Cristian Virgili

Corso di Immagini e Multimedialità

Vito.roberto@uniud.it
Cristian.virgili@uniud.it

Che cos'è JAVA 3D?

Java 3D è una Application Programming Interface (API) per la creazione di programmi (applet) JAVA per la resa di immagini 3D

Viene utilizzato per lo sviluppo e la presentazione di contenuti 3D

- Progettato per "Write once, run anywhere"
- Multipiattaforma
- Ambienti Multi display
- Multiple input devices

Cos'è JAVA 3D?

Eleviamo la base di programmazione

- Pensiamo ad oggetti . . . *non a vertici*
- Focalizziamoci sul contenuto . . . *non sul processo di rendering*

Perché Java3D?

E' più facile di altre API si impara più in fretta
è basato su Java (non bisogna prima imparare C/C++ o qualche altro linguaggio)

Si vogliono insegnare **i principi tipici di una API grafica** non “una particolare API”
lascia nascosti molti dettagli superflui in un corso di primo livello

ESEMPIO

Compiti precedenti

Di cosa ho bisogno per sviluppare in JAVA 3D?

Editor Java Eclipse:
http://www.eclipse.org/downloads/

Libreria Java3D 1.5.2:
Per MAC: http://bit.ly/2F0CsV0
Per PC: http://bit.ly/2oxpnfB

Java 3D API:
http://download.java.net/media/java3d/javadoc/1.5.2/

Per MAC OS X 10.13+
Java 6 per mac os x
https://support.apple.com/kb/dl1572?locale=it_IT
Librerie J3d che trovate sul sito di elearning
https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2018-12/R/eclipse-inst-mac64.dmg

Java 3d: concetti BASE

Il pacchetto principale è:

- javax.media.j3d

Altri pacchetti utilizzati nei programmi Java3D:

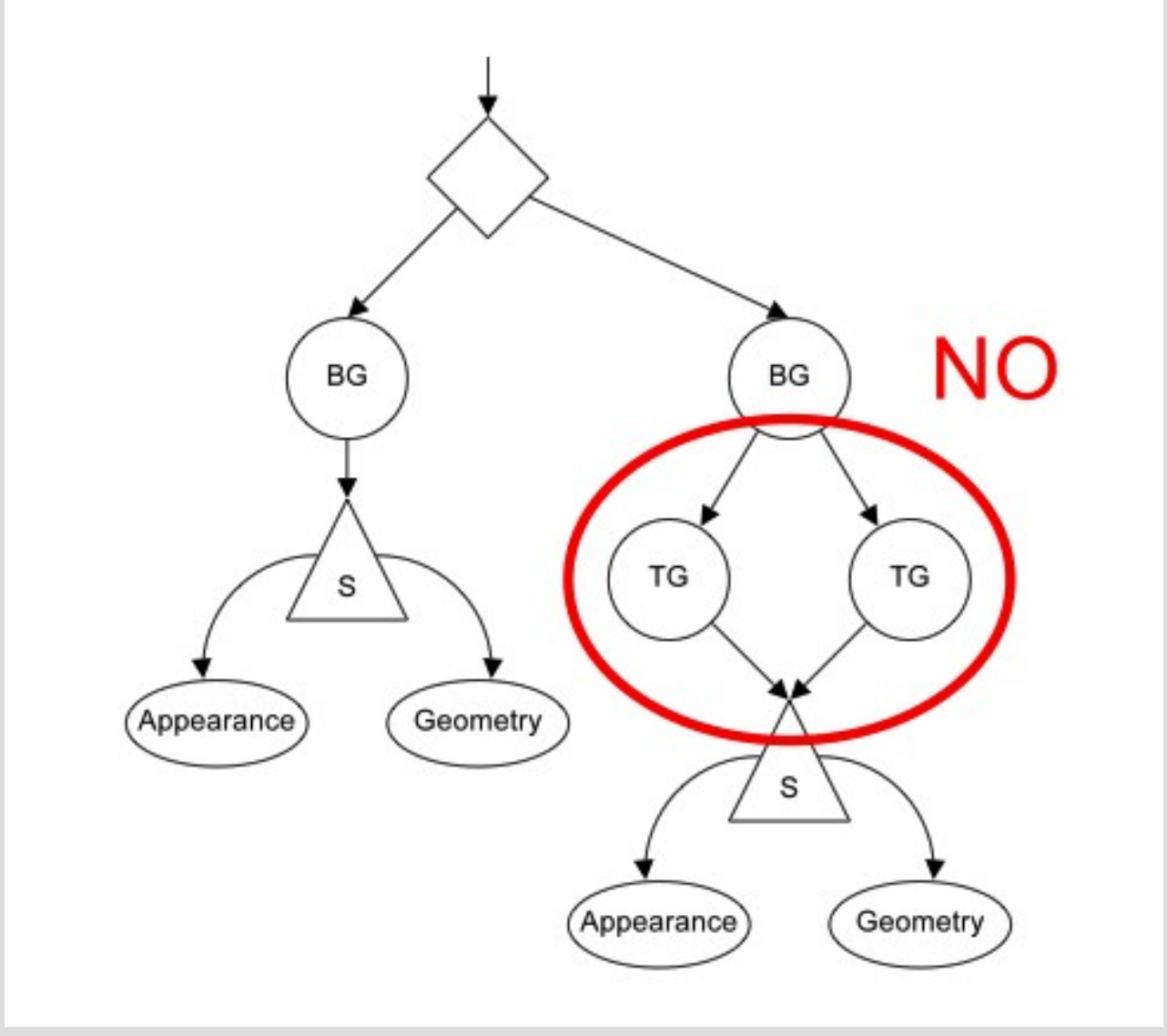
- com.sun.j3d.utils
- javax.vecmath

Java 3D: Concetti base

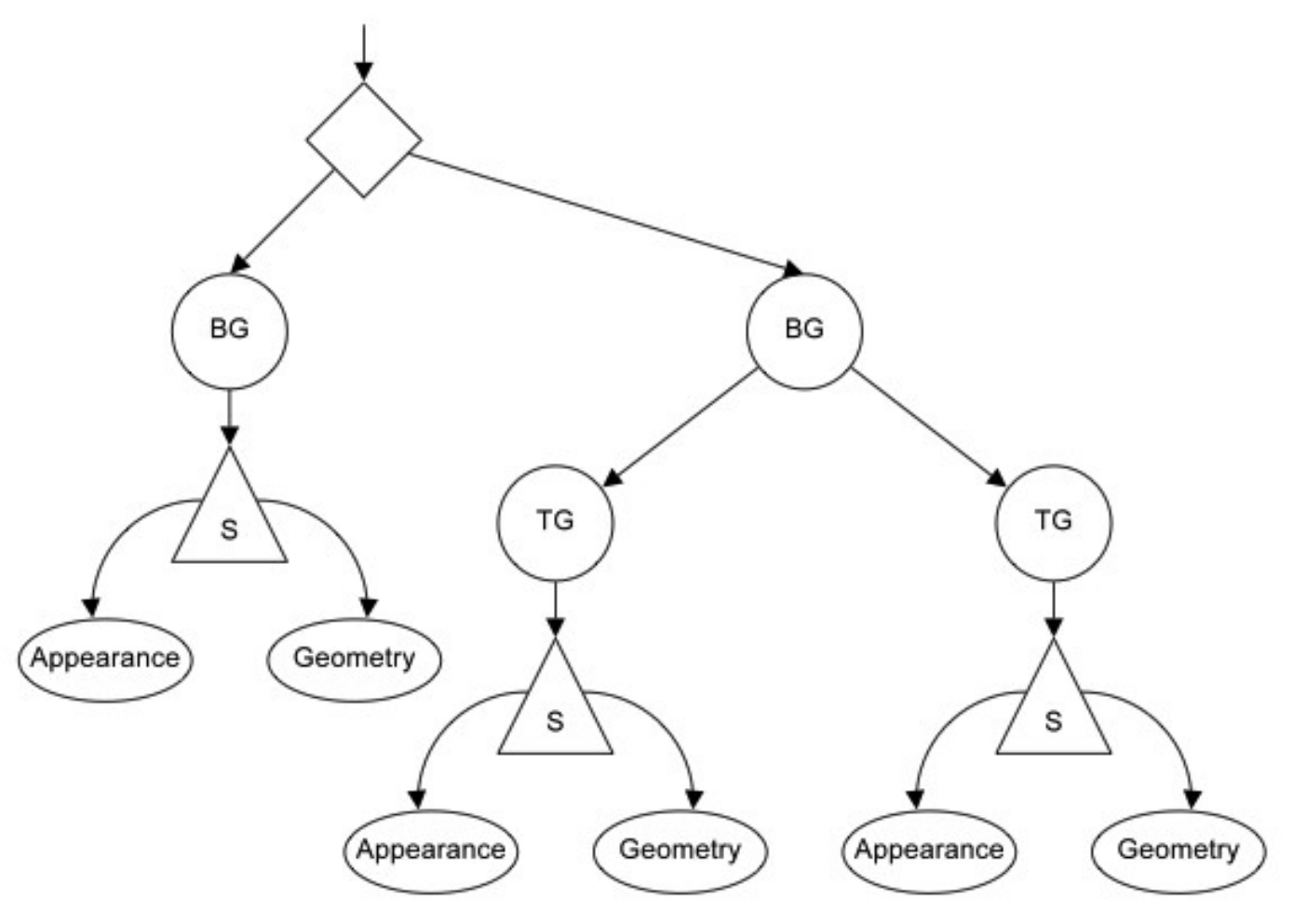
Java3D descrive i mondi 3D attraverso lo **scene graph**.

