

**Proyecto Parcial, Segundo Parcial**

**Integrantes:** Jennifer Elizabeth Yáñez Duran - 189117

Carlos German Encinas Lizárraga - 189229

Bryan Alexis Mora Palafox – 189039

Karen Daniela López Bañuelos – 189368

Jesús Eduardo Gastélum Pedroza – 189289

Andrés Ramon Reyes Galván - 189236

**Profesor**. Jesús Francisco Caro

Viernes 24 de abril del 2020, CD. OBREGON SON.

El proyecto parcial de la materia de Tópicos de Física es en base a lo que es WebGL. Antes que nada, hablemos sobre que es, WebGL es una abreviatura a Web Graphics Libray la cual nos sirve para dar una especificación estándar que define una API implementada en JavaScript para la renderización de gráficos en 3D dentro de cualquier navegador web. Esta misma integra completamente sobre todos los estándares web del navegador, permitiendo la aceleración hardware(física) de la GPU y el procesamiento de imágenes y efecto como parte misma del canvas de la web, también se ha de mencionar que todos los elementos de WebGL se pueden combinar con otros elementos HTML y estar compuestos con otras partes de la página o del background. Su forma de funcionar es a base de un código de control escrito en JavaScript la cual nos permite que podamos utilizar la implementación nativa de OpenGL cuya es 2.0, con un lenguaje similar a C o C++, la creación, diseño y la gestión de WebGL es sin ánimos de lucro proveniente de Khronos Grupo.

Para comenzar hablemos de lo básico sobre el contexto de GL, primero que nada, al inicio del proyecto lo que se debe lograr es lograr aprender a encontrar y lograr comunicarse con nuestro contexto pues con ello podemos traer lo que sería el graphics library en un navegador. Para lograrlo debe hacerse a través de un canvas con un contexto de WebGL 2.

Sobre todo, se necesita un JavaScript ya que es quien va a controlar todo lo va a ser nuestra aplicación de WebGL, también está en que siempre en la escritura base se puso lo que fue el enlace del JavaScript a la página principal con la que se correrá el trabajo realizado y va por el otro lado igual, se debe enlazar el JS a lo que es la principal. Para que podamos declarar el funcionamiento básico de la herramienta, es dentro del JS se debe declarar que el “gl” debe ser igual al canvas ya vinculado desde la principal y que deba buscar el tipo de contexto ya dicho el cual es “WebGL2”, una vez ya declarado el funcionamiento básico se debe declarar con que color va estar haciendo el refresh a cada uno de los frames y para eso se debe mandar primero un clear color y después de eso limpiar el Buffer de los bit de color previstos por los frames. El clear Color debe tener como pinte un negro y es más que nada por gusto, para ello se necesita darle la ausencia de color la cual sería representándola en la librería como un 0 0 0 a excepción del alfa que se tendría que dejar en 1 para que la transparencia no sea completamente transparente y para que en cada pantalla logre quitar la pintada cada vez que vuelta a usar el tinte se debe declarar el Buffer Color Bit para retomar nuevamente la nueva pintada, una vez lograda la acción podremos iniciarla página principal vinculada al JS para poder visualizar el canvas dentro de la página, con ello listo se pudo iniciar lo que es el proyecto de las demás partes.

Hay que tomar en cuenta que con lo logrado anteriormente no solo estamos alistando la zona de trabajo para continuar con los demás puntos, sino que también se está demostrando el cómo es que funciona el llamado de la aceleración por computadora de los gráficos, dando de vista como con la solicitud que damos, nos da de respuesta la devuelta del canvas solicitada.

Por segunda lo que debemos hacer es compilar un shader. Primero que nada los shader son una parte de la computadora que le indican a una geometría o a todo un conjunto de vértices que forman una geometría y en cómo se va a comportar la computadora en la manera de renderizarlo, en la forma que toma los vértices y con ello forma lo que es un frame, o también podemos ponerlo de manera básica y simple, el cómo toma una captura y lo pasa a la pantalla y como va suceder eso, esa es la definición rápida de los shaders, dentro del trabajo no podremos visualizarlo sin estos, ya que como todo, debemos de tener una textura para lo demás que haremos, y para ello tenemos que declararlo y luego comenzar su proceso de validación y para ello debemos crear otra página y otro JS para el puro shader y la vinculación funciona de la misma manera a como fue con la primera pero solo cambiando el contexto, primero que nada debemos declarar el shader, el objetivo que se tiene en mente es la realización de un triángulo y la razón es porque en la geometría el triángulo es la geometría mínima necesaria que existe en el mundo de la computación, necesita tres vértices para poder formar una geometría.

