Grafos de escena

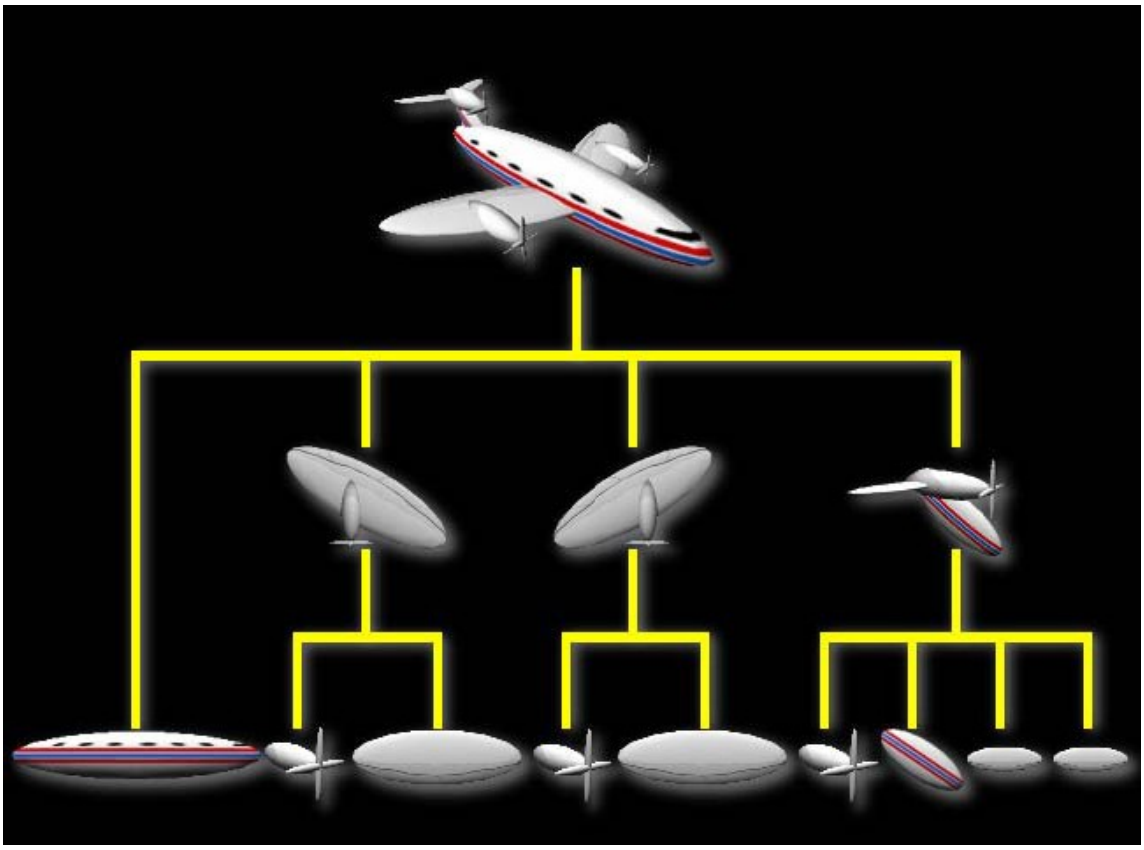
Un *grafo de escena* es un grafo dirigido acíclico de nodos que contiene los datos que definen un escenario virtual y controlan su proceso de dibujado. Contiene descripciones de bajo nivel de la geometría y la apariencia visual de los objetos, así como descripciones de alto nivel referentes a la organización espacial de la escena, datos específicos de la aplicación, transformaciones, etc. Los *grafos de escena* almacenan la información del escenario virtual en diferentes tipos de **nodos.**

Existen nodos que almacenan la información geométrica y actúan como nodos hijos dentro del *grafo de escena;* el resto de los nodos suelen aplicar algún tipo de modificación sobre el segmento de jerarquía que depende de ellos, bien sea estableciendo agrupaciones, aplicando alguna transformación afín o realizando algún tipo de selección sobre alguna de sus ramas hijas. El proceso de dibujado consiste en realizar un recorrido de dicho grafo, aplicando las operaciones indicadas por cada tipo de nodo.

