Tecnológico de Monterrey

Campus Estado de México
Campus Guadalajara

Gráficas Computacionales

Ley de Ohm en una aplicación web

Profesor: Luis Palomino Ramírez

Livier Andrade Barajas	A01377333
Daniel Marcelo Zavala Hernandez	A01169414
Guillermo Herrera Acosta	A01400835

Introducción

El objetivo final de nuestro curso en gráficas computacionales es realizar una aplicación web con un enfoque educativo aplicando los conocimientos de tecnologías de gráficas computacionales como lo es WebGL. En nuestro caso, la aplicación a desarrollar será acerca de la Ley de Ohm. Para ello se realizó una previa investigación sobre dicho tópico, así como también, un estudio requerido sobre la documentación de Three Js y WebGL.

Ley de Ohm

Definiciones básicas:
George Simon Ohm, formuló en 1827 la que se conoce como Ley de Ohm, Una de las leyes fundamentales de la electrónica.

Primero definió matemáticamente las tres magnitudes físicas principales de la electrónica las cuales son:

- **Voltaje:** Es una magnitud física, con la cual podemos cuantificar la diferencia de potencial eléctrico

- **Corriente:** Es el flujo de carga eléctrica que recorre un material

- **Resistencia:** Es la oposición al flujo de corriente eléctrica a través de un conductor

Con esto, enuncia lo siguiente:
La corriente que circula por un circuito eléctrico varía de manera directamente proporcional a la diferencia de potencial, e inversamente proporcional con la resistencia del circuito. Dando así origen a la siguiente ecuación:

$$I = \frac{V}{R}$$

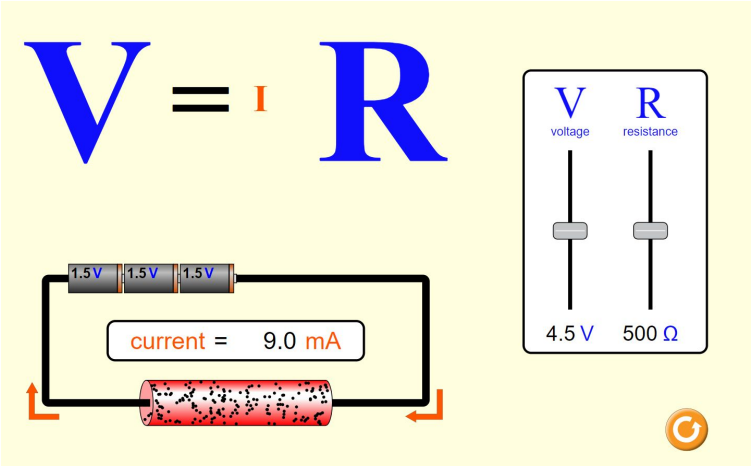
La ley de Ohm como ecuación, sirve como una receta algebraica para calcular la corriente si se conoce la diferencia de potencial eléctrico y la resistencia. Sin embargo, aunque esta ecuación sirve como una poderosa receta para resolver problemas, es mucho más que eso. Esta ecuación indica las dos variables que afectarían la cantidad de corriente en un circuito. La corriente en un circuito es directamente proporcional a la diferencia de potencial eléctrico impresa en sus extremos e inversamente proporcional a la resistencia total ofrecida por el circuito

