ESAT

Programación gráfica

WebGPU y OpenGL

Nombre Alumno/s: Pau Badia, Marc Folgado, Carlos García, Rubén Gil y Luis Miguel Jiménez
Nombre Asignatura: Programación gráfica
Nombre Profesor: Iván Sancho
Fecha : 22/02/24

índice

- ¿Qué es WebGPU?
- Plataformas compatibles
- Dispositivos móviles
- Shaders y compilacion
- Multithreading
- Performance
- Retrocompatibilidad
- Graphic pipeline
- Primeros pasos

¿Qué es WebGPU?

- Diseñada para permitir a los desarrolladores acceder al hardware de gráficos de manera más eficiente a través del navegador web
- Mayor rendimiento y una mayor flexibilidad

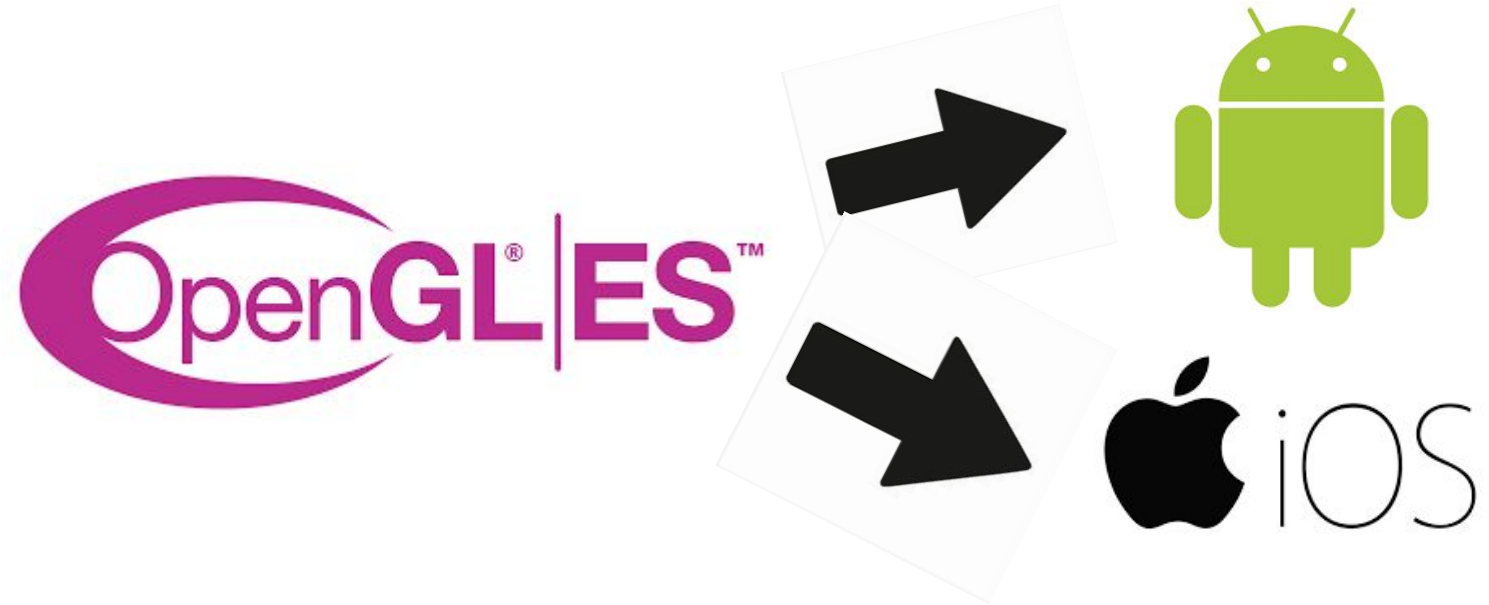
Plataformas Compatibles

- Diseñado para ser una API multiplataforma.
- Compatible con las principales plataformas de navegadores web.




Dispositivos móviles

- Variante de OpenGL llamada OpenGL ES, la cual es completamente compatible con dispositivos móviles.
- La disponibilidad de WebGPU en dispositivos móviles depende de la capacidad del navegador.



Shaders

- Lenguaje propio para programar Shaders:

  WGSL (WebGPU Shading Language). Basado ligeramente en **GLSL** (OpenGL Shading Language).

- Modernizado y optimizado para ser ejecutado en navegador.

WGSL y GLSL

- WGSL cuenta con un "tipado" muy fuerte en la declaración de variables. Heredado de la sintaxis de **Rust**.

