

**Escuela Superior Politécnica Del Litoral**

**Gráficos por Computadora**

Proyecto Final

**Proyecto Segundo Parcial WEBGL**



**Estudiantes:**

*Lozano Jara Estefania*

*Mora Fausto*

*San Lucas César*

I Término 2015-2016  
Septiembre 12-2015

**Detalles** **del Proyecto**

Nuestro proyecto consiste en desarrollar nuestros conocimientos aprendidos en la materia de gráficos por computadora, usando WEBGL puro.

Los requerimientos realizados en nuestro proyecto son:

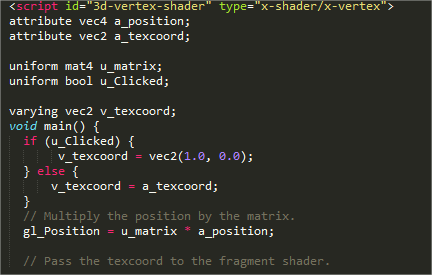
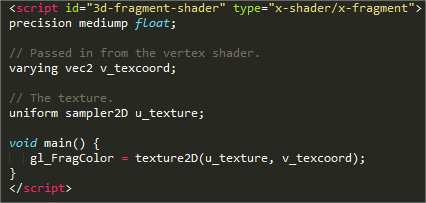
1. Deberá modificarse el primer proyecto del curso. Se utilizará tan solo WebGl para su desarrollo.
2. Se podrán mover los objetos.
3. Debe dibujarse un poste en el centro del plano y agregar una animación de “rotación” que permita que todos los objetos giren alrededor del mismo.
4. Los objetos tendrán apariencia de sólidos opacos.
5. Deberá poder moverse la cámara.
6. Los objetos tendrán textura.
7. Deberá poderse realizar zoom in y zoom out.
8. **Deberá modificarse el primer proyecto del curso. Se utilizará tan solo WebGl para su desarrollo.**

En el primer parcial tuvimos que hacer cuatro sólidos: La esfera, el cubo, la pirámide y el toroide. Y poder hacer tres operaciones: Escalar, Rotar y Trasladar. Ahora tenemos que hacer lo mismo

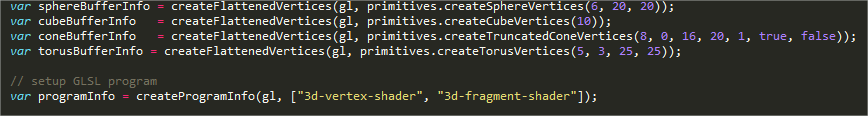
**Desarrollo**

En WebGL tuvimos que cambiar todo lo que habíamos hecho, ya que el parcial pasado habíamos usado Three.js. Ahora tuvimos que adquirir conocimientos acerca de cómo WebGL pinta los vértices y a su vez cómo les da color.

Los conocimientos adquiridos fueron acerca de los Programas: 3d-vertex-shader y 3d-fragment-shader. El primero se encarga de las propiedades de un vértice tal como la posición, el radio, etc. Mientras que el segundo se encarga del color del vértice (o si se le quisiera agregar textura). Nuestro código para estos dos programas es el siguiente. Es de destacar que las variables de estos dos programas se pueden acceder mediante el llamado a la variable “gl” que es el canvas pero en WebGl.

Ahora tuvimos que crear los uniforms para cada objeto que necesitabamos en este caso, para cada objeto solido. Y luego se llama al program Info que usa los programas anteriores

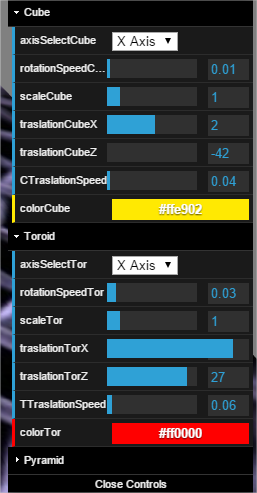
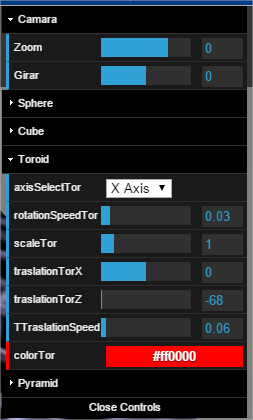


1. **Se podrán mover los objetos**

Los objetos deberán poderse mover o trasladar en los ejes permitidos.

