

***Documentación de Proyecto***

*Tópicos de Física*

Aileen Balderrama

Raúl Campbell

Fátima Millanes

Alexia Ruiz

Cd. Obregón, Sonora, 25 de abril de 2020

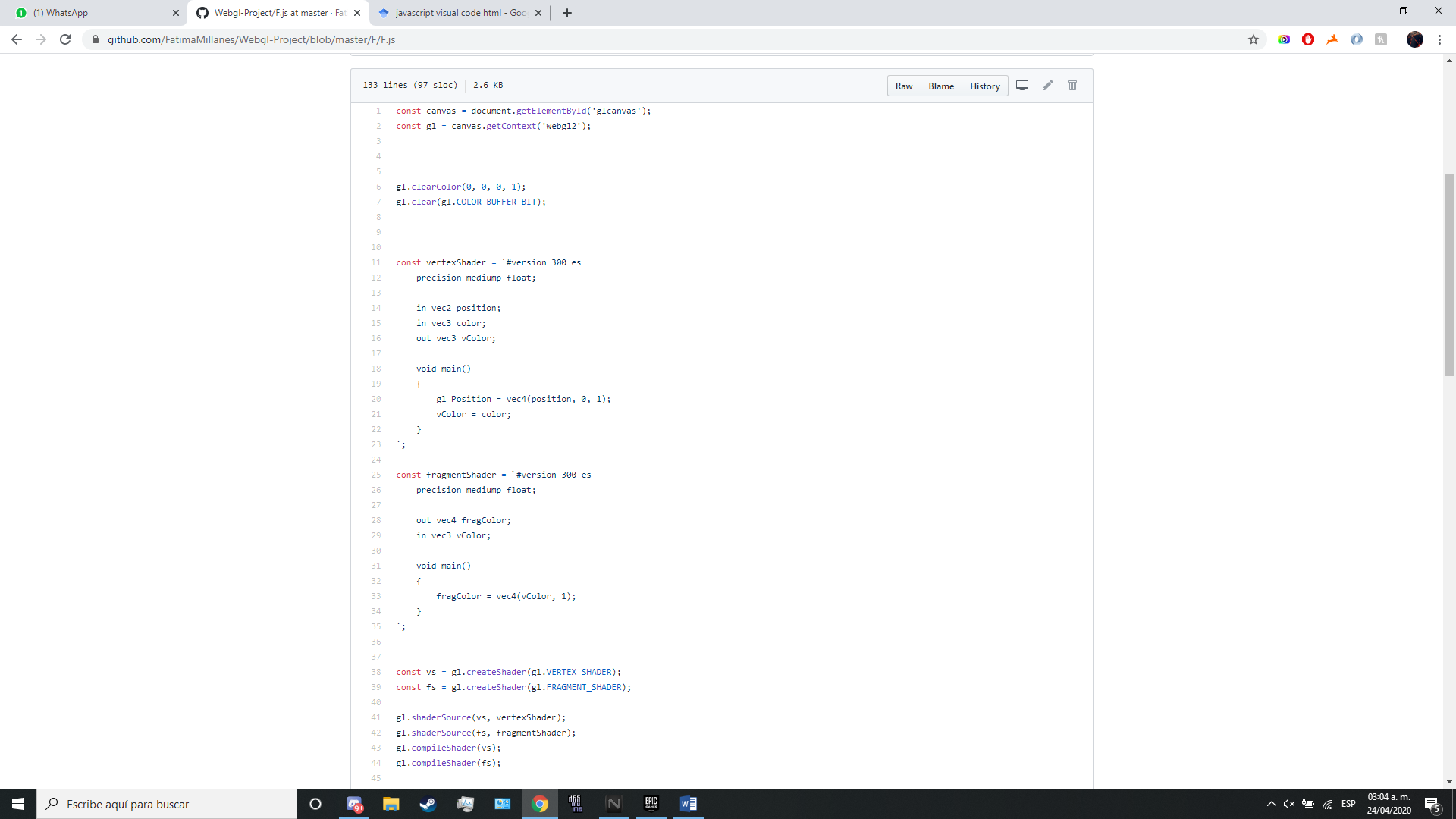
**Documentación**

***Nota: Todas las imágenes aquí presentadas son evidencias del proyecto.***

**Que es WebGL**

WebGL (Web Graphics Library) es el nuevo estándar para gráficos 3D en la Web, está diseñado para representar gráficos 2D y gráficos 3D interactivos. Se deriva de la biblioteca ES 2.0 de OpenGL, que es una API 3D de bajo nivel para teléfonos y otros dispositivos móviles. WebGL proporciona una funcionalidad similar de ES 2.0 (sistemas integrados) y funciona bien en hardware moderno de gráficos 3D.

