite
XAudio2

Origins of WebGL

로그인하지 않음 토론 기여 계정 만들기 로그인

위키백과
우리 모두의 백과사전

문서 토론 읽기 편집 역사 보기 검색

OpenGL

위키백과, 우리 모두의 백과사전.

OpenGL(**Open** Graphics **L**ibrary, 오픈지엘)은 1992년 실리콘 그래픽스사에서 만든 2차원 및 3차원 그래픽스 표준 API 규격으로, 프로그래밍 언어 간 플랫폼 간의 교차 응용 프로그래밍을 지원한다. 이 API는 약 250여개 가량의 함수 호출을 이용하여 단순한 기하도형에서부터 복잡한 삼차원 장면을 생성할 수 있다. OpenGL은 현재 CAD, 가상현실, 정보시각화, 비행 시뮬레이션 등의 분야에서 활용되고 있다. 또한 컴퓨터 게임 분야에서도 널리 활용되고 있으며, 마이크로소프트사의 Direct3D와 함께 컴퓨터 그래픽 세계를 양분하고 있다. Direct3D와는 달리, 표준안이 여러 관련 업체의 토론과 제안으로 이루어지기에 버전 업데이트는 느린 편이다. OpenGL을 사용하여 개발된 대표적인 게임은 이드 소프트웨어의 퀘이크, 둠3 시리즈이다. 현재 비영리 기술 컨소시엄인 크로노스 그룹에 의하여 관리되고 있다.

OpenGL

개발자	크로노스 그룹 (원 개발자는 실리콘 그래픽스)
최근 버전	4.5 / 2014년 8월 11일, 792일 경과
운영 체제	크로스 플랫폼
플랫폼	크로스 플랫폼
언어	C
종류	API
웹사이트	www.opengl.org

목차 [숨기기]

1 역사
2 설계
3 예제
4 같이 보기
4.1 OpenGL 지원 라이브러리
4.2 기타 3D 그래픽스 API
4.3 기타 2D 그래픽스 API
5 각주
6 바깥 고리

역사 [편집]

대문
사용자 모임
요즘 화제
최근 바뀜
모든 문서 보기
임의 문서로
도움말
기부

도구
여기를 가리키는 문서
가리키는 글의 바뀜
파일 올리기
특수 문서 목록
고유 링크
문서 정보
위키데이터 항목
이 문서 인용하기

인쇄/내보내기
책 만들기
PDF로 다운로드
인쇄용 판

다른 프로젝트
위키미디어 공용
위키책

다른 언어
العربية

Origins of WebGL

OpenGL ES Overview

The Standard for Embedded Accelerated 3D Graphics

OpenGL® ES is a royalty-free, cross-platform API for rendering advanced 2D and 3D graphics on embedded and mobile systems - including consoles, phones, appliances and vehicles. It consists of a well-defined subset of desktop OpenGL suitable for low-power devices, and provides a flexible and powerful interface between software and graphics acceleration hardware.

OpenGL ES API Versions at a Glance

OpenGL ES 3.2 - Additional OpenGL functionality

The latest in the series, OpenGL ES 3.2 added additional functionality based on the Android Extension Pack for OpenGL ES 3.1, which brought the mobile API's functionality significantly closer to it's desktop counterpart - OpenGL.

OpenGL ES 3.1 - Bringing Compute to Mobile Graphics

Despite being only a bump in the minor revision of the API, OpenGL ES 3.1 was an enormous milestone for the API, as it added the ability to do general purpose compute in the API, bringing compute to mobile graphics.

OpenGL ES 3.0 - Enhanced Graphics

OpenGL ES 3.0 was another evolutionary step for OpenGL ES, notably including multiple render targets, additional texturing capabilities, uniform buffers, instancing and transform feedback.