- Lo scene graph è un *grafo* diretto aciclico (DAG).
- I *nodi* contengono informazioni sugli oggetti della scena.
- Gli *archi* rappresentano relazioni tra gli oggetti.
- Lo scene graph generalmente è suddiviso in due parti:
  - una descrizione della *scena*;
  - una descrizione della *posizione* dell'osservatore nel mondo virtuale

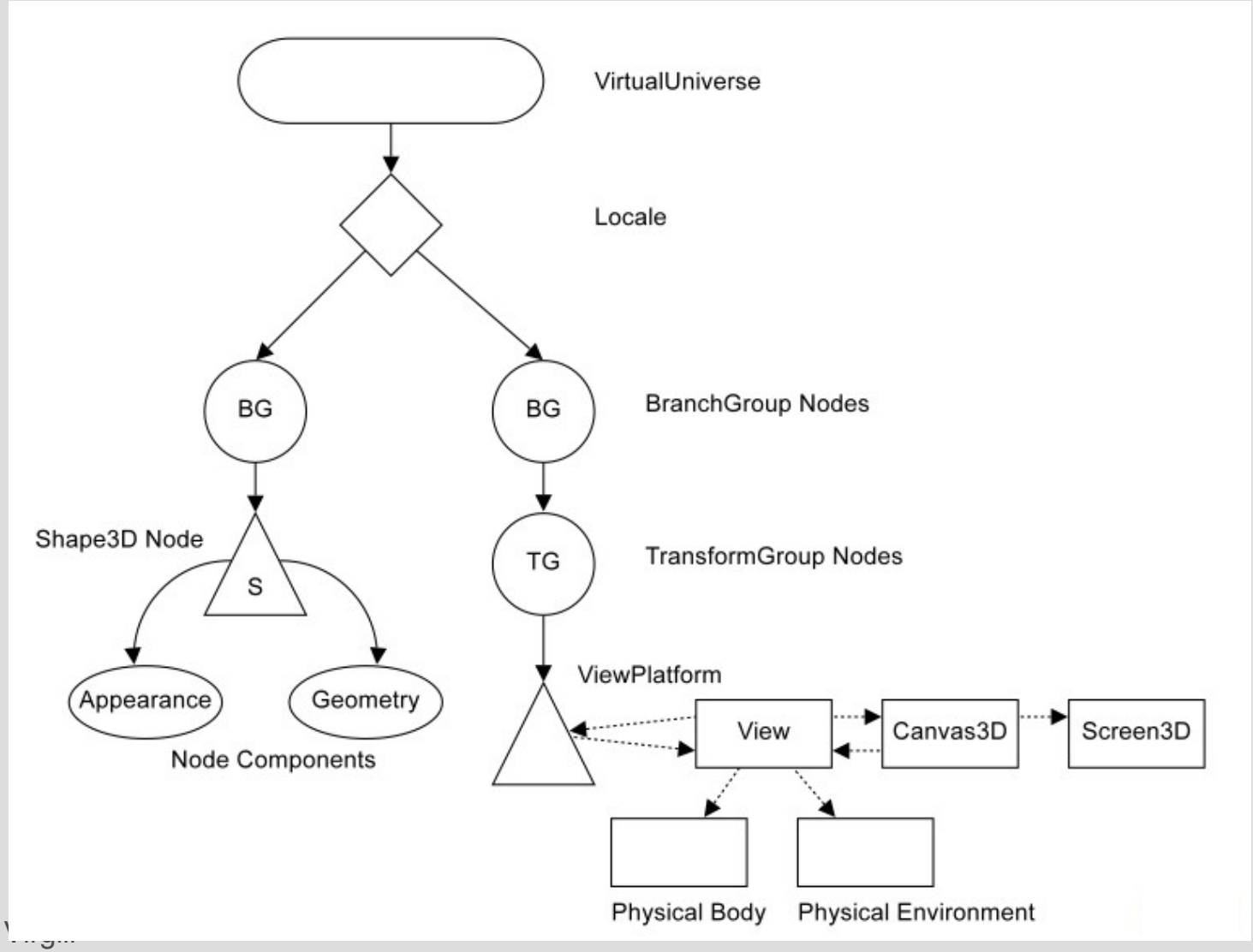
Grafo scorretto



Grafo corretto



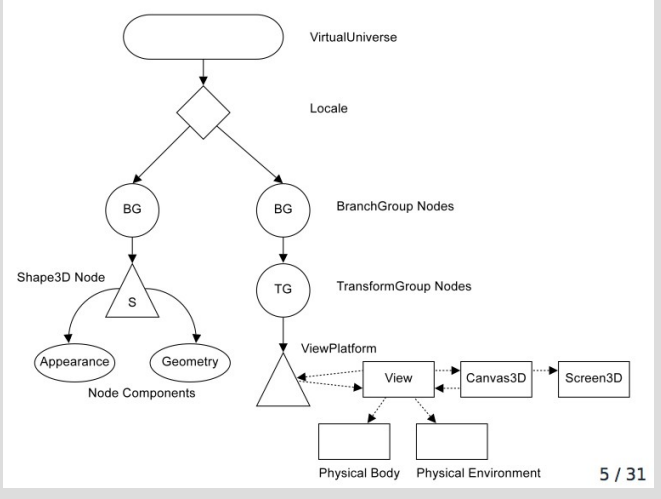
Scene Graph


VirtualUniverse
Locale
BG
BG
BranchGroup Nodes
Shape3D Node
S
TG
TransformGroup Nodes
Appearance
Geometry
Node Components
ViewPlatform
View
Canvas3D
Screen3D
Physical Body
Physical Environment

Elementi di uno Scene Graph
Virtual Universe

- Ogni applicazione Java3D ha un'istanza di una classe **VirtualUniverse**.

- È la radice dello scene graph.

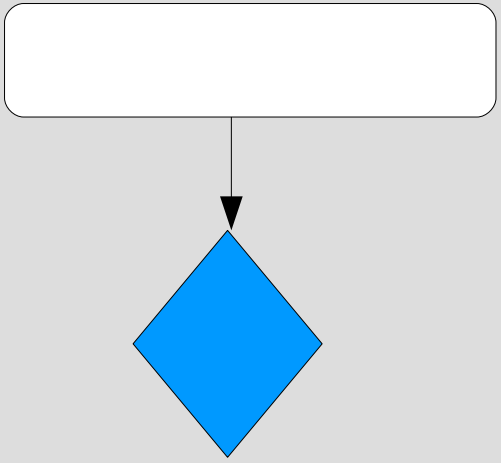


Elementi di uno Scene Graph
Locale

- Un *VirtualUniverse* contiene almeno un nodo ***Locale***.

- È un punto di riferimento nel *VirtualUniverse*: definisce l'origine in coordinate ad alta risoluzione (256 bit).

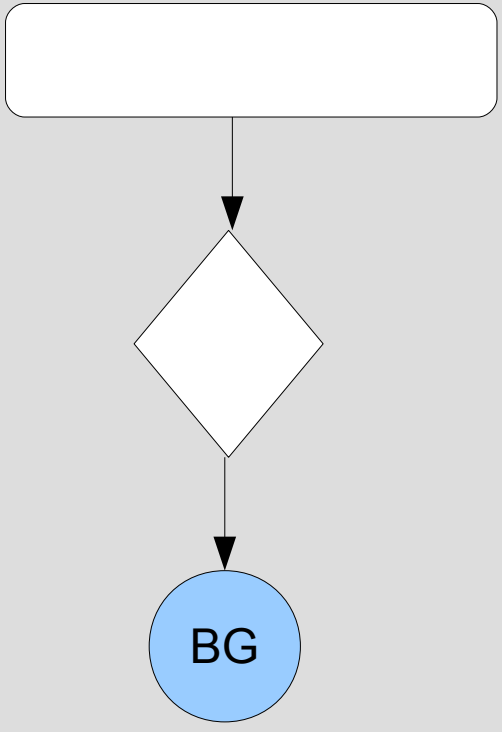
- **Funziona come contenitore (container) di una collezione di sottografi.**



Elementi di uno Scene Graph
BranchGroup

BranchGroup è l'unico *tipo* di nodo collegabile a un nodo *Locale*.

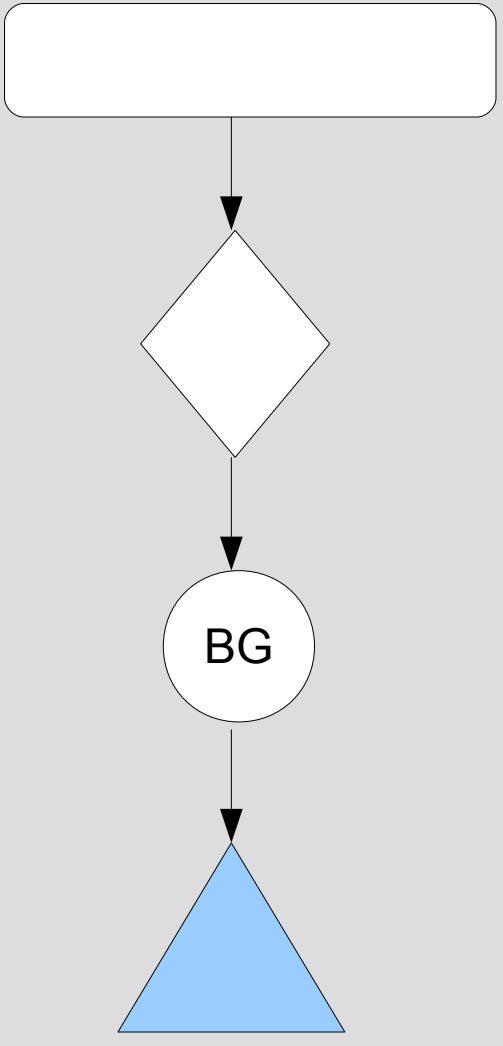
Servono a raccogliere nodi figli e a permettere l'inserimento e la rimozione in modo dinamico.