Es una API de JavaScript que se puede usar con HTML5. El código WebGL está escrito dentro de la etiqueta <canvas> de HTML5. Es una especificación que permite a los navegadores de Internet acceder a las unidades de procesamiento gráfico (GPU) en las computadoras donde se usaron.



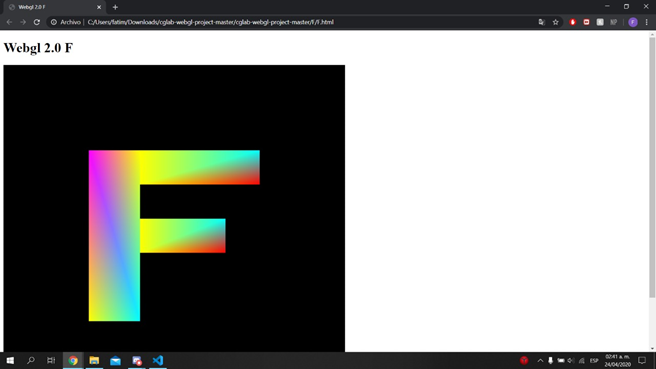
WebGL maneja la geometría de manera estándar, independientemente de la complejidad y el número de puntos que pueden tener las superficies. Hay dos tipos de datos que son fundamentales para representar la geometría de cualquier objeto 3D: vértices e índices.

**Como funciona WebGl**

Al igual que cualquier otro sistema 3D, tendrá ejes x, y y z en WebGL, donde el eje z significa profundidad. Las coordenadas en WebGL están restringidas a (1, 1, 1) y (-1, -1, - 1). Significa que si considera que la pantalla proyecta gráficos WebGL como un cubo, una esquina del cubo será (1, 1, 1) y la esquina opuesta será (-1, -1, -1). WebGL no mostrará nada que se dibuje más allá de estos límites

Un valor positivo de z indica que el objeto está cerca de la pantalla / visor, mientras que un valor negativo de z indica que el objeto está lejos de la pantalla. Del mismo modo, un valor positivo de x indica que el objeto está en el lado derecho de la pantalla y un valor negativo indica que el objeto está en el lado izquierdo. Del mismo modo, los valores positivos y negativos de y indican si el objeto está en la parte superior o inferior de la pantalla.

Para dibujar objetos 2D o 3D, la API de WebGL proporciona dos métodos, a saber, drawArrays () y drawElements (). Estos dos métodos aceptan un parámetro llamado modo mediante el cual puede seleccionar el objeto que desea dibujar. Las opciones proporcionadas por este campo están restringidas a puntos, líneas y triángulos.



**Ventajas de WebGl**

*Programación de JavaScript:* las aplicaciones WebGL están escritas en JavaScript. Con estas aplicaciones, puede interactuar directamente con otros elementos del documento HTML. También puede usar otras bibliotecas de JavaScript (por ejemplo, JQuery) y tecnologías HTML para enriquecer la aplicación WebGL.

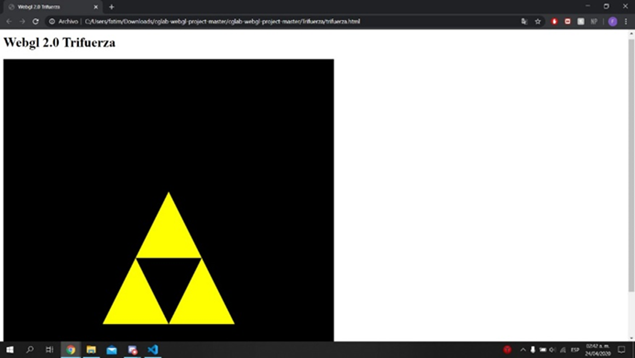
*Mayor compatibilidad con navegadores móviles:* WebGL también es compatible con navegadores móviles como iOS Safari, Android Browser y Chrome para Android.

*Código abierto:* WebGL es un código abierto. Puede acceder al código fuente de la biblioteca y comprender cómo funciona y cómo se desarrolló.

*No es necesario compilar*: JavaScript es un componente mitad programación y mitad HTML. Para ejecutar este script, no es necesario compilar el archivo. En su lugar, puede abrir directamente el archivo utilizando cualquiera de los navegadores y verificar el resultado. Dado que las aplicaciones WebGL se desarrollan utilizando JavaScript, no hay necesidad de compilar aplicaciones WebGL también.

