

Milaneze Francesco – www.redbaron85.com
Java 3D – Guida di base

Java 3D

Guida di base

Francesco “RedBaron85” Milanese
www.redbaron85.com – 2010

Java 3D è un ricco set di API di programmazione di livello medio-alto che consente di creare facilmente applicazioni con grafica tridimensionale interattiva.

Questa guida parte dalle basi teoriche dello Scene Graph per poi trattare, mediante esempi pratici, la creazione, il posizionamento e l'animazione di oggetti nell'universo virtuale di Java 3D, oltre alla gestione della loro resa visiva, l'interazione mediante mouse e tastiera, l'animazione ed altro ancora.

Francesco Milanese ha tenuto un ciclo di seminari sull'argomento presso l'Università degli Studi di Catania durante l'A.A. 2008-09.

E' un Blender Foundation Certified Trainer ed ha realizzato e gestisce il sito www.redbaron85.com, dove pubblica periodicamente tutorial e guide riguardanti la Computer Grafica 3D.

SOMMARIO

Java 3D	12
LA COMPUTER GRAFICA 3D	12
LE API DI PROGRAMMAZIONE	13
JAVA 3D	14
DI COSA ABBIAMO BISOGNO ?	14
STRUTTURA DELLA GUIDA	15
UN SEMPLICE ESEMPIO 2D “FINTO 3D”: WITU.....	17
Lo Scene Graph. Il Simple Universe	19
STRUTTURE ED ELEMENTI DELLO SCENE GRAPH	20
ELEMENTI DELLO SCENE GRAPH – IL VIRTUAL UNIVERSE	20
ELEMENTI DELLO SCENE GRAPH – IL LOCALE	21
ELEMENTI DELLO SCENE GRAPH – LA SUPERCLASSE NODE	21
ELEMENTI DELLO SCENE GRAPH – I GRUPPI	22
ELEMENTI DELLO SCENE GRAPH – LA SUPERCLASSE LEAF	22
ELEMENTI DELLO SCENE GRAPH – LA SUPERCLASSE NODECOMPONENT	22
RELAZIONI PADRE-FIGLIO E RELAZIONI REFERENCE	23
IL SIMPLE UNIVERSE	23
COMPILARE UN BRANCH GRAPH	25
LE CAPABILITY	25
OGGETTI “VIVI”	26
IL SISTEMA DI RIFERIMENTO DI JAVA 3D	26
UN PRIMO ESEMPIO: UN UNIVERSO VUOTO	27
SECONDO ESEMPIO: POPOLARE L’UNIVERSO	28
Il Simple Universe in dettaglio	30
PERSONALIZZARE IL SIMPLE UNIVERSE	30
IL LOCALE	31
ELEMENTI DEL VIEW BRANCH GRAPH. OPERAZIONI.	32
LE NORMALI	34
IL CLIPPING	35
L’ESEMPIO “ESEMPIOSIMPLEUNIVERSE”	36
BOUND E SCOPE: INTRODUZIONE	36
BOUND E BOUNDINGLEAF	37
SCOPE, CON UN PICCOLO INTRUSO	38
IL BACKGROUND	39
ANCORA SU BOUND E BACKGROUND	39
Le classi “matematiche” in Java 3D	42
MATEMATICA 3D: CENNI	42
LE CLASSI MATEMATICHE IN JAVA E IN JAVA 3D	45

Gruppi. Oggetti visuali di base e da file	48
I GRUPPI: BRANCH E TRANSFORM	49
GLI OGGETTI VISUALI	50
GEOMETRIE DI BASE	51
GEOMETRIE DI BASE – COLORCUBE	51
GEOMETRIE DI BASE – BOX	52
GEOMETRIE DI BASE – CONE	52
GEOMETRIE DI BASE – CYLINDER	53
GEOMETRIE DI BASE – SPHERE	53
GEOMETRIE DI BASE – TEXT2D	54
GEOMETRIE DI BASE – TEXT 3D	55
GEOMETRIE DA FILE	57
Geometry e le sue sottoclassi	58
GEOMETRY E LE SUE SOTTOCLASSI: PANORAMICA	59
GEOMETRYARRAY E LE SUE SOTTOCLASSI	59
SOTTOCLASSI DI GEOMETRY ARRAY	60
SOTTOCLASSI DI GEOMETRYARRAY – POINTARRAY	61
SOTTOCLASSI DI GEOMETRYARRAY – LINEARRAY	62
SOTTOCLASSI DI GEOMETRYARRAY – TRIANGLEARRAY	63
SOTTOCLASSI DI GEOMETRYARRAY – QUADARRAY	64
SOTTOCLASSI DI GEOMETRYARRAY – GEOMETRYSTRIPARRAY	64
SOTTOCLASSI DI GEOMETRYSTRIPARRAY – LINESTRIPARRAY	65
SOTTOCLASSI DI GEOMETRYSTRIPARRAY – TRIANGLESTRIPARRAY	66
SOTTOCLASSI DI GEOMETRYSTRIPARRAY – TRIANGLEFANARRAY	67
INDEXEDGEOMETRYARRAY E LE SUE SOTTOCLASSI	68
GEOMETRYINFO	69
GEOMETRY INFO: UN ESEMPIO	71
RASTER	73
Le trasformazioni	74
TRASLAZIONI	75
ROTAZIONI	76
RIDIMENSIONAMENTI (SCALING)	77
IL PIVOTING	77
TRASLAZIONI E PIVOTING	78
ROTAZIONI E PIVOTING	79
SCALING IN CASCATA	81
UN ESEMPIO COMPLETO	82
L'interazione	83
I BEHAVIOR	84
CONDIZIONI DI WAKEUP	86
CREARE DEI SEMPLICI BEHAVIOR: ESEMPI	87
KEYNAVIGATORBEHAVIOR	88
BOUNDINGLEAF E BOUND DI DIMENSIONE INFINITA	90
BEHAVIOR PER IL MOUSE	92
MOUSE BEHAVIOR	93

IL PICKING	93
PICKMOUSEBEHAVIOR	94
MOUSEBEHAVIOR e PICKMOUSEBEHAVIOR	95
PICKINGCALLBACK	95
USERDATA. ESTENDERE LE CLASSI DI JAVA 3D	97
Le luci	99
I COLORI DEGLI OGGETTI IN JAVA 3D	100
LE LUCI NELLO SCENE GRAPH	100
LA CLASSE LIGHT	100
AMBIENT LIGHT	101
DIRECTIONAL LIGHT	102
POINT LIGHT	103
SPOT LIGHT	104
L'ATTENUAZIONE DELLE LUCI	106
BOUNDING LEAF	107
SCOPE	107
LE OMBRE	109
L'aspetto visivo degli oggetti	111
COLORI DELLE GEOMETRIE	112
I MODELLI DI OMBREGGIATURA	113
APPEARANCE E I SUOI ATTRIBUTI	113
ATTRIBUTI DI APPEARANCE – POINT ATTRIBUTES	115
ATTRIBUTI DI APPEARANCE – LINE ATTRIBUTES	115
ATTRIBUTI DI APPEARANCE – POLYGON ATTRIBUTES	116
ATTRIBUTI DI APPEARANCE – COLORING ATTRIBUTES	116
ATTRIBUTI DI APPEARANCE – TRANSPARENCY ATTRIBUTES	117
UTILIZZARE PIU' TECNICHE: UN ESEMPIO	118
ALTERNATE APPEARANCE	119
MATERIAL	121
LUCI, ASPETTO E NORMALI DELLE GEOMETRIE	122
LE TEXTURES E IL TEXTURE MAPPING	123
CARICARE UNA TEXTURE	124
IMPOSTARE UNA TEXTURE IN UN APPEARANCE	124
IMPOSTARE LE COORDINATE DI UNA TEXTURE	125
TEXCOORDGENERATION	125
IMPOSTARE LE COORDINATE DELLE TEXTURES MANUALMENTE	126
ESEMPI CON LE TEXTURES	128
BOUNDARY MODE	128
PIXEL E TEXEL	129
Animazioni	130
ANIMAZIONI DIPENDENTI DAL PUNTO DI VISTA DELL'OSSERVATORE	131
BILLBOARD	132
ORIENTEDSHAPE3D	133
ANIMAZIONI LOD	135
ANIMAZIONI TIME-BASED	137

