Java 3D

Laboratório de Computação
Jorge Barbosa

2002

Java 3D

- API para desenvolver aplicações 3D, com capacidade de descrever ambientes virtuais complexos
- Fornece um conjunto de construções de alto nível para:
  - Descrever objectos 3D (geometria, aparência, transformação, comportamento)
  - Construir o grafo necessário para efectuar *rendering* da cena criada

Objectivos

- **Desempenho:** utiliza outros API's a um nível inferior que implementam funções gráficas optimizadas e adaptados a cada arquitectura. Ex: DirectX ou OpenGL.
- **Fácil utilização:** programação OO de alto nível.
- **Compatibilidade:** existe suporte para vários formatos de dados: programas CAD específicos, VRML, etc.

Utilização

- Modelo de programação:
  - Um programa consiste em criar instâncias de classes do Java3D, ligando-as posteriormente numa estrutura em árvore, a qual se designa por ***Scene Graph***.
- Modelo de execução:
  - Na execução, o Java 3D inicia um ciclo infinito, percorrendo os nós do *Scene Graph*; efectua os comportamentos descritos e o *rendering* dos objectos visíveis.

packages

javax.media.j3d - classes principais

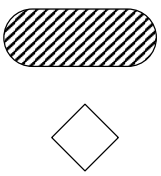
javax.vecmath - classes úteis para definir a geometria dos objectos na cena (Point*, Color*, Vector*, TexCoord*, etc)
(* : 3b, 3f, 3d, 4b, 4f, 4d, etc) [Tutorial 2-16]

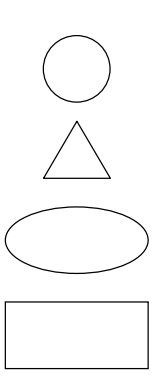
com.sun.j3d.utils - 4 categorias: loaders, classes de ajuda na criação da cena, geometria (Box, Sphere, ColorCube) e outros utilitários (e.g. KeyNavigatorBehavior, etc)

Classes Símbolo no grafo

Javax.media.j3d
VirtualUniverse
Locale
View
PhysicalBody
PhysicalEnvironment
Screen3D
Canvas3D (extends awt.Canvas)
SceneGraphObject
- Node
- Group
- Leaf
- NodeComponent

