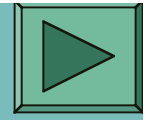


Università degli Studi di Camerino

Dipartimento di Matematica e Informatica

GRAFICA COMPUTAZIONALE

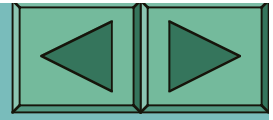
“Java/Java 3D”

Corso di laurea in Matematica / Informatica

Prof. Riccardo Piergallini Dott. Pennesi Roberto

A. A. 2001-2002

Linguaggio JAVA



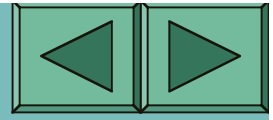
- Progetto di ricerca della Sun Microsystems (1992).
- Ambiente operativo snello e affidabile.
- Portabile e distribuito in tempo reale.
- Finalizzato ai servizi di rete.

SITO: http://java.sun.com ("Java Tutorial")

Caratteristiche Principali:

- Semplicità (simile a C++);
- Object Oriented
- Linking Dinamico (JVM)
- MultiThread
- Portatilità (Bytecode)
- Gestione della Memoria (GC)
- Gestione Eventi
- Sicurezza

Linguaggio Object Oriented

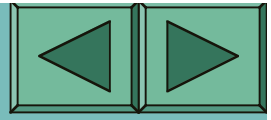


Oggetto: Modello di programmazione software, il quale ha uno stato nelle variabili istanza e un comportamento definito dai metodi

Classe: è un prototipo (costrutto software), che definisce le variabili e i metodi comuni a tutti gli oggetti di un certo tipo

Requisiti:

- Incapsulamento: dare la possibilità di rendere riservata o astratta parte dell'informazione;
- Polimorfismo: Stessi messaggi inviati a oggetti differenti producono effetti diversi in base all'oggetto coinvolto;
- Ereditarietà: Si possono definire (estendere) classi sulla base di altre classi;
- Coll. Dinamico: gli oggetti potrebbero provenire da qualsiasi parti; spedire messaggi ad oggetti senza conoscerne il tipo.



Ambiente di Lavoro

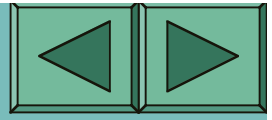
- Hardware:

  Personal Computer (Pentium, 32MB Ram)
  Scheda video SVGA (con almeno 1 Mbyte)

- Software:

  - Windows (95, 98, NT, ME, ecc)
  - DirectX / OpenGL
  - Java 2 JDK ver. 1.3.*
  - Java3D API ver. 1.2.*
  - RealJ (opzionale)

Esempi (Java):



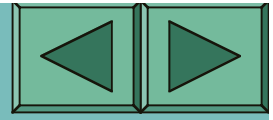
Applicazione

```
/**
 * The HelloWorldApp class implements an application that
 * simply displays "Hello World!" to the standard output.
 */
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); //Display the string.
    }
}
```

Applet

```
import java.applet.Applet;
import java.awt.Graphics;

public class HelloWorld extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello world!", 50, 25);
    }
}
```



Promemoria: Creazione …. Esecuzione:

Creazione (Finestra Prompt di Dos)

c:\mysource-java> **edit** nomeapp.java

Compilazione

c:\mysource-java> **javac** nomeapp.java

NB. 1) Il path deve contenere "jdk1.3.1_01\bin"
2) La variabile amb.: classpath=c:\jdk1.3.1_01\lib;.;

Esecuzione

c:\mysource-java> **java** nomeapp

Help: http://java.sun.com/docs/books/tutorial/getStarted/cupojava/win32.htm

Considerazioni

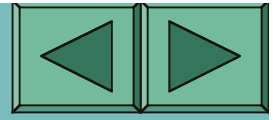
Specificatori di accesso:

-**Public**: visibilità totale.
-**Private**: visibilità limitata alla classe.
-**Protected**: visibilità alla gerarchia di classe e al package.
(Default : visibilità limitata al package)

Package principali del JDK:

- **java.lang**: classi base del linguaggio(Object, System, …)
- **java.io**: classi per la gestione input/output.
- **java.util**: classi di utilità (Date, Random,…).
- **java.net**: classi di supporto alle applicazioni di rete (Url, ..).
- **java.applet**: classe Applet, ….
- **java.awt**: Abstract Windowing Toolkit.