ALPHA	139
INTERPOLATOR E LE SUE SOTTOCLASSI	143
SOTTOCLASSI DI INTERPOLATOR – COLOR INTERPOLATOR	144
SOTTOCLASSI DI INTERPOLATOR – TRANSPARENCY INTERPOLATOR	145
SOTTOCLASSI DI INTERPOLATOR – SWITCH VALUE INTERPOLATOR	146
SOTTOCLASSI DI INTERPOLATOR – TRANSFORM INTERPOLATOR	147
SISTEMI DI RIFERIMENTO: LOCALE E GLOBALE	148
SOTTOCLASSI DI TRANSFORM INTERPOLATOR – POSITION INTERPOLATOR	149
SOTTOCLASSI DI TRANSFORM INTERPOLATOR – ROTATION INTERPOLATOR .	150
SOTTOCLASSI DI TRANSFORM INTERPOLATOR – SCALE INTERPOLATOR	151
SOTTOCLASSI DI TRANSFORM INTERPOLATOR – PATH INTERPOLATOR	152
AXISANGLE4D E QUATERNIONI	153
SOTTOCLASSI DI PATH INTERPOLATOR – POSITION PATH INTERPOLATOR	155
SOTTOCLASSI DI PATH INTERPOLATOR – ROTATION PATH INTERPOLATOR	156
SOTTOCLASSI DI PATH INTERPOLATOR – ROTPOSPATH INTERPOLATOR	156
SOTTOCLASSI DI PATH INTERPOLATOR – ROTPOSSCALEPATH INTERPOLATOR	157
ANIMAZIONI “A COMANDO”	158
ANIMAZIONI MORPH	160
Extra	163
LE COLLISIONI IN JAVA 3D	164
IL METODO GET TRIGGERING PATH E L’OGGETTO SCENE GRAPH PATH	165
SCRIVERE SULLA CANVAS 3D	167
I THREAD IN JAVA	168
ANIMAZIONI MEDIANTE THREAD: UN SEMPLICE ESEMPIO	175
SET TEXTURE MODE DI TEXTURE ATTRIBUTES	178
Appendice	180
APPLICAZIONE JAVA 3D DI BASE	181
WITU.java	183
WITUapp.java	188
PRIMO ESEMPIO	189
SECONDO ESEMPIO	191
NOMINAL VIEWING PLATFORM	193
ESEMPIO SIMPLE UNIVERSE	195
ESEMPIO BACKGROUND	198
BACKGROUND CREATO CON SFERA ENORME	200
BACKGROUND CLASSICO	202
ESEMPIO COLOR CUBE	205
ESEMPIO BOX	207
ESEMPIO CONE	208
ESEMPIO CYLINDER	209
ESEMPIO SPHERE 1	210
ESEMPIO SPHERE 2	211
ESEMPIO TEXT 2D	212
ESEMPIO TEXT 3D	214
ESEMPIO POINT ARRAY	216

ESEMPIO LINE ARRAY QUATTRO VERTICI	218
ESEMPIO LINE ARRAY TRE VERTICI	220
ESEMPIO SISTEMA DI RIFERIMENTO	222
ESEMPIO TRIANGLE ARRAY	225
ESEMPIO QUAD ARRAY	227
ESEMPIO LINE STRIP ARRAY	230
ESEMPIO TRIANGLE STRIP ARRAY	233
ESEMPIO TRIANGLE FAN ARRAY	236
ESEMPIO INDEXED LINE ARRAY	239
ESEMPIO INDEXED QUAD ARRAY	241
ESEMPIO GEOMETRY INFO	243
ESEMPIO RASTER	246
DOWNTOWN	248
ROTAZIONE PIVOT 1	250
ROTAZIONE PIVOT 2	252
ESEMPIO SCALING IN CASCATA	255
ESEMPIO ROTO TRAS SCALE	257
ESEMPIO BEHAVIOR 1	259
MIO BEHAVIOR 1	261
ESEMPIO BEHAVIOR 2	264
MIO BEHAVIOR 2	266
ESEMPIO BEHAVIOR NAVIGAZIONE	269
ESEMPIO NAV BEHAVIOR 2	271
MIO BEHAVIOR TASTIERA	273
ESEMPIO MOUSE BEHAVIOR	276
ESEMPIO PICKING	278
ESEMPIO PICKING CALLBACK	280
MIA CLASSE DI PICKING CALLBACK	282
ESEMPIO PICKING CALLBACK 2	283
MIA CLASSE DI PICKING CALLBACK 2	285
MIO TRANSFORM GROUP	286
ESEMPIO LUCE AMBIENTALE	287
ESEMPIO LUCE DIREZIONALE	289
ESEMPIO POINT LIGHT	292
ESEMPIO SPOT LIGHT	295
ESEMPIO POINT LIGHT CON SCOPE	298
ESEMPIO COLORI DELLE GEOMETRIE 1	301
ESEMPIO COLORI DELLE GEOMETRIE 2	304
ESEMPIO POINT ATTRIBUTES	306
ESEMPIO LINE ATTRIBUTES	308
ESEMPIO POLYGON ATTRIBUTES	310
ESEMPIO COLORING ATTRIBUTES	312
ESEMPIO TRANSPARENCY ATTRIBUTES	314
ESEMPIO BORDI IN RILIEVO	316
ESEMPIO ALTERNATE APPEARANCE	319
ESEMPIO ALTERNATE APPEARANCE APP 2	322
ESEMPIO MATERIAL	326
ESEMPIO TEXTURE 1	329