3D 웹 그래픽 기술

Origins of WebGL

안드로이드 / iOS 그래픽 가속

베이스 라이브러리

OpenGL ES 1.0 > OpenGL 1.3
OpenGL ES 1.1 > OpenGL 1.5
OpenGL ES 2.0 > OpenGL 2.0
OpenGL ES 3.0 > OpenGL 4.3

버전 일람

- OpenGL ES 1.0 : 2003년 7월 (OpenGL 1.3 기반)
- OpenGL ES 1.1 : 2004년 9월 (OpenGL 1.5 기반)
- OpenGL ES 2.0 : 2007년 3월 (OpenGL 2.0~3.0 기반)
- OpenGL ES 3.0 : 2012년 8월 (OpenGL 4.3 기반)
- OpenGL ES 3.1 : 2014년 3월
- OpenGL ES 3.2 : 2015년 8월

3D 웹 그래픽 기술

Support of WebGL

- 최신 브라우저 (Modern browsers)에서 대부분 지원
- 하지만, GPU 제공 부분은 제공 장치에 따라 다름

Desktop browsers [edit]

- Google Chrome – WebGL has been enabled on all platforms that have a capable graphics card with updated drivers since version 9, released in February 2011.[16][17] By default on Windows, Chrome uses the ANGLE (Almost Native Graphics Layer Engine) renderer to translate OpenGL ES to Direct X 9.0c or 11.0, which have better driver support.[18] On Linux and Mac OS X the default renderer is OpenGL however.[19] It is also possible to force OpenGL as the renderer on Windows.[18] Since September 2013, Chrome also has a newer Direct3D 11 renderer, which however requires a newer graphics card.[20][21]
- Mozilla Firefox – WebGL has been enabled on all platforms that have a capable graphics card with updated drivers since version 4.0.[22] Since 2013 Firefox also uses DirectX on the Windows platform via ANGLE.[18]
- Safari – Safari 6.0 and newer versions installed on OS X Mountain Lion, Mac OS X Lion and Safari 5.1 on Mac OS X Snow Leopard implemented support for WebGL, which was disabled by default before Safari 8.0.[23][24][25][26][27]
- Opera – WebGL has been implemented in Opera 11 and 12, although was disabled by default in 2014.[28][29]
- Internet Explorer – WebGL is partially supported in Internet Explorer 11.[30][31][32][33] It initially failed the majority of official WebGL conformance tests, but Microsoft later released several updates. The latest 0.94 WebGL engine currently passes ~97% of Khronos tests.[34] WebGL support can also be manually added to earlier versions of Internet Explorer using third-party plugins such as IEWebGL.[35]
- Microsoft Edge – The initial stable release supports WebGL version 0.95 (context name: "experimental-webgl") with an open source GLSL to HLSL transpiler.[36]

Support of WebGL

- 모바일 브라우저의 경우에도 대부분 지원

Mobile browsers [edit]

- BlackBerry 10 – WebGL is available for BlackBerry devices since OS version 10.00[37]
- BlackBerry PlayBook – WebGL is available via WebWorks and browser in PlayBook OS 2.00[38]
- Android Browser – Basically unsupported, but the Sony Ericsson Xperia range of Android smartphones have had WebGL capabilities following a firmware upgrade.[39] Samsung smartphones also have WebGL enabled (verified on Galaxy SII (4.1.2) and Galaxy Note 8.0 (4.2)). Supported in Google Chrome that replaced Android browser in many phones (but is not a new standard Android Browser).
- Internet Explorer – WebGL is available on Windows Phone 8.1
- Firefox for mobile – WebGL is available for Android and MeeGo devices since Firefox 4.[40][41]
- Firefox OS[41]
- Google Chrome – WebGL is available for Android devices since Google Chrome 25 and enabled by default since version 30.[42]
- Maemo – In Nokia N900, WebGL is available in the stock microB browser from the PR1.2 firmware update onwards.[43]
- MeeGo – WebGL is unsupported in the stock browser "Web." However, it is available through Firefox.[41]
- Microsoft Edge – WebGL is available on Windows 10 Mobile.[44]
- Opera Mobile – Opera Mobile 12 supports WebGL (on Android only).[45]
- Sailfish OS – WebGL is supported in the default Sailfish browser.[46]
- Tizen – WebGL is supported[47]
- Ubuntu Touch
- WebOS
- iOS – WebGL is available for mobile Safari, in iOS 8.[48]