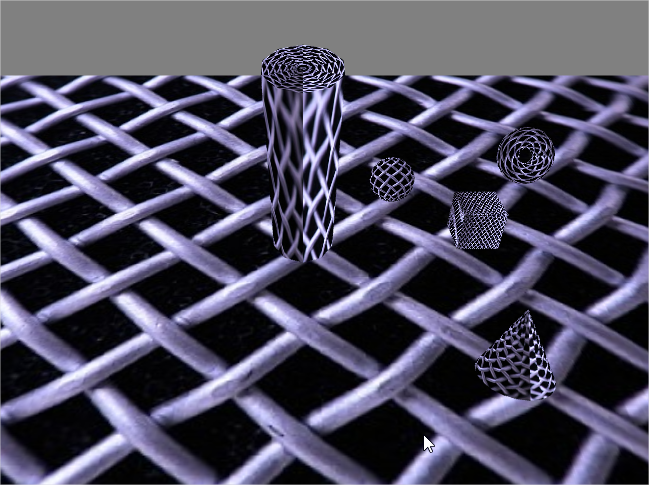
**Desarrollo**

Para realizar éste requerimiento hicimos uso de controles que nos permitieran mover a los objetos.



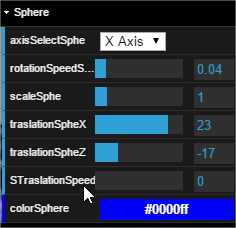
Y funciona de la siguiente manera:

* 1. Cuando se inicie a la aplicación por defecto todos los objetos están rotando.

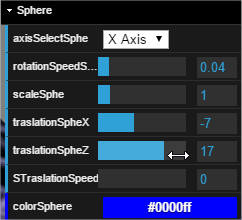


* 1. Para poder desplazar un objeto tendremos que decirle primero que deje de rotar. Y lo hacemos poniendo a cero la siguiente variable:

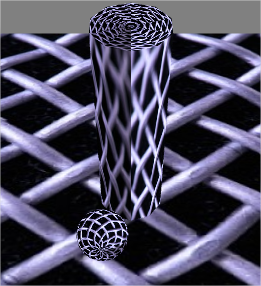
Ejemplo: Sphere



* 1. Una vez hecho esto ahí si podremos modificar las variables de X y Z en translación



* 1. Ahora la sphera está en una nueva posición donde podrá girar



1. **Debe dibujarse un poste en el centro del plano y agregar una animación de “rotación” que permita que todos los objetos giren alrededor del mismo.**

Para esto debemos usar una primitiva que permita establecer el punto central de rotación