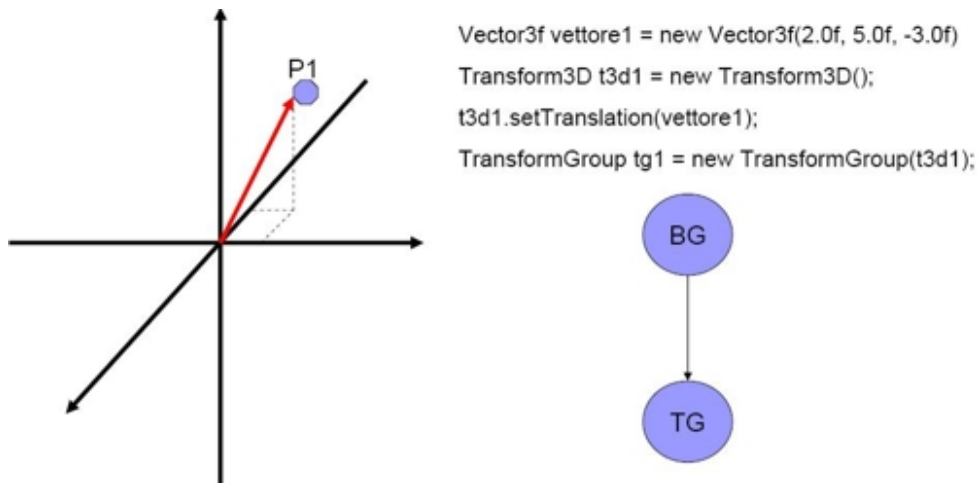
ESEMPIO TEXTURE 2	331
ESEMPIO TEXTURE 3	333
ESEMPIO TEXTURE 4	336
ESEMPIO BOUNDARY MODE	339
ESEMPIO BILLBOARD	342
ESEMPIO BILLBOARD 2	345
ESEMPIO ORIENTED SHAPE 3D	348
ESEMPIO LOD	351
ESEMPIO COLOR INTERPOLATOR	353
ESEMPIO TRANSPARENCY INTERPOLATOR	356
ESEMPIO SWITCH VALUE INTERPOLATOR	359
ESEMPIO POSITION INTERPOLATOR 1	361
ESEMPIO POSITION INTERPOLATOR 2	363
ESEMPIO POSITION INTERPOLATOR 3	366
ESEMPIO ROTATION INTERPOLATOR	369
ESEMPIO SCALE INTERPOLATOR	371
ESEMPIO QUATERNIONI	373
ESEMPIO POSITION PATH INTERPOLATOR	376
ESEMPIO ROTPOSPATHINTERPOLATOR	379
ESEMPIO ROTPOSSCALEPATHINTERPOLATOR	382
ESEMPIO ICBM --- MAIN	385
ESEMPIO ICBM --- GUI	390
ESEMPIO MORPH BEHAVIOR	397
BEHAVIOR COLLISIONE	400
SIMULATORE --- MAIN	401
SIMULATORE --- THREAD CONTROLLO 1	402
SIMULATORE --- THREAD FRAME 1	403
SIMULATORE --- FRAME SIMULATORE 1	404
SIMULATORE --- MIO BEHAVIOR TASTIERA	406
SIMULATORE --- SCENA 3D	410
PIANETA TERRA	412

TRASLAZIONI E PIVOTING

La **traslazione** è il caso più semplice: se aggiungiamo un **TransformGroup** (che chiameremo ad esempio tg1), con una sua **trasformazione di traslazione** (che chiameremo ad esempio t3d1), in cima al **Content Branch Graph**, la **traslazione** verrà effettuata *relativamente all'origine del sistema* (il **pivot**, in questo caso, è l'origine).

Il **vettore di traslazione** di t3d1 parte quindi dall'origine e **identifica un nuovo punto di arrivo nello spazio**. Chiamiamo tale punto P1.

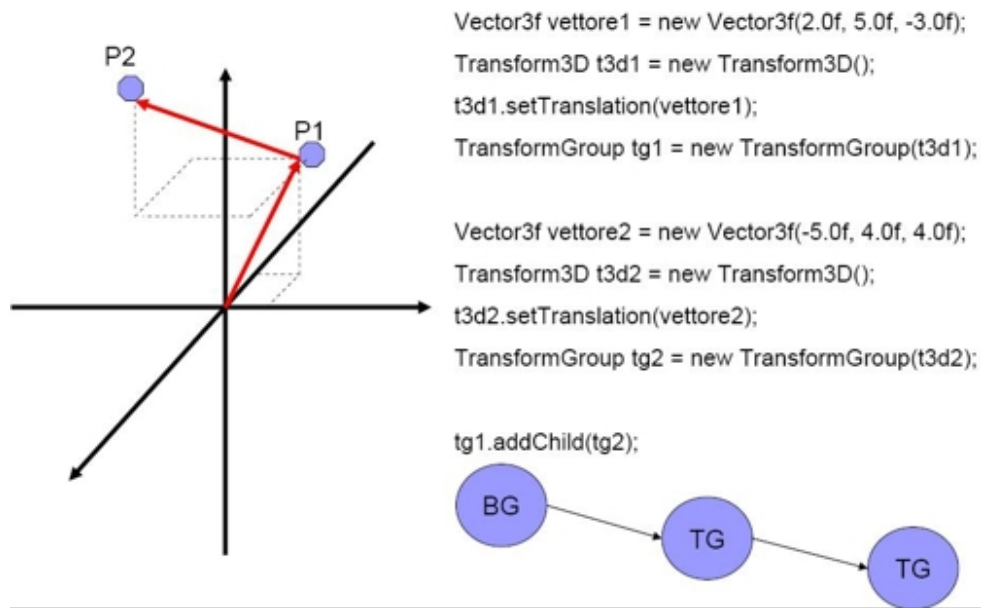
Un nuovo **oggetto visuale** collegato, come **figlio**, a tg1, verrà quindi **centrato** in P1.



Se a tg1 colleghiamo ora, come **figlio**, un nuovo **TransformGroup** (tg2), con una sua **trasformazione di traslazione** (t3d2), il **vettore** di t3d2 partirà a P1.

Le **coordinate di arrivo** di t3d2, *in riferimento all'origine dell'universo virtuale di Java3D*, vengono calcolate con la **somma vettoriale dei vettori** di t3d1 e t3d2 ed identificano un **secondo punto P2** (coordinate *globali* di P2: t3d1 + t3d2; *coordinate locali*, cioè *in riferimento al nodo padre* di P2: t3d2).

Un **oggetto visuale** collegato, come **figlio**, a tg2, verrà **centrato** in P2.



ROTAZIONI E PIVOTING

Finora abbiamo visto la **rotazione** di oggetti che si trovavano nell'origine; in quel caso, il **pivot della trasformazione** coincideva con il **centro dell'oggetto**.

Quando però un oggetto viene **traslato**, bisogna fare attenzione a dove **posizionare la trasformazione di rotazione**, al fine di evitare effetti indesiderati.

Sostanzialmente, quando bisogna **ruotare** un oggetto bisogna porsi la seguente domanda: **'rispetto a cosa lo stiamo ruotando'** ?

Ebbene, se dobbiamo **ruotare** un oggetto **intorno al suo centro**, possiamo applicare la **trasformazione** di rotazione **direttamente** al TG che controlla la posizione dell'oggetto, cioè suo **padre**.

Nell'**esempio 'RotazionePivot1'** è possibile vedere un cubo spostato in avanti rispetto al punto di vista dell'osservatore di 5 unità e ruotato, intorno all'asse Y , di $\pi/4$ radianti; per far questo, si è reso necessario creare due TG: il primo per traslare il cubo, il secondo per **ruotarlo intorno al suo centro**.

Per ruotare l'oggetto intorno ad un altro punto, **un punto esterno**, cosa bisogna fare ?

Soffermiamoci su come vengono colorate le due sfere.

La sfera nell'origine ha colore **diffuso** VERDE (0, 1, 0), la luce ha colore CIANO (0, 1, 1), dunque il **colore visibile** della sfera sarà VERDE, eccezion fatta per la regione del **riflesso speculare**: il colore speculare, che di default è bianco, verrà reso come CIANO, cioè il colore della fonte luminosa, che ha canale R nullo.

La seconda sfera ha colore **diffuso** ROSSO (1, 0, 0), dunque non subirà l'influenza della luce sul colore diffuso e verrà resa nera, eccezion fatta per la regione del riflesso speculare, che verrà resa color CIANO, per i motivi esposti precedentemente.