WebGL Version

Khronos Working Draft

WebGL

KHRONOS GROUP

WebGL Specification

Editor's Draft Wed Oct 3 16:43:01 2018 -0700

This version:
https://www.khronos.org/registry/webgl/specs/latest/1.0/
WebIDL: https://www.khronos.org/registry/webgl/specs/latest/1.0/webgl.idl
Latest version:
https://www.khronos.org/registry/webgl/specs/latest/1.0/
WebIDL: https://www.khronos.org/registry/webgl/specs/latest/1.0/webgl.idl
Previous version:
https://www.khronos.org/registry/webgl/specs/1.0.3/
WebIDL: https://www.khronos.org/registry/webgl/specs/1.0.3/webgl.idl
Editors:
Dean Jackson (Apple Inc.)
Jeff Gilbert (Mozilla Corp.)

Abstract

This specification describes an additional rendering context and support objects for the HTML 5 canvas element [CANVAS]. This context allows rendering using an API that conforms closely to the OpenGL ES 2.0 API.

https://www.khronos.org/registry/webgl/specs/latest/1.0/

WebGL Version

Khronos Working Draft

WebGL

KHRONOS GROUP

WebGL 2.0 Specification

Editor's Draft Fri Oct 26 21:28:35 2018 -0400

This version:
https://www.khronos.org/registry/webgl/specs/latest/2.0/
WebIDL: https://www.khronos.org/registry/webgl/specs/latest/2.0/webgl2.idl
Latest version:
https://www.khronos.org/registry/webgl/specs/latest/2.0/
WebIDL: https://www.khronos.org/registry/webgl/specs/latest/2.0/webgl2.idl
Previous version:
https://www.khronos.org/registry/webgl/specs/1.0.2/
WebIDL: https://www.khronos.org/registry/webgl/specs/1.0.2/webgl.idl
Editors:
Dean Jackson (Apple Inc.)
Jeff Gilbert (Mozilla Corp.)

Abstract

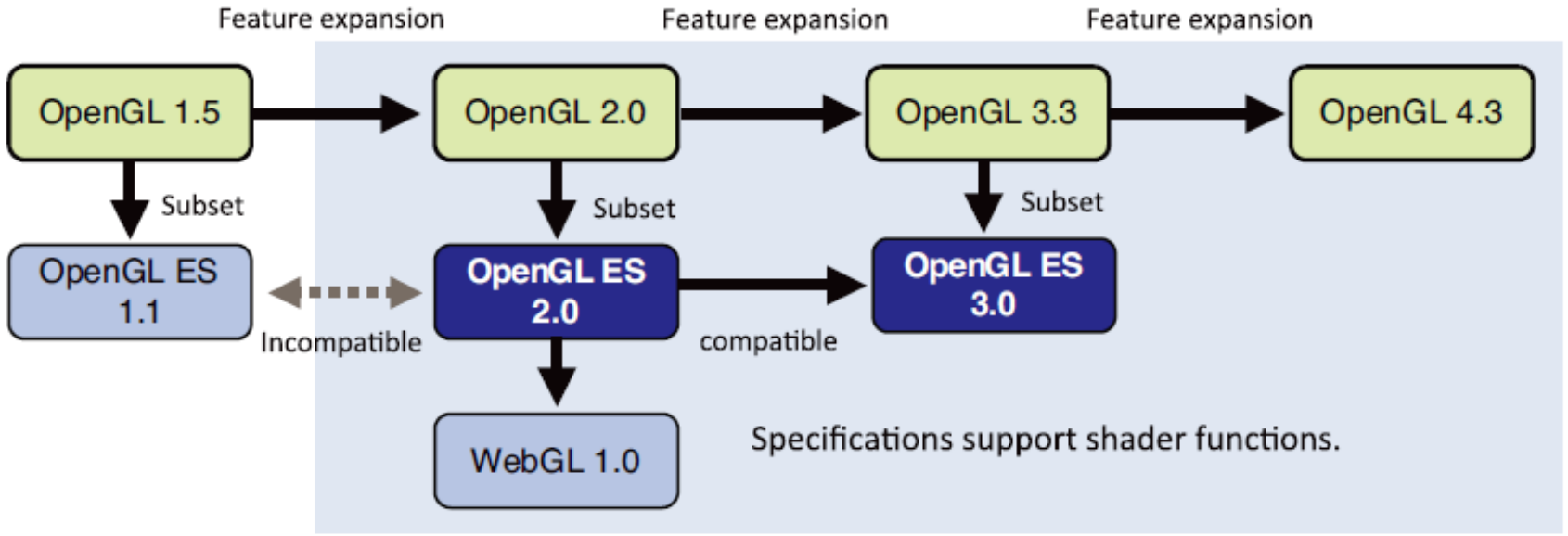
This is Version 2.0 of the WebGL Specification.