El *Grafo de Escena* tiene como funciones principales:

- Contribuir a establecer una organización lógica de la escena.
- Establecer dependencias jerárquicas entre distintos sistemas de referencia.
- Posibilitar el proceso de selección entre múltiples niveles de detalle.
- Posibilitar el proceso automático de *Culling* (eliminación automática de los objetos que se encuentran fuera del campo de visión).
- Facilitar el control de la escena por parte del usuario.
- Hacer más cómodo el acceso a las librerías gráficas de bajo nivel (OpenGl en este caso).

En la siguiente imagen se puede apreciar la descomposición de un objeto en sus diferentes componentes, de manera agrupada, lo cual sería una aproximación al *grafo de escena* que lo definiría.



Tipos básicos de Nodos.

En la actualidad existen varias librerías gráficas de alto nivel, y cada uno de sus *grqfos de escena* presenta sus propias particularidades. Sin embargo, existe un conjunto básico de nodos que, a veces con distintos nombres, se encuentran presentes en todos ellos:

• **Nodo de Geometría.** Almacenan la información poligonal de los objetos, también almacenan informaciones referentes a su apariencia, tales como material, textura, etc. Usualmente actúan como nodos hoja.

• **Nodo Grupo.** Se emplean para agrupar varios nodos hijos, bien sea a nivel meramente organizativo, o para facilitar el proceso de culling jerárquico.

• **Nodo Nivel de Detalle.** Usualmente llamados nodos *LOD* (Level of Detail). Seleccionan uno de sus hijos, basándose en la distancia entre el objeto con múltiples niveles de detalle y el punto de vista.

• **Nodo de Transformación Afín.** Permite aplicar una matriz de transformación que afectara a ubicación espacial de sus nodos hijos. Son necesarios para la definición de objetos móviles y también para la creación de estructuras articuladas.

• **Nodo de Switch.** Permiten realizar una selección entre sus nodos hijos.

También es usual que el usuario tenga cierta capacidad para personalizar el comportamiento de los nodos, para ello los nodos suelen tener la capacidad de almacenar datos genéricos que necesite el usuario, y también rutinas de callback escritas por el usuario que son invocadas junto con el código interno de gestión del nodo.

OpenSceneGraph

OpenSceneGraph (OSG) es un toolkit gráfico de alto nivel y portable para el desarrollo de aplicaciones gráficas de alto rendimiento tales como simuladores de vuelo, juegos, realidad virtual o visualización científica. Está orientado a objetos y construido a partir de la librería gráfica OpenGL, esto libera al desarrollador de implementar y optimizar llamadas gráficas de bajo nivel, y provee muchas utilidades adicionales para un rápido desarrollo de aplicaciones gráficas.

El corazón del grafo de escena ha sido diseñado para tener mínimas dependencias de una plataforma específica, requiriendo poco más que C++ estándar y OpenGL. Esto ha permitido al grafo de escena ser rápidamente portado a un gran número de plataformas (originalmente desarrollado en IRIX, portado a Linux, Windows, FreeBSD, Mac OSX, Solaris, HP-UX e incluso PlayStation2).

Todo el código de *OpenSceneGraph* esta publicado bajo la *OpenSceneGraph* Public License (permite a proyectos de código abierto y cerrado utilizarla, modificarla y distribuirla libremente). Open Scene Graph soporta view frustum culling, occlusion culling, small feature culling, nodos con nivel de detalle (LOD), clasificación de estado, vertex arrays y listas de dibujado como parte del

corazón del grafo de escena.

OpenSceneGraph es uno de los grafos de escena disponibles de mayor rendimiento. Este rendimiento iguala a otros grafos de escena como OpenGL Performer o Vega Scene Graph. Open Scene Graph opta por soluciones muy parecidas a OpenGL Performer. Por contra, no soporta multiproceso, característica que soporta OpenGL Performer.