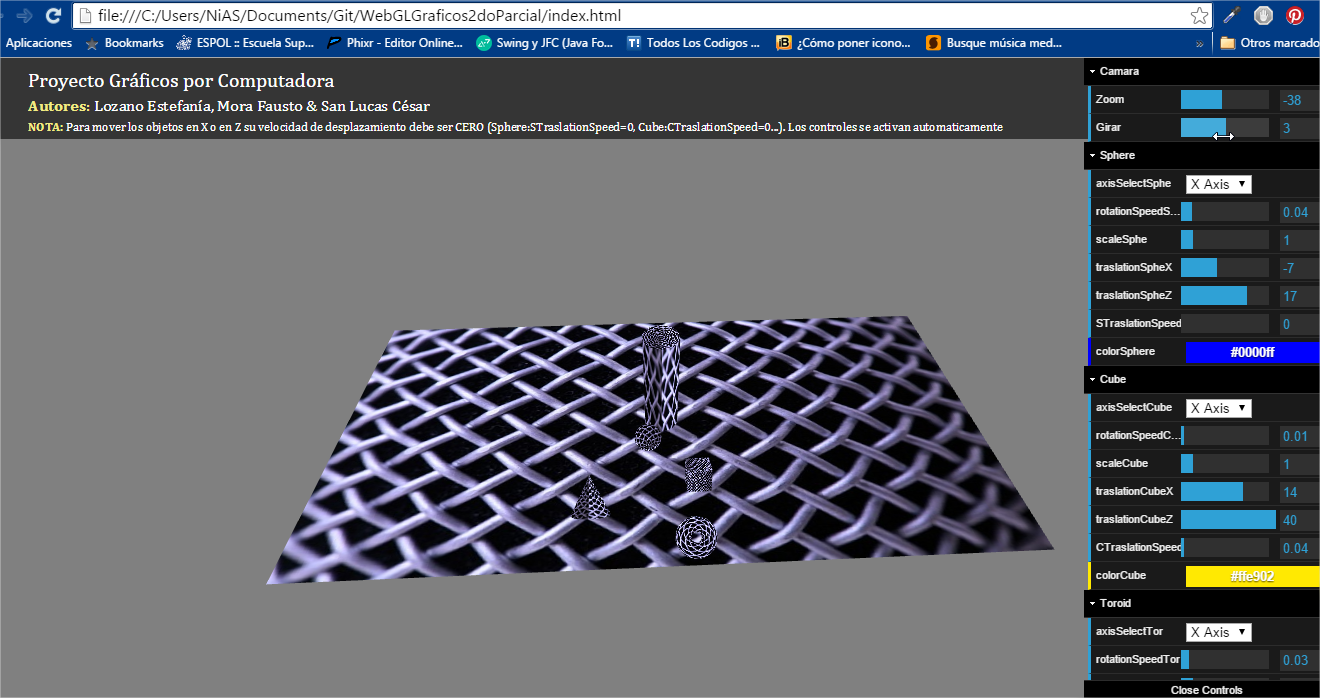
**Desarrollo**

Tuvimos que crear un poste con un uniform cilíndrico. De la siguiente forma:





Así es como quedo:

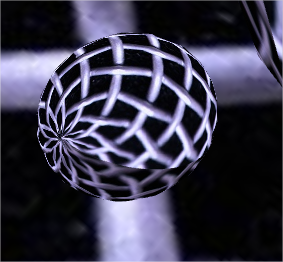
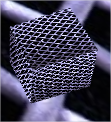
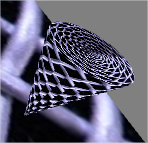


1. **Los objetos tendrán apariencia de sólidos opacos.**

Los sólidos a usarse deberán verse opacos.

**Desarrollo**

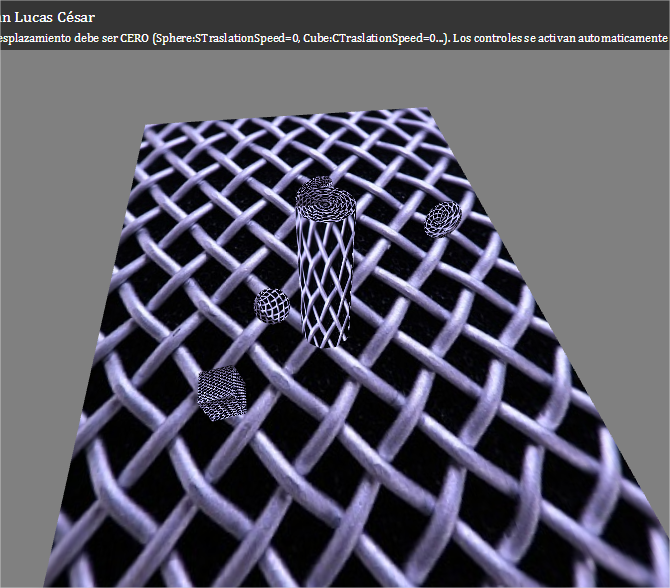
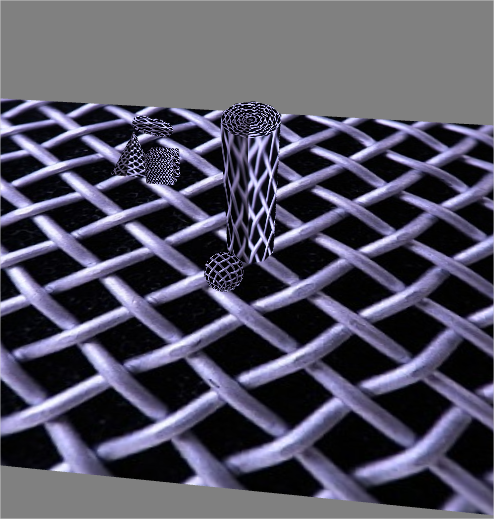
Cada objeto que se realizó tiene la forma de un sólido opaco

1. **Deberá poder moverse la cámara.**

**Desarrollo**

La cámara se puede rotar 180 grados. Y lo hace sincrónicamente, sin saltos ni fallas.