WGSL	GLSL
mat2x2<f32>	mat2
mat3x2<f32>	mat3x2
mat4x2<f32>	mat4x2
mat2x3<f32>	mat2x3
mat3x3<f32>	mat3
mat4x3<f32>	mat4x3
mat2x4<f32>	mat2x4
mat3x4<f32>	mat3x4
mat4x4<f32>	mat4

From GLSL to WGSL: the future of shaders on the Web · Damien Seguin - Technical Lead / Computational Designer / Creative Developer / Generative Artisan | https://dmnsgn.me

WGSL y GLSL

- Declaración de funciones diferente debido a la influencia de Rust

+ WGSL:

```
fn saturate(x: f32) -> f32 {
  return clamp(x, 0.0, 1.0);
}
```

+ GLSL:

```
float saturate(float x) {
  return clamp(x, 0.0, 1.0);
}
// or
#define saturate(x) clamp(x, 0.0, 1.0)
// but more on that later
```

From GLSL to WGSL: the future of shaders on the Web · Damien Seguin - Technical Lead / Computational Designer / Creative Developer / Generative Artisan | https://dmnsgn.me

Multithreading

- WebGPU no cuenta con un soporte oficial. Pero existen investigaciones al respecto.
- OpenGL lo permite de forma indirecta

Multithreading

Multithreading WebGPU

- Propuesta de acceso a un mismo “**device**” desde varios “**threads**” de Javascript.
- Objetos de WebGPU compartidos entre “threads”

Multithreading OpenGL

- Mediante el uso de las **“Shared List”**
- Permite compartir elementos entre múltiples contextos de OpenGL. Útil para trabajar con múltiples “**threads**”

Retrocompatibilidad OpenGL

- Fuerte enfoque en la retrocompatibilidad.
- Incluye funciones obsoletas.
- Funciones obsoletas siguen estando disponibles para garantizar su correcto funcionamiento.

Retrocompatibilidad WebGPU

- Diseñado para aprovechar al máximo las capacidades del hardware gráfico.
- Es posible que las aplicaciones existentes escritas para APIs gráficas más antiguas necesiten adaptaciones.

Performance

- WebGPU:
  - Aprovecha las funcionalidades de las GPU modernas.
  - Ejecución de nuevos algoritmos directamente en GPU y no en JavaScript
  - Uso de Compute Shaders (Machine Learning)

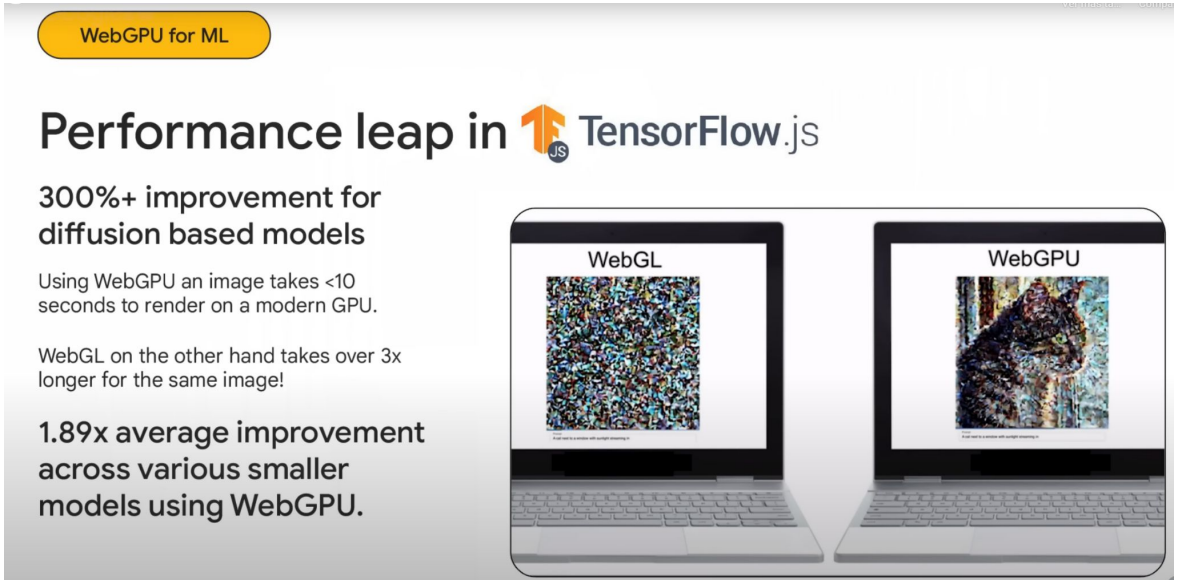
Compute Shaders

- WebGPU:
  - Disponibles de forma nativa
- OpenGL:
  - Disponibles a partir de la versión 4.3