Open Scene Graph esta formado por los siguientes espacios de nombres:

- **osg :** Es el núcleo de la librería OSG, y proporciona las clases básicas del grafo de escena tales como Nodes, Status, y Drawables, así como clases matemáticas y otras.
- **osgDB:** osgDB proporciona soporte para leer y escribir grafos de escena, proporcionando un framework para plugins y clases para manejo de ficheros.
- **osgFX :** Es una extensión del núcleo del grafo de escena para proporcionar un framework de efectos especiales.
- **osgGA :** osgGA (osg GUI Abstraction) proporciona herramientas para ayudar a los desarrolladores para trabajar con distintos sistemas de ventanas.
- **osgIntrospection :** Proporciona un entorno de programación que permite la consulta en tiempo de ejecución de las propiedades y los métodos relacionados con las librerias OSG.
- **osgParticle :** osgParticle amplia el núcleo del grafo de escena para soportar efectos de partículas.
- **osgProducer :** Es una librería de utilidades que integra OpenProducer para proporcionar clases de viewer de propósito general.
- **osgSim :** osgSim extiende el núcleo del grafo de escena para soportar Nodes y Drawables que especifiquen la simulación visual, tales como soporte para un punto de luz navegacional y transformaciones de grados de libertado del estilo OpenFlight.
- **osgTerrain :** Librería de utilidades que proporciona soporte para la generación de bases de datos de terreno.
- **osgText :** Extiende el núcleo del grafo de escena para dar soporte a texto de alta calidad.
- **osgUtil :** Proporciona clases de utilidad de propósito general, tales como recorridos de update, cull, y/o Draw, operadores de grafo de escena como son optimisation, tri stripping, y tessellation.
- **osgUtx :** osgUtx es un entorno de programacion para la evaluacion de aplicaciones.

OpenSceneGraph emplea los siguientes tipos básicos de nodos:

- Node : La clase base para todas la clases que derivan de *Node*.
  - Group : Agrupa varios nodos hijos.
    - Transform : Clase base para aplicar una transformación al subgrafo.
      - MatrixTransform : Transformación de una matriz 4x4.
      - PositionAttitudeTransform : Transformación que usa un Vector de tres coordenadas (*Vec3*) para la posición, y una rotación de Cuaternion (*Quat*) para la actitud, y un *Vec3* para el pivote.
      - DOFTransform : Nodo de transformación de grados de libertad.
    - Geode : Es un nodo hoja que almacena la información geométrica de un objeto.

- Billboard : Rota una geometría de modo que siempre aparezca orientada hacia el punto de vista. Es especialmente útil para representar objetos que tienen una simetría axial, como pueden ser árboles, farolas, etc.
- LOD : Se emplea para gestionar distintos niveles de detalle de un objeto
- Impostor : añade soporte para el cacheado jerárquico de imágenes.
- Switch : Es un nodo que puede tener varios hijos y permite que el usuario seleccione cuales de ellos quiere que sean dibujados.
- Sequence : Es un nodo que puede tener varios hijos, los cuales va mostrando de forma secuencial. Se utiliza para representar secuencias animadas. Una secuencia consiste en una lista ordenada de hijos, cada uno de los cuales con una duración asignada. Es posible hacer que la secuencia se ejecute de inicio a fin, de fin a inicio, que se repita cíclicamente, etc.
- LightSource : Posición un objeto *Light* en la escena
- ClipNode : Posiciona un objeto *ClipPlane* en la escena
- Projection : Sobrecarga la matriz de proyección.
- OccluderNode : Permite colocar en la escena planos y cajas para definir oclusiones entre objetos.

librería osgVP

Core

El objetivo de osgVP-CORE es que las clases de esta libreria mapeen, haciendo uso de JNI, la libreria OpenSceneGraph. Es decir, las clases de esta libreria se implementan haciendo llamadas nativas a las mismas clases en OSG, el mapeo es directo, uno a uno. Esceptuando las clases matematicas que han sido implementadas en su totalidad en Java.