Help: C:\jdk1.3.1_01\docs\api\index.html

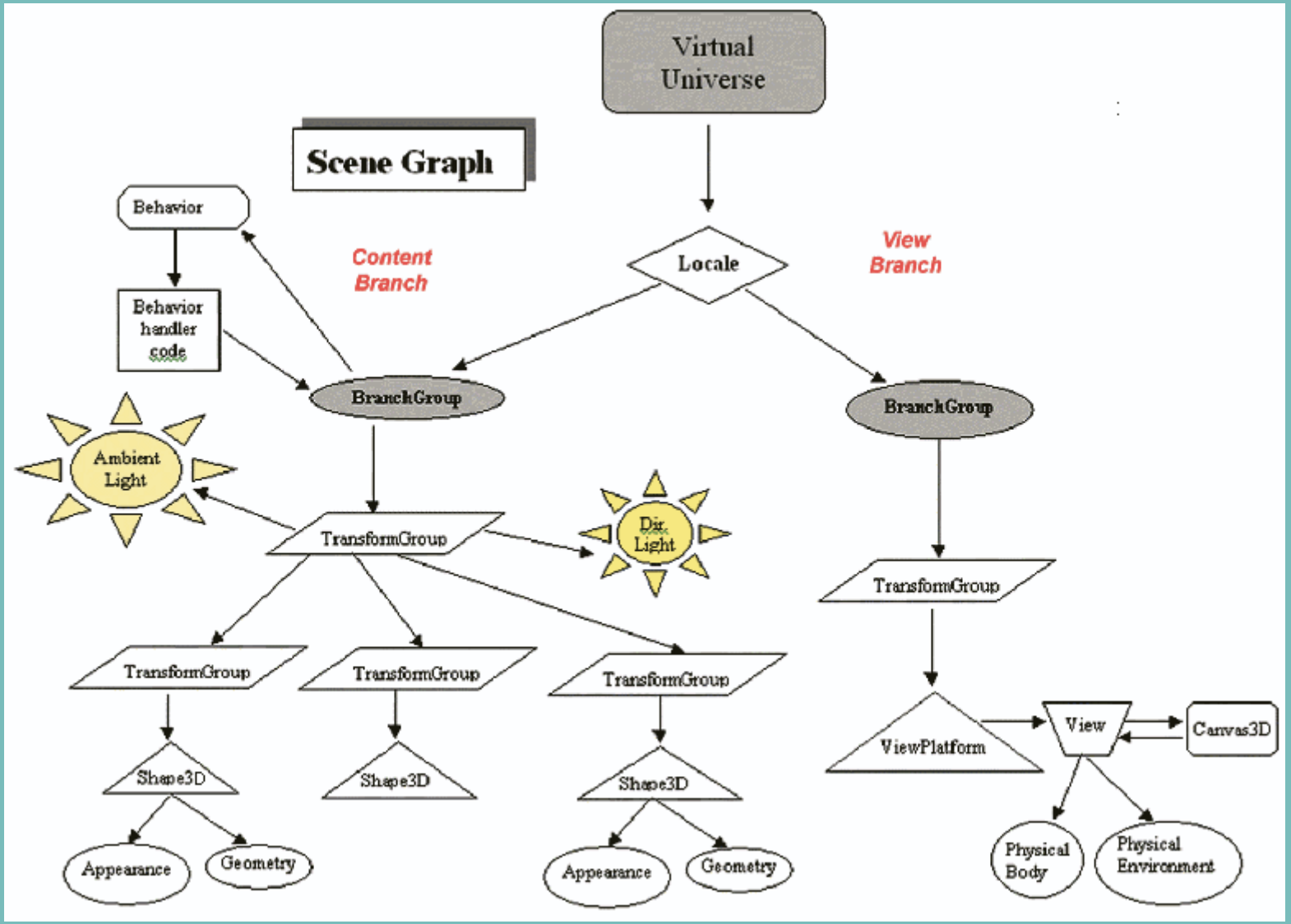


Java 3D:

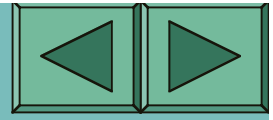
- Creato dalla Sun Microsystem
- API ufficiale di Java per il 3D; è una gerarchia di classi per lo sviluppo di software che mostri scene tridimensionali.
- Necessita del pacchetto JDK (Java2)

- Delega il rendering finale a OpenGL
- Composizione della scena: "Scene Graph".
  Struttura ad albero, dove appendere tutti gli oggetti della scena 3D.