Elementi di uno Scene Graph
Leaf

Leaf è la *superclasse* usata per specificare forme, aspetti, suoni e comportamenti.

Non può avere figli, ma può avere riferimenti ad oggetti **NodeComponent.**

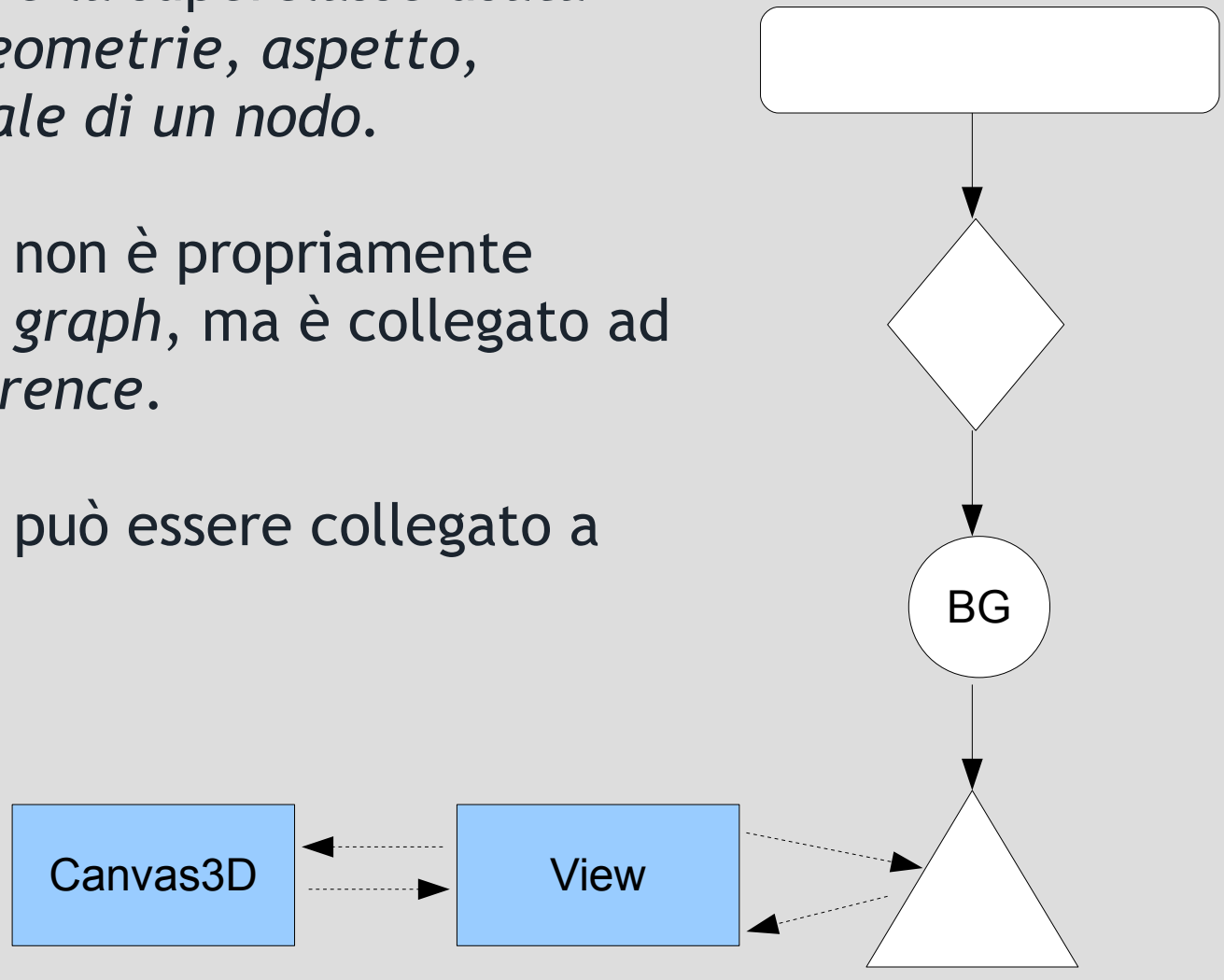


Elementi di uno Scene Graph
NodeComponent

NodeComponent è la superclasse usata per specificare *geometrie*, *aspetto*, *texture e materiale di un nodo*.

NodeComponent non è propriamente parte dello *scene graph*, ma è collegato ad esso tramite *reference*.

NodeComponent può essere collegato a più di un nodo.