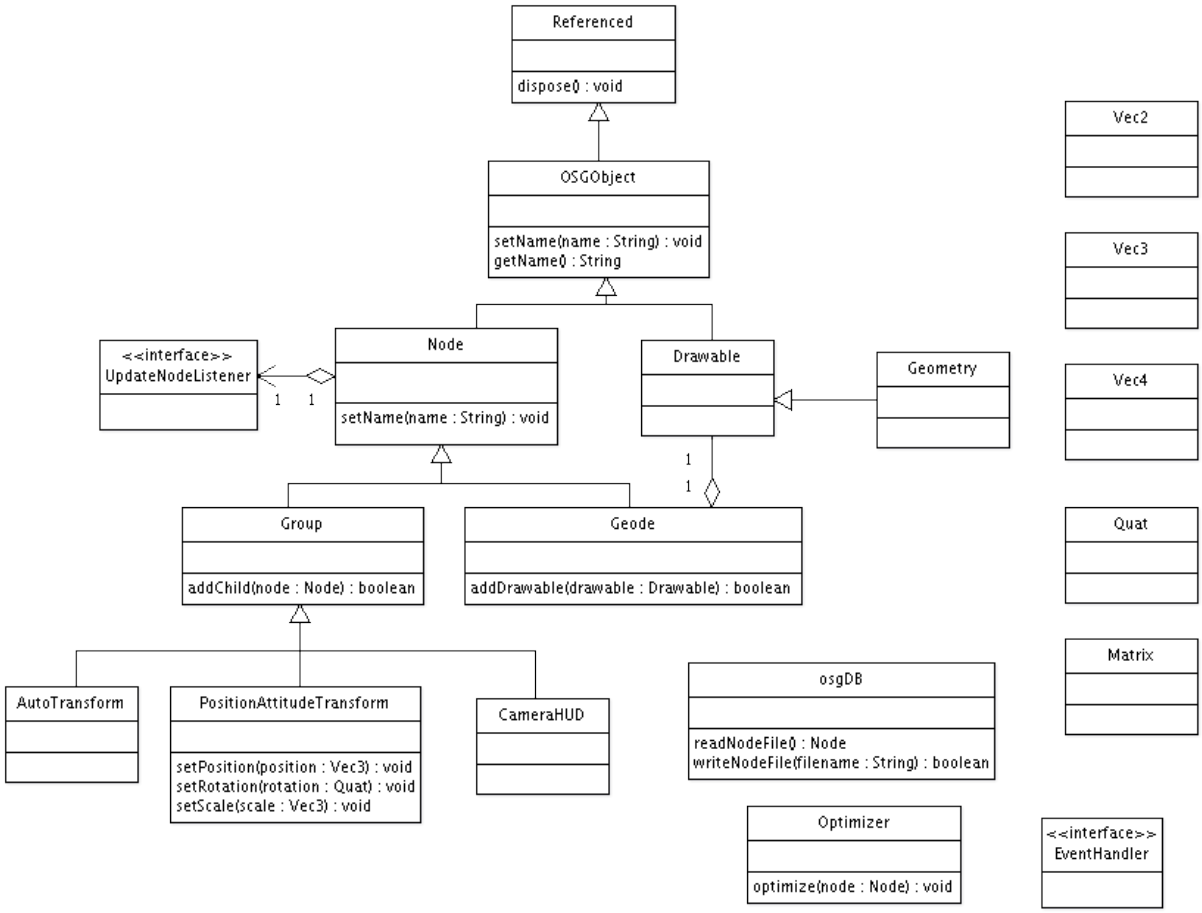
Esta libreria integra los elementos necesarios para la construccion y optimizacion del grafo de escena. Asi como una gran cantidad de herramientas matematicas (Vec{2,3,4},Matrix y Quad).

Tambien posee las clases AutoTransform y PositionAttitudeTransform que son capaces de realizar transformaciones geometricas sobre Nodos o Grupos.

Vec{2,3,4},Matrix y Quad son clases que dan soporte matematico para el manejo de vectores y para el calculo de matrices y cuaterniones.

OsgDB, clase estatica, es capaz de cargar y gruardar grafos de escena. Asi como de realizar la carga de diversos formatos soportados por OSG.

Geode (GEOmetry NoDE) y Geometry. Ambos dos son nodos que contienen geometrias y pueden ser desde una simple linea hasta un modelo tridimensional.



Viewer

La libreria osgVP-Viewer crea un visor de grafos de escena de OSG dentro de una aplicacion java, utilizando un Jpanel o un Canvas intregrado. Esta libreria utiliaza JOGL para creun un contexto de render y se apoya en la libreia nativa de C++ osg-viewer. Por lo tanto esta libreria maneja la visualizacion de escenas OSG mediante llamadas JNI.

