29-3-2017

Programación Orientada a Objetos

Alumno: Erick Efrain Vargas Romero

Profesor: Tecla Parra Roberto

Tema: Java 3D.

Número de práctica: 3

Fecha: 29-03-2017

Grupo: 2CM4

# **Objetivo**

Aplicar los conocimientos que se han adquirido en el salón de clase sobre Java 3D. Entre los conocimientos que son aplicados la rotación, traslación, creación de figuras, etc. Es realmente necesario para trabajar en esta práctica el conocer todos estos elementos.

# **Desarrollo**

Primeramente, para esta práctica es necesario hacer uso del constructor en el cual utilizaremos primero un BranchGroup, posteriormente crearemos algunos objetos tipo Appearance uno para cada planeta y el sol, una vez realizado lo anterior creamos un objeto tipo TextureLoader en el cual le daremos la ruta en la cual se encuentra la textura a aplicar, esto se realiza para las texturas de los demás planetas y el sol, finalmente en nuestras variables tipo Appearance es aplicada la textura.

BranchGroup group **=** **new** BranchGroup**();**

Appearance appsol **=** **new** Appearance**();**

Appearance appearth **=** **new** Appearance**();**

Appearance appmarte **=** **new** Appearance**();**

Appearance appjupiter **=** **new** Appearance**();**

TextureLoader tex **=** **new** TextureLoader**(**"c:\\textura\\TIERRA.JPG"**,** **null);**

appearth**.**setTexture**(**tex**.**getTexture**());**

tex **=** **new** TextureLoader**(**"C:\\textura\\SOL.JPG"**,** **null);**

appsol**.**setTexture**(**tex**.**getTexture**());**

tex **=** **new** TextureLoader**(**"C:\\textura\\MARTE.JPG"**,null);**

appmarte**.**setTexture**(**tex**.**getTexture**());**

tex **=** **new** TextureLoader**(**"C:\\textura\\JUPITER.JPG"**,null);**

appjupiter**.**setTexture**(**tex**.**getTexture**());**

Ya que se han aplicado las texturas es necesario crear esferas, las cuales utilizaremos para mostrar los planetas, como es obvio cada planeta de diferente tamaño, y el de mayor tamaño es el sol, además a las esferas le pasamos como parámetro las apariencias que fueron creadas anteriormente.

Sphere earth **=** **new** Sphere**(**0.063f**,** Primitive**.**GENERATE\_NORMALS **|** Primitive**.**GENERATE\_TEXTURE\_COORDS**,** 32**,** appearth**);**

Sphere sol **=** **new** Sphere**(**0.35f**,** Primitive**.**GENERATE\_NORMALS **|** Primitive**.**GENERATE\_TEXTURE\_COORDS**,** 32**,** appsol**);**

Sphere marte **=** **new** Sphere**(**0.043f**,** Primitive**.**GENERATE\_NORMALS **|** Primitive**.**GENERATE\_TEXTURE\_COORDS**,** 32**,** appmarte**);**

Sphere jupiter **=** **new** Sphere**(**0.093f**,** Primitive**.**GENERATE\_NORMALS **|** Primitive**.**GENERATE\_TEXTURE\_COORDS**,** 32**,** appjupiter**);**

Una vez hecho lo anterior es necesario darle rotación a cada planeta y el sol, esto creando objetos tipo TransformGroup, posteriormente, debemos darle también a cada planeta una traslación, la cual debemos de definir también una distancia a la cual se estará trasladando el planeta y la rapidez con la cual se va a trasladar, una vez realizado esto será necesario añadir todo esto a nuestro objeto BranchGroup.

TransformGroup solRotXformGroup **=** rotate**(**sol**,** **new** Alpha**(-**1**,** 1250**));**

TransformGroup earthTransXformGroup **=** translate**(**earthRotXformGroup**,** **new** Vector3f**(**0.0f**,** 0.0f**,** 0.50f**));**

TransformGroup earthRotGroupXformGroup **=** rotate**(**earthTransXformGroup**,** **new** Alpha**(-**1**,** 6000**));**

Cuando lo anterior se ha hecho ahora debemos de crear una configuración gráfica, la cual será igualada a un universo simple y obtendremos su configuración preferida, ahora procedemos a crear un objeto tipo Canvas3D a la cual le pasamos como parámetro la configuración gráfica que se ha creado, finalmente, le damos tamaño a nuestro canvas.