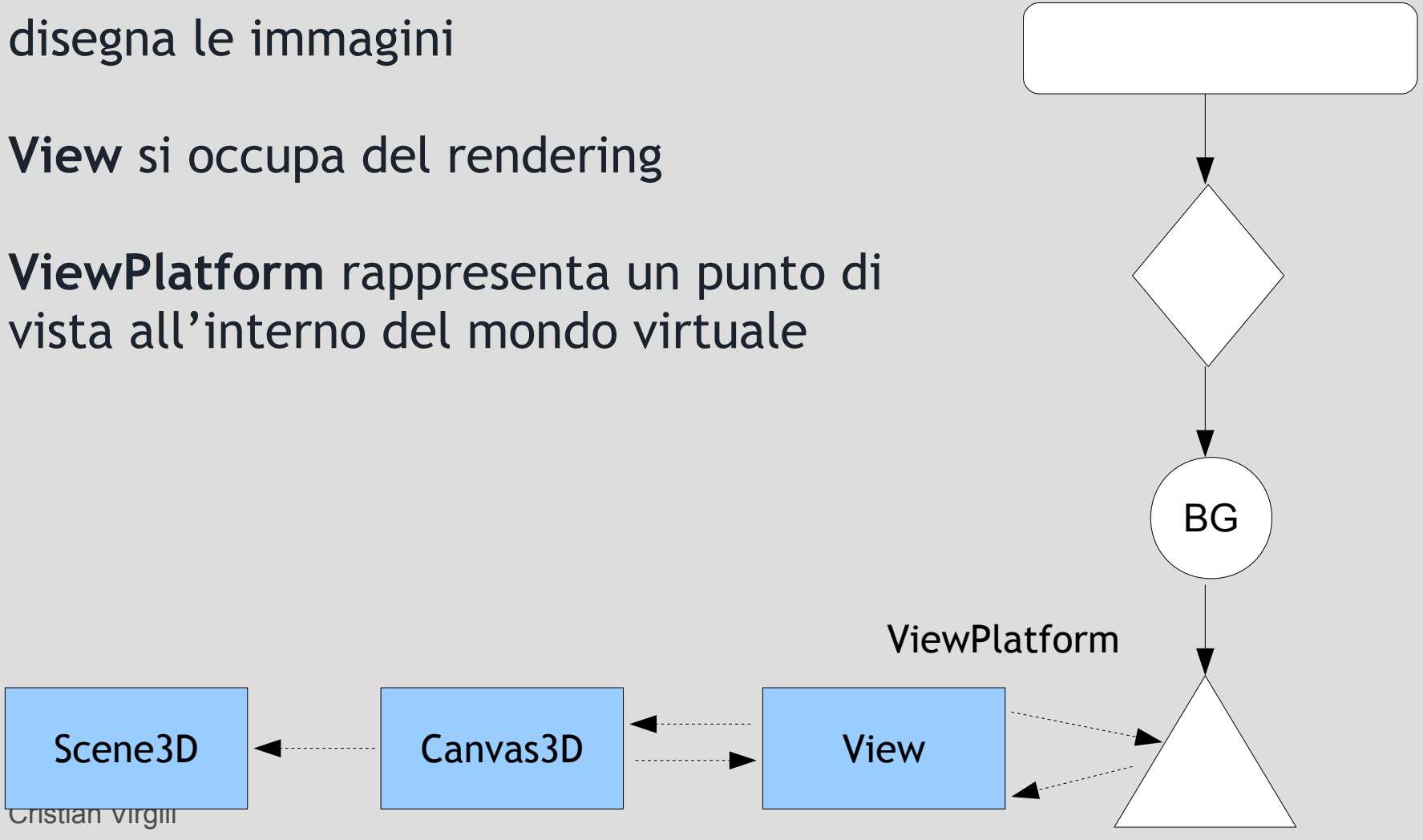


Elementi di uno Scene Graph
Canvas3D, View, ViewPlatform

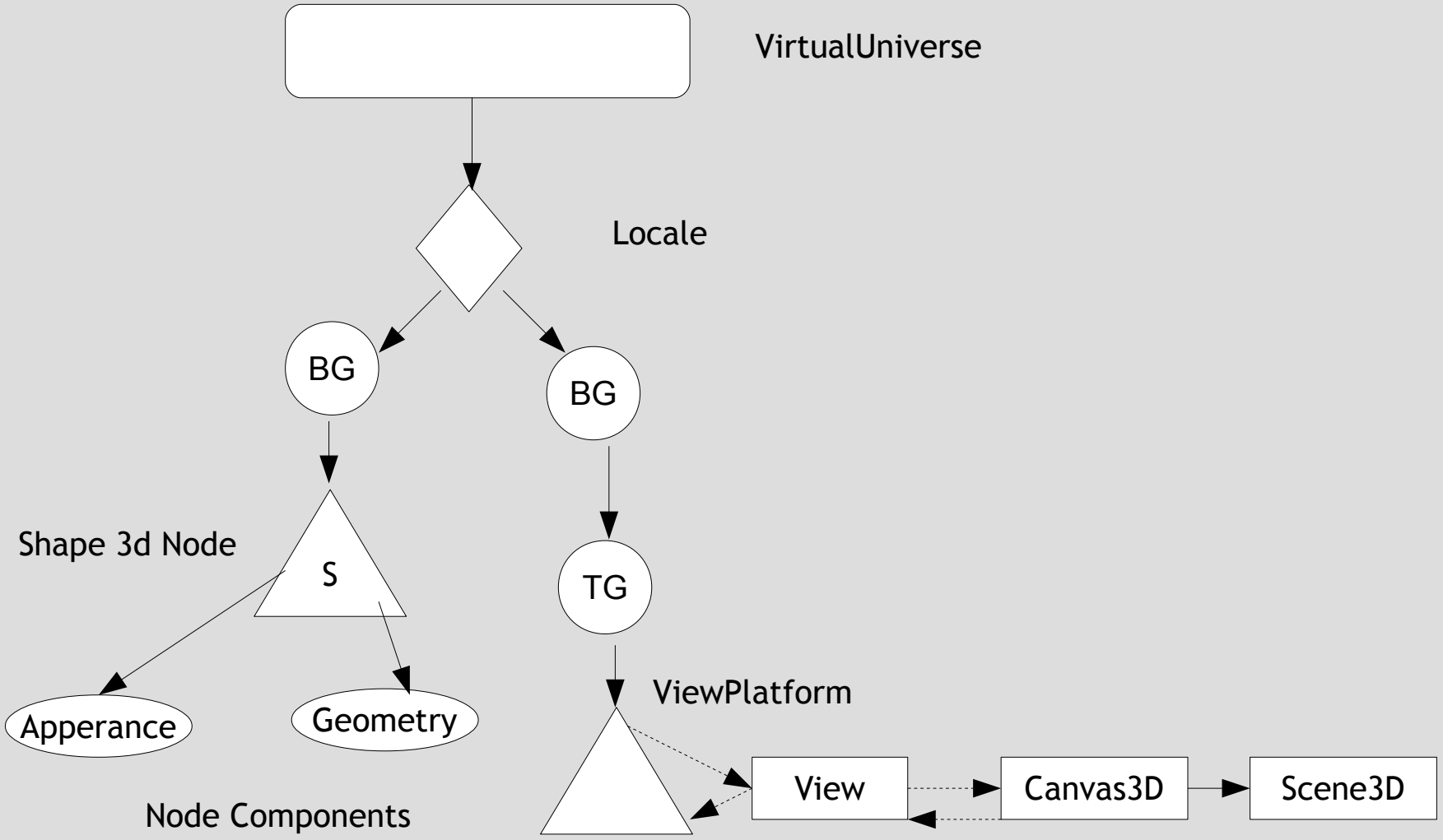
Canvas3D è la finestra in cui Java3D disegna le immagini

View si occupa del rendering

ViewPlatform rappresenta un punto di vista all'interno del mondo virtuale



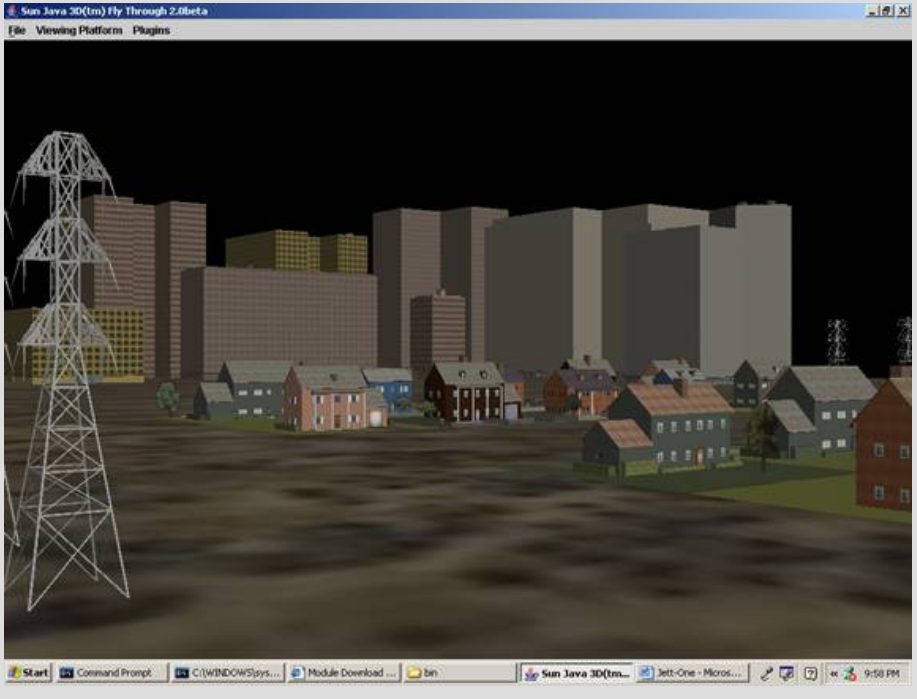
Elementi di uno Scene Graph
Esempio



Scene Graph

Scene graphs sono tipicamente suddivisi in due branch graphs:

Content branch: forme, luci, e altri contenuti
Tipicamente ci sono branches multipli per locale

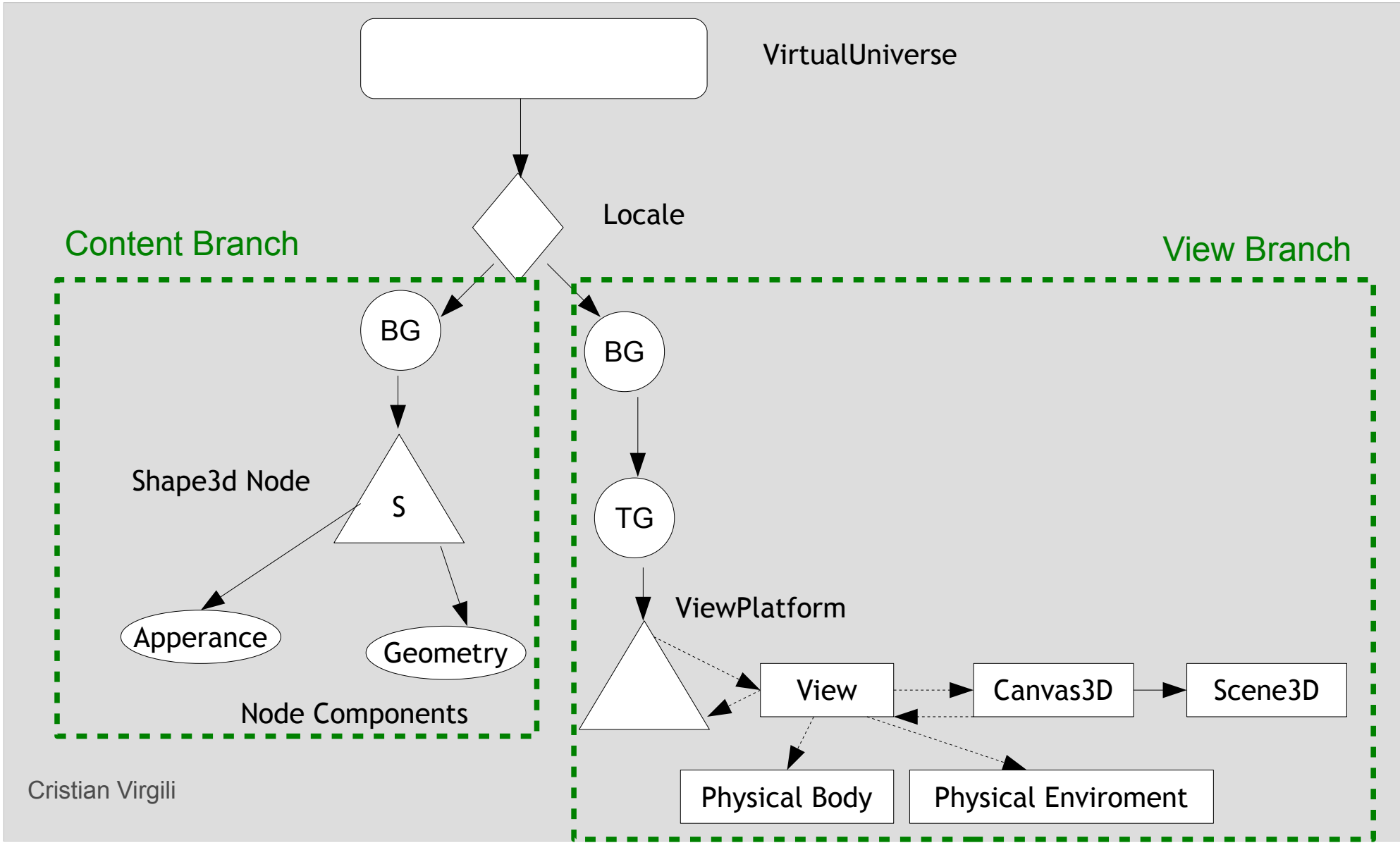


Scene Graph

e

View branch: informazioni di visualizzazione
Tipicamente **uno** per universo

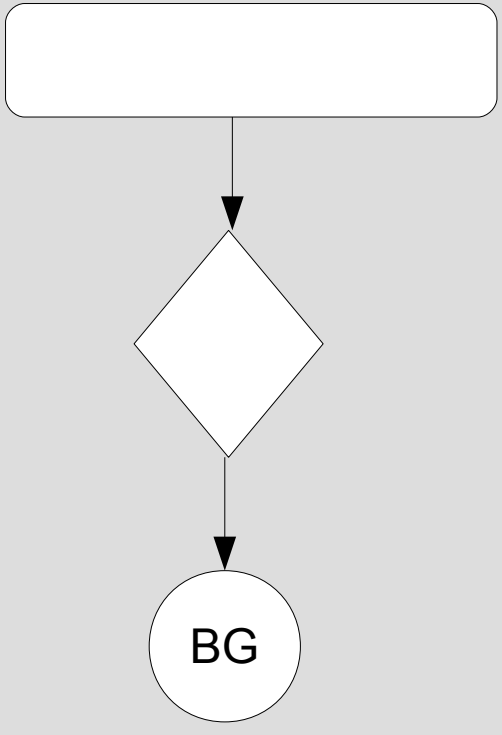
Elementi di uno Scene Graph Esempio



Costruiamo un Universo Virtuale

Gerarchia delle classi:

java.lang.Object
- **javax.media.j3d.VirtualUniverse**
- **javax.media.j3d.Locale**
- javax.media.j3d.Node
  - javax.media.j3d.Group
  - **javax.media.j3d.BranchGroup**