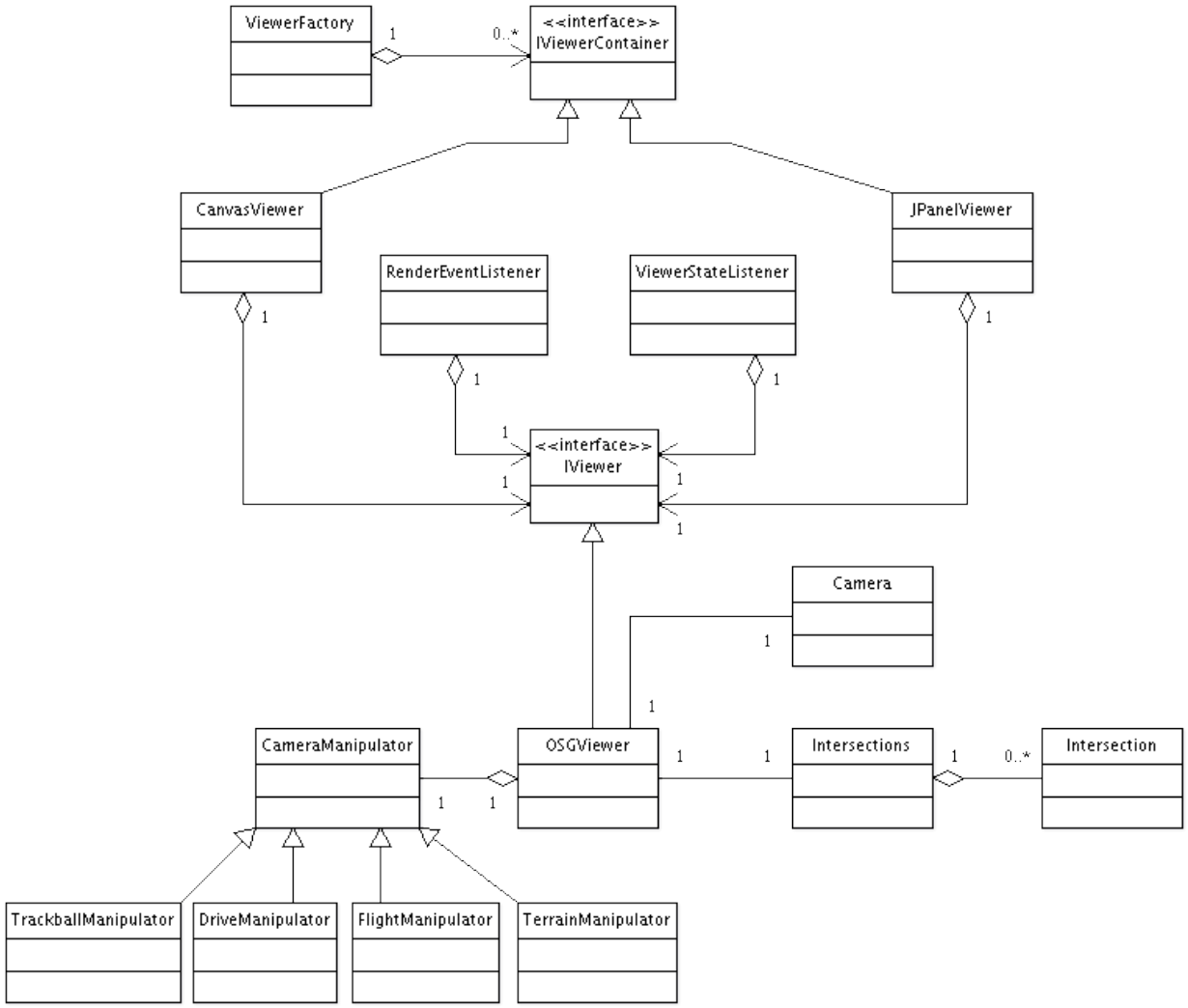
La libreria puede visualizar cualquier grafo de escena creado con OSG o bien creado desde Java.

La clase ViewerFractory, estatica, es capaz de crear un viewer de tipo Canvas (CanvasViewer) o Jpanel(JpanelViewer).

La clase camera es una simplificacion de la camara basica de OSG dejando unos metodos muy senciollos para su edicion.

Intersection clase utilizada para el almacenamiento de puntos de interseccion con la escena.

ViewerStateListener es un listener de teclado que facilita la depuracion de la visualizacion. Permitiendo: apagar/encender luces, activar/desactivar modo alambrico, etc.