This specification describes an additional rendering context and support objects for the HTML 5 canvas element [CANVAS]. This context allows rendering using an API that conforms closely to the OpenGL ES 3.0 API.

https://www.khronos.org/registry/webgl/specs/latest/2.0/

3D 웹 그래픽 기술

WebGL



- **WebGL 1.0 -> OpenGL ES 2.0 기반**
- **WebGL 2.0 -> OpenGL ES 3.0 기반**
- **Shader function / Shaders : 컴퓨터 프로그래밍 (C와 유사한 특수 프로그램 언어)**
  정교한 시각 효과를 프로그래밍

3D 웹 그래픽 기술

Shading language

- OpenGL ES 2.0에서 사용된 shading language는 OpenGL shading language 기반
- OpenGL ES 2.0 shading language
- WebGL은 OpenGL ES 2.0 기반 -> ∴ GLSL ES 사용

GLSL

위키백과, 우리 모두의 백과사전.

GLSL(OpenGL Shading Language, OpenGL 셰이딩 언어)는 C 언어를 기초로 한, 상위 레벨 셰이딩 언어이다. **GLslang**로도 알려져 있다. HLSL과 유사한 이 언어는 어셈블리 언어나 하드웨어에 의존한 언어를 사용하지 않고, 개발자가 그래픽스 파이프라인을 직접 제어할 수 있도록 OpenGL ARB(Architecture Review Board)가 책정하였다.

GLSL은 프로파일이 있어서 개발자는 Cg로 개발한 코드를 바로 변환할 수도 있다.

목차 [숨기기]

특징 [편집]

- 매킨토시나 윈도, 리눅스 등의 여러 운영 체제 간의 호환성을 지원한다.
- GLSL을 지원하는 어떠한 제조사의 그래픽 카드에서도 동작하는 셰이더를 쓸 수 있다.
- 하드웨어 제조사는 그래픽 카드 장치 드라이버 내에 GLSL 컴파일러를 포함할 수 있도록 지원한다. 이 덕분에, 그 그래픽 카드의 마이크로아키텍처에 최적화된 코드를 만들 수 있다.

Cg (프로그래밍 언어)

위키백과, 우리 모두의 백과사전.

Cg 또는 **C for Graphics**는 엔비디아가 마이크로소프트와 협력하여 개발한 상위 레벨 셰이딩 언어이다. 마이크로소프트의 HLSL과 매우 비슷하다.

Cg는 C 언어를 기반으로 하며, C 언어와 같은 문법을 가졌지만 일부 기능들은 변형되고 그래픽 처리 장치를 프로그래밍하기에 적절한 새로운 자료형이 추가되었다.