Costruiamo un Universo Virtuale
Content Branch

Costruiamo un universe

```
VirtualUniverse myUniverse = new VirtualUniverse( );
```

Costruiamo un locale

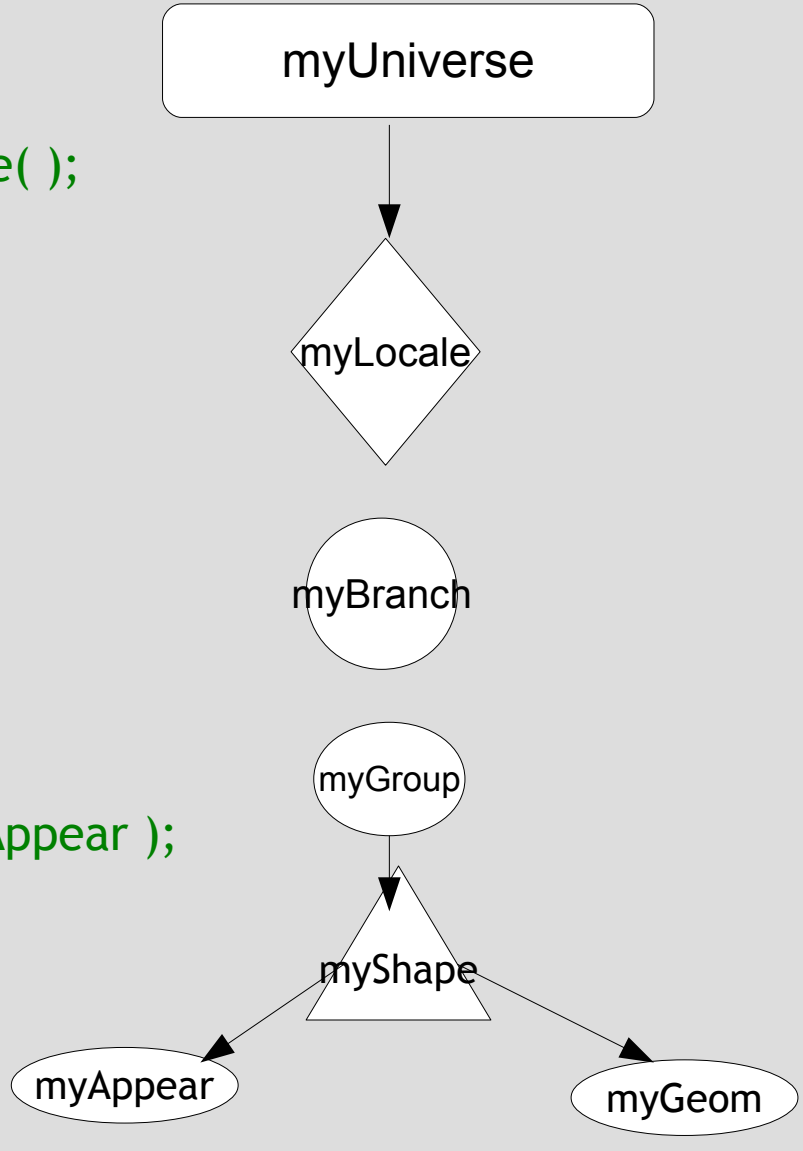
```
Locale myLocale = new Locale( myUniverse );
```

Costruiamo un branch group

```
BranchGroup myBranch = new BranchGroup( );
```

Costruiamo nodi e gruppi di nodi

```
Shape3D myShape = new Shape3D( myGeom, myAppear );
Group myGroup = new Group( );
myGroup.addChild( myShape );
```



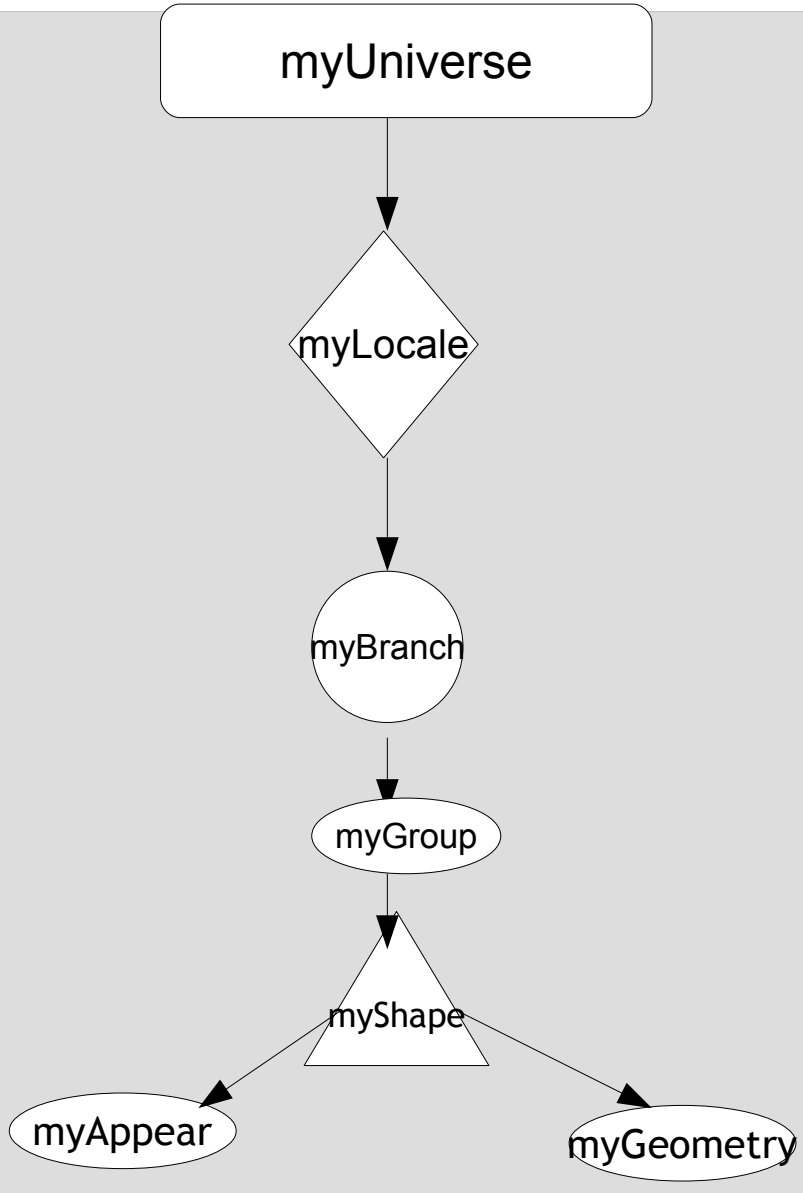
Costruiamo un Universo Virtuale
Content Branch