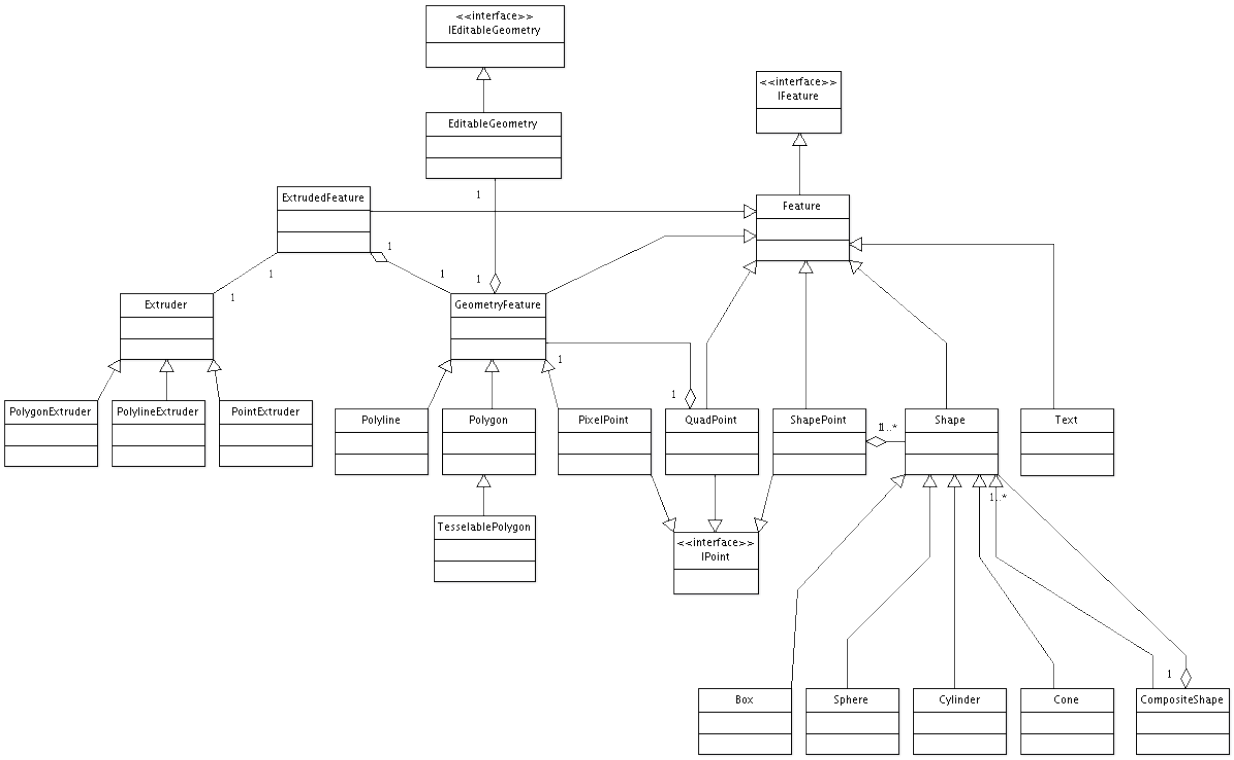
Features

OSGvp-Features es una libreria de dibujo vectorial capaz de mostrar texto, puntos, lineas, poligonos , figuras geometricas sencillas y figuras extruidas.

Todas las features soportan transparencia y cambios de color, asi comola adicion o sustraccion de vertices a la geometria existente.

La libreria esta dotada tambien de la herramientas necesarias para extruir formas geometricas difentertes tecnicas.

El manejo de la libreria Java no es directo sobre las clases OSG(como ocurre en la libreria de osgvp-core), estas clases se construyen a partir de la libreria osgvpfeatures, implementada en C++.



Planets

OSGvp-Planets permite crear grafos de escena especificos para planetas apoyandose en las utilidades que le aporta la libreria OSGvp-core. Esta libreria se encarga de exportar funcionalidad de lal ibreria josgplanets nativa de C++ utilizando la tennologia JNI.