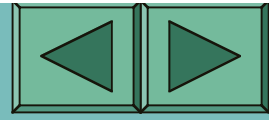
http://java.sun.com/products/java-media/3D/


Virtual Universe
Scene Graph
Behavior
Behavior handler code
Content Branch
Locale
View Branch
BranchGroup
BranchGroup
Ambient Light
TransformGroup
Dir. Light
TransformGroup
TransformGroup
TransformGroup
TransformGroup
Shape3D
Shape3D
Shape3D
Appearance
Geometry
Appearance
Geometry
ViewPlatform
View
Canvas3D
Physical Body
Physical Environment

Componenti principali:



- VirtualUniverse: radice dell'albero (Punto di riferimento)
- Locale: stabilisce la posizione all'interno dell'universo
- BranchGroup: puntatori all'inizio dei due rami;
- "Content Branch": contiene gli oggetti della scena;
- "View Branch": riguarda la "vista" della scena;
- ViewPlatform: indica dove la scena deve essere vista;
- View: indica come la scena deve essere vista;
- Canvas3D: definisce la finestra nativa per OpenGL;
- TransformGroup: gruppo di trasformazione. Ad esso vengono applicati oggetti di tipo "Transform3D".



Oggetti Geometrici:

Shape3D: dispositivo per inserire oggetti geometrici sulla scena.
Ad esso fanno riferimento oggetti di tipo “Geometry” ed “Appearance”.

Definizione della Geometria: “Geometry”

Metodi di utilità: setCoordinate(), setColor(), ecc….

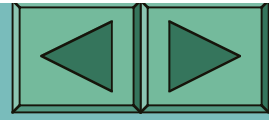
Passaggio di parametri:

- By-COPYNG: Duplicando le informazioni della geometria.
- BY-REFERENCE: facendo riferimento ai vettori base.

Primitive Class:

-Box -Sphere
-Cylinder -Cone

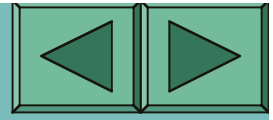
Classe utility: ColorCube



Make Scene ("view branch")

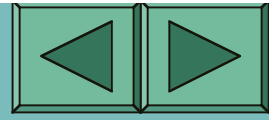
1. Create VirtualUniverse object and Locale object
2. Create ViewPlatform object and View object
   - Create Canvas3D.
   - Create PhysicalBody, Physical Environment
   - Attach above object
3. Create BranchGroup, TransformGroup for ViewPlatform and attach BranchGroup to Locale

- Use SimpleUniverse() function!!

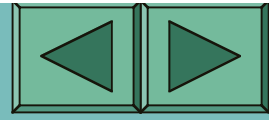


Make Scene ("content branch")

4. Construct Shape3D and TransformGroup
5. Create BranchGroup and attach above TransformGroups to this BranchGroup
6. Attach BranchGroup to Locale



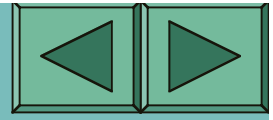
```
import java.applet.Applet;
import java.awt.BorderLayout;
import java.awt.Frame;
import java.awt.event.*;
import com.sun.j3d.utils.applet.MainFrame;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.ColorCube;
import javax.media.j3d.*;
import javax.vecmath.*;
public class HelloJava3Da extends Applet {
   public HelloJava3Da() {
      setLayout(new BorderLayout());
      Canvas3D canvas3D = new Canvas3D(null);
      add("Center", canvas3D);
     SimpleUniverse simpleU = new  SimpleUniverse(canvas3D);
     simpleU.getViewingPlatform().setNominalViewingTransform();
      BranchGroup scene = createSceneGraph();
     simpleU.addBranchGraph(scene);
   }
  public BranchGroup createSceneGraph() {
      BranchGroup objRoot = new BranchGroup();
      objRoot.addChild(new ColorCube(0.4));
      return objRoot;
   }
   public static void main(String[] args) {
      Frame frame = new MainFrame(new HelloJava3Da(), 256, 256);
   }
}
```



Transform

- Transform3D: transform object represented internally as a 4x4 double-precision floating point matrix

