WebGL OES_EGL_image_external Extension Specification

Name

OES_EGL_image_external

Contact

WebGL working group (public_webgl 'at' khronos.org)

Contributors

Byungseon Shin(sun.shin@lge.com), LG Electronics

Andrey Volykhin(andrey.volykhin@lge.com), LG Electronics

Version

Last modified date: October 11, 2016
Revision: 3

Number

WebGL extension #k

Dependencies

Written against the WebGL API 1.0 specification.

Overview

This extension exposes the OES_EGL_image_external functionality to WebGL.

The following WebGL-specific behavioral changes apply:

- Defines a new texture target `TEXTURE_EXTERNAL_OES`.
- Provides a mechanism for binding `HTMLVideoElement`'s EGLImage to external texture targets.

Consult the above extension for documentation, issues and new functions and enumerants.

When this extension is enabled:

- Add support for `OES_EGL_image_external` texture binding of HTMLVideoElement.
- When a *fragment* shader enables, requires, or warns `OES_EGL_image_external` with an `#extension` directive:
  - `samplerExternalOES` is a built-in type.
  - `vec4 texture2D(samplerExternalOES sampler, vec2 coord)` is a built-in function.
- The GLSL macro `OES_EGL_image_external` is defined as `1`.

IDL

```
[NoInterfaceObject]
interface OESEGLImageExternal {
    const GLenum TEXTURE_EXTERNAL_OES             = 0x8D65;
    const GLenum SAMPLER_EXTERNAL_OES             = 0x8D66;
    const GLenum TEXTURE_BINDING_EXTERNAL_OES     = 0x8D67;
    const GLenum REQUIRED_TEXTURE_IMAGE_UNITS_OES = 0x8D68;

    [RaisesException] void EGLImageTargetTexture2DOES(
        GLenum target, HTMLVideoElement video);
};
```

Sample Code

This a fragment shader that samples a video texture.

```
#extension GL_OES_EGL_image_external : require
precision mediump float;
varying vec2 v_texCoord;

//uniform sampler2D uSampler;
uniform samplerExternalOES uSampler;

void main(void) {
  gl_FragColor = texture2D(uSampler, v_texCoord);
}
```

This shows application that renders video using proposed extension.

```
var videoElement = document.getElementById("video");
var videoTexture = gl.createTexture();

function update() {
    var ext = gl.getExtension('OES_EGL_image_external');
    if(ext !=== null){
        gl.bindTexture(ext.TEXTURE_EXTERNAL_OES, videoTexture);
        ext.EGLImageTargetTexture2DOES(ext.TEXTURE_EXTERNAL_OES, videoElement);
        gl.bindTexture(ext.TEXTURE_EXTERNAL_OES, null);
    }
}

function render() {
    gl.clearColor(0.0, 0.0, 1.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);

    gl.bindBuffer(gl.ARRAY_BUFFER, squareVerticesBuffer);
    gl.vertexAttribPointer(vertexPositionAttribute, 3, gl.FLOAT, false, 0, 0);

    gl.activeTexture(gl.TEXTURE0);
    gl.bindTexture(ext.TEXTURE_EXTERNAL_OES, videoTexture);
    gl.uniform1i(gl.getUniformLocation(shaderProgram, "uSampler"), 0);

    gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
}
```

Application renders each video frames into WebGL canvas based on game-loop pattern.

```
update();

while (true) {
   processInput();
   render();
}
```

Conformance Tests

Issues

Revision History

Revision 1, 2016/10/11

- Initial revision.

Revision 2, 2016/10/11

- Refine sample code.

Revision 3, 2016/10/11

- Remove comment on WebGL API 2.0 dependency
- Refine sample code as an implicit texture updating mode