Aggiungiamoli al BG

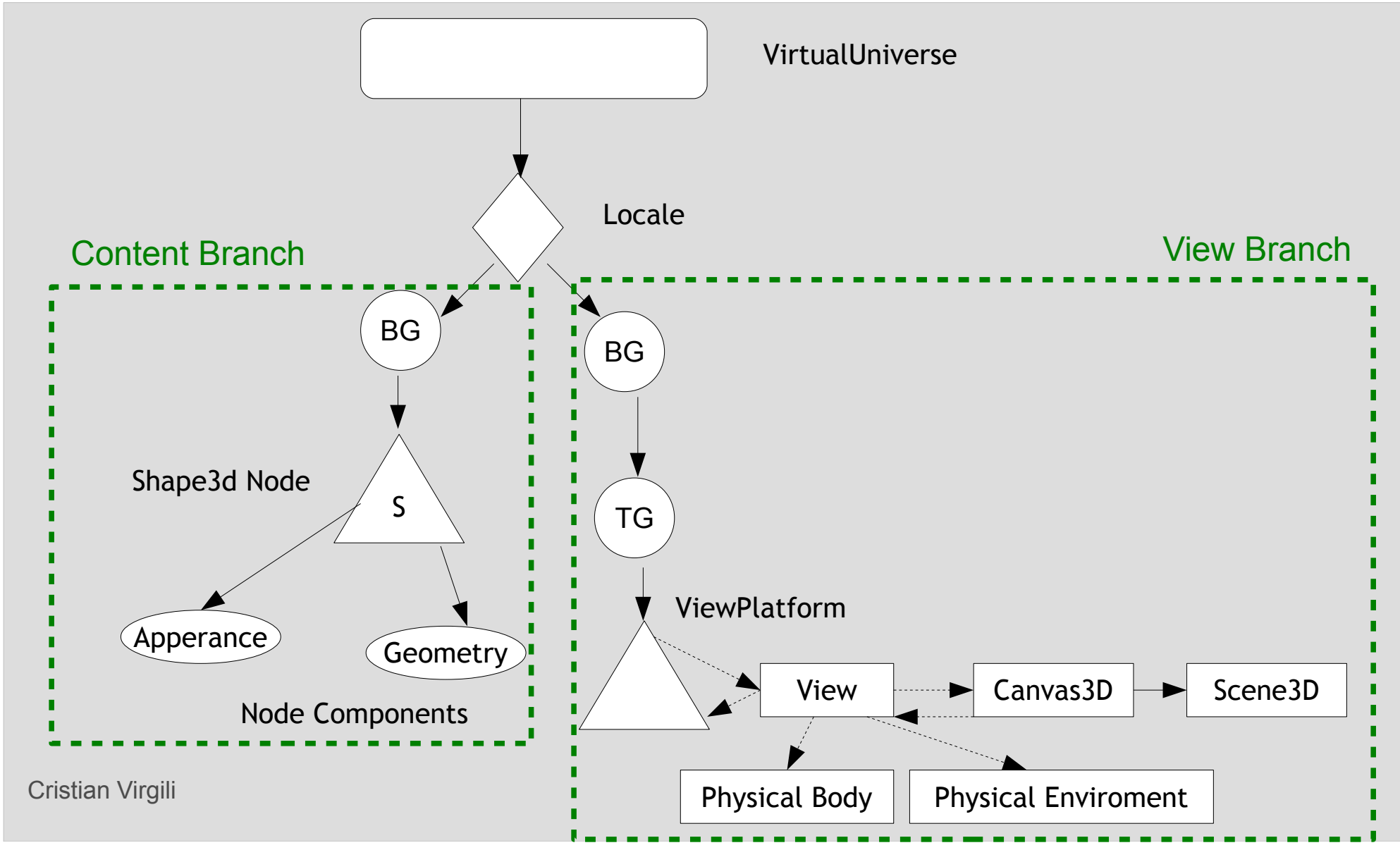
```
myBranch.addChild( myGroup );
```

Aggiungiamo il BG al Locale

```
myLocale.addBranchGraph( myBranch );
```



Scene Graph
Esempio



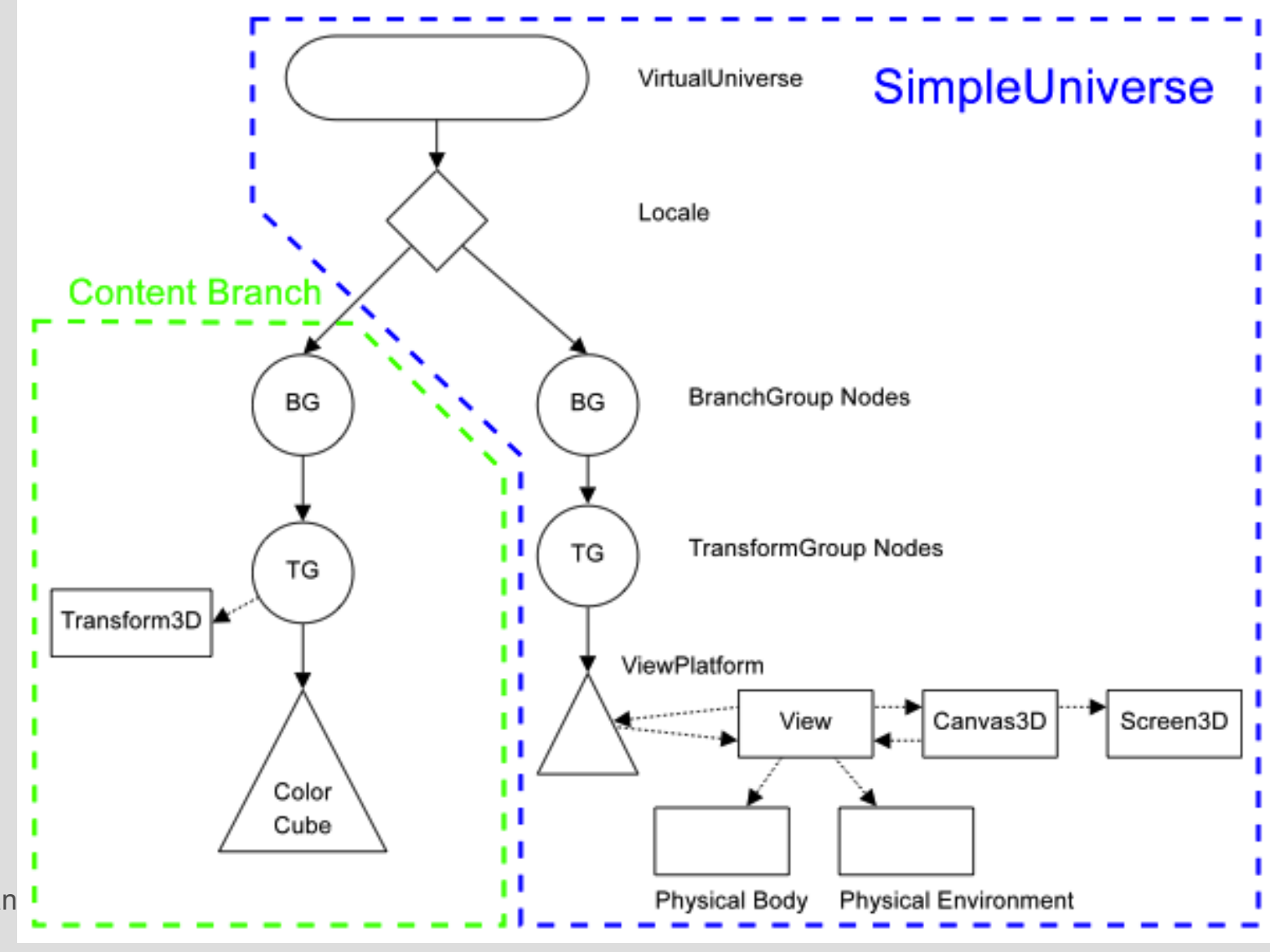
View Branch
(Il SIMPLE UNIVERSE)

Ricapitolando: quando vogliamo creare un universo 3D ex novo dobbiamo istanziare un Virtual Universe, uno o più oggetti Locale, un **View Branch Graph** (con tutti gli oggetti, le proprietà ed i metodi collegati), un Content Branch Graph e tutti gli elementi che compongono la scena 3D...

...troppo complicato ?!?

In effetti sì, ed è per questo che Java 3D mette a disposizione una utility, una classe chiamata **SimpleUniverse**, che può fare la maggior parte del lavoro per noi, occupandosi di Virtual Universe, Locale e View Branch Graph, lasciandoci liberi di operare sul Content Branch Graph.

Il SIMPLE UNIVERSE



Simple Universe

Il risparmio, in termini di tempo e di codice, è evidente se si confrontano i passi necessari per creare una scena 3D da zero e con l'ausilio del **SimpleUniverse**; da zero, infatti, dovremmo:

- creare una Canvas3D ed un Virtual Universe, e collegarli tra di loro;
- creare un Locale e collegarlo al Virtual Universe;
- costruire un View Branch Graph, collegandogli gli oggetti View, View-Platform, PhysicalBody, PhysicalEnvironment ed impostandone i parametri;
- costruire uno o più Content Branch Graph;
- compilare tutti i Branch Graph;
- collegare i Content Branch Graph al Locale;

mentre, con l'aiuto di SimpleUniverse, sarà sufficiente:

- creare una Canvas3D ed un SimpleUniverse, e collegarli tra di loro;
- (OPZIONALE) personalizzare il SimpleUniverse;
- costruire uno o più Content Branch Graph;
- compilare tutti i Branch Graph;
- collegare i Content Branch Graph al Locale del SimpleUniverse (già presente, in quanto creato implicitamente dal SimpleUniverse).

Configuriamo ECLIPSE

Scarichiamo e installiamo una USER LIBRARIES in Eclipse.

Libreria Java3D 1.5.2:
Per MAC: http://bit.ly/2F0CsV0
Per PC: http://bit.ly/2oxpnfB
Java 3D API:
http://download.java.net/media/java3d/javadoc/1.5.1/index.html