Transform3D

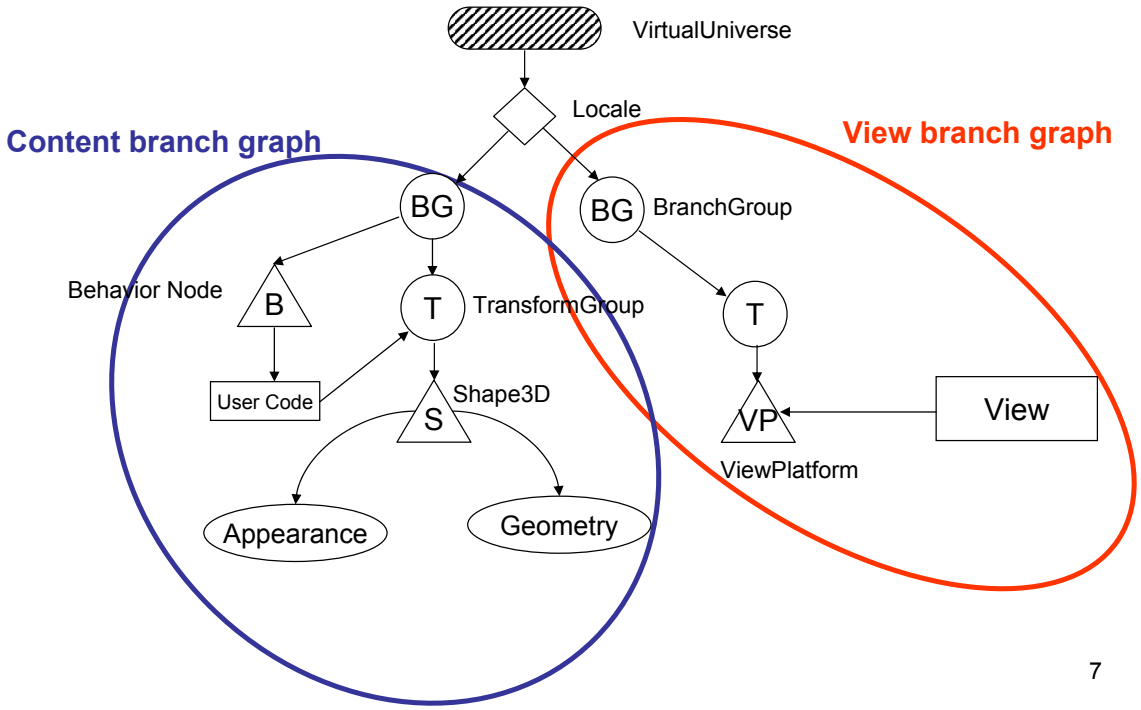




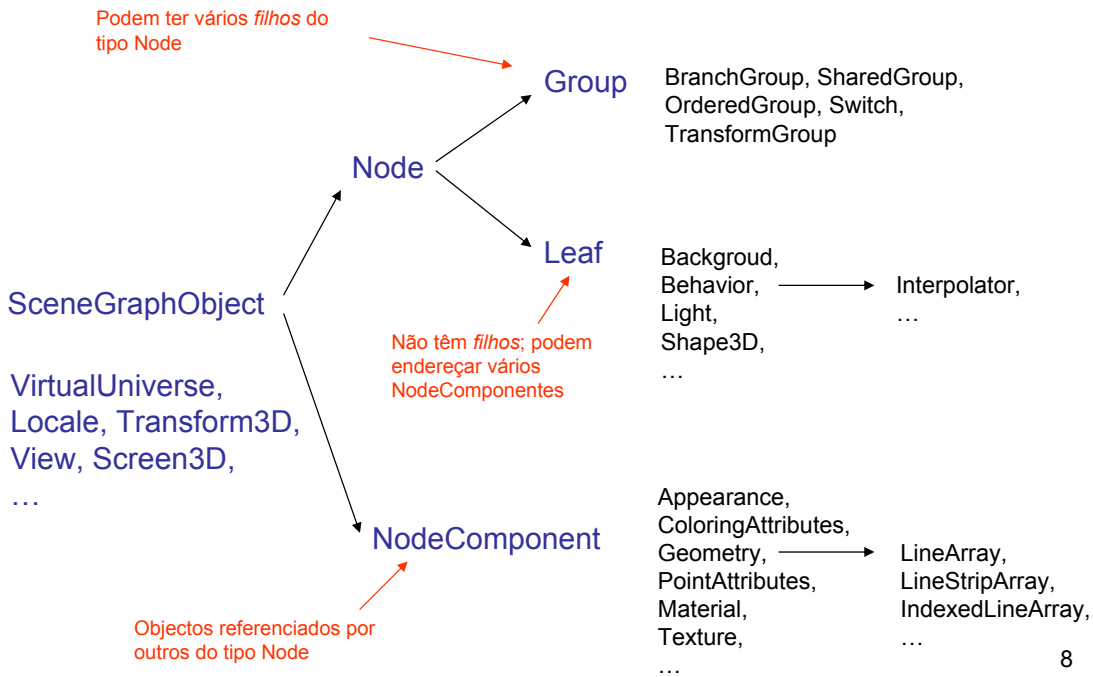
Classes para definir o *content graph*

Outros

Exemplo de um grafo

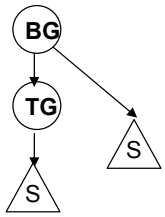


Javax.media.j3d



javax.media.j3d

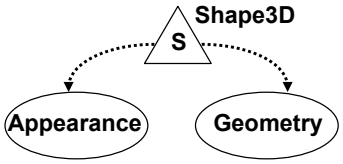
Group Subclasses de *Group* são usadas para organizar o Scene Graph. Podem ter vários nós como filhos, quer do tipo *Group* quer do tipo *Leaf*.



Exemplo de subcalsses:
•TransformGroup
•BranchGroup

Leaf Subclasses de *Leaf* são usadas para representar os objectos que constituem a cena, como geometria, luz e som.
Estes nós não têm filhos; apenas podem referenciar objectos do tipo *NodeComponent*.