	Sfera 1	Sfera 2		Sfera 1	Sfera 2
Colore diffuso proprio	0.0--1.0--0.0	1.0--0.0--0.0	Colore speculare proprio	1.0--1.0--1.0	1.0--1.0--1.0
Colore luce	0.0--1.0--1.0	0.0--1.0--1.0	Colore luce	0.0--1.0--1.0	0.0--1.0--1.0
Colore diffuso finale	0.0--1.0--0.0	0.0--0.0--0.0	Colore speculare finale	0.0--1.0--1.0	0.0--1.0--1.0

POINT LIGHT

Il funzionamento di una **PointLight** è in tutto e per tutto simile a quello di una **lampadina a bulbo**: una sorgente di luce **omnidirezionale**, la cui **intensità luminosa** decresce (se lo si desidera, in **Java3D**) all'aumentare della **distanza**.

Costruttori:

- `PointLight()`;
- `PointLight(Color3f color, Point3f position, Point3f attenuation)`;
- `PointLight(boolean lightOn, Color3f color, Point3f position, Point3f attenuation)`.

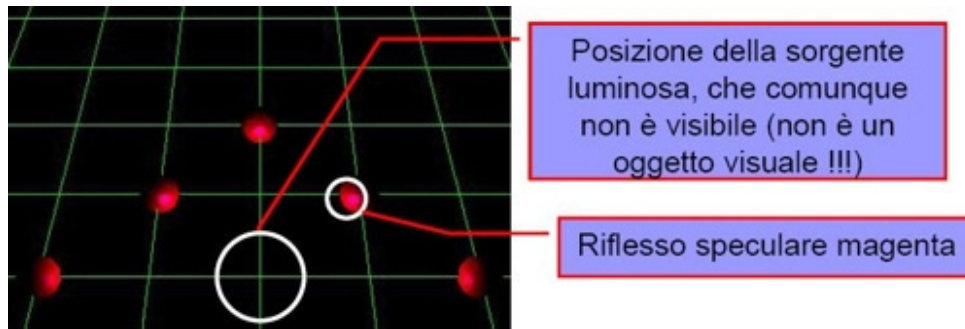
L'**attenuazione** è calcolata con un'equazione i cui **parametri** possono essere modificati; a tal proposito, **PointLight** mette a disposizione due **metodi**:

- `setAttenuation(float constant, float linear, float quadratic)`;
- `setAttenuation(Point3f attenuation)`.

Una **PointLight** influisce sui **colori diffuso e speculare degli oggetti** che si trovano nel suo **bound d'azione**.

Nel file di esempio *PointLight* abbiamo una **sorgente luminosa PointLight** di colore MAGENTA posta nell'**origine**.

Le 5 **sfere** presenti hanno colore diffuso ROSSO e colore speculare BIANCO (default), quindi con la luce MAGENTA il colore diffuso finale sarà il ROSSO e quello speculare il MAGENTA.



SPOT LIGHT

La **classe SpotLight** è figlia di **PointLight**: in effetti, è una **fonte di luce** che agisce in una determinata **direzione** e all'interno di un **cono d'azione**.

Costruttori:

- `SpotLight()` ;
- `SpotLight(Color3f color, Point3f position, Point3f attenuation, Vector3f direction, float spreadAngle, float concentration)` ;
- `SpotLight(boolean lightOn, Color3f color, Point3f position, Point3f attenuation, Vector3f direction, float spreadAngle, float concentration)` .

La **direzione** è espressa mediante un **vettore**.

Nel determinare l'**intensità luminosa** in un dato punto del **cono d'azione** non entra in gioco solo la **distanza** dalla **sorgente**, ma anche l'**angolo** formato dal **vettore direzione** e dal **vettore** passante per la **sorgente** e il punto interno al cono preso in esame.

Il valore di **concentration** varia da 0.0 (distribuzione uniforme) a 128.0 (luce concentrata).

AXISANGLE4D

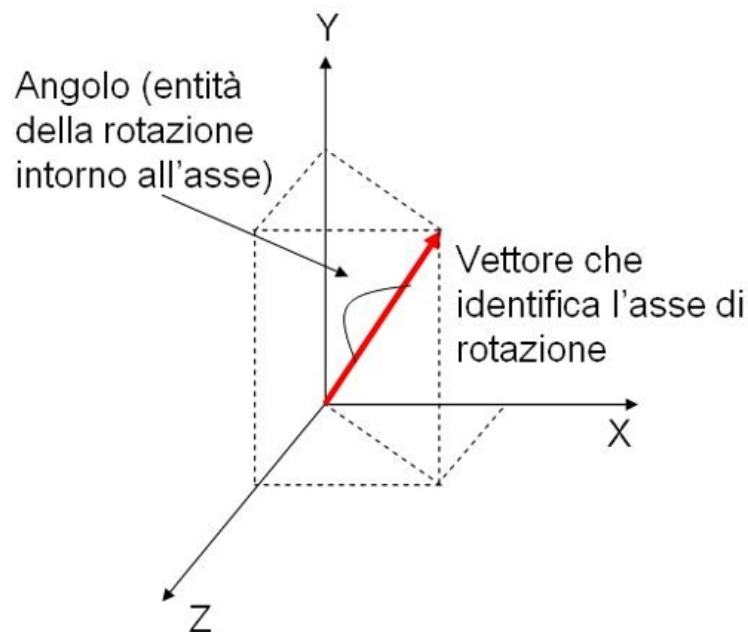
Un **AxisAngle4d** è un oggetto **AxisAngle**, utilizzato per descrivere le **rotazioni** nello **spazio 3D** con 4 valori double: 3 componenti di un **vettore** e un **angolo**.

In generale, con le tre componenti double identifichiamo un **vettore** che, da un punto di partenza, porta ad una destinazione, mediante i valori x , y e z trattati singolarmente (cioè secondo le regole della somma vettoriale).

Il **vettore** però, in questo caso, non verrà utilizzato per identificare un punto, ma un **asse di rotazione**.

L'asse di rotazione viene **calcolato sulla base delle coordinate locali dell'oggetto**.

Il **valore dell'angolo** rappresenta l'entità della **rotazione** intorno a tale **asse**.



Costruttori:

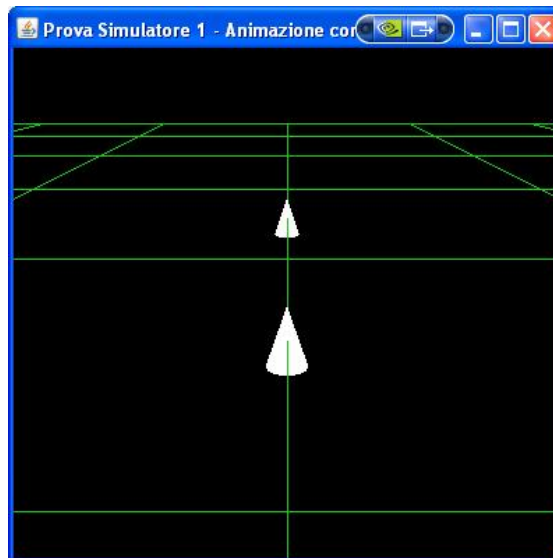
- `AxisAngle4d;`
- `AxisAngle4d(Vector3d axis, double angle);`
- `AxisAngle4d(double[] a);`
- `AxisAngle4d(double x, double y, double z, double angle);`
- `AxisAngle4d(AxisAngle4d a1);`
- `AxisAngle4d(AxisAngle4d a1);`

Tasto	Effetto
W	Avanza di 10 unità
A	Trasla a sinistra di 10 unità
S	Trasla indietro di 10 unità
D	Trasla a destra di 10 unità
Freccia ←	Ruota verso sinistra di 10 gradi
Freccia →	Ruota verso destra di 10 gradi
Freccia ↑	Ruota verso l'alto di 10 gradi
Freccia ↓	Ruota verso il basso di 10 gradi
+	Aumenta la potenza del reattore di 10 unità
-	Diminuisce la potenza del reattore di 10 unità

E' quindi possibile **modificare la posizione e l'orientamento** dell'utente/osservatore mediante i tasti **freccia** e **WASD**.