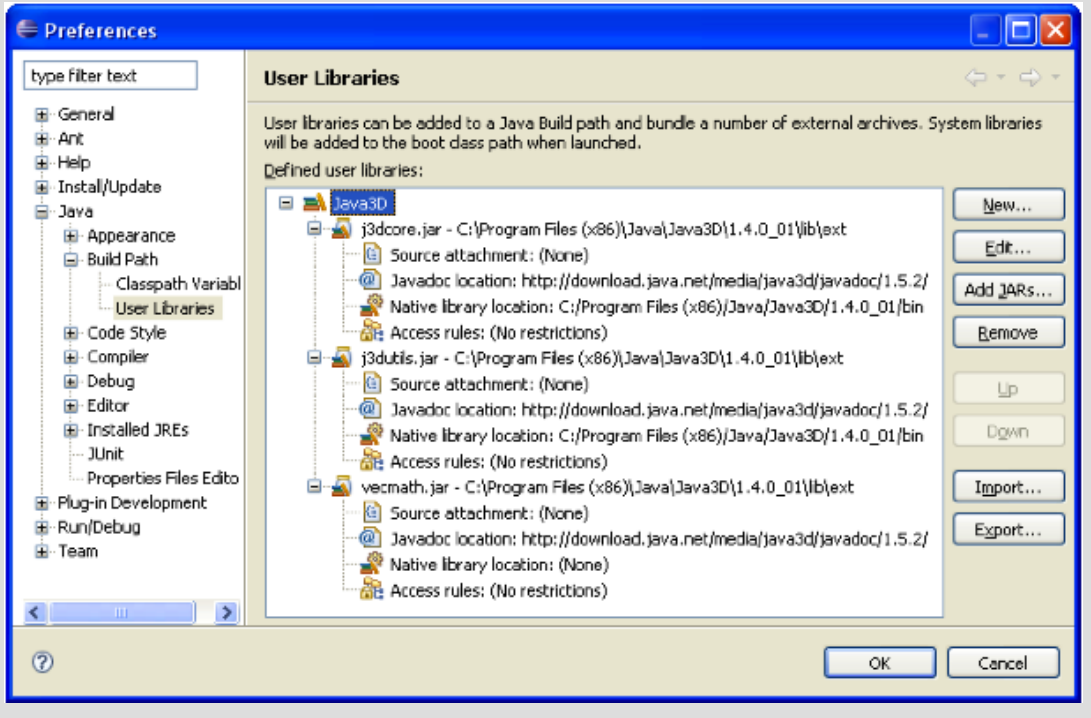
Per MAC OS X 10.13+
Java 6 per mac os x
https://support.apple.com/kb/dl1572?locale=it_IT

Librerie J3d che trovate sul sito di elearning

https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2018-12/R/eclipse-inst-mac64.dmg

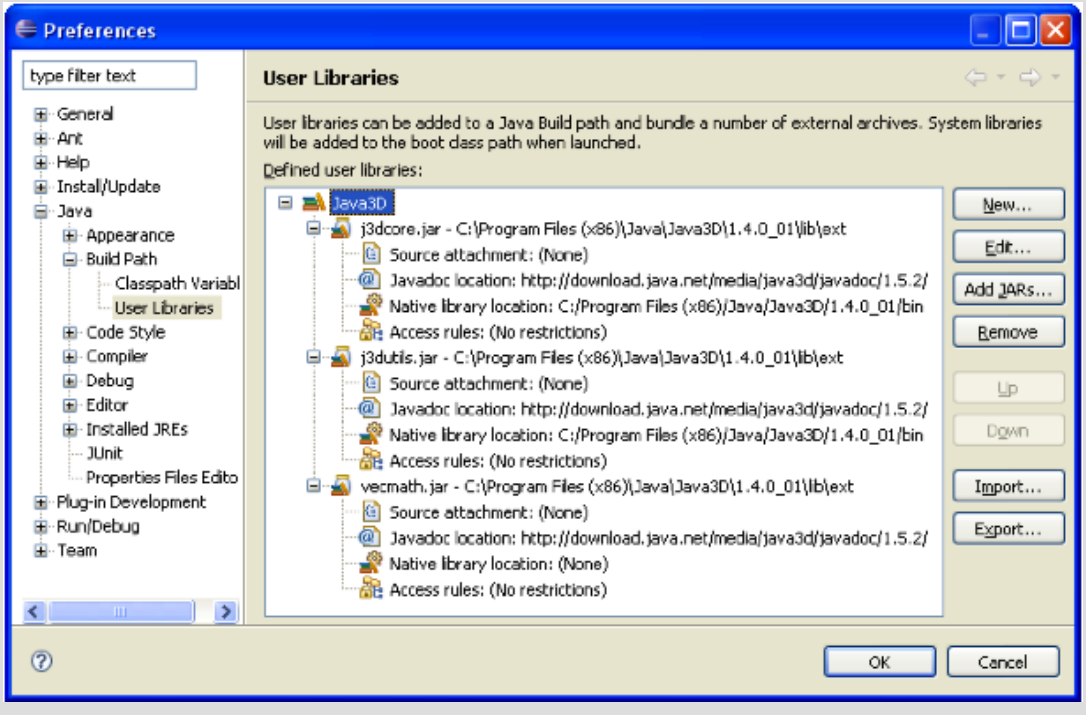
Configuriamo il nostro IDE per l'utilizzo di JAVA 3D

Java 3D API:
http://download.java.net/media/java3d/javadoc/1.5.2/index.html



Prima applicazione

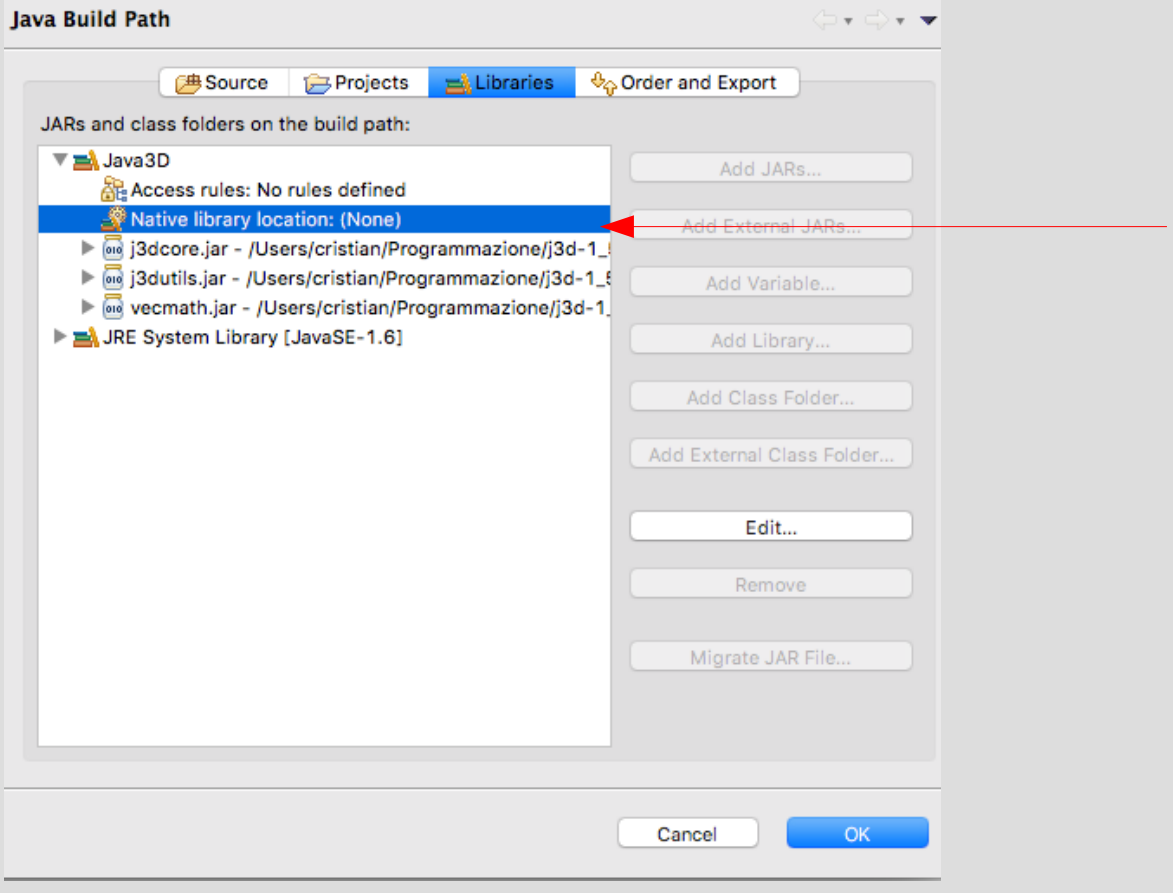
Utilizziamo Eclipse per la costruzione della nostra applicazione JAVA 3d.
Nella figura sottostante viene mostrato un esempio di come deve essere impostato Eclipse.
I percorsi ovviamente dipendono dal sistema operativo in uso



Ulteriori configurazioni

JAVA 6 per MAC OS X: https://support.apple.com/kb/dl1572?locale=it_IT

Per windows se non potete mettere le DLL in system32, utilizzate il campo “native library”



Prima applicazione
Struttura

1. *Creare un oggetto Canvas3D.*

2. *Creare un oggetto SimpleUniverse che faccia riferimento all'istanza di Canvas3D.*

3. *Personalizzare l'oggetto SimpleUniverse.*

4. *Costruire un BranchGroup e popolarlo con il contenuto dell'universo (content branch).*

5. *Compilare il content branch.*

6. *Inserire il content branch nell'oggetto Locale del SimpleUniverse.*

Prima Applicazione

```
import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.GraphicsConfiguration;
import javax.media.j3d.BranchGroup;
import javax.media.j3d.Canvas3D;
import javax.media.j3d.Transform3D;
import javax.media.j3d.TransformGroup;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.geometry.ColorCube;
import com.sun.j3d.utils.universe.SimpleUniverse;

public class HelloJava3D extends Applet {

    public HelloJava3D() {
        setLayout(new BorderLayout());
        Transform3D t = new Transform3D();
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        add("Center", canvas3D);
        BranchGroup scene = createSceneGraph();
        scene.compile();

        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewingPlatform().setNominalViewingTransform();
        simpleU.addBranchGraph(scene);
    }
     // Funzione che crea il sottografo
     public BranchGroup createSceneGraph() {
       BranchGroup node = new BranchGroup();
       TransformGroup TG = createSubGraph(); //funzione implementata successivamente
       node.addChild(TG); //aggiunge l'oggetto TG come figlio del BrachGruop
      return node;
     }

      public TransformGroup createSubGraph(){

            TransformGroup transform = new TransformGroup(); // crea oggetto TG
            transform.addChild(new ColorCube(0.3));  //aggiungo al TG come figlio il cubo
            return transform;
         }

    public static void main(String[] args) {
        new MainFrame(new HelloJava3D(), 1024, 768);
    }
```

GROUP: raggruppamenti

Group è la superclasse di tutti i nodi di raggruppamento.