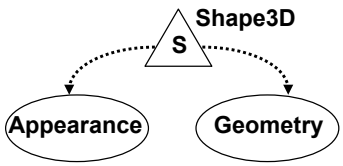
Exemplo de subcalsse:
•Shape3D



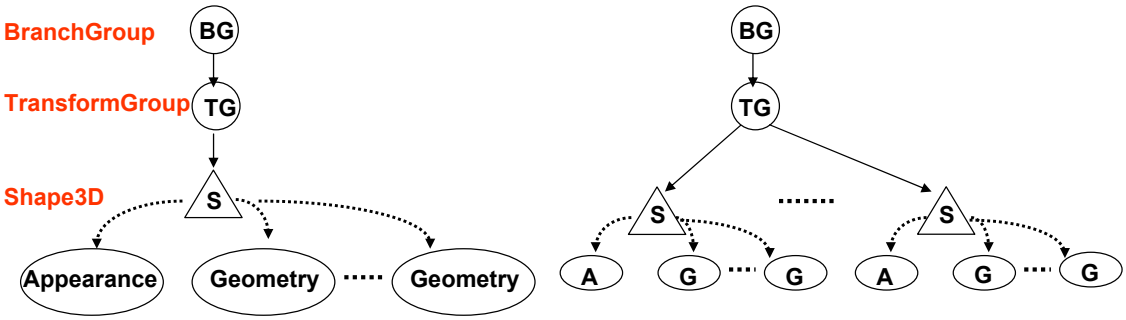
javax.media.j3d

NodeComponent *Superclasse* para classes de geometria e aparência. Objectos deste tipo não são inseridos no grafo, apenas são referênciados por nós do tipo *Leaf*.

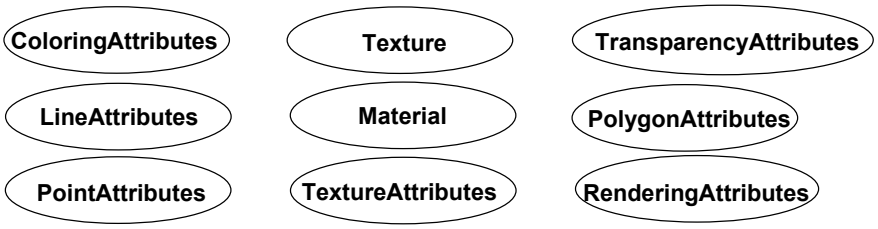
Exemplo de subcalsses:
•Geometry
•Appearance
•LineAttributes
•PointAttributes
•ColoringAttributes
•PolygonAttributes
•Material
•Texture
...



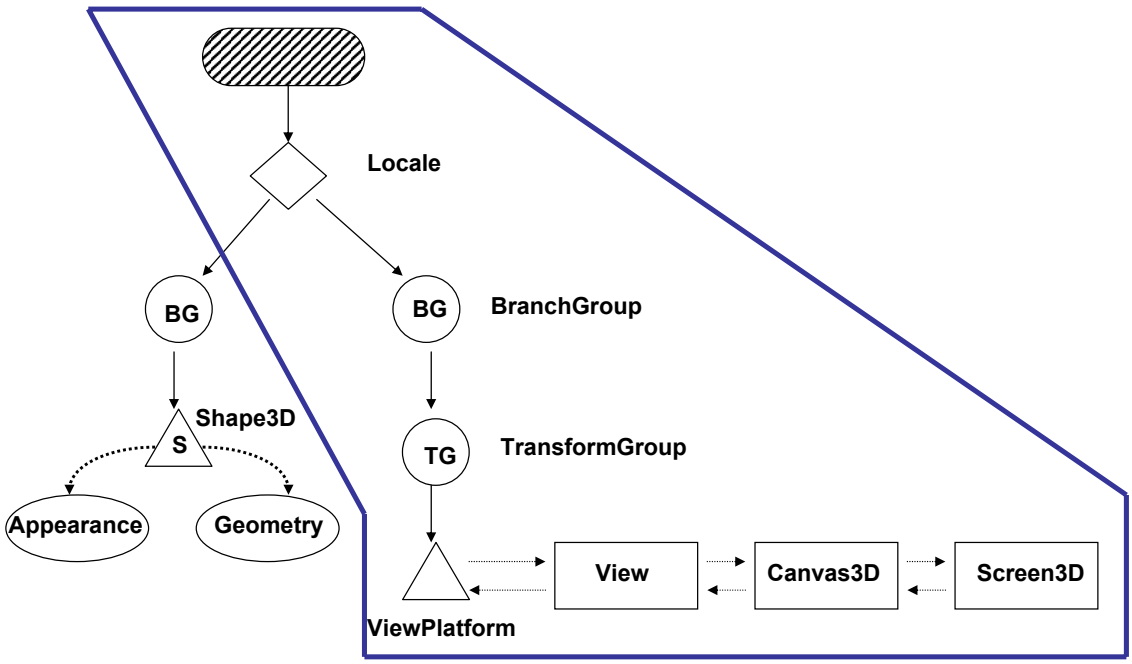
Especificação de um Objecto na cena



Objectos endereçáveis por objectos Appearance (descrevem atributos):