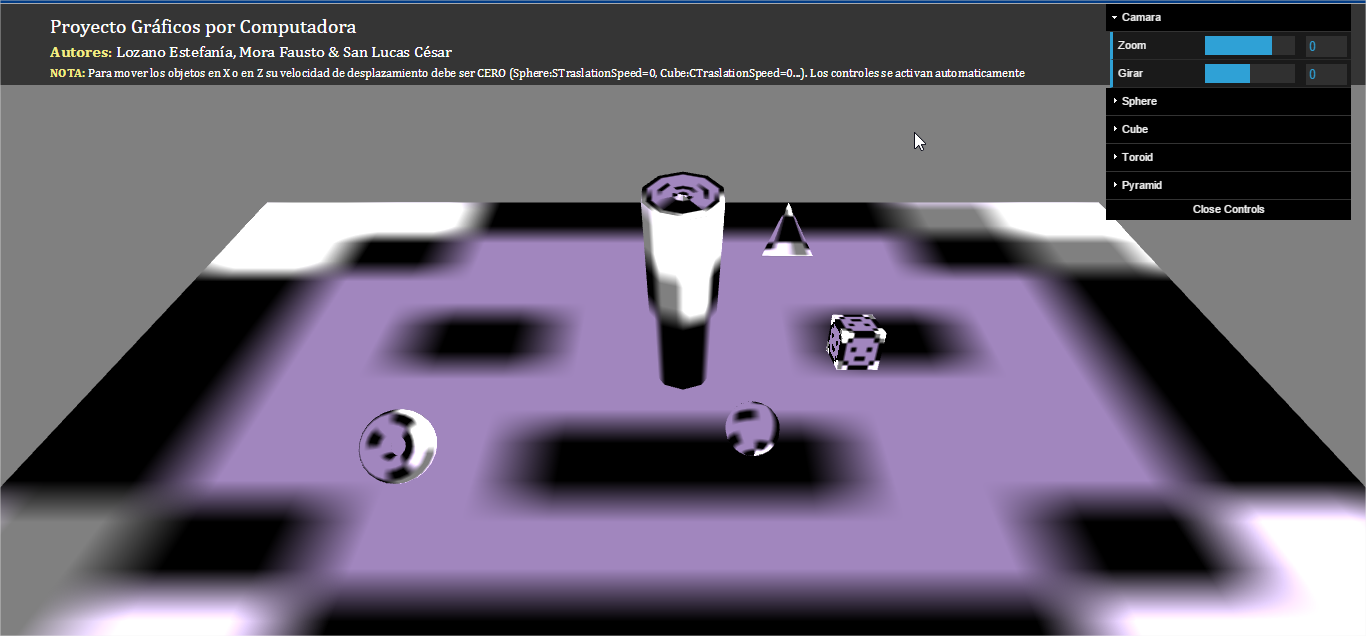
 

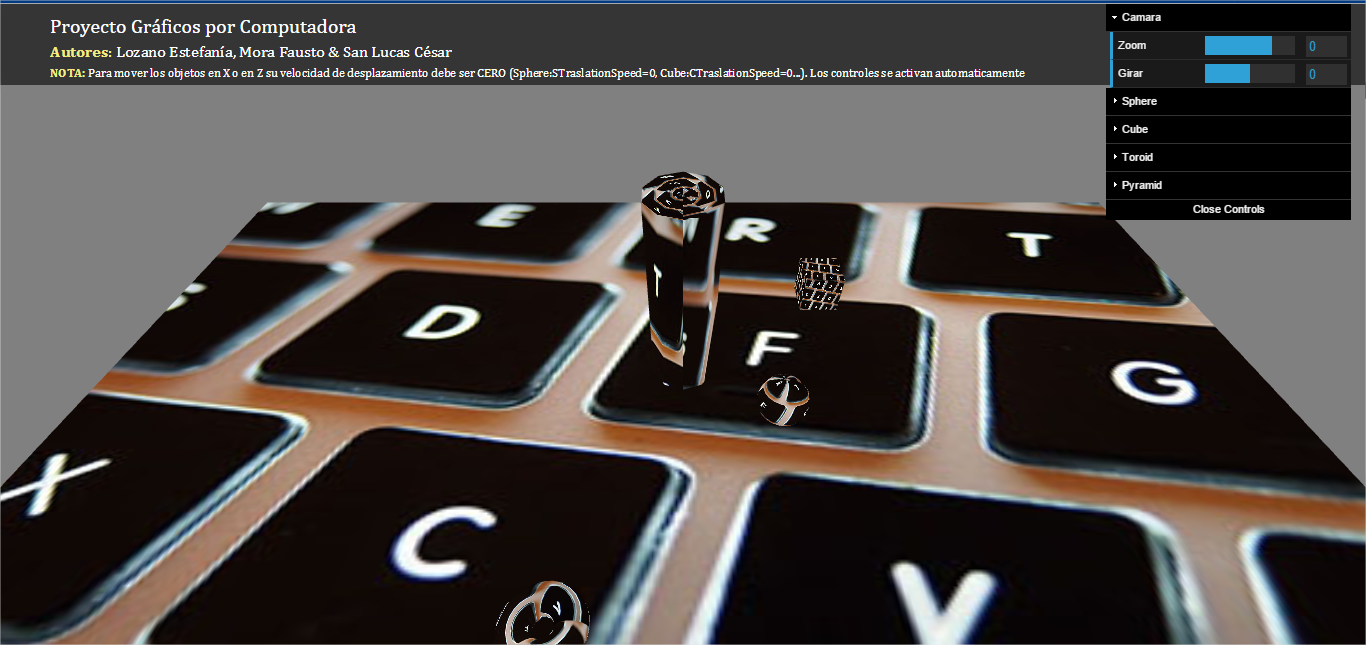
1. **Los objetos tendrán textura.**

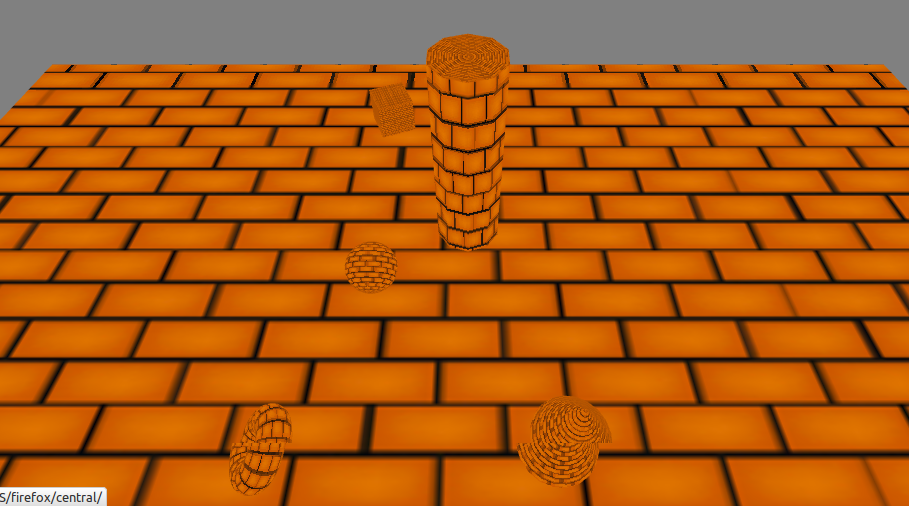
Ésta fue una de las partes más complicadas, ya que la forma de usar texturas de WEBGL es bastante compleja. Tiene una gran importancia la definición y la forma en que se hayan