A questo movimento, che dipende dall'utente, va sommata la **traslazione in avanti** effettuata autonomamente dal programma se il valore di **'potenza reattore'** è maggiore di 0 (valore che può essere incrementato o decrementato, in una scala che va da 0 a 110, mediante i tasti + e -; il valore **massimo** è stato impostato a 110 per motivi 'storici', in quanto questa era la velocità raggiungibile con i **postbruciatori** accesi nella maggior parte dei **simulatori di volo**).

Provate, ad esempio, a **ruotare l'osservatore** verso il basso e a destra e a premere il tasto +.



Il flusso di esecuzione del programma è un po' particolare ma serve ad illustrare come **combinare Behavior** per **l'interazione** da tastiera (con animazioni proprie) e **thread concorrenti** (con un **thread** che può modificare alcune **proprietà dell'universo virtuale**).

SET TEXTURE MODE DI TEXTURE ATTRIBUTES

Tra gli **attributi** dell'**aspetto visivo** degli oggetti vi è *TextureAttributes*, che si occupa di definire in dettaglio alcuni **parametri** del processo di **texture mapping**.

Tali **parametri** sono:

- **Texture Mode** , che esamineremo successivamente;
- **Combine Mode** , che si occupa di definire l'operazione 'combine' quando il parametro precedente, *TextureMode*, è impostato, appunto, a 'COMBINE';
- **CombineColorSource, CombineColorFunction e CombineScaleFactor**, che si occupano di parametri supplementari legati sempre all'operazione 'combine';
- **Transform** , che specifica la trasformazione utilizzata per trasformare (traslare, ruotare, ridimensionare) le coordinate delle texture prima di applicare la texture ad un oggetto;
- **Blend Color** , che specifica un colore uniforme;
- **Perspective Correction** , che permette di scegliere il modello di correzione prospettica utilizzato;
- **Texture Color Table** , che definisce una look up table da impostare prima di applicare il texture mode.

In questa sezione prenderemo in esame solo il primo parametro, **TextureMode**.

TextureMode si occupa di definire **come verranno combinati i colori dell'oggetto e quelli della texture** da applicare all'oggetto.

Le **opzioni** possibili sono:

- **MODULATE** : combina il colore dell'oggetto con quello della texture;
- **DECAL** : la texture viene applicata sul colore dell'oggetto 'come un'etichetta adesiva' (decal); interessante per via della gestione implicita delle trasparenze eventualmente presenti nella texture;
- **BLEND** : combina il colore 'blend color' della texture con quello dell'oggetto;
- **REPLACE** : il colore dell'oggetto viene sostituito da quello della texture;

```
    PolygonAttributes polyAttrib = new PolygonAttributes();  
    polyAttrib.setCullFace(PolygonAttributes.CULL_NONE);  
    polyAttrib.setBackFaceNormalFlip(true);  
    textAppear.setPolygonAttributes(polyAttrib);  
    */  
    tgRotazione.addChild(testo2D);  
    objRoot.addChild(tgRotazione);  
    return objRoot;  
    }  
}
```

ESEMPIO TEXT 3D

// Milanese Francesco --- Java 3D - Guida di base --- www.redbaron85.com

/*

Questo esempio mostra come inserire nel Content Branch Graph un oggetto "nativo" di Java3D, un Testo 3D, ruotato ma non spostato, che é una Shape3D a tutti gli effetti dove bisogna specificare Geometry e Appearance (e l'esempio mostra come fare).

*/

```
import java.awt.*;
```

```
import javax.swing.*;
```

```
import javax.vecmath.*; // Stavolta questo ci serve per Point3f, all'interno di Text3D
```

```
import javax.media.j3d.*;
```

```
import com.sun.j3d.utils.universe.*;
```

```
import com.sun.j3d.utils.geometry.*;
```

```
public class EsempioText3D extends JFrame
```

```
{
```

```
    public static void main(String [] args)
```

```
    {
```

```
        EsempioText3D et3d = new EsempioText3D();
```

```
        et3d.setVisible(true);
```

```
    }
```

```
    public EsempioText3D()
```

```
    {
```

```
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        this.setSize(300, 300);
```

```
        this.setLocation(100, 100);
```

```
        this.setTitle("Java 3D, esempio Text3D. Milanese Francesco, www.redbaron85.com");
```

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

```
        Canvas3D c3D = new Canvas3D(config);
```

```
        SimpleUniverse sU = new SimpleUniverse(c3D);
```

```
        sU.getViewerPlatform().setNominalViewingTransform();
```

```
        BranchGroup scena = creaLoSceneGraph();
```

```
        scena.compile();
```

```
        sU.addBranchGraph(scena);
```

```
        this.getContentPane().add(c3D, BorderLayout.CENTER);
```

```
    }
```

```
    public BranchGroup creaLoSceneGraph()
```

```
    {
```

```
        BranchGroup objRoot = new BranchGroup();
```

```
        Transform3D t3D = new Transform3D();
```

```
        t3D.setTranslation(new Vector3f(0.0f, 0.0f, -3.0f));
```

```
        TransformGroup tgTraslazione = new TransformGroup(t3D);
```

```
        Transform3D rotazione1 = new Transform3D();
```

```
        rotazione1.rotY(Math.PI/4.0);
```

```
        Transform3D rotazione2 = new Transform3D();
```

```
        rotazione2.rotX(-Math.PI/11.0);
```

```
        rotazione2.mul(rotazione1);
```

```
        TransformGroup tgRotazione = new TransformGroup(rotazione2);
```

```
        // INIZIA LA FASE DI CREAZIONE DEL TESTO 3D: PRIMA SI CREA UN APPEARANCE,
```

```
        // POI LA GEOMETRIA, INFINE UN OGGETTO SHAPE3D CON SIFFATTE APPEARANCE E
```

```
GEOMETRY:
```

```

    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.setTranslation(new Vector3f(0.0f, 0.0f, -1.0f));
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    else if(codiceEvento == KeyEvent.VK_D) // Tasto D: spostamento laterale
verso destra
    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.setTranslation(new Vector3f(1.0f, 0.0f, 0.0f));
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    else if(codiceEvento == KeyEvent.VK_UP) // Freccia in alto: ROTAZIONE
verso l'alto (rotazione intorno all'asse X)
    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.rotX(Math.PI/18);
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    else if(codiceEvento == KeyEvent.VK_DOWN) // Freccia in alto:
ROTAZIONE verso il basso (rotazione intorno all'asse X)
    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.rotX(-Math.PI/18);
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    else if(codiceEvento == KeyEvent.VK_LEFT) // Freccia in alto: ROTAZIONE
verso sinistra (rotazione intorno all'asse Z)
    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.rotY(Math.PI/18);
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    else if(codiceEvento == KeyEvent.VK_RIGHT) // Freccia in alto:
ROTAZIONE verso destra (rotazione intorno all'asse Z)
    {
        TGtarget.getTransform(t3dAux);
        t3dAux2.rotY(-Math.PI/18);
        t3dAux.mul(t3dAux2);
        TGtarget.setTransform(t3dAux);
    }
    }
}
// RESETTO il trigger di questo behavior
this.wakeupOn(new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED));
}
}

```

ESEMPIO MOUSE BEHAVIOR

// Milanesi Francesco --- Java 3D - Guida di base --- www.redbaron85.com

/*

Questo esempio mostra come interagire con la scena 3D mediante i tasti del mouse:

- tenendo premuto il tasto sinistro, muovendo il mouse si provoca la rotazione dell'oggetto;
 - tenendo premuto il tasto centrale, muovendo il mouse si provoca la traslazione dell'oggetto lungo l'asse Z;
 - tenendo premuto il tasto destro, muovendo il mouse si provoca la rotazione dell'oggetto sul piano XY.
- Qui si fa uso dei Behavior per il mouse "nativi" di Java3D: MouseTranslate, MouseRotate, MouseZoom.