- Transform3D object:
  - rotate
  - translate
  - scale

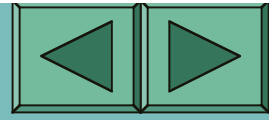


Example:

```
public BranchGroup createSceneGraph() {
        // Create the root of the branch graph
        BranchGroup objRoot = new BranchGroup();
        // rotate object has composited transformation matrix
        Transform3D rotate = new Transform3D();
        Transform3D tempRotate = new Transform3D();
        rotate.rotX(Math.PI/4.0d);
        tempRotate.rotY(Math.PI/5.0d);
        rotate.mul(tempRotate);
       TransformGroup objRotate = new  TansformGroup(rotate);
        objRoot.addChild(objRotate);
        objRotate.addChild(new ColorCube(0.4));

         objRoot.compile();
    return objRoot;
     }
```

Packages (API Java3D):



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

Tutorial Java 3D:

http://java.sun.com/products/java-media/3D/collateral/

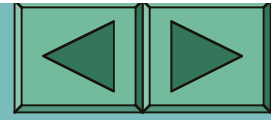
Documentazione API 3D:

http://C:\jdk1.3.1_01\html\index.html

Demo 3D:

http://C:\jdk1.3.1_01\demo\java3d\index.html

Lights



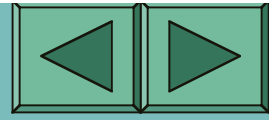
Oggetti per aumentare la qualità ed il realismo delle scene.

Tipi:

- **AmbientLight**: una luce diffusa che illumina uniformemente tutti gli oggetti;
- **DirectionalLight**: raggi paralleli che puntano in una direzione;
- **PointLight**: i raggi sono emessi da un punto e si irradiano in tutte le direzioni;
- **SpotLight**: i raggi sono emessi da un punto e si irradiano dentro un cono;

Bounds (BoundingBox): Regione in cui l'azione ha effetto.

Behavior:



I behavior sono delle classi di Java3D per gestire sofisticati processi in una scena 3D. Esistono dei metodi, catalogati e pronti ad entrare in azione non appena si verificano determinati eventi, in gergo:

WakeUp Condition = condizione di rivesglio.

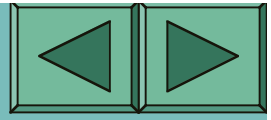
Bounds: `Ambiente (regione) di azione`

Metodi della classe astratta Behavior:

-initialize(): viene attivato all'inizio del processo;

-processStimulus(): viene chiamato ogni volta che si verifica un evento specificato da WakeUpOn(). Terminato il lavoro assegnatogli, reimposta le nuove condizioni di risveglio.

MouseBehavior class:



-MouseRotate: rotate visual object in place

-MouseTranslate: translate the visual object in the image plate

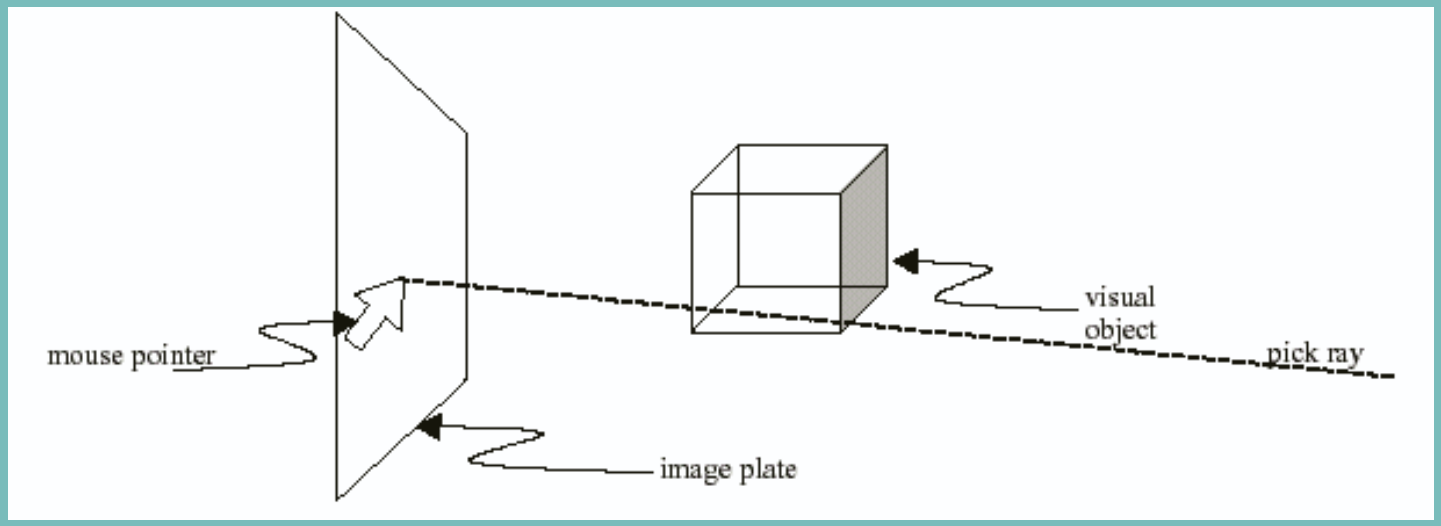
-MouseZoom: translate the visual object in a plane orthogonal to the image plate.