Por lo que para dar iniciación a la creación primero debemos declarar que utilizaremos el WebGL pero en si se utiliza el OpenGL ES y esta es más que nada una versión especial del render para que se pueda trabajar con todo tipo de procesadores de bajo recurso y al estar totalmente optimizado lo vuelve algo más complicado de poder trabajar con él, y esto lo hacemos debido a que nunca se sabe con qué equipo se va a trabajar y es mejor tenerlo en lo más bajo. También trabajamos con lo que es el vertex shader ya que este se encarga de posicionar los vértices de un conjunto de vértices para poder formar lo que sería una geometría, una vez agregado el vertex podemos pasar al fragment que una vez puesto los vértices este es el que les da color a los vértices. Hay que tomar en cuenta que solo se trabaja con un vector de dos componentes los cuales son X y Y, junto con esto debemos declarar nuestra función principal del shader ya que si no lo hacemos el shader no tendrá hacia donde accesar para poder ejecutar su contenido. El objetivo que tiene esto es generar un Script de salga de una mezcla entre el vertex y el fragment, con estos ambos haremos lo que sería un in and out donde la entrada iría del vertex y la otra seria en el fragment de donde también se recibiría como una salida, por lo que en el fragment colocaríamos la salida que sería nuestra variable out y esta misma sería un vector 4 por que el color es RGBA abreviado de Red, Green, Blue y Alpha, este mismo lo pondríamos neutral, es decir blanco por lo que lo colocamos cono 1, 1, 1, 1. Cuando trabajamos con esto debemos también declarar la precisión de nuestro proyecto, entre mayor precisión tenga el trabajo más requerimientos pedirá y entre menos menos, por lo que en este estaremos trabajando a precisión medio, todo esto deberemos meterlo en un source code donde necesitaremos dos cosas, una el shader vacío y la localización del source que ligaremos con el shader vacío, ambos deben compilarse hacia vertex shader. Una vez conseguido se tuvo que generar el proyecto donde se deben conjugar al fragment y al vertex, en si debemos tachar el shader al programa, cuando lo ligamos no podemos dejarlo así, sino más bien se debe comprobar la compilación para que no se encuentre un error ahí por lo que realizamos un Link status y que nos mande un error y la razón del porque cada vez que evite el inicio del shader, si no se marcó ningún error en el inicio del proceso significa que ya se puede empezar a usar el shader.

En tercera tenemos el paso para utilizar nuestro shader y visualizarlo con geometría, para comenzar lo que se hizo fue realizar un arreglo la cual nos trae las posiciones en las que se va a dibujar el triángulo, principalmente colocando las coordenadas de la geometría en el WebGL. Cuando se crea el array se necesita crear un buffer vacío para poder reservar un espacio en memoria para poder traerse una variable del shader, aparte de eso se debe substraer la información al shader, es decir enviarle data al shader para que así el buffer trabaje con eso mismo. Para finalizar se puso en plano el modo en el que utilizarían los array, por cual va comenzar y el total de vértices que hay para que al correrlo nos del triángulo que buscamos, ya con eso llegamos al final principal buscado que es la realización del triángulo, pero podemos generar otro tipo de figuras a través del mismo triangulo como un cuadrado con la función de trianglestrip donde agregamos más, pudiendo agregar más vértices en la geometría pero hay muchas maneras de lograrlo

Por último, tenemos la posibilidad de hacer aún más y en este caso es llegar hacer más figuras e incluso llegar hacer un círculo, para ello se necesita lo que es un triangle fan y lo que hace es conectar vértices, pero lo hace agarrando el punto del centro como un pivote, es decir dónde van a conectarse los demás vértices, debido a que funciona como punto de intersecciones.