In questo esempio si fa uso del KeyNavigatorBehavior di Java3D per navigare nello Spazio Virtuale.

*/

```
import java.awt.*;
import javax.swing.*;
import javax.vecmath.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.behaviors.keyboard.*;
import com.sun.j3d.utils.behaviors.mouse.*;

public class EsempioMouseBehavior extends JFrame
{
    public static void main(String[] args)
    {
        EsempioMouseBehavior emb = new EsempioMouseBehavior();
        emb.setVisible(true);
    }

    public EsempioMouseBehavior()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocation(100, 100);
        this.setSize(400, 400);
        this.setTitle("Esempio Mouse Behavior; Milanesi Francesco; www.redbaron85.com");
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        this.getContentPane().add(canvas3D, BorderLayout.CENTER);
        BranchGroup scene = createSceneGraph();
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewer().getView().setBackClipDistance(10000);
        simpleU.getViewer().getView().setFrontClipDistance(0.1);
        TransformGroup vpTG = simpleU.getViewingPlatform().getViewPlatformTransform();
        Transform3D t3d = new Transform3D();
        t3d.setTranslation(new Vector3f(0.0f, 2.0f, 4.0f));
        vpTG.setTransform(t3d);
        KeyNavigatorBehavior keyNavBeh = new KeyNavigatorBehavior(vpTG);
        keyNavBeh.setSchedulingBounds(new BoundingSphere(new Point3d(), 1000.0));
        scene.addChild(keyNavBeh);
        scene.compile();
        simpleU.addBranchGraph(scene);
    }

    private BranchGroup createSceneGraph()

```

```
// ma con trasparenza al 0.5 (su una scala da 0 a 1)
objRoot.addChild(new com.sun.j3d.utils.geometry.Box(0.4f, 0.6f, 0.2f,
creaAppearanceTransparencyAttributes());
// Aggiungo un pavimento di riferimento alla scena
objRoot.addChild(this.creaRiferimento());
// Restituisco il tutto
return objRoot;
}

private Appearance creaAppearanceTransparencyAttributes()
{
    // Creo un nuovo oggetto Appearance
    Appearance app = new Appearance();
    // Creo un nuovo oggetto TransparencyAttributes, specificando il metodo di rendering della trasparenza e
    l'ammontare della stessa
    TransparencyAttributes ta = new TransparencyAttributes(TransparencyAttributes.NICEST, 0.5f);
    // Imposto come TransparencyAttributes dell'Appearance l'oggetto TransparencyAttributes appena definito
    app.setTransparencyAttributes(ta);
    // Restituisco l'Appearance
    return app;
}

private Shape3D creaRiferimento()
{
    LineArray landGeom = new LineArray(44, GeometryArray.COORDINATES|GeometryArray.COLOR_3);
    float l = -50.0f;
    for(int c=0; c<44; c+=4)
    {
        landGeom.setCoordinate(c+0, new Point3f(-50.0f,0.0f,l));
        landGeom.setCoordinate(c+1, new Point3f(50.0f,0.0f,l));
        landGeom.setCoordinate(c+2, new Point3f(1,0.0f,-50.0f));
        landGeom.setCoordinate(c+3, new Point3f(1,0.0f,50.0f));
        l += 10.0f;
    }
    Color3f c = new Color3f(0.1f,0.8f,0.1f);
    for(int i=0; i<44; i++) landGeom.setColor(i,c);
    return new Shape3D(landGeom);
}
}
```