Example: (MouseBehaviorApp.java)

```
.....
BranchGroup objRoot = new BranchGroup();
TransformGroup objTransform = new TransformGroup();
objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
objTransform.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
objRoot.addChild(objTransform);
objTransform.addChild(new ColorCube(0.4));

MouseRotate myMouseRotate = new MouseRotate();
myMouseRotate.setTransformGroup(objTransform);
myMouseRotate.setSchedulingBounds(new BoundingSphere());
objRoot.addChild(myMouseRotate);
.....
```

Picking



PickMouseBehavior Class:

- PickRotateBehavior
- PickTranslateBehavior
- PickZoomBehavior

Example: - MousePickApp.java
- jdk1.3.1_01\demo\java3d\PickTest\IntersectTest.java

Libreria Grafica **OpenGL**

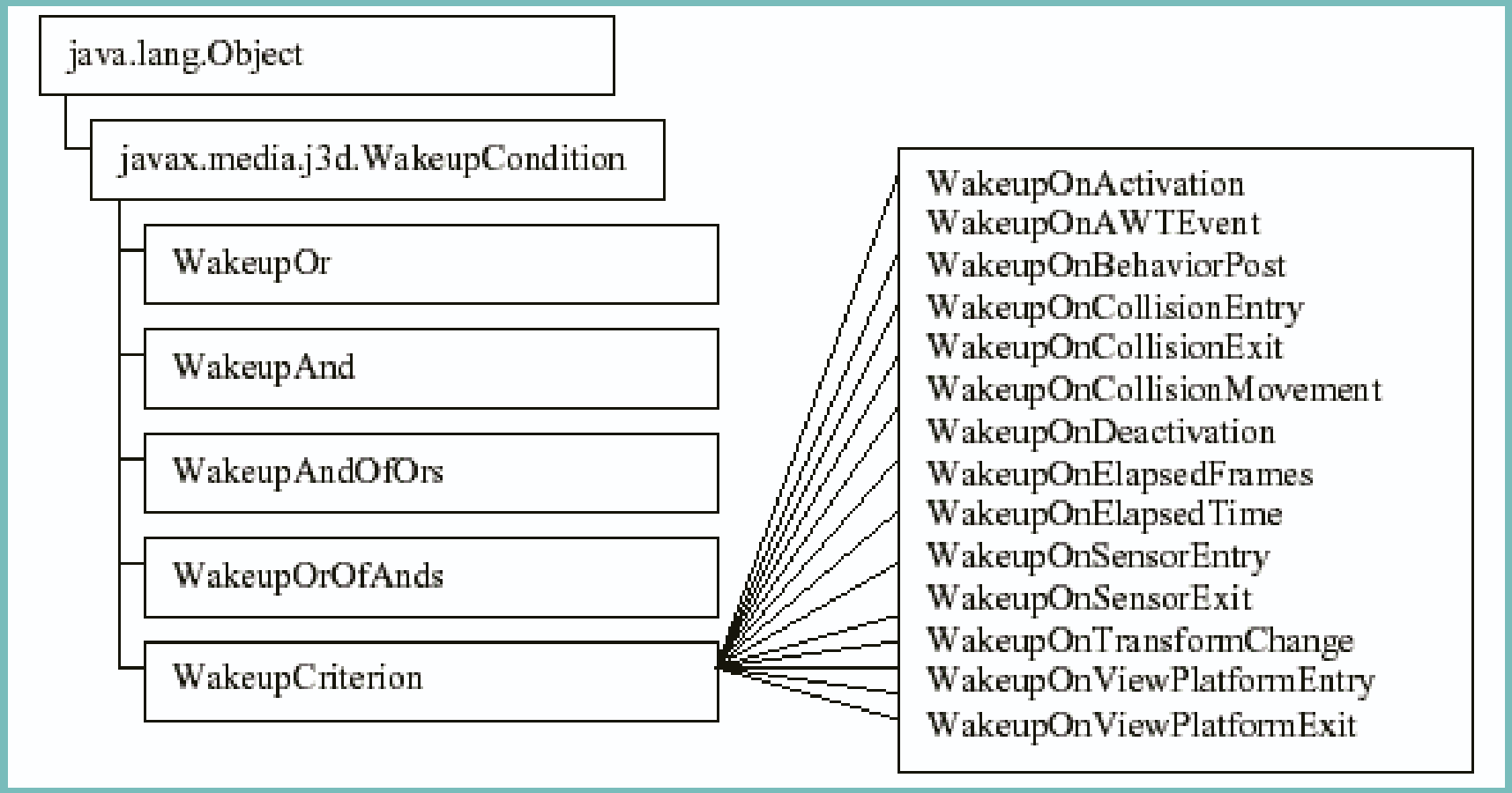
Librerie Grafiche: interfacce software per l'hardware grafico di output. "OpenGL": sono un centinaio di funzioni che consentono la specifica di oggetti ed operazioni, per la creazione di immagini grafiche.