Para lograr el circulo lo que se necesita es mandarle lo que se va a tener por dentro y debido a que ese círculo esta subdividido por los 360° se debe subdividir en el número de lado que se necesitan que tenga, puede variar dependiendo lo fino que se quiera obtener el circulo, pero usualmente se necesitan alrededor de 28. Al igual que las demás figuras se necesita que se mande directamente toda la información necesaria para poder formarlo, al tratarse de un círculo y ya no solo de una figura formada por vértices, implica que tengamos que agregarle más funciones de lo normal es decir que necesitamos la información del circulo, los arreglos como el radio, sus vértices, las coordenadas y el color, cuando se tiene definido como será el cirulo deberá llenarse con matemática o en otras palabras la matemática que se ocupa para realizar el circulo, lo principal que debe calcular es la circunferencia debido a que es el ángulo en el que se está moviendo pues se puede ir atrapando el área en el que se va avanzar en las posiciones que estarán en el círculo, es decir 28.

Para poder formular la circunferencia del circulo necesitaremos sacar que la constante es igual a dos veces el PI por el número que va sobre el número total de vértices, para así poder obtener el área inicial pues al tratarse de un círculo nos movemos de forma circular y hay que encontrar el inicio el cual es un puntito en el área y para ello se necesita la X y la Y o en otras palabras sacar la X siendo coseno y Y siendo seno. Una vez sacadas se debe insertar los arreglos a la información del circulo sacada anteriormente, una vez insertadas nos queda enviar los datos sacado del circulo, mandando el primer vértice como el ancla principal y todos los demás vértices dándonos como resultado el circulo.

Se pueden hacer cualquier tipo de objeto o en si geometría de muchas maneras y cada una dependerá de la manera en que se decida hacerse, hay veces que hay maneras de hacer una y poder hacer otra de la misma manera en que se logró inicialmente y hay otros casos en el que se tiene que iniciar nuevamente o cambiar muchas cosas pues hay veces que se puede hacer corto pero con el sacrificio de no poder transfórmalo después, todo dependerá de quien esté haciéndolo.

En el caso de dibujar la letra F con triángulos se puede hacer en maneras diferentes pero una de ellas y la más fácil es realizando varios triangulo de diferentes formas y jugando con la localización de cada una hasta poder encontrar la posición exacta de cada uno hasta poder formar la F, para ello se necesitarían vincular muchos vértices a bases diferentes para poder formar triángulos diferentes y que cada uno tenga su propia regla donde no interponga con uno y ningún triangulo sea considerado como principal y no acapare a los demás, teniendo siempre en cuenta las reglas principales y las funciones principales necesarias para lograse.

En el caso de animar colores en los vértices primero que nada debemos aplicar color en ellos, en el caso cada vértice posee una posición y un color, todos son procesados utilizando una interpolación, creando automáticamente gradientes suaves, donde el shader del vértice tiene un blanco solido a cada pixel, así que supongamos que queremos desplegar un gradiente a cada una de las esquinas, por lo que primero se debe establecer el color de las cuatro vértices por lo que se debe crear un arreglo de vértices de colores. Para poder designar los colores a utilizar el vertex shader necesita ser actualizado para extraer color apropiado del buffer de color.