ESEMPIO BORDI IN RILIEVO

// Milanese Francesco --- Java 3D - Guida di base --- www.redbaron85.com

```
import java.awt.*;
import javax.swing.*;
import javax.vecmath.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.behaviors.keyboard.*;

public class BordiInRilievo extends JFrame
{
    public static void main(String[] args)
    {
        BordiInRilievo bir = new BordiInRilievo();
        bir.setVisible(true);
    }

    public BordiInRilievo()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocation(100, 100);
        this.setSize(400, 400);
        this.setTitle("Esempio bordi in rilievo; Milanese Francesco; www.redbaron85.com");
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        this.getContentPane().add(canvas3D, BorderLayout.CENTER);
        BranchGroup scene = createSceneGraph();
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewer().getView().setBackClipDistance(10000);
        simpleU.getViewer().getView().setFrontClipDistance(0.1);
        TransformGroup vpTG = simpleU.getViewingPlatform().getViewPlatformTransform();
        Transform3D t3d = new Transform3D();
        t3d.setTranslation(new Vector3f(0.0f, 0.3f, 0.0f));
        vpTG.setTransform(t3d);
        KeyNavigatorBehavior keyNavBeh = new KeyNavigatorBehavior(vpTG);
        keyNavBeh.setSchedulingBounds(new BoundingSphere(new Point3d(), 1000.0));
        scene.addChild(keyNavBeh);
        scene.compile();
        simpleU.addBranchGraph(scene);
    }

    private BranchGroup createSceneGraph()
    {
        BranchGroup objRoot = new BranchGroup();
        objRoot.addChild(this.creaRiferimento());
        objRoot.addChild(creaConi(2.0f, 5.0f));
        return objRoot;
    }

    // Metodo per la creazione di un TransformGroup e dei con i suoi figli
    private TransformGroup creaConi(float radius, float height)
    {
        // Creazione di un TransformGroup con trasformazione di traslazione
    }
}
```

```
// (per permetterci di visualizzare subito i coni)
Vector3f vettore = new Vector3f(0.0f, 0.0f, -10.0f);
Transform3D t3d = new Transform3D();
t3d.setTranslation(vettore);
TransformGroup tg = new TransformGroup(t3d);

// Creo, con un metodo esterno, un'appearance per la visualizzazione delle linee
// (modalità di resa dei poligoni POLYGON.LINE) con spessore 2 e colore blu
Appearance blueColorAppearance = getBlueColorAppearance();

// Creo, con un metodo esterno, un'appearance per la visualizzazione delle facce
// con colore rosso e valore di opacità 0.5
Appearance redColorAppearance = getredColorAppearance();

// Creo due coni con raggio e altezza dati ed impostando per il primo un'appearance
// rossa semitrasparente (ottenuta con getredColorAppearance),
// per il secondo l'appearance "fil di ferro" blu ottenuta con getBlueColorAppearance()
Cone cono = new Cone(radius, height);
cono.setAppearance(redColorAppearance);
Cone secondoCono = new Cone(radius, height);
secondoCono.setAppearance(blueColorAppearance);

// Aggiungo i due coni al TransformGroup e restituisco il Transform Group
tg.addChild(cono);
tg.addChild(secondoCono);
return tg;
}

// Creo un'appearance "fil di ferro" blu
private Appearance getBlueColorAppearance()
{
    // Creo un oggetto Appearance
    Appearance blueColorAppearance=new Appearance();
    // Creo gli attributi di resa dei poligoni per l'appearance: imposto cull_none
    // e resa "fil di ferro" o "struttura" (POLYGON_LINE)
    PolygonAttributes polygonAttributes = new PolygonAttributes(PolygonAttributes.POLYGON_LINE,
PolygonAttributes.CULL_NONE, 0.0f);
    blueColorAppearance.setPolygonAttributes(polygonAttributes);
    // Creo gli attributi per colorare l'appearance, in questo caso imposto il colore blu
    ColoringAttributes blueColoring=new ColoringAttributes();
    blueColoring.setColor(0.0f,0.0f,1.0f);
    blueColorAppearance.setColoringAttributes(blueColoring);
    // Imposto la modalità di resa delle linee per questa Appearance, mettendo come spessore "2.0f"
    LineAttributes lineAttrib=new LineAttributes();
    lineAttrib.setLineWidth(2.0f);
    blueColorAppearance.setLineAttributes(lineAttrib);
    // Restituisco l'appearance così definita
    return blueColorAppearance;
}

// Creo un'appearance trasparente rossa
private Appearance getredColorAppearance()
{
    // Creo un oggetto Appearance
    Appearance redColorAppearance=new Appearance();
    // Creo l'attributo per gestire la trasparenza di questa appearance,
    // mettendo come valore di opacità 0.5
```

```
        TransparencyAttributes transparencyAttributes = new
TransparencyAttributes(TransparencyAttributes.NICEST, 0.5f);
        redColorAppearance.setTransparencyAttributes(transparencyAttributes);
        // Creo gli attributi per colorare l'appearance, in questo caso imposto il colore rosso
        ColoringAttributes redColoring=new ColoringAttributes();
        redColoring.setColor(1.0f,0.0f,0.0f);
        redColorAppearance.setColoringAttributes(redColoring);
        // Restituisco l'appearance così definita
        return redColorAppearance;
    }

    private Shape3D creaRiferimento()
    {
        LineArray landGeom = new LineArray(44, GeometryArray.COORDINATES|GeometryArray.COLOR_3);
        float l = -50.0f;
        for(int c=0; c<44; c+=4)
        {
            landGeom.setCoordinate(c+0, new Point3f(-50.0f,0.0f,l));
            landGeom.setCoordinate(c+1, new Point3f(50.0f,0.0f,l));
            landGeom.setCoordinate(c+2, new Point3f(1,0.0f,-50.0f));
            landGeom.setCoordinate(c+3, new Point3f(1,0.0f,50.0f));
            l += 10.0f;
        }
        Color3f c = new Color3f(0.1f,0.8f,0.1f);
        for(int i=0; i<44; i++) landGeom.setColor(i,c);
        return new Shape3D(landGeom);
    }
}
```

```
DistanceLOD oggettoLoD = new DistanceLOD(arrayDiDistanze);
oggettoLoD.addSwitch(oggettoSwitch);
oggettoLoD.setSchedulingBounds(new BoundingSphere());
// Aggiungo l'oggetto DistanceLoD e l'oggetto Switch alla scena
objRoot.addChild(oggettoLoD);
objRoot.addChild(oggettoSwitch);
// Aggiungo alla scena una "griglia" di riferimento" ( un passo: 10 unità Java3D)
objRoot.addChild(this.disegnaGriglia());
// Restituisco lo Scene Graph così definito
return objRoot;
}

private Shape3D disegnaGriglia() {
    LineArray rif = new LineArray(132, GeometryArray.COORDINATES | GeometryArray.COLOR_3);
    float l = -50.0f;
    for(int x=0; x<44; x+=4)
    {
        rif.setCoordinate(x+0, new Point3f(-50.0f, 0.0f, l));
        rif.setCoordinate(x+1, new Point3f(50.0f, 0.0f, l));
        rif.setCoordinate(x+2, new Point3f(l, 0.0f, -50.0f));
        rif.setCoordinate(x+3, new Point3f(l, 0.0f, 50.0f));
        l+=10.0f;
    }
    l = -50.0f;
    for(int y=44; y<88; y+=4)
    {
        rif.setCoordinate(y+0, new Point3f(-50.0f, l, 0.0f));
        rif.setCoordinate(y+1, new Point3f(50.0f, l, 0.0f));
        rif.setCoordinate(y+2, new Point3f(l, -50.0f, 0.0f));
        rif.setCoordinate(y+3, new Point3f(l, 50.0f, 0.0f));
        l+=10.0f;
    }
    l = -50.0f;
    for(int z=88; z<132; z+=4)
    {
        rif.setCoordinate(z+0, new Point3f(0.0f, -50.0f, l));
        rif.setCoordinate(z+1, new Point3f(0.0f, 50.0f, l));
        rif.setCoordinate(z+2, new Point3f(0.0f, l, -50.0f));
        rif.setCoordinate(z+3, new Point3f(0.0f, l, 50.0f));
        l+=10.0f;
    }
    Color3f xc = new Color3f(0.1f, 0.9f, 0.1f);
    Color3f yc = new Color3f(0.9f, 0.1f, 0.1f);
    Color3f zc = new Color3f(0.1f, 0.1f, 0.9f);
    for(int x=0; x<44; x++) rif.setColor(x,xc);
    for(int y=44; y<88; y++) rif.setColor(y,yc);
    for(int z=88; z<132; z++) rif.setColor(z,zc);
    return new Shape3D(rif);
}
}
```

ESEMPIO COLOR INTERPOLATOR

// Milanesi Francesco --- Java 3D - Guida di base --- www.redbaron85.com

/*

Questo esempio mostra come generare animazioni time-based, indipendenti dalle azioni dell'utente. In particolare, qui il colore del materiale di un oggetto verrà modificato mediante ColorInterpolator.

In questo esempio si fa uso del KeyNavigatorBehavior di Java3D per navigare nello Spazio Virtuale.

