

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**2CM3**

**PROFESOR: TECLA PARRA ROBERTO**

**PRÁCTICA 3 SISTEMA SOLAR (PLANETARIO)**

**VÁZQUEZ MORENO MARCOS OSWALDO 2016601777**



**FECHA DE ENTREGA: 15 DE MARZO DE 2018**

**OBJETIVO**

Realizar un programa en lenguaje de programación Java complementario al visto en clase agregando 2 planetas más al código ya plasmado y para cada planeta:

-Crear una apariencia.

-Cargar una textura a partir del archivo de una imagen.

-Poner la textura en la apariencia.

-Crear una esfera con el radio y la apariencia correspondiente al planeta.

-Rotar la esfera sobre su propio eje a la velocidad correspondiente al planeta (duración del día).

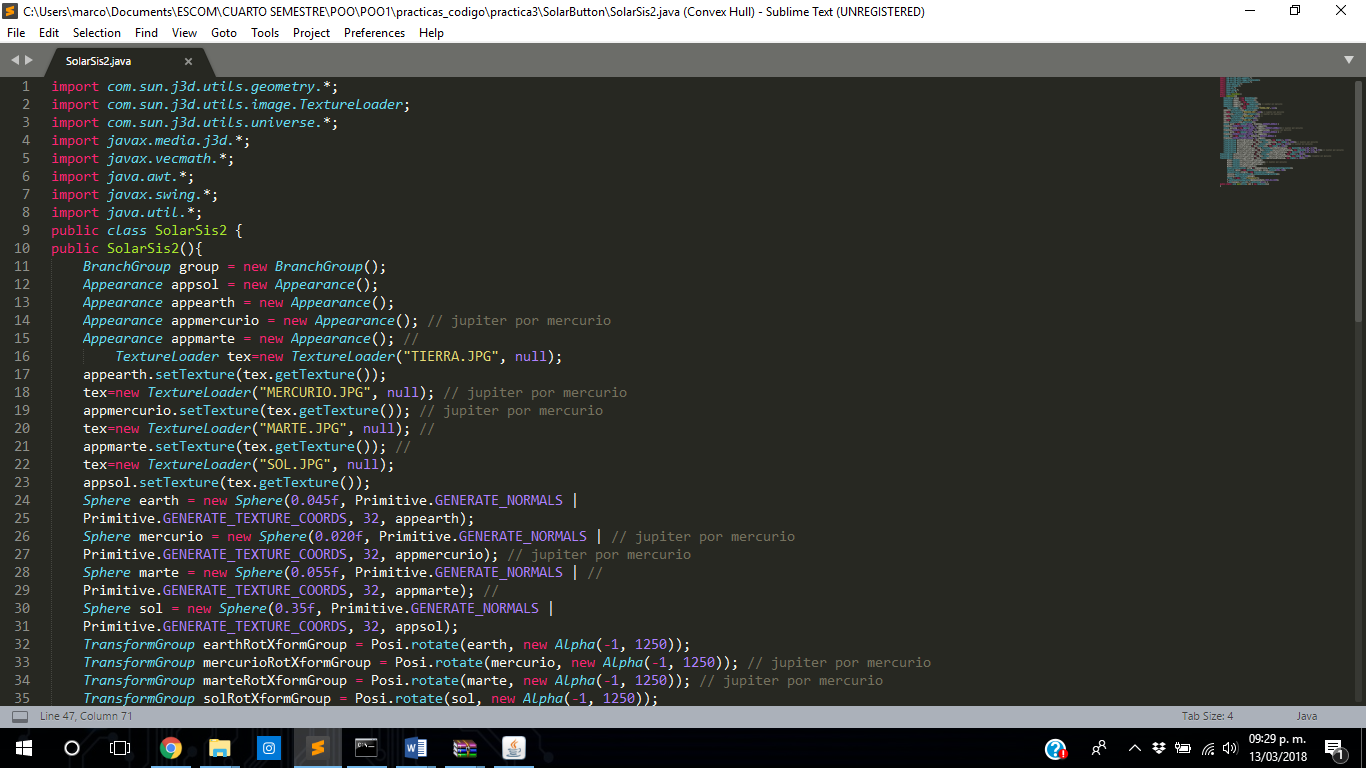
-Alejar la esfera del sol (la posición del sol es el origen).

-Rotar la esfera alrededor del sol a la velocidad correspondiente al planeta (duración del año).