construido el vertex y los shader. Además debimos usar un encoder del source de las imagenes para poder representar cualquier textura proveniente de una imagen.

**Desarrollo**

Una vez entendido el concepto pudimos darle diferentes texturas.







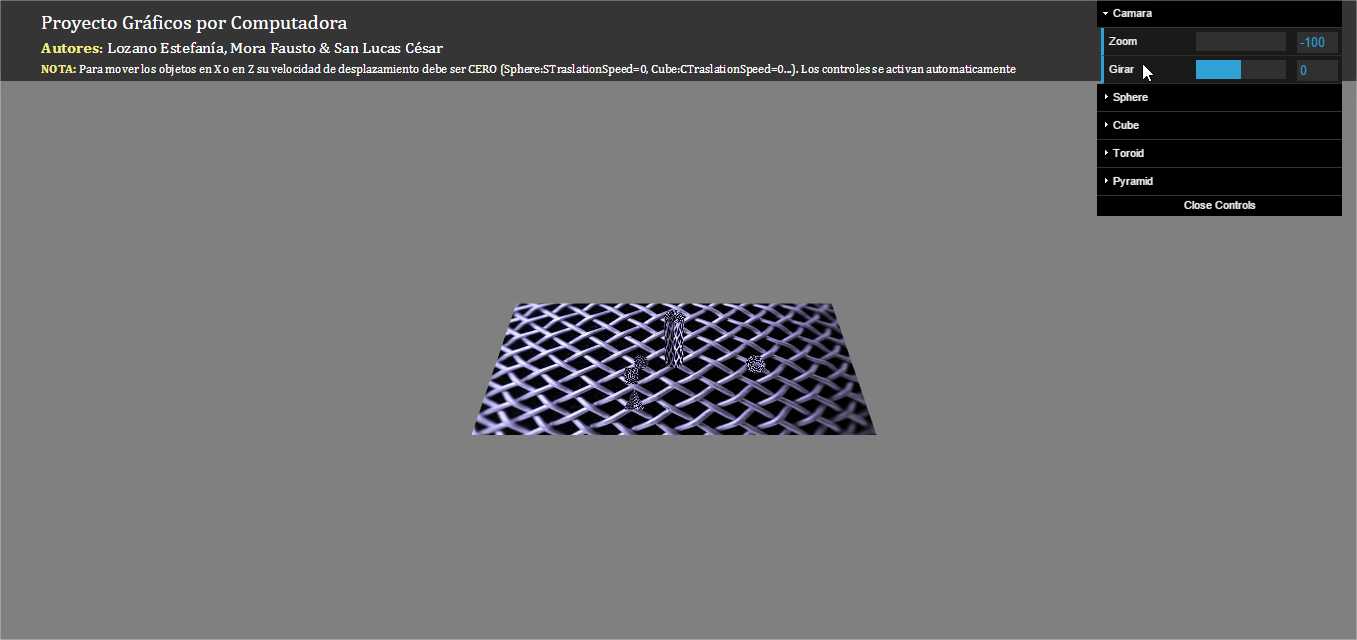
1. **Deberá poderse** **realizar zoom in y zoom out**.