*/

```
import java.awt.*;
import javax.swing.*;
import javax.vecmath.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.behaviors.keyboard.*;

public class EsempioColorInterpolator extends JFrame
{
    public static void main(String[] args)
    {
        EsempioColorInterpolator eci = new EsempioColorInterpolator();
        eci.setVisible(true);
    }

    public EsempioColorInterpolator()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocation(100, 100);
        this.setSize(400, 400);
        this.setTitle("EsempioColorInterpolator; Milanesi Francesco; www.redbaron85.com");
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        this.getContentPane().add(canvas3D, BorderLayout.CENTER);
        BranchGroup scene = createSceneGraph();
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewer().getView().setBackClipDistance(10000);
        simpleU.getViewer().getView().setFrontClipDistance(0.1);
        TransformGroup vpTG = simpleU.getViewingPlatform().getViewPlatformTransform();
        Transform3D t3d = new Transform3D();
        t3d.setTranslation(new Vector3f(0.0f, 0.3f, 2.0f));
        vpTG.setTransform(t3d);
        KeyNavigatorBehavior keyNavBeh = new KeyNavigatorBehavior(vpTG);
        keyNavBeh.setSchedulingBounds(new BoundingSphere(new Point3d(), 1000.0));
        scene.addChild(keyNavBeh);
        scene.compile();
        simpleU.addBranchGraph(scene);
    }

    private BranchGroup createSceneGraph()
    {
        BranchGroup objRoot = new BranchGroup();
        objRoot.addChild(this.conoCambiaColore());
        objRoot.addChild(this.creaRiferimento());
    }
}
```

```
objRoot.addChild(this.creaUnaLucePointLight());
return objRoot;
}

private BranchGroup conoCambiaColore()
{
    // Creo la radice di questo gruppo
    BranchGroup objRootInterno = new BranchGroup();

    // Creo un Appearance e un Material
    Appearance app = new Appearance();
    Material mtl = new Material();
    // Devo specificare quale colore di questo materiale é il colore TARGET del Position Interpolator; devo,
    inoltre, impostare le Capability
    mtl.setDiffuseColor(new Color3f(1.0f, 1.0f, 1.0f));
    mtl.setColorTarget(Material.DIFFUSE);
    mtl.setCapability(Material.ALLOW_COMPONENT_WRITE);
    // Imposto il materiale così definito come materiale dell'Appearance
    app.setMaterial(mtl);

    // Creo l'Alpha: 1 secondo per ogni fase
    Alpha mioAlpha = new Alpha(-1, Alpha.INCREASING_ENABLE | Alpha.DECREASING_ENABLE, 0,
    0, 1000, 0, 1000, 1000, 0, 1000);

    // Creo il Target: un TransformGroup, con le capability impostate
    TransformGroup tg = new TransformGroup();
    tg.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

    // Creo l'Interpolator, ne imposto i dati e il bound d'azione
    ColorInterpolator ci = new ColorInterpolator(mioAlpha, mtl, new Color3f(1.0f, 1.0f, 1.0f), new
    Color3f(1.0f, 1.0f, 0.0f));
    ci.setSchedulingBounds(new BoundingSphere());

    // Creo il Cono, gli associo l'Appearance con il relativo Material, e lo aggancio al TransformGroup
    Cone mioCono = new Cone(2.0f, 5.0f);
    mioCono.setAppearance(app);
    tg.addChild(mioCono);

    // Assemblo la scena
    objRootInterno.addChild(tg);
    objRootInterno.addChild(ci);

    // Restituisco il gruppo
    return objRootInterno;
}

private Shape3D creaRiferimento()
{
    LineArray landGeom = new LineArray(44, GeometryArray.COORDINATES|GeometryArray.COLOR_3);
    float l = -50.0f;
    for(int c=0; c<44; c+=4)
    {
        landGeom.setCoordinate(c+0, new Point3f(-50.0f,0.0f,1));
        landGeom.setCoordinate(c+1, new Point3f(50.0f,0.0f,1));
        landGeom.setCoordinate(c+2, new Point3f(1,0.0f,-50.0f));
        landGeom.setCoordinate(c+3, new Point3f(1,0.0f,50.0f));
        l += 10.0f;
    }
}
```

```
objRoot.addChild(this.creaRiferimento());
return objRoot;
}

private BranchGroup scalingCubo()
{
    // Creo la radice di questo gruppo
    BranchGroup bgScaling = new BranchGroup();

    // Sposto leggermente in alto il cubo
    Transform3D t3d1 = new Transform3D();
    t3d1.rotZ(Math.PI/2);
    t3d1.setTranslation(new Vector3f(0.0f, 1.0f, 0.0f));
    TransformGroup tg1 = new TransformGroup(t3d1);

    // Creo l'Alpha: lo scaling durerà in totale 4 secondi, solo increasing enable
    Alpha mioAlpha = new Alpha(-1, Alpha.INCREASING_ENABLE, 0, 0, 4000, 0, 0, 0, 0);

    // Creo il Target: un TransformGroup, con le capability impostate
    TransformGroup tg2 = new TransformGroup();
    tg2.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);

    // Creo l'Interpolatore, ne imposto i dati e il bound d'azione.
    ScaleInterpolator si = new ScaleInterpolator(mioAlpha, tg2);
    si.setSchedulingBounds(new BoundingSphere());

    // Creo il cubo e lo aggancio al TG dell'intrpolazione
    ColorCube cuboColorato = new ColorCube(1.0f);
    tg2.addChild(cuboColorato);

    // Assemblo la scena
    tg1.addChild(tg2);
    tg1.addChild(si);
    bgScaling.addChild(tg1);

    // Restituisco il gruppo
    return bgScaling;
}

private Shape3D creaRiferimento()
{
    LineArray landGeom = new LineArray(44, GeometryArray.COORDINATES|GeometryArray.COLOR_3);
    float l = -50.0f;
    for(int c=0; c<44; c+=4)
    {
        landGeom.setCoordinate(c+0, new Point3f(-50.0f,0.0f,l));
        landGeom.setCoordinate(c+1, new Point3f(50.0f,0.0f,l));
        landGeom.setCoordinate(c+2, new Point3f(l,0.0f,-50.0f));
        landGeom.setCoordinate(c+3, new Point3f(l,0.0f,50.0f));
        l += 10.0f;
    }
    Color3f c = new Color3f(0.1f,0.8f,0.1f);
    for(int i=0; i<44; i++) landGeom.setColor(i,c);
    return new Shape3D(landGeom);
}
}
```

ESEMPIO QUATERNIONI

// Milanesi Francesco --- Java 3D - Guida di base --- www.redbaron85.com

/*

Questo esempio mostra come "configurare" ed utilizzare i quaternioni per specificare degli orientamenti nello spazio3D, facendo uso di oggetti visuali statici per mostrare gli effetti.

In questo esempio si fa uso del KeyNavigatorBehavior di Java3D per navigare nello Spazio Virtuale.

*/

```
import java.awt.*;
import javax.swing.*;
import javax.vecmath.*;
import javax.media.j3d.*;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.*;
import com.sun.j3d.utils.behaviors.keyboard.*;

public class EsempioQuaternioni extends JFrame
{
    public static void main(String[] args)
    {
        EsempioQuaternioni eq = new EsempioQuaternioni();
        eq.setVisible(true);
    }

    public EsempioQuaternioni()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocation(100, 100);
        this.setSize(400, 400);
        this.setTitle("Esempio Quaternioni; Milanesi Francesco; www.redbaron85.com");
        GraphicsConfiguration config = SimpleUniverse.getPreferredConfiguration();
        Canvas3D canvas3D = new Canvas3D(config);
        this.getContentPane().add(canvas3D, BorderLayout.CENTER);
        BranchGroup scene = createSceneGraph();
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewer().getView().setBackClipDistance(10000);
        simpleU.getViewer().getView().setFrontClipDistance(0.1);
        TransformGroup vpTG = simpleU.getViewingPlatform().getViewPlatformTransform();
        Transform3D t3d = new Transform3D();
        t3d.setTranslation(new Vector3f(0.0f, 0.3f, 2.0f));
        vpTG.setTransform(t3d);
        KeyNavigatorBehavior keyNavBeh = new KeyNavigatorBehavior(vpTG);
        keyNavBeh.setSchedulingBounds(new BoundingSphere(new Point3d(), 1000.0));
        scene.addChild(keyNavBeh);
        scene.compile();
        simpleU.addBranchGraph(scene);
    }

    private BranchGroup createSceneGraph()
    {
        BranchGroup objRoot = new BranchGroup();
        objRoot.addChild(this.creaRiferimento());
    }
}
```