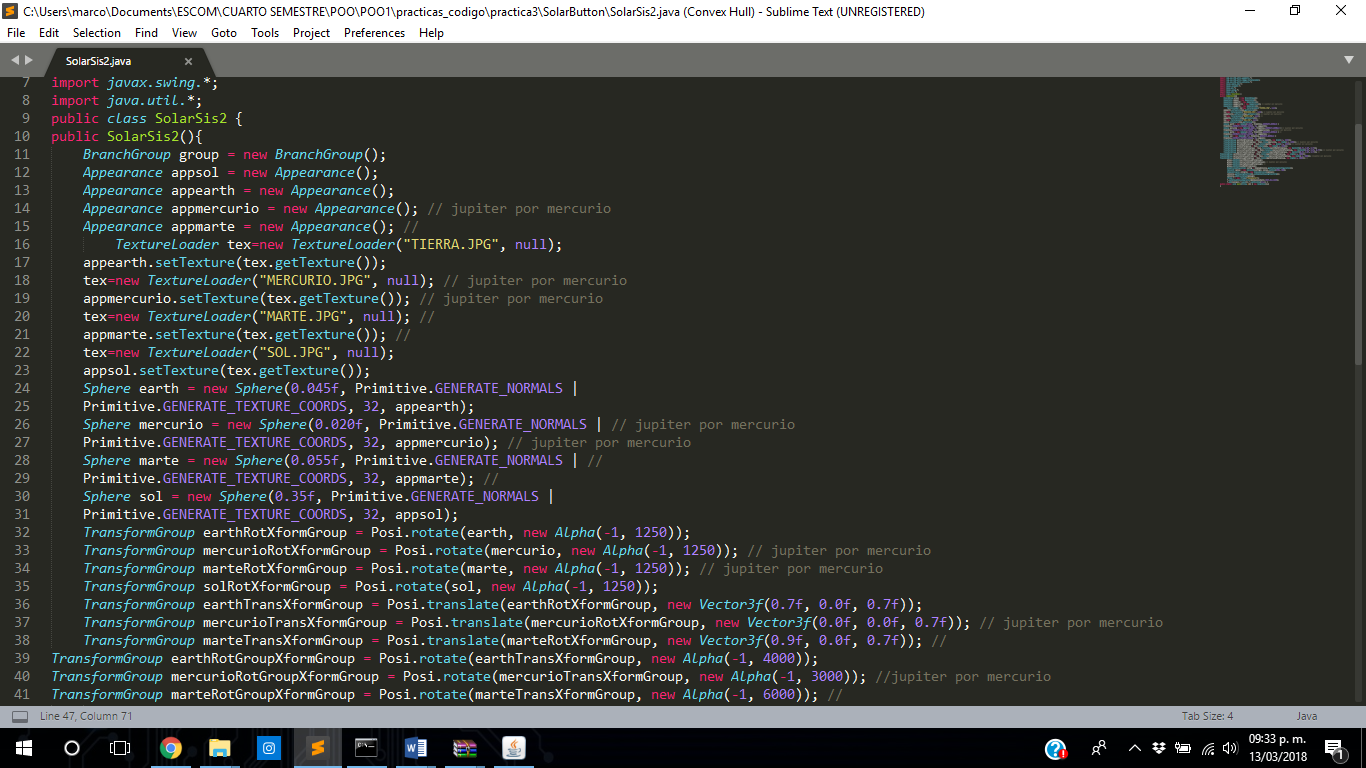
-Agregarla al BranchGroup.