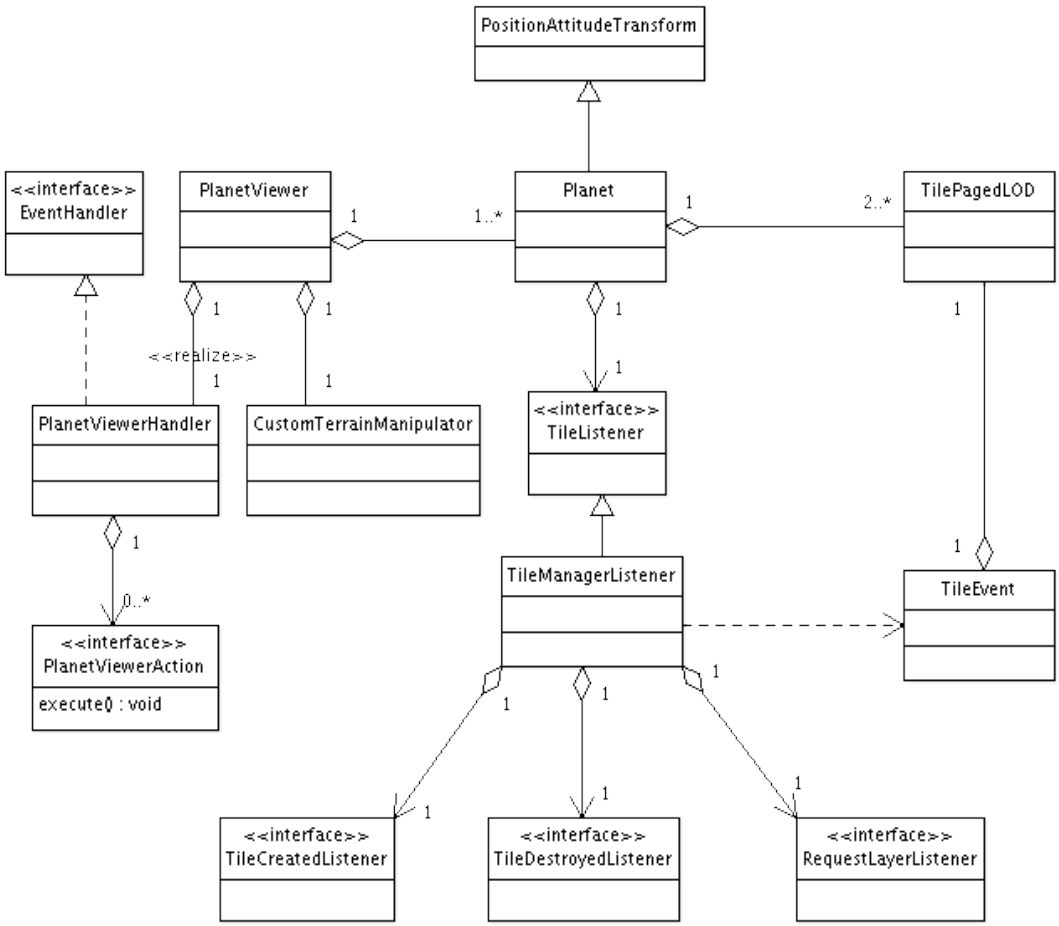
Planet es la clase para el manejo de planetas. Tiene multitud de parametros necesarios para su instanciacion. Como puede ser el nombre, tipo (plano o esferico), el sistema de coordenadas, el radio polar, radio ecuatorial, extension y su posicion en el espacio. Y ademas el planeta tiene asociado capas de texturas y MDT (modelo digital del terreno).

Un planeta esta compuesto por una serie det Tiles (baldosas) con diferentes niveles de detalle. Cada Tile esta representado por la clase TilePageLod.

La generacion de geometria para dichos tiles, asi como el manejo en memoria de dicha estructura esta manejado automaticamente por esta libreria. El manejo de capas de textura y de elevacion esta completamente controlado desde la parte de Java. Para realizar esto es necesario implementar las siguientes interfaces TileCreatedListener, TileDestroyedListener y RequestLayerListener. Las dos primeras nos permiten capturar eventos cuando se crea o se destruye un tile. Y la tercera nos permite volver a solicitar informacion para un tile ya creado.