Performance

Machine Learning

- Mejora de rendimiento en GPUs modernas

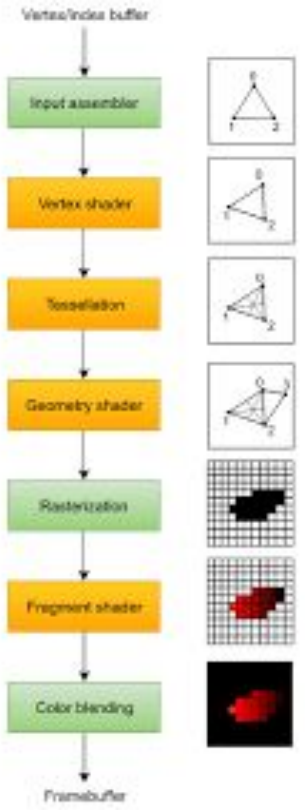


[WebGPU desbloquea el acceso a las GPU modernas para Javascript (io.google)]

Graphic pipeline

Graphic pipeline OpenGL

- Altamente configurable y programable.
- Input Assembler
- Vertex Shader
- Tessellation
- Geometry Shader
- Rasterization
- Fragment Shader
- Color Blending



Introduction - Vulkan Tutorial (vulkan-tutorial.com)

Graphic pipeline WebGPU

- Diseñado para aprovechar las características específicas de la arquitectura de hardware moderna y los estándares web.
- Mayor control sobre el acceso a la GPU.
- Diseñado para ser compatible con múltiples plataformas y dispositivos.
- Se integra con las políticas de seguridad y privacidad de la web moderna.

Primeros Pasos

1º INICIALIZACIÓN DE WebGPU:

- 1.1º Creación de un Canvas en HTML

```
<!doctype html>

<html>
  <head>
    <meta charset="utf-8">
    <title>WebGPU Life</title>
  </head>
  <body>
    <canvas width="512" height="512"></canvas>
    <script type="module">
      const canvas = document.querySelector("canvas");

      // Your WebGPU code will begin here!
    </script>
  </body>
</html>
```

- 1.2º Solicitud de dispositivo y adaptador

```
if (!navigator.gpu) {
  throw new Error("WebGPU not supported on this browser.");
}
```

```
const adapter = await navigator.gpu.requestAdapter();
if (!adapter) {
  throw new Error("No appropriate GPUAdapter found.");
}
```

```
const device = await adapter.requestDevice();
```

Primeros Pasos

- 1.3º Configuración del lienzo

```
const context = canvas.getContext("webgpu");
const canvasFormat = navigator.gpu.getPreferredCanvasFormat();
context.configure({
  device: device,
  format: canvasFormat,
});
```

- 1.4º Borrar lienzo usando un codificador

```
const encoder = device.createCommandEncoder();
```

```
const pass = encoder.beginRenderPass({
  colorAttachments: [{
     view: context.getCurrentTexture().createView(),
     loadOp: "clear",
     storeOp: "store",
  }]
});
```

Primeros Pasos

- 1.5º Finalizar renderización y “CommandBuffer”

```
pass.end();
```

```
const commandBuffer = encoder.finish();
```

```
device.queue.submit([commandBuffer]);
```

```
device.queue.submit([encoder.finish()]);
```

- 1.6º Elegir un color

```
const pass = encoder.beginRenderPass({
  colorAttachments: [{
    view: context.getCurrentTexture().createView(),
    loadOp: "clear",
    clearValue: { r: 0, g: 0, b: 0.4, a: 1 }, // New line
    storeOp: "store",
  }],
});
```

Primeros Pasos

2º Dibujar Geometría

- 2.1º Definir vértices

```
const vertices = new Float32Array([
//   X,    Y,
  -0.8, -0.8, // Triangle 1 (Blue)
   0.8, -0.8,
   0.8,  0.8,
```

- 2.2º Crear buffer de vértices y añadir los vértices

```
const vertexBuffer = device.createBuffer({
  label: "Cell vertices",
  size: vertices.byteLength,
  usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
```

```
device.queue.writeBuffer(vertexBuffer, /*bufferOffset=*/0, vertices);
```