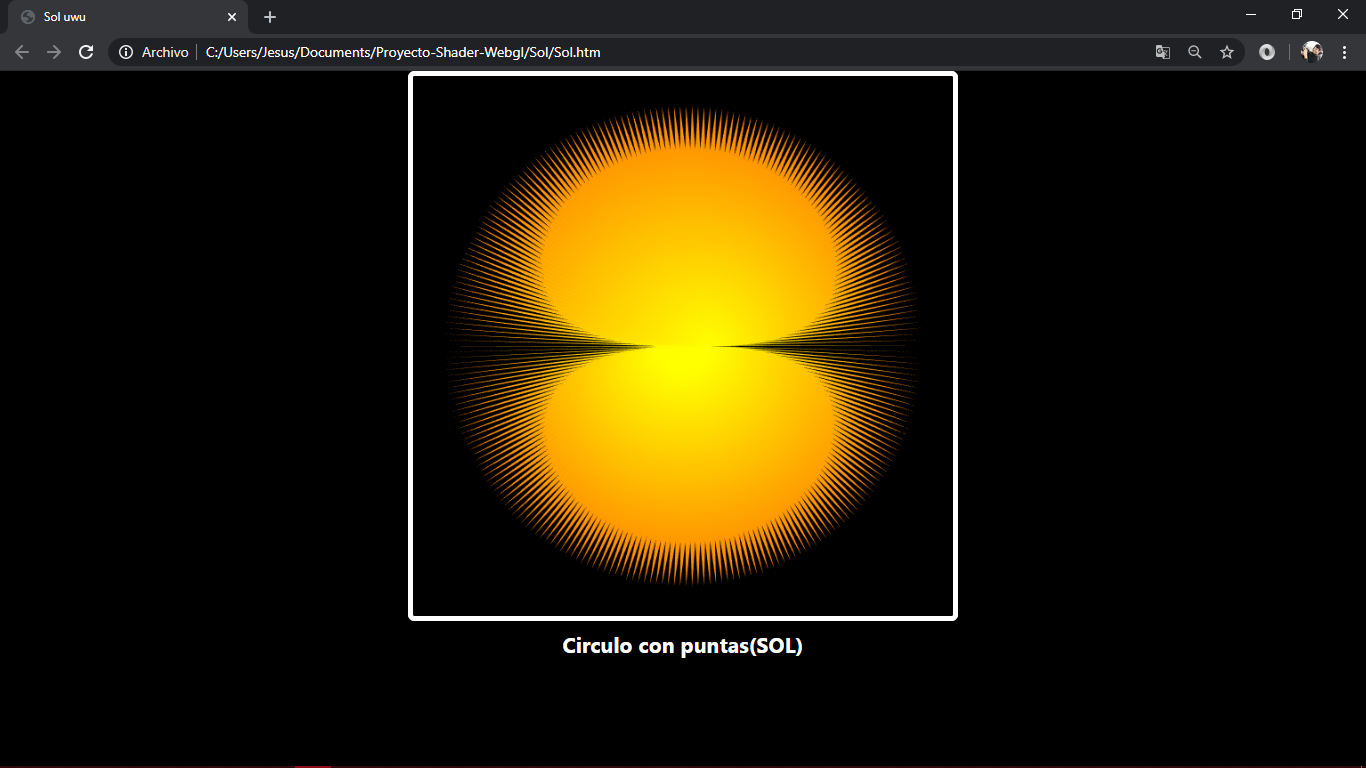
La diferencia principal en esto es que, por cada vértice, se tenga que establecer el color correspondiente desde el arreglo del color, según se vaya tomando el color para cada pixel, simplemente se necesita cambiar, cada fragmento simplemente recibe el color interpolado basado en su posición relativa a los vértices, en lugar de un valor establecido-

Para lograr animar un color debemos cambiar las partículas en cada frame por cada segundo o bien se puede poner los colores en rotación, pero serian por skip en vez de por frame por lo que el cambio sería más brusco y lento.

Para el símbolo de la trifuerza se puede utilizar la regla de triple en fondo, donde utilizamos que todo el fondo represente a dos triángulos que se van repitiendo de forma consecutiva dando como resultado un efecto panal donde se vea la tri fuerza, o bien podemos crear tres triángulos con los mismos datos, pero colocados en diferentes posiciones que nos den como resultado la trifuerza.

En el caso del circulo con forma de sol, teniendo un círculo como base podemos agregar una fórmula de seno en el mismo circulo u otro que sobresalga para que en la orilla de este se vea las subidas y bajadas, mientras más corta sea la unión más finas serían las puntas, o bien podemos utilizar el efecto diente el cual sería más sencillo pero mucho más tardado y largo pues lo que se haría seria que cada una de las puntas tenga su propio dato o en otras palabras hacer que cada una de las puntas tenga sus propios datos y hacerlas una por una buscando también la posición correcta en toda el área.