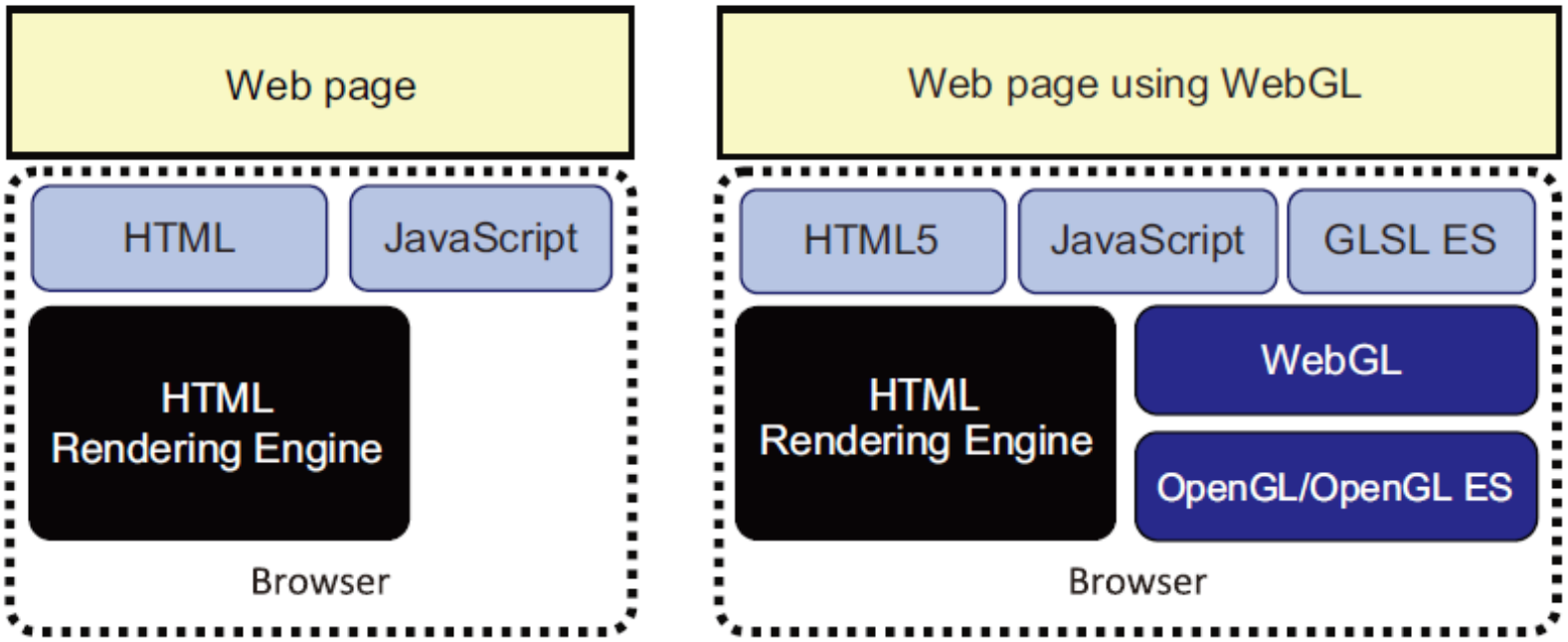
바깥 고리 [편집]

- (영어) Cg 웹페이지

분류: 셰이더 언어

Structure of WebGL Applications

- HTML 동적 웹페이지 : HTML, JavaScript
- WebGL 페이지 : GLSL ES (쉐이더 언어) 추가



3D 웹 그래픽 기술

WebGL 기초

- WebGL = Rasterization 엔진
- 웹 브라우저 내부에서 실행되지만 하드웨어 GPU에서 실행
- GPU에서 실행되는 코드 = 함수 쌍 형태 (form of pairs of functions)
- Vertex Shader / Fragment Shader
  > C/C++처럼 엄격한 형태를 가지는 GL Shader Language(GLSL) 작성
- Vertex Shader : 버텍스 위치 계산
- Fragment Shader : Vertex Shader로 그려진 Primitive들에 대해 색을 계산

3D 웹 그래픽 기술

Shader 데이터

- 함수(vertex/fragment)들이 접근 해야하는 모든 데이터는 GPU에서 제공
- Shader가 데이터를 받는 방법에 대해 4가지
  (Attribute/Buffer, Uniform, Texture, Varying)

- Attribute / Buffer
  > Buffer : 2진 데이터 배열 ,
  위치, 법선, Texture 좌표, Vertext 색상 등 포함 (이 외에도 자유롭게 넣을 수 있음)
  >Attribute: 어떻게 Buffer에서 데이터를 가지고 오며, Vertex Shader에서 제공할지 방법을 지정하는데 사용