Primeros Pasos

- 2.3º Diseño de vértices

```
const vertexBufferLayout = {
  arrayStride: 8,
  attributes: [{
    format: "float32x2",
    offset: 0,
    shaderLocation: 0, // Position, see vertex shader
  }],
};
```

- 2.4º Módulo de “Shaders” y canalización de renderizaciones

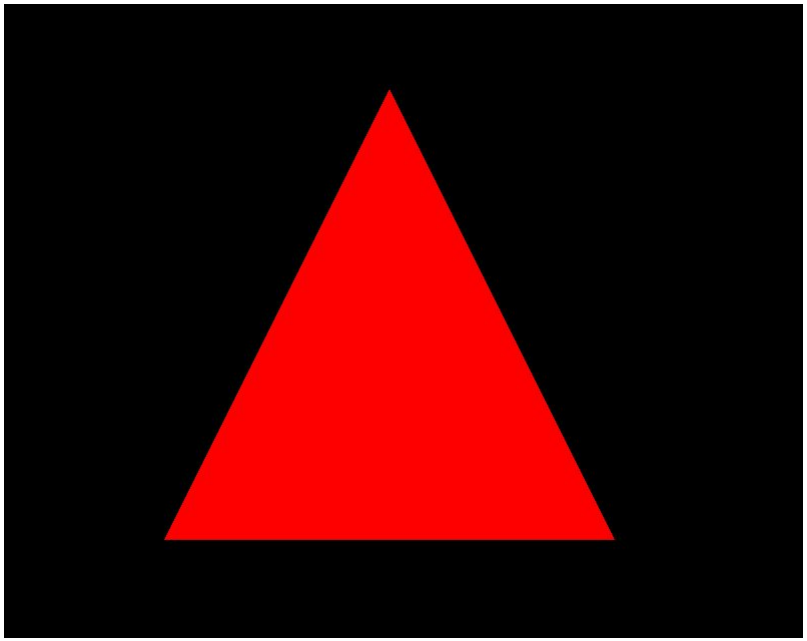
```
const cellShaderModule = device.createShaderModule({
  label: 'Cell shader',
  code: `
    @vertex
    fn vertexMain(@location(0) pos: vec2f) ->
      @builtin(position) vec4f {
      return vec4f(pos, 0, 1);
    }

    @fragment
    fn fragmentMain() -> @location(0) vec4f {
      return vec4f(1, 0, 0, 1);
    }
  `
});
```

```
const cellPipeline = device.createRenderPipeline({
  label: "Cell pipeline",
  layout: "auto",
  vertex: {
    module: cellShaderModule,
    entryPoint: "vertexMain",
    buffers: [vertexBufferLayout]
  },
  fragment: {
    module: cellShaderModule,
    entryPoint: "fragmentMain",
    targets: [{
      format: canvasFormat
    }]
  }
});
```

Primeros Pasos

- 2.5º Dibujar el triángulo en pantalla

```
pass.setPipeline(cellPipeline);
pass.setVertexBuffer(0, vertexBuffer);
pass.draw(vertices.length / 2); // 6 ve
```



Posibles problemas

- Se requiere una versión mínima (tanto de Android como de Chrome) para poder utilizar WebGPU en dispositivos móviles

- Error al solicitar el adaptador, configuración del lienzo o creación de buffers.