**Referencias:**

**MDN (2019). “Utilizar los shaders para aplicar color en WebGL”. Recuperado el 24 de abril del 2020. Página web:** [**https://developer.mozilla.org/es/docs/Web/API/WebGL\_API/Tutorial/Using\_shaders\_to\_apply\_color\_in\_WebGL**](https://developer.mozilla.org/es/docs/Web/API/WebGL_API/Tutorial/Using_shaders_to_apply_color_in_WebGL)

**Vahid (2019). “Change the color in WebGL + JavaScript”. Recuperado el 24 de abril del 2020. Página web:** [**https://stackoverflow.com/questions/53571798/change-the-color-in-webgl-javascript**](https://stackoverflow.com/questions/53571798/change-the-color-in-webgl-javascript)

**MDN (2019). “Animating textures in WebGL” Recuperado el 24 de abril del 2020. Página web:** [**https://developer.mozilla.org/es/docs/Web/API/WebGL\_API/Tutorial/Animating\_textures\_in\_WebGL**](https://developer.mozilla.org/es/docs/Web/API/WebGL_API/Tutorial/Animating_textures_in_WebGL)

**WebGl (2010). “Añadir color”. Recuperado el 24 de abril del 2020. Página web:** [**https://sites.google.com/site/desarrolloenwebgl/tutorial-1/3-colores**](https://sites.google.com/site/desarrolloenwebgl/tutorial-1/3-colores)

**WebGl (2010). “Primer programa”. Recuperado el 24 de abril del 2020. Página Web:** [**https://sites.google.com/site/desarrolloenwebgl/tutorial-1/2---primer-programa**](https://sites.google.com/site/desarrolloenwebgl/tutorial-1/2---primer-programa)

**Unity (2018). “Empezar con el desarrollo de WebGL”. Recuperado el 24 de abril del 2020. Página Web:** [**https://docs.unity3d.com/es/current/Manual/webgl-gettingstarted.html**](https://docs.unity3d.com/es/current/Manual/webgl-gettingstarted.html)

**TutorialsPoint (2020). “WebGl-Color”. Recuperado el 24 de abril del 2020. Página Web:** [**https://www.tutorialspoint.com/webgl/webgl\_colors.htm**](https://www.tutorialspoint.com/webgl/webgl_colors.htm)

**Ethane (2017). “WebGL-change colours for between objects”. Recuperado el 24 de abril del 2020. Página web:** [**https://stackoverflow.com/questions/42325472/webgl-change-colours-for-between-objects**](https://stackoverflow.com/questions/42325472/webgl-change-colours-for-between-objects)

**Keith (NT). “Build complex 3D models with WebGL”. Recuperado el 24 de abril del 2020. Página web:** [**https://egghead.io/lessons/webgl-apply-varying-colors-per-vertex-to-webgl-triangles**](https://egghead.io/lessons/webgl-apply-varying-colors-per-vertex-to-webgl-triangles)

**Reebs (2019). “How to create different color points in WebGL”. Recuperado el 24 abril del 2020. Página web:** [**https://www.reddit.com/r/opengl/comments/9qqe7z/how\_to\_create\_different\_color\_points\_in\_webgl/**](https://www.reddit.com/r/opengl/comments/9qqe7z/how_to_create_different_color_points_in_webgl/)

**Math (NT) “First Examples”. Recuperado el 24 de abril del 2020. Página web:** [**http://math.hws.edu/graphicsbook/c6/s2.html**](http://math.hws.edu/graphicsbook/c6/s2.html)

**MDN (2019) “A basic 2D WebGL animation example”. Recuperado el 24 de abril del 2020. Página Web:** [**https://developer.mozilla.org/en-US/docs/Web/API/WebGL\_API/Basic\_2D\_animation\_example**](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API/Basic_2D_animation_example)

**Gonzales (2015) “Dibujando con algoritmos”. Recuperado el 24 de abril del 2020. Página Web:** [**https://thebookofshaders.com/05/?lan=es**](https://thebookofshaders.com/05/?lan=es)

**Gonepane (2015) “Quien quiere un Mundo perfecto”. Recuperad el 24 de abril del 2020. Página web:** [**https://www.noeslomismo.org/2015/11/quien-quiere-un-mundo-perfecto.html**](https://www.noeslomismo.org/2015/11/quien-quiere-un-mundo-perfecto.html)

**Jesús Caro. (2020). WebGL. abril 24,2020, de CG Lab Sitio web:** [**https://www.youtube.com/watch?v=ZKGZ3qQKv\_0&list=PLpalPuQ7rTKWtZwSBrk\_rjkc4nDTo92yk**](https://www.youtube.com/watch?v=ZKGZ3qQKv_0&list=PLpalPuQ7rTKWtZwSBrk_rjkc4nDTo92yk)