**DESARROLLO**

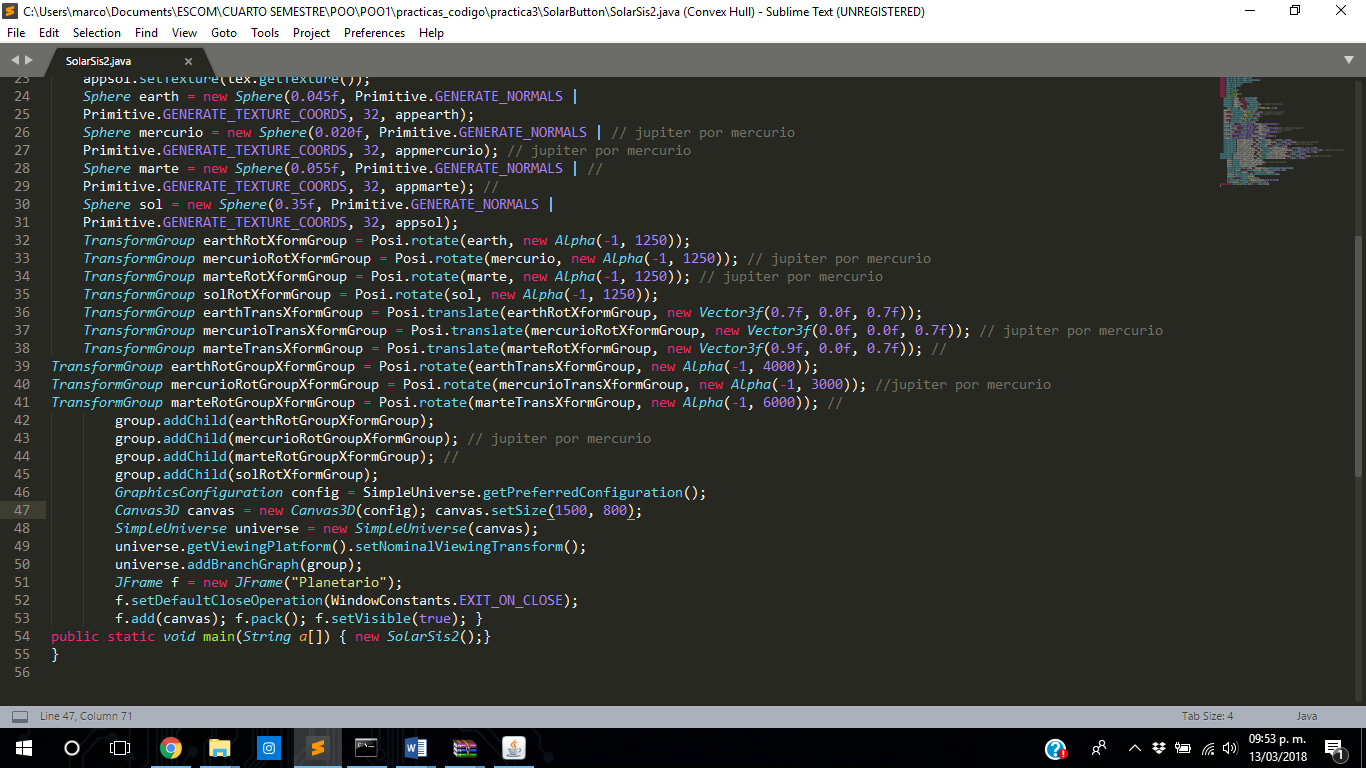
Primeramente, agregamos todas las librerías y extensiones tanto de java como de java 3D para poder llevar acabo nuestra simulación de un planetario, las cuales serán distintas a las anteriormente usadas y ahora comienzan a integrar partes de el java 3D con “com.sun”.



Después, declaramos las variables de tipo *BranchGroup, Appearance y TextureLoader* las cuales van a agregar al grupo las variables gráficas que se vayan declarando para así estar en un mismo agrupamiento, la apariencia para que tenga algo de sentido el planeta, que se vea que es una imagen y no una simple esfera haciendo movimientos de rotación y traslación, por último en este apartado tenemos el TextureLoader el cual va a asignar la imagen con su extensión a cada uno de los planetas y el sol.



Continuando tenemos en la línea 18 la asignación de textura mediante la imagen a los objetos siendo las esferas, siendo así hasta la línea 23 puesto que en estas líneas se agregó a mercurio y marte, se hizo esto ya que no me agrada mucho el planeta Venus y decidí tener así más espacio entre mercurio y la tierra, generando en las líneas siguientes la textura acorde al tipo de figura pensada.



Continuando con la implementación tenemos en la línea 32 hasta la 35 del código la transformación en el espacio determinado para mostrar el programa funcionando ya que tanto a la tierra, sol, mercurio y marte puedes colocarlos en una misma sintonía o en este caso en la misma línea, pero no en la misma orbita, teniendo así en estas líneas la rotación sobre su propio eje.

Después en las líneas de la 36 a la 38 vamos a definir el tamaño, distancia y velocidad de mercurio, tierra y marte respecto al Sol, ya que mercurio es más rápido, seguido de la Tierra y al último marte ya que entre más alejado está el planeta del sol, menos rápido hacer su rotación, el último de los dígitos es para que todos los planetas estén sobre un mismo plano y no unos más arriba de otros.

Y como en todo buen programa de java que hemos hecho se agregaron cada uno de los planetas con sus especificaciones anteriores de la línea 42 a la 45.

Finalmente, las últimas líneas de código son para definir el JFrame que se encuentra englobando a todo el funcionamiento del planetario.

**CONCLUSIÓN**

Con la implementación de la práctica número 3 se puede concluir que sin duda esto es algo nuevo para mí, estoy impactado y emocionado ya que ver un resultado grafico te hace creer que estás evolucionando como programador, sin embargo, es una práctica bastante sencilla en donde solo debes de jugar con el código ver qué es lo que hacer y modificarlo de modo que puedas agregar los siguientes dos planetas que se necesitan.

Por otro lado, es un buen y gran inicio en Java 3D puesto que se vienen mucho mejores cosas por aprender.

Además, se logró satisfactoriamente el objetivo de agregar y mostrar los dos planetas adicionales, llevando a cabo una correcta rotación y traslación de cada uno de los planetas y su real movimiento como se muestra a continuación.