Schema esemplificativo

Primitive Geometriche:

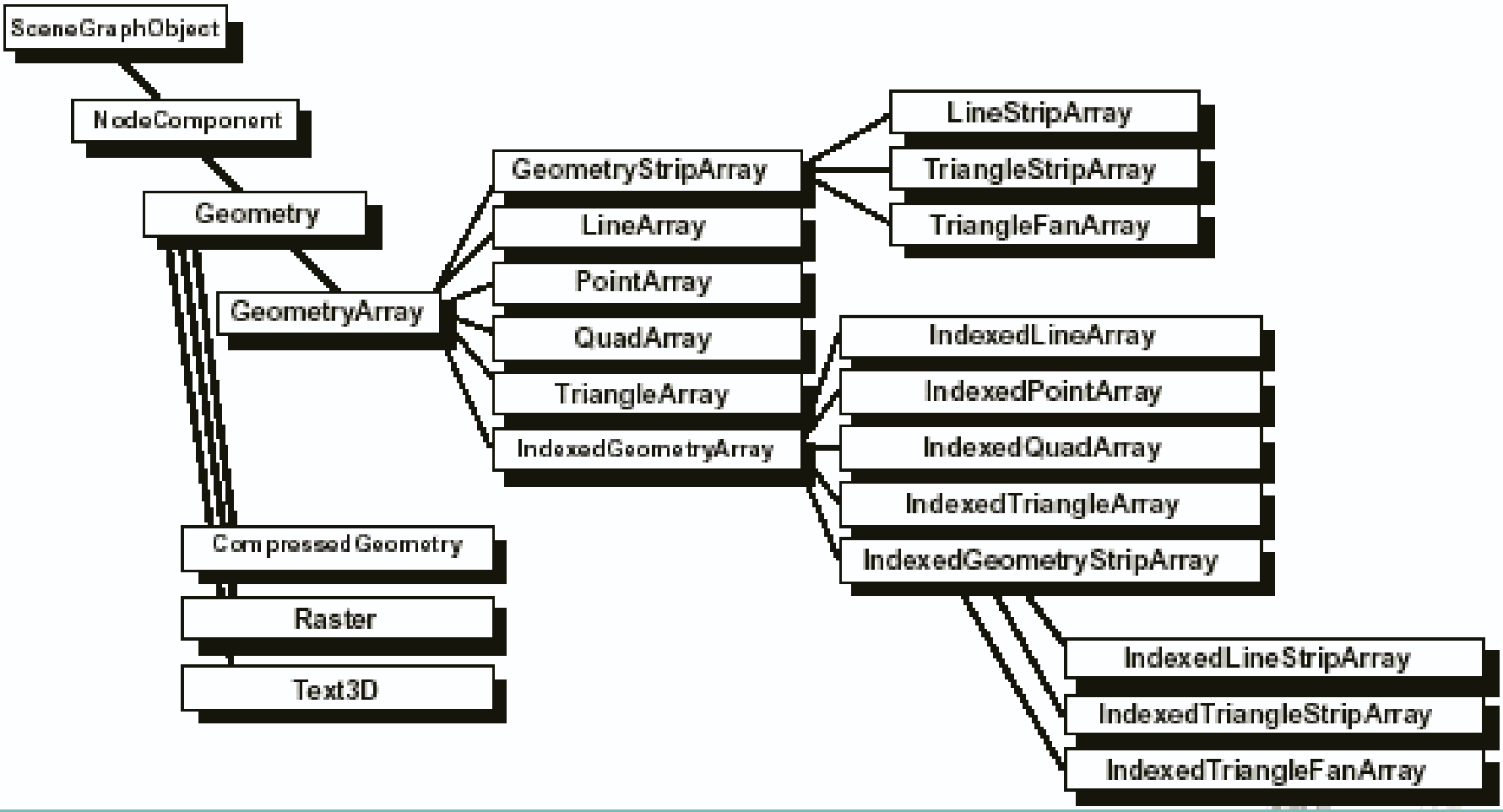
Proprietà : TIPO e VERTICI; i vertici costituiscono l'informazione geometrica, mentre il tipo indica come questi vertici devono essere connessi per formare la primitiva.