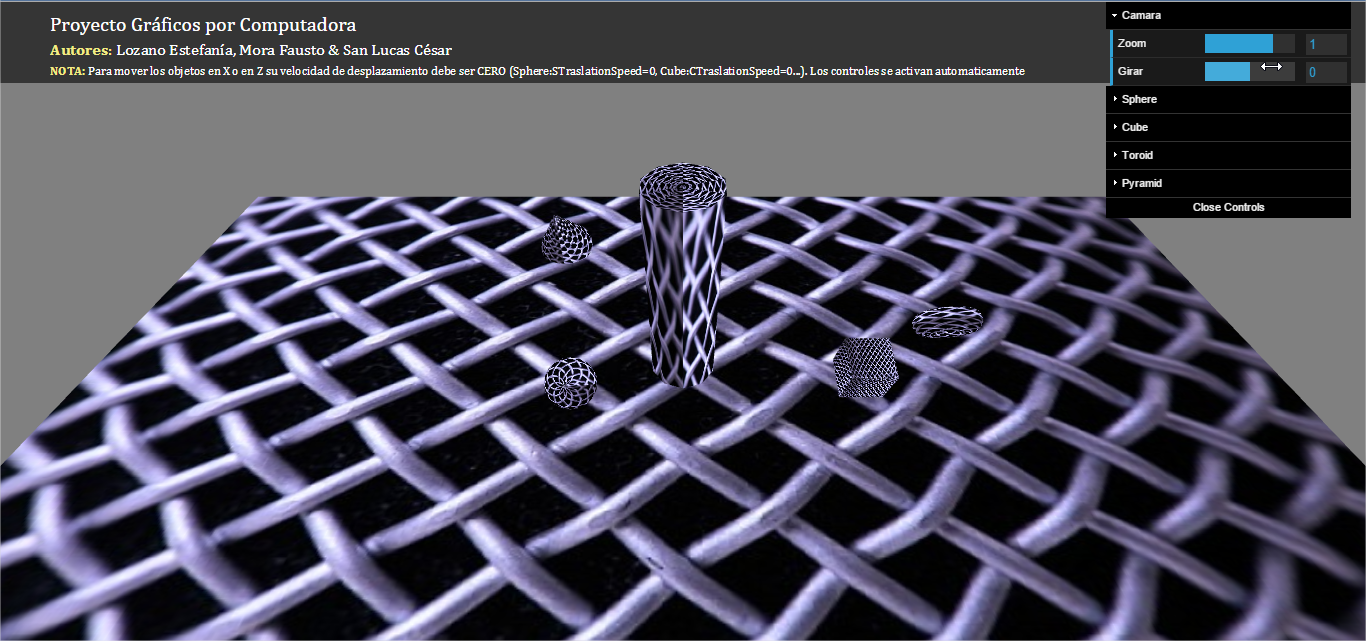
El proyecto debe poder alejar y acercar la cámara en la posición en la que se encuentre.

**Desarrollo**

Hicimos que el proyecto haga zoom in y zoom out desde cualquier parte que se encuentre

ZOOM - OUT





ZOOM - IN