GraphicsConfiguration config**=**SimpleUniverse**.**getPreferredConfiguration**();**

Canvas3D canvas **=** **new** Canvas3D**(**config**);**

canvas**.**setSize**(**400**,** 400**);**

Ahora debemos de crear un objeto tipo SimpleUniverse, le pasaremos como parámetro nuestro objeto canvas, a este objeto tipo SimpleUniverse, utilizando el método addBranchGraph le daremos como parámetro nuestro BranchGroup previamente creado.

SimpleUniverse universe **=** **new** SimpleUniverse**(**canvas**);**

universe**.**getViewingPlatform**().**setNominalViewingTransform**();**

universe**.**addBranchGraph**(**group**)**

Ahora debemos utilizar vectores, a este objeto tipo Vector, le añadiremos elementos, los cuales son los nombres de nuestros planetas, este vector se le añadirá a un JComboBox que como es obvio debe ser declarado antes, y al igual que los objetos tipo Button o JButton que han sido utilizados hasta ahora se le añade un ActionListener.

Vector nomPlanet **=** **new** Vector**();**

nomPlanet**.**addElement**(**"Sol"**);**

nomPlanet**.**addElement**(**"Tierra"**);**

jcb **=** **new** JComboBox**(**nomPlanet**);**

jcb**.**addActionListener**(this);**

También para esta práctica utilizaremos JPanel en vez de un Applet, a este JPannel se le añadirá nuestro JComboBox posteriormente se creará un objeto tipo JFrame que tendrá como título Planetario, además a este JFrame, se le posiciona en el centro nuestro objeto tipo Canvas3D y también nuestro JPanel, finalmente se establece como visible.

JPanel jp **=** **new** JPanel**();**

jp**.**add**(**jcb**);**

JFrame f **=** **new** JFrame**(**"Planetario"**);**

f**.**setDefaultCloseOperation**(**WindowConstants**.**EXIT\_ON\_CLOSE**);**

f**.**setLayout**(new** BorderLayout**());**

f**.**add**(**"Center"**,** canvas**);**

f**.**add**(**"South"**,** jp**);**

f**.**pack**();**

f**.**setVisible**(true);**

Ahora se ha trabajado en dos métodos ambos retornarán un objeto tipo TransformGroup el primero se llama rotate y recibe dos parámetros un nodo y un objeto tipo Alpha, lo que se hace en este método es dar rotación, como se ha mencionado anteriormente, esto lo hacemos primeramente creando un grupo de transformación, y también creando un objeto de tipo RotationInterpolator, al cual le pasaremos dos parámetros, el primero es el alpha que recibe nuestro método rotate y el segundo es nuestro objeto TransformGroup previamente creado, con el objeto RotationInterpolator estableceremos el ángulo de rotación tanto inicial como final que tendrán nuestros planetas.

TransformGroup rotate**(**Node node**,** Alpha alpha**)** **{**

TransformGroup xformGroup **=** **new** TransformGroup**();**

xformGroup**.**setCapability**(**TransformGroup**.**ALLOW\_TRANSFORM\_WRITE**);**

RotationInterpolator interpolator **=** **new** RotationInterpolator**(**alpha**,** xformGroup**);**

interpolator**.**setSchedulingBounds**(new** BoundingSphere**(new** Point3d**(**0.0**,** 0.0**,** 0.0**),** 1.0**));**

xformGroup**.**addChild**(**interpolator**);**

xformGroup**.**addChild**(**node**);**

**return** xformGroup**;**

**}**

También tenemos un método llamado translate, el cual recibe dos parámetros un objeto tipo Node y un objeto tipo Vector3f, cabe mencionar que el objeto Node hace referencia a la figura con la que trabajaremos, para nuestro caso, una esfera, y el segundo parámetro es una clase matemática, que especifica que un vector está en tres dimensiones prácticamente.

TransformGroup translate**(**Node node**,** Vector3f vector**)** **{**

Transform3D transform3D **=** **new** Transform3D**();**

transform3D**.**setTranslation**(**vector**);**

TransformGroup transformGroup **=** **new** TransformGroup**();**

transformGroup**.**setTransform**(**transform3D**);**

transformGroup**.**addChild**(**node**);**

**return** transformGroup**;**

**}**

# **Conclusión**

Podemos concluir que Java tiene infinidad de posibilidades, desde crear aplicaciones llamémosles “simples” hasta aplicaciones realmente complejas en cuanto a implementación, pero que son realmente útiles, en el caso de Java 3D podríamos incluso hacer simulaciones haciendo uso de sus grandes bondades