*Administración automática de memoria:* JavaScript admite la administración automática de memoria. No hay necesidad de asignación manual de memoria. WebGL hereda esta característica de JavaScript.

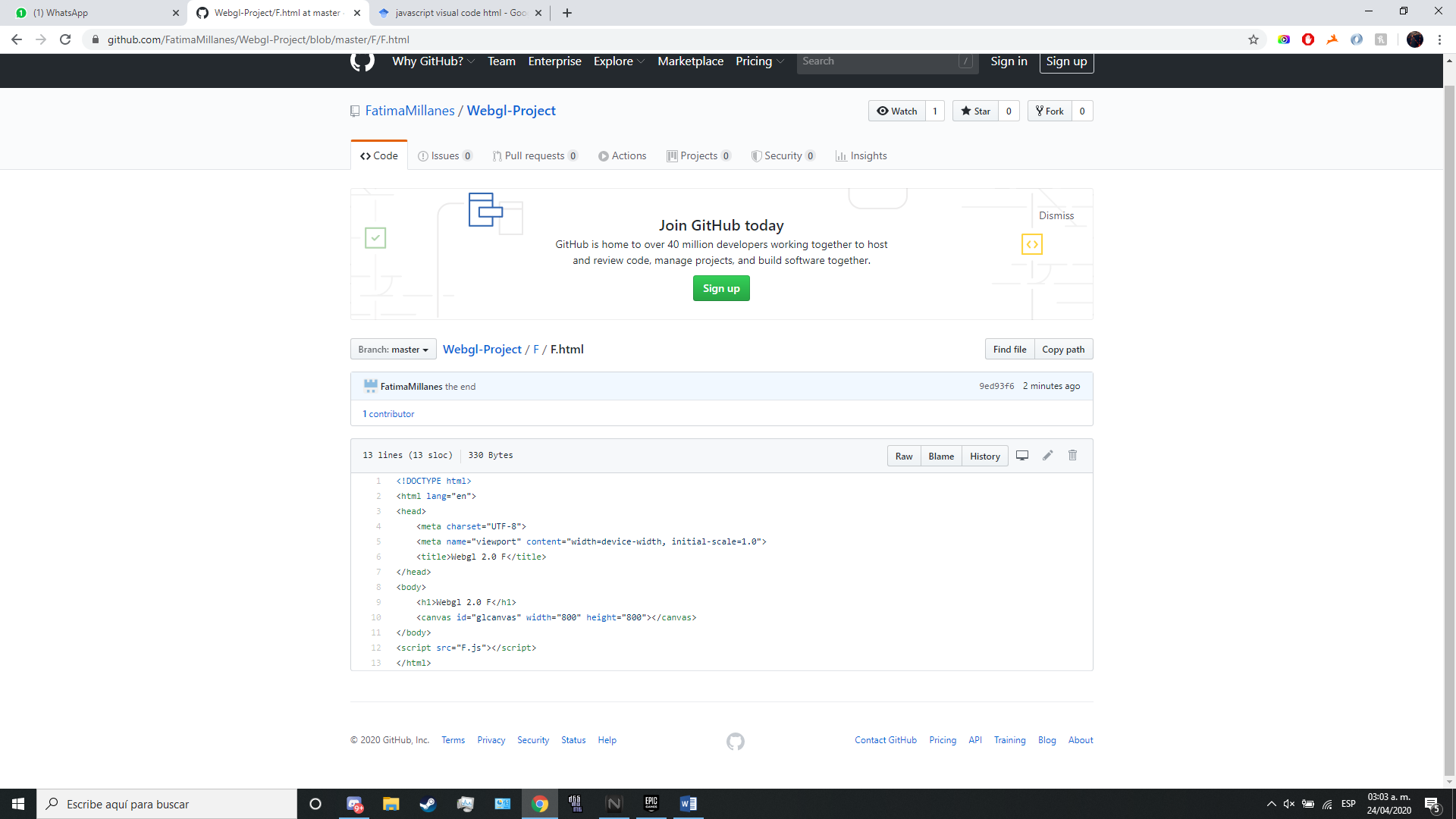
*Fácil de configurar*: dado que WebGL está integrado en HTML 5, no hay necesidad de una configuración adicional. Para escribir una aplicación WebGL, todo lo que necesita es un editor de texto y un navegador web.