Posibles problemas

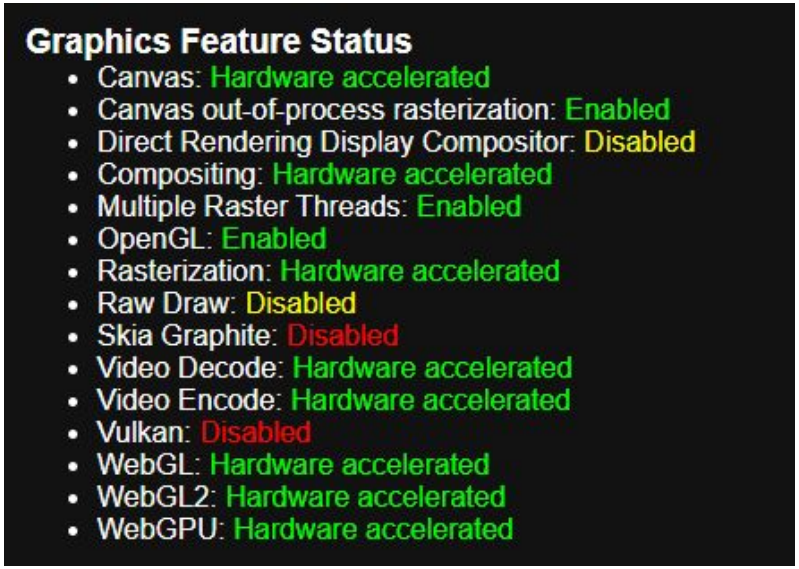
El adaptador de GPU es nulo

- Puedes saberlo cuando el adaptador que obtienes al llamar a `requestAdapter()` es nulo.
- Puede ocurrir porque:
  - WebGPU aún no es compatible con esta plataforma.
  - El hardware de la GPU se incluyó específicamente en la lista de entidades bloqueadas.
  - No hay ningún adaptador de GPU que coincida con las opciones que se pasaron en `requestAdapter()`.

Posibles problemas

Lentitud vs WebGL

- El uso de la aceleración por Hardware puede estar desactivado en el navegador. Se puede revisar en **chrome://gpu**