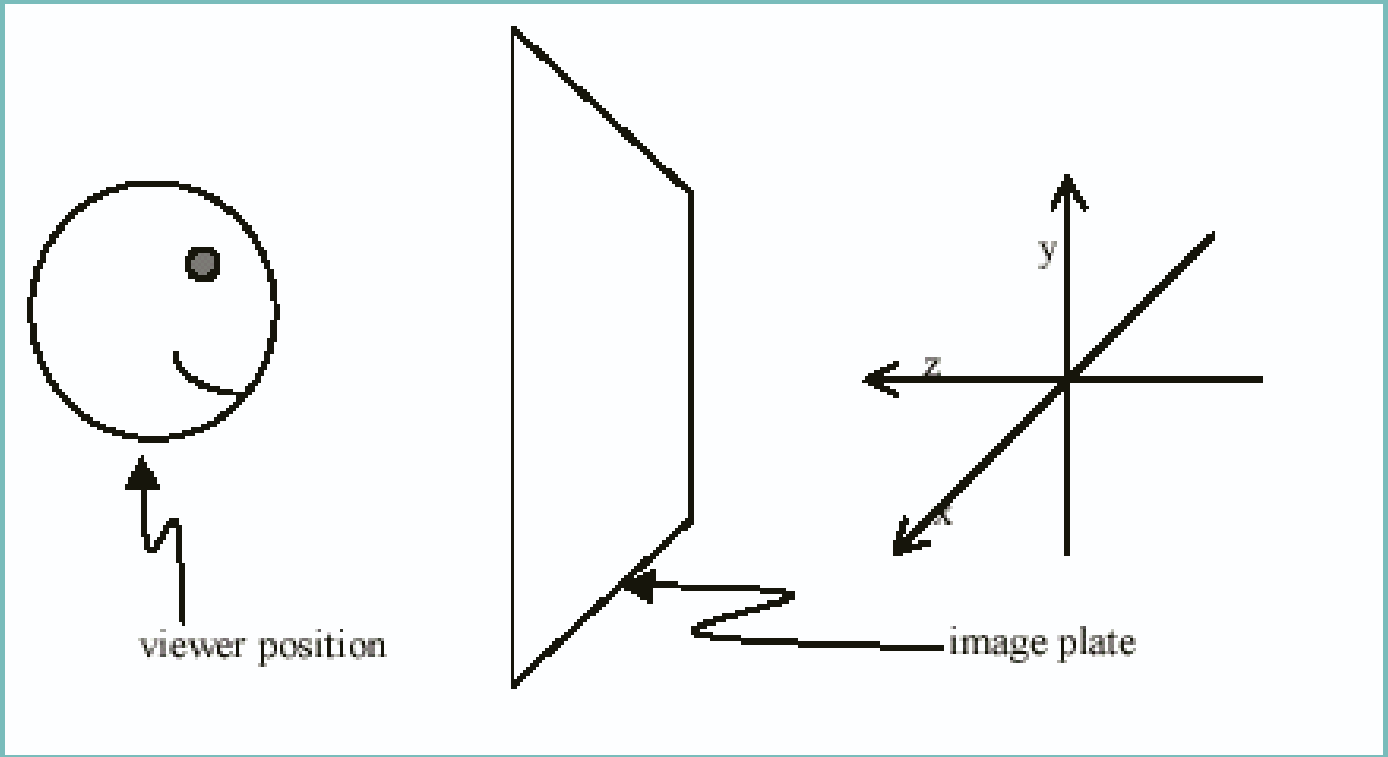
Condizioni di Risveglio

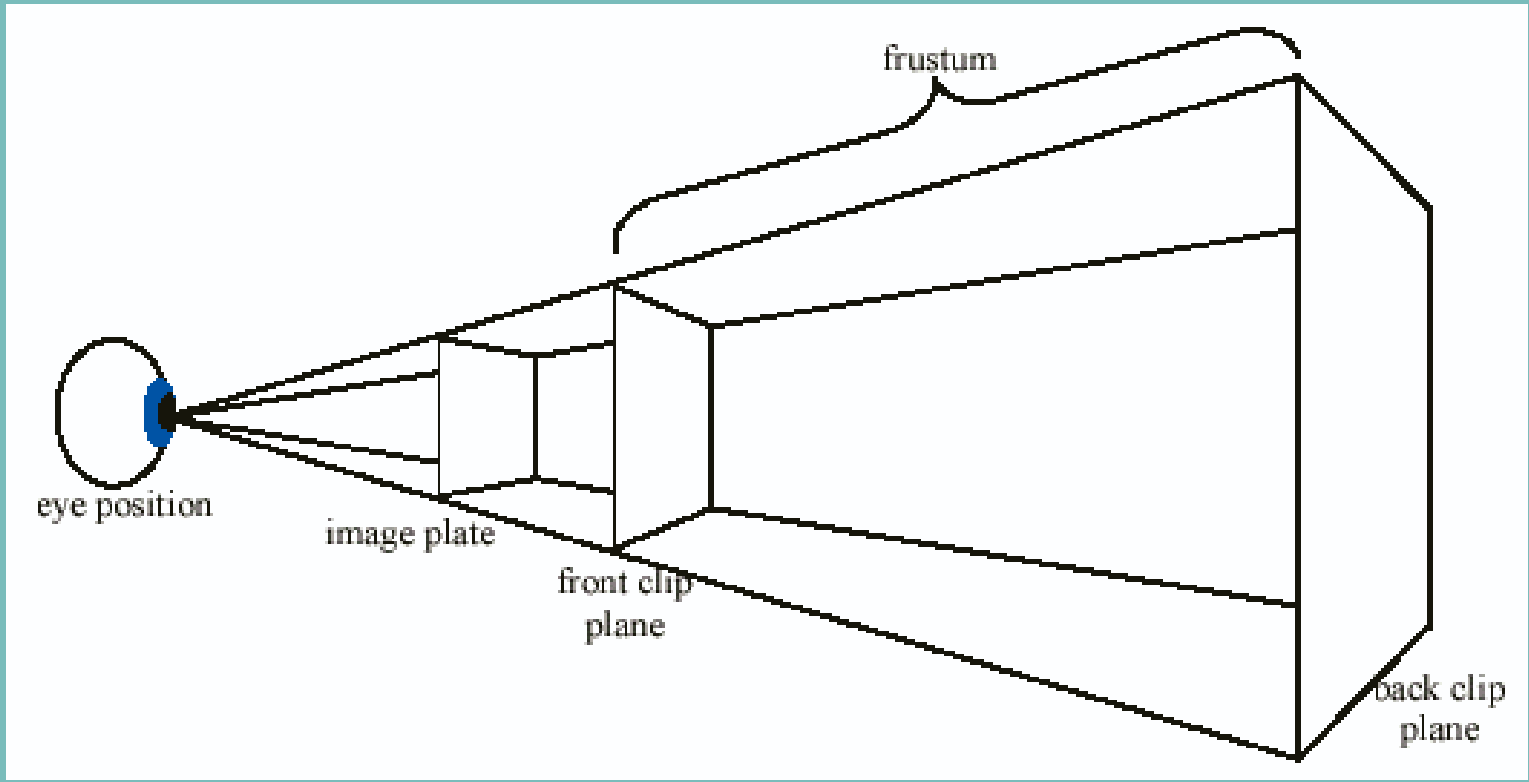


Struttura software:

Geometry Object Hierarchy




y
z
x
viewer position
image plate


frustum
eye position
image plate
front clip
plane
back clip
plane