**Vertices**

Los vértices son los puntos que definen las esquinas de los objetos 3D. Cada vértice está representado por tres números de punto flotante que corresponden a las coordenadas x, y y z del vértice. A diferencia de OpenGL, WebGL no proporciona métodos API para pasar vértices independientes a la canalización de representación, por lo tanto, necesitamos escribir todos nuestros vértices en una matriz de JavaScript y luego construir un búfer de vértices WebGL con él.

Los objetos de almacenamiento intermedio del vertice almacenan datos sobre los vértices, mientras que los objetos de almacenamiento intermedio de índice almacenan datos sobre los índices. Después de almacenar los vértices en matrices, los pasamos a la canalización de gráficos WegGL usando estos objetos Buffer.



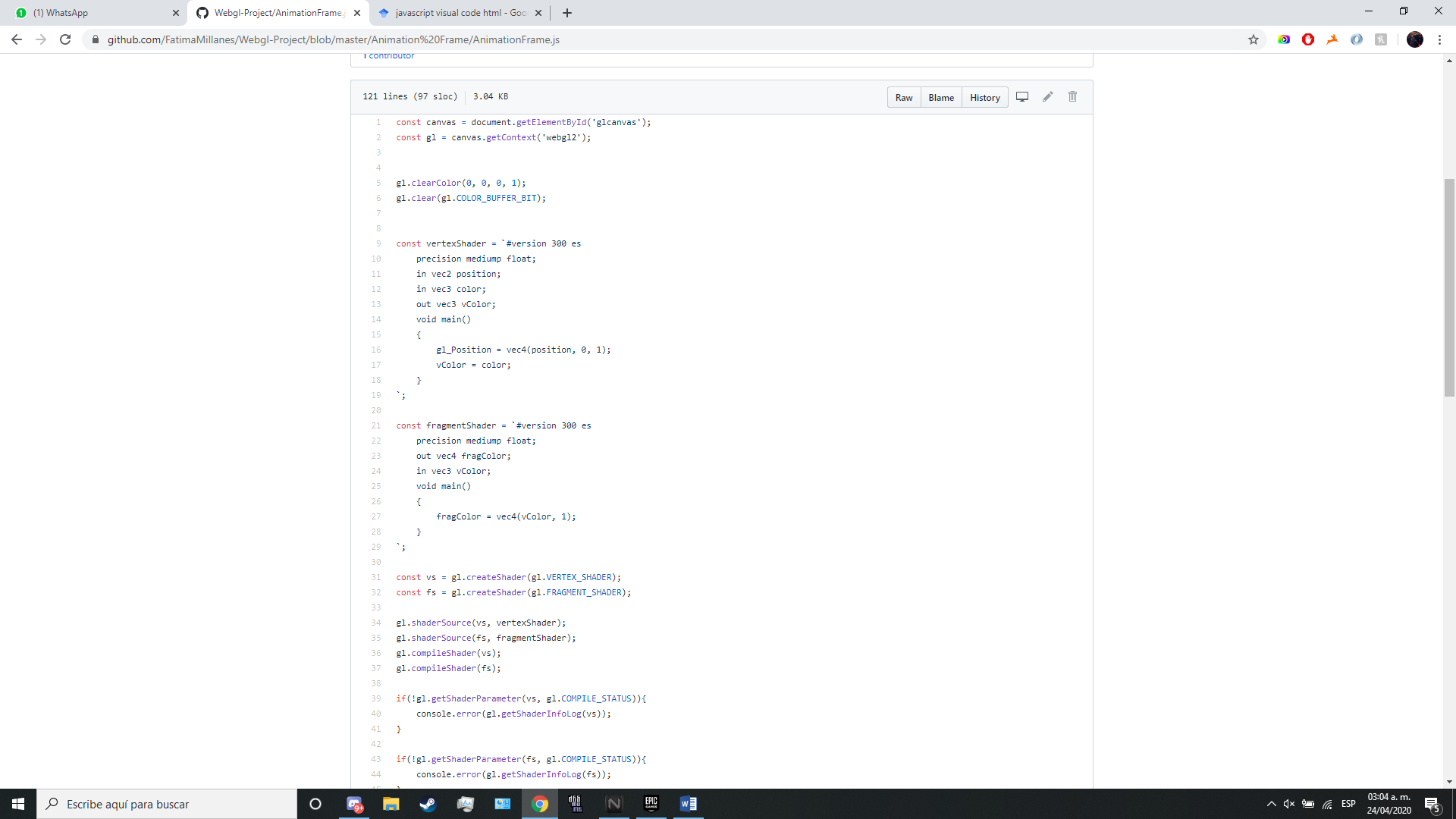
**Buffers**

Los buffers son las áreas de memoria de WebGL que contienen los datos. Hay varios búferes, a saber, búfer de dibujo, búfer de cuadro, búfer vetex y búfer de índice. El búfer de vértices y el búfer de índice se utilizan para describir y procesar la geometría del modelo.