Posibles problemas

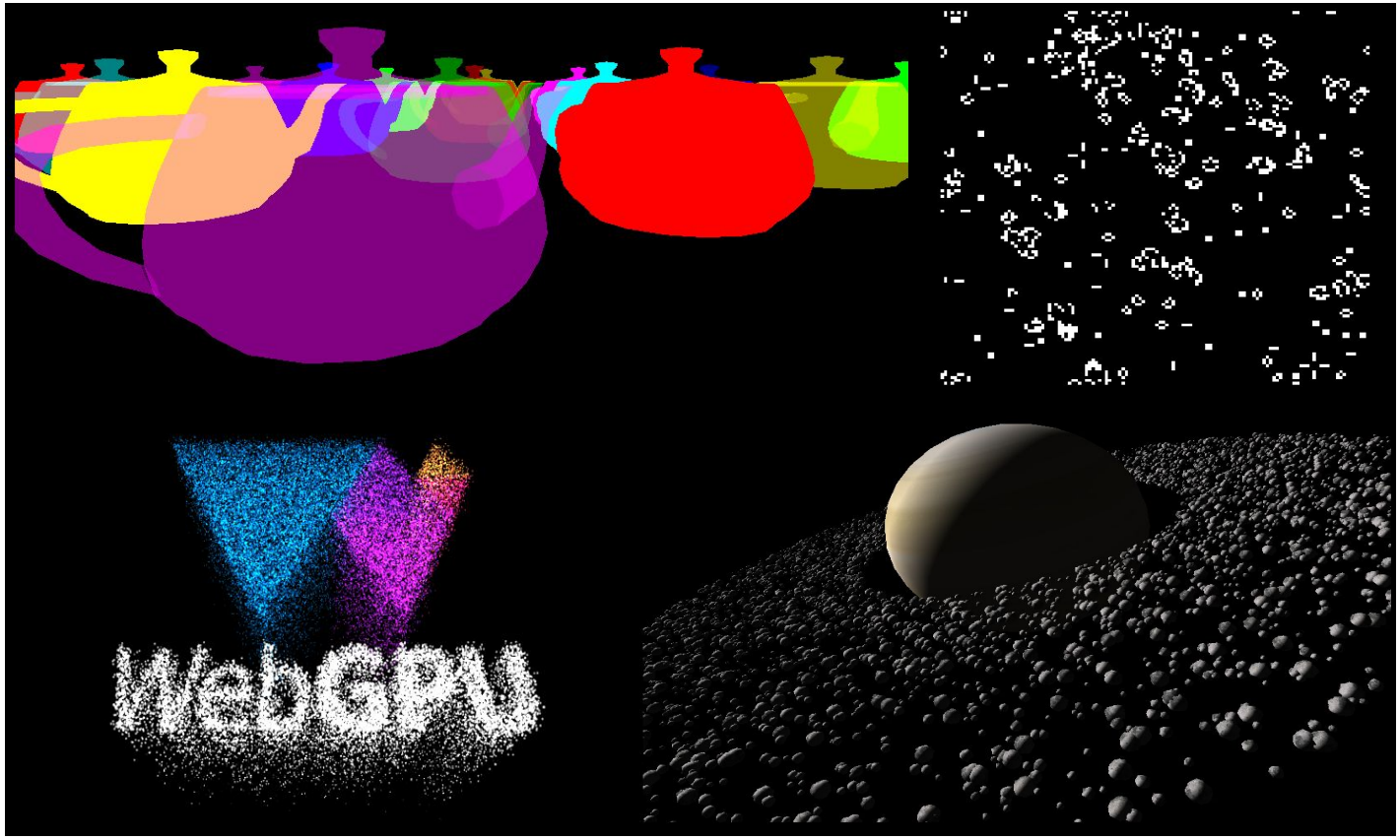
Solicitud de memoria fallida

- La solicitud de memoria a la GPU puede fallar y generar un error de **“out-of-memory”**.
- Este error se puede detectar de la siguiente manera.

```
async function tryCreateBuffer(device: GPUDevice, descriptor: GPUBufferDescriptor): Promise<GPUBuff
  device.pushErrorScope('out-of-memory');
  const buffer = device.createBuffer(descriptor);
  if (await device.popErrorScope() !== null) {
    return null;
  }
  return buffer;
}
```

Ejemplos

Hello Triangle - WebGPU Samples



Bibliografía

EUROPA PRESS, 2023. Google anuncia para Chrome la API WebGPU, que brinda gráficos 3D de alto rendimiento. *portaltic* [en línea]. [consulta: 18 febrero 2024]. Disponible en: https://www.europapress.es/portaltic/software/noticia-google-anuncia-chrome-api-webgpu-brinda-graficos-3d-alto-rendimiento-20230407134401.html.

OpenGL Insights. *Openglinsights.com* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://openglinsights.com/.

Reddit.com [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://www.reddit.com/r/rust/comments/136k95m/quiero_hablar_sobre_webgpu/es/.

Rendering pipeline overview. *Khronos.org* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview.

USHER, W., [sin fecha]. From 0 to glTF with WebGPU: The first triangle - updated for chrome 113 release. *Willusher.io* [en línea]. [consulta: 18 febrero 2024]. Disponible en: https://www.willusher.io/graphics/2023/04/10/0-to-gltf-triangle.

WIKIPEDIA CONTRIBUTORS, [sin fecha]. OpenGL ES. *Wikipedia, The Free Encyclopedia* [en línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=OpenGL_ES&oldid=155347632.

OpenGL ES. *Android Developers* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://developer.android.com/guide/topics/graphics/opengl?hl=es-419.

The multi explainer, [sin fecha]. S.l.: s.n. Disponible en: https://github.com/gpuweb/gpuweb/wiki/The-Multi-Explainer

WebGPU shading language. *Github.io* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://gpuweb.github.io/gpuweb/wgsl/.

Bibliografía

SEGUIN, D., 2021. From GLSL to WGSL: the future of shaders on the Web. *Dmnsgn.me* [en línea]. [consulta: 18 febrero 2024]. Disponible en: https://dmnsgn.me/blog/from-glsl-to-wgsl-the-future-of-shaders-on-the-web/.

WebGPU desbloquea el acceso a las GPU modernas para Javascript. *.Io.google* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://io.google/2023/program/0da196f5-5169-43ff-91db-8762e2c424a2/intl/es/.

LearnOpenGL - introduction. *Learnopengl.com* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://learnopengl.com/Guest-Articles/2022/Compute-Shaders/Introduction.

Deferred Rendering - WebGPU Samples. *Github.io* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://webgpu.github.io/webgpu-samples/samples/deferredRendering.

OpenGL and multithreading. *Khronos.org* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://www.khronos.org/opengl/wiki/OpenGL_and_multithreading.

Steamworks SDK (steamworks documentation). *Steamgames.com* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://partner.steamgames.com/doc/sdk.

Web API overview (steamworks documentation). *Steamgames.com* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://partner.steamgames.com/doc/webapi_overview.

UNITY TECHNOLOGIES, [sin fecha]. Configuración del SDK de android. *Unity3d.com* [en línea]. [consulta: 18 febrero 2024]. Disponible en: https://docs.unity3d.com/es/530/Manual/android-sdksetup.html.

Shader compilation. *Khronos.org* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://www.khronos.org/opengl/wiki/Shader_Compilation.

Bibliografía

WebGPU. *Www.w3.org* [en línea], [sin fecha]. [consulta: 18 febrero 2024]. Disponible en: https://www.w3.org/TR/webgpu/.

WebGPU explainer. (n.d.). Github.Io. Retrieved February 25, 2024, from https://gpuweb.github.io/gpuweb/explainer/.

GitHub, F. B. (2024, February 7). *WebGPU: Sugerencias y soluciones para la solución de problemas*. Chrome for Developers. https://developer.chrome.com/docs/web-platform/webgpu/troubleshooting-tips?hl=es-419

www.esat.es