- Uniform
  > Shader Program을 실행하기 전 선언하는 유용한 전역 변수

- Texture
  > Shader Program이 무작위로 접근할 수 있는 데이터 배열
  > 이미지 데이터, 색상 등

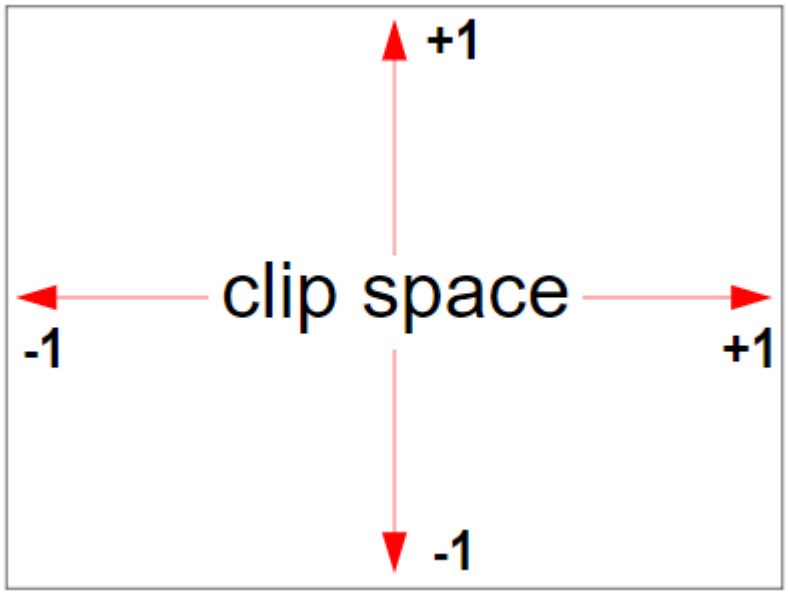
- Varying
  > Vertex Shader가 Fragment Shader에 데이터를 넘기는 방법

WebGL 맛보기

- 프로그래머가 할 일
  : Clip 공간 좌표 / 색상

- Vertex Shader -> Clip 공간 좌표 제공
- Fragment Shader -> 색상 제공

- Clip 공간 좌표는 canvas 크기에 상관 없이 항상 -1에서 +1까지 사용



WebGL 맛보기

1. Vertex Shader

```
// Attribute는 Buffer로부터 데이터를 받습니다.
attribute vec4 a_position;

// 모든 shader는 main 함수를 가지고 있습니다.
void main() {
  // gl_Position은 Vertex Shader가 설정을 담당하는 특수 변수입니다.
  gl_Position = a_position;
}
```

2. Fragment Shader

```
// Fragment Shader는 기본 정밀도를 가지고 있지 않기 때문에 하나를 선언해야 합니다.
// mediump(중간 정도 정밀도)은 기본값으로 좋습니다.
precision mediump float;

void main() {
  // gl_FragColor는 Fragment Shader의 설정을 담당하는 특수 변수입니다.
  gl_FragColor = vec4(1, 0, 0.5, 1); // 붉은-보라색 반환
}
```

WebGL 맛보기

3. HTML canvas 요소 필요

```
<canvas id="c"></canvas>
```

4. JavaScript 에서 canvas 요소 활용

```
var canvas = document.getElementById("c");
var gl = canvas.getContext("webgl");
if (!gl) {
// webgl을 쓸 수 없어요!
...
}
```

WebGL 맛보기

5. Shader 생성, GLSL 업로드, Shader 컴파일 함수 생성

```
function createShader(gl, type, source) {
  var shader = gl.createShader(type);
  gl.shaderSource(shader, source);
  gl.compileShader(shader);

  var success = gl.getShaderParameter(shader, gl.COMPILE_STATUS);
  if (success) {
    return shader;
  }

  console.log(gl.getShaderInfoLog(shader));
  gl.deleteShader(shader);
}
```