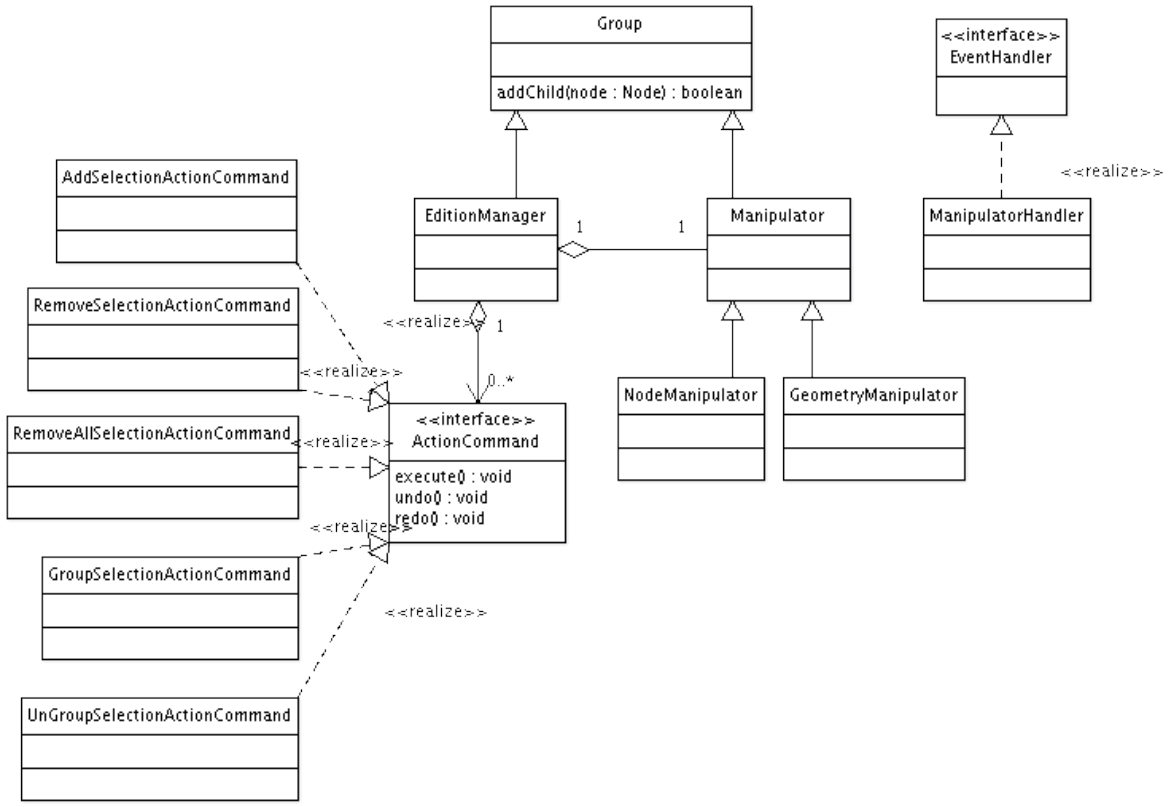
PlanetViewer es una especializacion de la clase Viewer de la libreria OSGvp-Viewer. Creada especialmente para la visualizacion de planetas. Por ejemplo: para el manejo de multiples superficies. Es posible añadir multiples planetas con difierantes capas y diferentes eleaciones.

CustomTerrainManipulator es una clase de navegacion especialmente creada para planetas. Y permite configurar el comportamiento de los eventos de teclado y raton.



Manipulators

OSGvp-Manipulators se encarga de la edicion de las transformacione asociadas a objetos 3D y tambien de las modificaciones realizadas sobre las geometrias.



Entorno de desarrollo

Paso 1
Descomprimir el fichero workShop_3D.rar (al ser posible en un directorio sin espacios).

Paso 2
Ejecutar eclipse y decirle que busque el workspace en la carpeta donde hemos descomprimido el archivo workShop_3D.rar

Paso 3
Importar los proyectos incluidos en workShop_3D. Ir a archivo->import, seleccionar el directorio workShop_3D y marcar los proyectos:

- binaries
- Sample3D

Sistema de building

Paso 1
Ir a al menu del eclipse Run-> External Tools → Open external tools dialog..
y Seleccionar la external tool “maven add-maven-repo”

Paso 2
Reiniciar el eclipse.

Paso 3
Ir a al menu del eclipse Run-> External Tools → Open external tools dialog..
y Seleccionar la external tool “maven eclipse”

Paso 5

Pinchar sobre el proyecto Sample3D y darle a F5

Y si no ha ocurrido ningun problema, ya deberiamos poder ejecutar el codigo de ejemplo.

Ejecutar el programa.

Paso 1
Ir a al menu del eclipse Run-> Open run dialog
y Seleccionar la “Encuentros UJI”

Paso 2
Y si no hay nigun problema deberia de ejecutarse el programa.