Ha esattamente un genitore, ma un numero arbitrario di figli.

Permette di aggiungere, togliere ed enumerare i figli.

GROUP: raggruppamenti

TransformGroup estende la classe Group.

Applica una determinata trasformazione a tutti i suoi figli.

Le trasformazioni vengono indicate da un oggetto *Transform3D*.

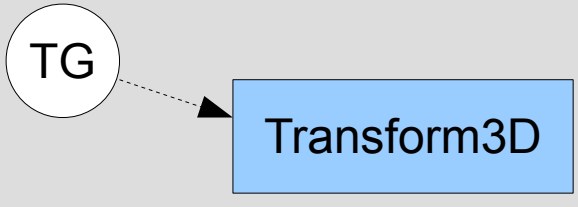
Trasformazioni

Trasform3D descrive una trasformazione nello spazio 3D.

Non è un figlio di *TransformGroup*.

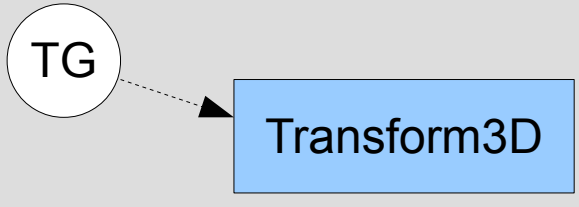
Permette di esprimere rotazioni, traslazioni, scalature, etc. (internamente è rappresentata da una matrice 4×4).

Prevede una serie di metodi per la creazione e la manipolazione delle trasformazioni.



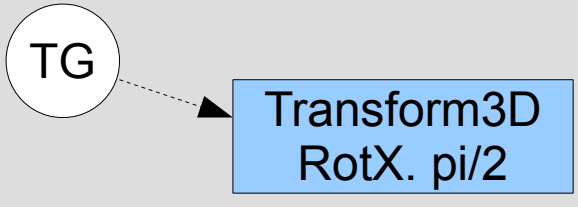
Trasformazioni esempio 1

```
//Dichiarazione e  istanziazione  dei
//nodi TransformGroup e   Transform3D

TransformGroup tg = new TransformGroup ( ) ;
Transform3D t3d = new Transform3D ( ) ;

// Angolo di   rotazione in radianti

double alpha = Math.PI/2;

// Creazione della matrice  di rotazione .

Matrix3d rotX = new Matrix3d(
   1.0d ,0.0d ,0.0d ,
   0.0d, Math.cos(alpha), -Math.sin(alpha),
   0.0d, Math.sin(alpha), Math.cos(alpha));

// Impostazione della della Matrice di rotazione.

t3d.set(rotX);

// Impostazione trasformazione.

tg.setTransform(t3d);
```



Trasformazioni esempio 2

```
//Dichiarazione e istanziazione dei nodi
// TransformGroup e Transform3D.

TransformGroup tg = new TransformGroup() ;
Transform3D t3d = new Transform3D ( ) ;

// Angolo di rotazione in radianti.

t3d.rotX( Math.PI/2);

//Impostazione della trasformazione.

tg.setTransform ( t3d ) ;
```



Transform3D

Costruttori:

```
Transform3D ( )
Transform3D(double[] matrix)
Transform3D(Matrix4d m1)
Transform3D(Transform3D t1)
```

Metodi:

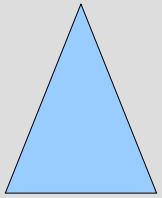
```
SetIdentity ()
setZero ( )
add(Transform3D t1)
sub(Transform3D t1)
invert()
transpose ( )
mul(double scalar)
mul(Transform3D t1)
normalize ( )
rotX(double angle)
setTranslation (Vector3d trans )
```

Leaf

Leaf è un nodo che non può avere figli (una foglia).

I nodi di tipo Leaf specificano:

- luci;
- forme;
- suoni.



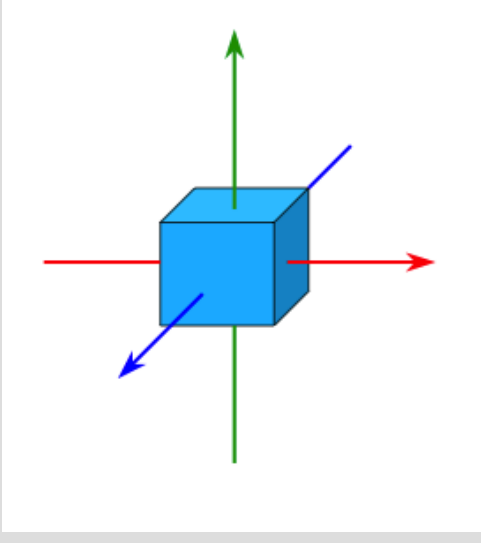
ColorCube (Leaf)

Cubo con facce di diversi colori.

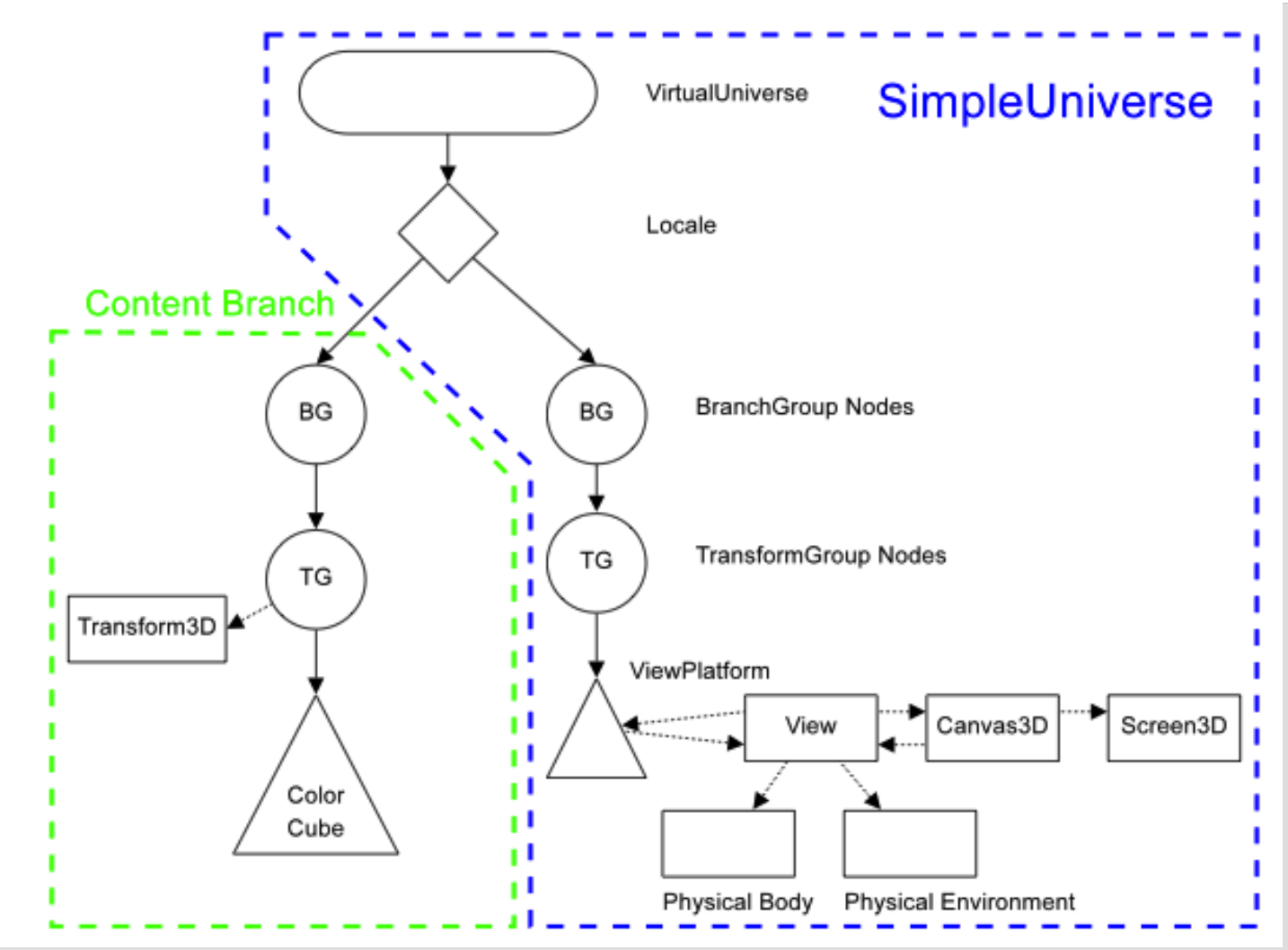
Centro in (0, 0, 0).

Lato di lunghezza 2, va da (−1, −1, −1) a (1, 1, 1).

Si può indicare un coefficiente di scalatura nel costruttore.



Esempio di un scene graph



Esempio di trasformazione

```
BranchGroup root = new BranchGroup();

Transform3D rotate = new Transform3D();
rotate.rotX(Math.PI/4.0d);


Transform3D rotateY = new Transform3D();
rotateY.rotY(Math.PI/4.0d);

//combinazione delle due trasformazioni
rotate.mul(rotateY);


TransformGroup rotateG = new TransformGroup(rotate);
rotateTG.addChild(new ColorCube(0.4));
root.addChild(rotateTG);
```

Risultato esempio