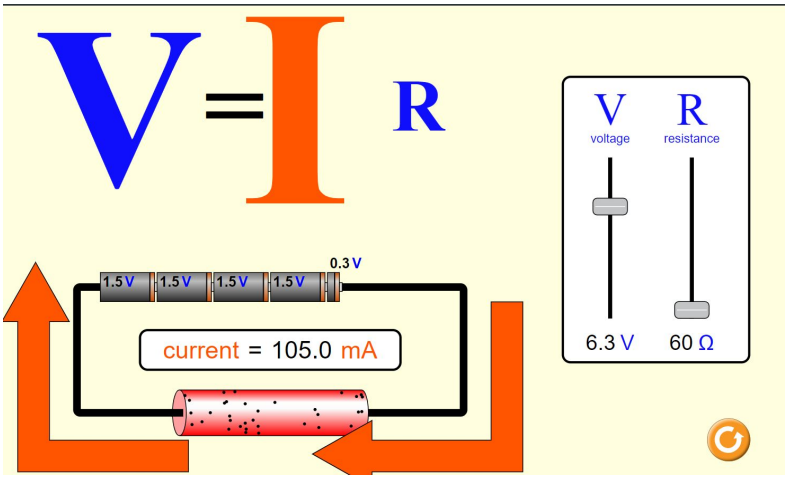
externo. Cuanto mayor sea el voltaje de la batería (es decir, la diferencia de potencial eléctrico), mayor será la corriente. Y cuanto mayor es la resistencia, menor es la corriente. La carga fluye a las velocidades más altas cuando aumenta el voltaje de la batería y disminuye la resistencia. De hecho, un aumento doble en el voltaje de la batería conduciría a un aumento doble en la corriente (si todos los demás factores se mantienen iguales). Y un aumento en la resistencia de la carga en un factor de dos causaría que la corriente disminuya en un factor de dos a la mitad de su valor original.

Simuladores

En nuestra búsqueda de simuladores encontramos que la gran mayoría de aplicaciones web eran calculadoras de corriente. La simulación que más nos llamó la atención y en la que nos basamos para realizar nuestra propia implementación, fue en la de PhET Interactive Simulations hecha por la Universidad de Colorado. En dicha simulación tenemos el dibujo de un pequeño circuito con baterías y una resistencia, al lado tenemos dos sliders que nos permiten controlar la entrada de voltaje y el valor de dicha resistencia, a partir de esos valores calcula la corriente utilizando la ley de Ohm. En dicha aplicación también podemos ver que las letras V, I , R que representan voltaje, corriente y resistencia respectivamente, cambian de tamaño según interactúan sus valores.

Se anexan capturas del simulador de referencia, la liga a dicho simulador se encuentra en la bibliografía.





Three Js y WebGL

Para nuestra aplicación web decidimos utilizar Three.js. el cual es un framework que corre como una capa de tipo plugin de WebGL. Three Js usa WebGL para dibujar en 3D. WebGL es un sistema de bajo nivel que ayuda a representar gráficamente geometrías básicas como puntos, líneas y triángulos. El hecho de que WebGL ofrezca herramientas gráficas básicas, se puede tomar como base para que un framework como Three Js pueda aprovechar para manejar implementaciones como escenas, luces, sombras, materiales, texturas, matemáticas en 3D, entre otras herramientas de un nivel más alto.

A continuación se muestra un listado con un breve resumen de la documentación utilizada en la realización de este proyecto.

- **Primitives :** Las primitivas son generalmente formas 3D que se generan en tiempo de ejecución con un montón de parámetros.
- **Materials:** Three.js proporciona varios tipos de materiales. Definen cómo aparecerán los objetos en la escena.
- **Textures :** Las texturas son generalmente imágenes que se crean con mayor frecuencia en algún programa de terceros como Photoshop o GIMP
- **Cameras :** La cámara más común en three.js es la PerspectiveCamera. Ofrece una vista en 3D donde las cosas en la distancia parecen más pequeñas que las cosas de cerca.
- **Custom Geometry:** El uso de Geometry vs BufferGeometry para piezas customizadas.
- **FontLoader :** Clase para cargar una fuente en formato JSON

Bibliografia

S.A(2017). Ohm's Law. Retrieved *July 23rd* from:
https://www.physicsclassroom.com/class/circuits/Lesson-3/Ohm-s-Law

S.A(2018). Ohm's Law. Retrieved *July 23rd* from :
http://hyperphysics.phy-astr.gsu.edu/hbase/electric/ohmlaw.html

phET(2019) Ohm's Law Simulator. Retrieved *July 24th* from:
https://phet.colorado.edu/sims/html/ohms-law/latest/ohms-law_en.html

phET(2019) Ohm's Law Simulator Resources. Retrieved *July 24th* from:
https://github.com/phetsims/ohms-law

Three. Js(2016) Three Js Fundamentals. Retrieved *July 24th* from:
https://threejsfundamentals.org

Three. Js(2018) Three Js FoantLoader. Retrieved *July 24th* from:
https://threejs.org/docs/#api/en/loaders/FontLoader