Tradicionalmente, las aplicaciones WebGL ponen datos de geometría en buffers. Luego usan atributos para suministrar automáticamente datos de vértices de esos búferes al sombreador de vértices donde el programador proporciona código para convertirlos en espacio de recorte.

La palabra tradicionalmente es importante. Es solo una tradición hacerlo de esta manera. De ninguna manera es un requisito. A WebGL no le importa cómo lo hacemos, solo le importa que nuestros sombreadores de vértices asignen coordenadas de espacio de clip a gl\_Position.

Por ejemplo, el búfer de cuadros es una parte de la memoria gráfica que contiene los datos de la escena. Este búfer contiene detalles como ancho y alto de la superficie (en píxeles), color de cada píxel, profundidad y búferes de plantilla.



**Vertex Shaders**

Mientras tanto, el vertex shader es el código del programa llamado en cada vértice. Se utiliza para transformar (mover) la geometría (ej .: triángulo) de un lugar a otro. Maneja los datos de cada vértice (datos por vértice) como coordenadas de vértice, normales, colores y coordenadas de textura.

En el código ES GL de vertex shader, los programadores tienen que definir atributos para manejar los datos. Estos atributos apuntan a un objeto búfer de vértices escrito en JavaScript.

Las siguientes tareas se pueden realizar utilizando sombreadores de vértices:

-Transformación del vértice.

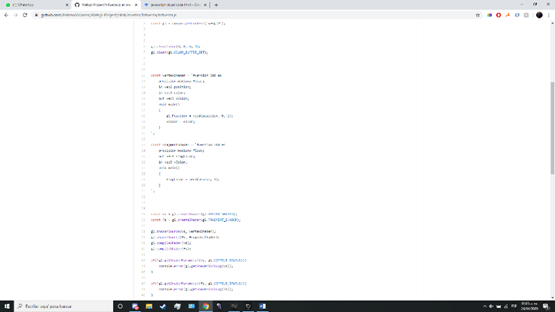
-Transformación normal y normalización.

-Generación de coordenadas de textura.

-Transformación de coordenadas de textura.

-Encendiendo.

-Aplicación de material de color.



**Bibliografía**

**Cimas Bravo, Ó. (2017). Editor y previsualizador de WebGL.**

**Sanz Asenjo, A. (2016). Tutorial web para el aprendizaje de WebGL.**

**Hernández Molpeceres, V. (2015). Elaboración de una librería sobre JavaScript para el desarrollo de juegos 2D compatible con XNA (Bachelor's thesis).**

**Rey, D., Ignacio, F., & Leonel, R. (2019). Representación gráfica interactiva de funciones matemáticas, figuras geométricas y animaciones en la Web: Proyecto MateFun.**

**Fraile de la Cruz, I. (2016). Videojuego de construcción y conducción sobre “Scalextric.**

**Hernández-Castro, F., & Monge-Fallas, J. (2018). Eficiencia comparativa en animaciones en javascript (nota técnica). Revista Tecnología en Marcha, ág-143.**

**Colomina Pardo, O., & Arques Corrales, P. (2011). Tema 3, parte 5: Animaciones y estilos dinámicos en JavaScript. Tecnologías Web.**

**Colomina Pardo, O., Arques Corrales, P., & Montoyo-Bojo, J. (2011). Tema 4: Frameworks JavaScript. jQuery. Tecnologías Web.**

**Vértice, E. (2009). Técnicas avanzadas de diseño web. Editorial Vértice.**

**García Salvadores, A. (2017). Desarrollo de un motor modular para desarrollo de videojuegos 2D (Bachelor's thesis).**

**Jaimez-González, C., & Vargas-Rodríguez, R. (2017). Editor web visual para HTML, CSS y JavaScript de apoyo a la docencia. Virtualidad, Educación y Ciencia, 8(14), 136-152.**

**en Red, A. D. S. I. Introducción al lenguaje HTML.**

**Hernández-Castro, F., & Monge-Fallas, J. (2018). Eficiencia comparativa en animaciones en javascript (nota técnica). Revista Tecnología en Marcha, ág-143.**

**Navarrete, T. (2006). El lenguaje JavaScript. Argentina.**

**Rodríguez, J. A. (2012). Manual de javascript.**