SimpleUniverse

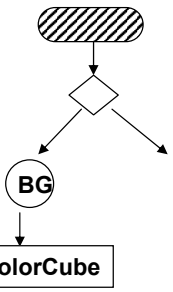


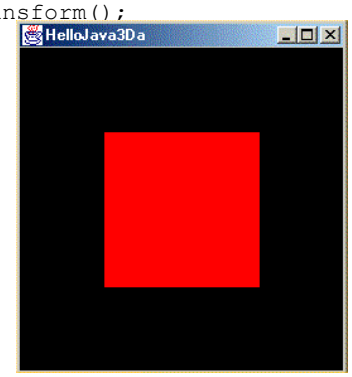
Construção de um programa

1. Criar um objecto Canvas3D
2. Criar um objecto SimpleUniverse o qual referencia Canvas3D
   a. Configurar o SimpleUniverse
3. Construir o Content Branch Graph
4. Compilar o Content Branch Graph
5. Inserir Content Branch Graph no objecto Locale do SimpleUniverse

Exemplo: HelloJava3Da

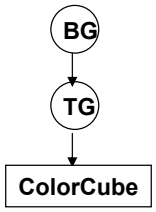
```
public HelloJava3Da() {
        setLayout(new BorderLayout());
        Canvas3D canvas3D = new Canvas3D(null);
        add("Center", canvas3D);

        BranchGroup scene = createSceneGraph();

        // SimpleUniverse is a Convenience Utility class
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);

       // This will move the ViewPlatform back a bit so the
       // objects in the scene can be viewed.
        simpleU.getViewingPlatform().setNominalViewingTransform();

        simpleU.addBranchGraph(scene);
    } // end of HelloJava3Da (constructor)

     public BranchGroup createSceneGraph() {
       // Create the root of the branch graph
       BranchGroup objRoot = new BranchGroup();

       objRoot.addChild(new ColorCube(0.4));

       return objRoot;
    } // end of CreateSceneGraph method of HelloJava3Da
```

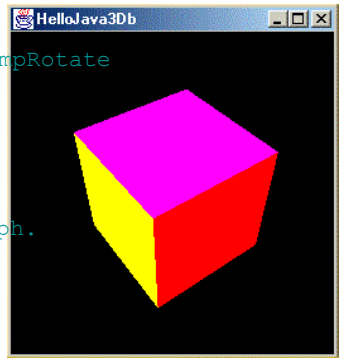




Rotação do cubo

```
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);              // rotate=rotate*tempRotate
 TransformGroup objRotate = new TransformGroup(rotate);

 objRoot.addChild(objRotate);
 objRotate.addChild(new ColorCube(0.4));
 // Let Java 3D perform optimizations on this scene graph.
  objRoot.compile();

 return objRoot;
} // end of CreateSceneGraph method of HelloJava3Db
```





Classes usadas no exemplo

- BranchGroup
  - Usada para criar subgrafos. As suas instâncias são os únicos objectos que podem ser ligados ao Locale.
  - compile() : o Java3D efectua optimizações em todo o subgrafo, mesmo que inclua outros BG .
  - Objectos BG podem ser retirados ou colocados no grafo em run time se **ALLOW_DETACH** for true .
  - Quando adicionado ao grafo, o subgrafo diz-se *Vivo*, i.e. os objectos passam a ser visualizados (*rendering)*
- BranchGroup capability
  - **ALLOW_DETACH**

Classes usadas no exemplo

- Transform3D
  - Representam transformações 3D como translações, rotações e escalamentos. Os nós TransformGroup copiam a matriz de transformação de um Transform3D para a sua estrutura interna. Os objectos Transform3D não são inseridos no grafo nem referenciados por qualquer um dos seus nós.
  - Alguns métodos disponíveis:
    - `set()`
    - `setTranslation()`
    - `setRotation()`
    - `setScale()`
    - `rotx()`
    - `roty()`
    - `rotz()`
    - `mul()`
    - `invert()`
    - `...`

Classes usadas no exemplo

- TransformGroup
  - Classe usada na construção do grafo para implementar as transformações necessárias nos nós que lhe estão ligados
  - Construtor: `TransformGroup(Transform3D t1)`
  - Alguns métodos disponíveis:
    - `getTransform(Transform3D t1)`
    - `setTransform(Transform3D t1)`
  - TransformGroup capabilities:
    - `ALLOW_TRANSFORM_READ`
    - `ALLOW_TRANSFORM_WRITE`

Classes usadas no exemplo

- Shape3D
  - Classe usada na construção do grafo para representar os objectos que constituem a cena. Estes nós só referenciam NodeComponents do tipo Geometry e Appearance.
  - Construtor: `Shape3D(Geometry geom, Appearance app)`
  - Alguns métodos disponíveis:
    - `setGeometry(Geometry geom)`
    - `setAppearance(Appearance app)`
  - TransformGroup capabilities:
    - `ALLOW_GEOMETRY_READ`
    - `ALLOW_GEOMETRY_WRITE`
    - `ALLOW_APPEARANCE_READ`
    - `ALLOW_APPEARANCE_WRITE`
    - `...`