WebGL 맛보기

6. Shader 함수 호출

```
var vertexShaderSource = document.getElementById("2d-vertex-shader").text;
var fragmentShaderSource = document.getElementById("2d-fragment-shader").text;

var vertexShader = createShader(gl, gl.VERTEX_SHADER, vertexShaderSource);
var fragmentShader = createShader(gl, gl.FRAGMENT_SHADER, fragmentShaderSource);
```

7. Vertex / Frament Shader 연결

```
var program = createProgram(gl, vertexShader, fragmentShader);
function createProgram(gl, vertexShader, fragmentShader) {
  var program = gl.createProgram();
  gl.attachShader(program, vertexShader);
  gl.attachShader(program, fragmentShader);
  gl.linkProgram(program);
   var success = gl.getProgramParameter(program, gl.LINK_STATUS);
  if (success) {
    return program;
  }
   console.log(gl.getProgramInfoLog(program));       gl.deleteProgram(program);
}
```

WebGL 맛보기

8 . Attribute 찾기

```
var positionAttributeLocation = gl.getAttribLocation(program, "a_position");
```

9 . Buffer 생성 (Attribute는 Buffer부터 데이터를 가지고 오기 때문)

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

10 . 전역 bind point를 통해 WebGL 많은 자원 조작
(bind point: WebGL 내부 전역 변수) bind point 할당 /자원 참조

```
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

// 2d point 3개
var positions = [ 0, 0,
                  0, 0.5,
                  0.7, 0,
                ];
gl.bufferData( gl.ARRAY_BUFFER, new Float32Array(positions), gl.STATIC_DRAW );
```

WebGL 맛보기- Rendering

- 화면을 그리기 전 canvas를 화면 크기와 일치하도록 조정 (drawingbuffer / width, height)

- Drawingbuffer : canvas 에 얼마나 많은 픽셀이 있는지
- Width, height : canvas 표시되는 크기

```
<canvas id="c" width="400" height="300"></canvas>


<canvas id="c"> </canvas>

var canvas = document.getElementById("c");
canvas.width = 400;
canvas.height = 300;
```

```
<canvas id="c" width="10" height="15"
        style="width:400px; height: 300px">
</canvas>
```

WebGL 맛보기- Rendering

- Canvas를 화면 크기와 일치하도록 조정

```
webglUtils.resizeCanvasToDisplaySize(gl.canvas)
```

```
gl.viewport(0, 0, gl.canvas.width, gl.canvas.height);
```

-1 <-> +1 clip >>> x: 0 <-> gl.canvas.width,
y: 0 <-> gl.canvas.height

- Canvas 화면 지우기

```
// Canvas 지우기
gl.clearColor(0, 0, 0, 0);
gl.clear(gl.COLOR_BUFFER_BIT);
```

- Shader Program 실행

```
gl.useProgram(program);
```

WebGL 맛보기- Rendering

- Attribute 활성화 / 데이터 output 방법 지정

```
gl.enableVertexAttribArray(positionAttributeLocation);
```

```
// position buffer 할당
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

// attribute에게 positionBuffer의 데이터를 어떻게 꺼낼지 지시 (ARRAY_BUFFER)
var size = 2;          // 실행될 때마다 2개 구성 요소 사용
var type = gl.FLOAT;   // 데이터는 32bit 소수점
var normalize = false; // 정규화되지 않은 데이터
var stride = 0;        // 0 = 반복할 때마다 size * sizeof(type)만큼 다음 위치로 이동
var offset = 0;        // buffer 시작점
gl.vertexAttribPointer(
  positionAttributeLocation,
  size,
  type,
  normalize,
  stride,
  offset
);
```

WebGL 맛보기- Rendering

- GLSL program 실행 요청

```
var primitiveType = gl.TRIANGLES;
var offset = 0;
var count = 3;
gl.drawArrays(primitiveType, offset, count);
```

```
var positions = [ 0, 0,
                  0, 0.5,
                  0.7, 0,
                ];
```

clip 공간 화면 공간

0, 0 -> 200, 150

0, 0.5 -> 200, 225

0.7, 0 -> 340, 150