Java 3D™ API

Profa. Isabel Harb Manssour

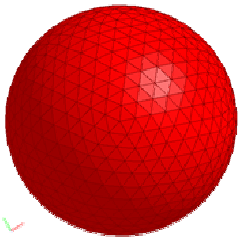
(http://www.inf.pucrs.br/~manssour/Java3D)

XVI SIBGRAPI – 12-15 de Outubro de 2003

3. Realismo, Interação e Animação

Realismo, Interação e Animação

- Para gerar imagens com realismo é necessário implementar várias técnicas que permitem reproduzir a realidade em termos de "aparência"
  - Cor
  - Efeitos de iluminação
  - Textura
  - Sombra
  - Outras

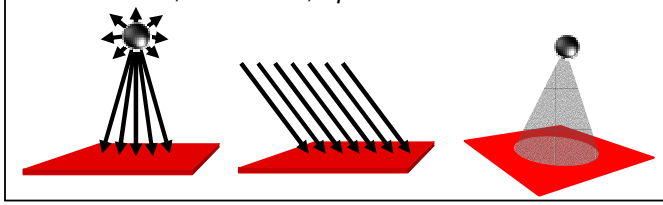


Realismo, Interação e Animação

- **Iluminação**
- Textura
- Interação
- Animação

Realismo, Interação e Animação
Iluminação

- Fonte de luz
  - Objeto que está emitindo energia radiante
- Alguns modelos de fonte de luz:
  - Pontual, Direcional, *Spots*



Realismo, Interação e Animação
Iluminação

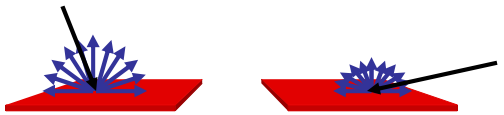
- Modelo de Reflexão (ou Modelo de Iluminação)
  - Descreve a interação da luz com uma superfície, em termos das propriedades da superfície e da natureza da luz incidente
- Objetivo
  - Exibir objetos tridimensionais no espaço de tela bidimensional que se aproximem da realidade
  - Fazer com que objetos do tipo espelho apresentem em sua superfície a imagem de outros objetos do universo

Realismo, Interação e Animação
Iluminação

- **Reflexão Ambiente (Luz Ambiente)**
  - Luz ambiente
    - Fonte de luz difusa, não direcional, resultante de múltiplas reflexões da luz com as superfícies
    - Incidente em uma superfície de todas as direções
  - Origina-se da interação da reflexão difusa com todas as superfícies da cena
  - Superfícies que não recebem raios de luz diretos da fonte são visíveis devido à luz ambiente

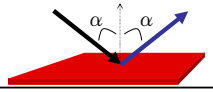
Realismo, Interação e Animação
Iluminação

- **Reflexão Difusa (Luz Difusa)**
  - Maioria dos objetos
    - Não emitem luz
    - Absorvem a luz do sol ou a luz emitida de uma fonte artificial, e refletem parte desta luz
    - Reflexão deve-se ao fato de haver uma interação molecular entre a luz incidente e o material da superfície
  - Cria o efeito de *degradé* nos objetos



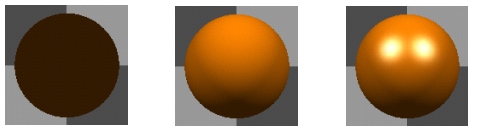
Realismo, Interação e Animação
Iluminação

- **Reflexão Especular (Luz Especular)**
  - A maioria das superfícies na vida real não são totalmente opacas
    - Geralmente, possuem algum grau de brilho
  - Superfície perfeitamente brilhosa: espelho
  - É a componente da luz que produz o ponto de brilho mais acentuado
  - Gera um brilho com a cor da luz e não com a cor do objeto



Realismo, Interação e Animação
Iluminação

- **Ambiente X Difusa X Especular**



Ambiente Difusa Especular

Realismo, Interação e Animação
Iluminação

- Classe *Light*
  - Classe abstrata que define um conjunto de parâmetros comum para todos os tipos de fonte de luz
    - Cor da luz
    - *Flag* (ON/OFF)
    - Região de influência, onde a luz está "ativa"

Realismo, Interação e Animação
Iluminação

- Classe *AmbientLight*
  - Resultado da luz refletida no ambiente (vem de todas as direções)
    - Não depende da orientação ou posição da superfície
  - Mesmos atributos de *Light*
  - Possui apenas um componente de reflexão ambiente
  - Construtores
    - ***AmbientLight*** *()*
    - ***AmbientLight*** *(boolean lightOn, Color3f color)*
    - ***AmbientLight*** *(Color3f color)*

Realismo, Interação e Animação
Iluminação

- Classe *DirectionalLight*
  - Especifica uma fonte de luz "orientada" com origem no infinito
  - Mesmos atributos de *Light* com a adição de um vetor que especifica a direção de onde a luz brilha
  - Contribui para as reflexões difusa e especular
  - Construtores
    - ***DirectionalLight*** *()*
    - ***DirectionalLight*** *(boolean lightOn, Color3f color, Vector3f direction)*
    - ***DirectionalLight*** *(Color3f color, Vector3f direction)*

Realismo, Interação e Animação
Iluminação

- Classe *PointLight*
  - Especifica uma fonte de luz em um ponto fixo que espalha raios de luz igualmente em todas as direções
  - Mesmos atributos de *Light* com a adição de parâmetros de localização e atenuação
  - Contribui para as reflexões difusa e especular
  - Construtores
    - ***PointLight*** *()*
    - ***PointLight*** *(boolean lightOn, Color3f c, Point3f position, Point3f attenuation)*
    - ***PointLight*** *(Color3f c, Point3f position, Point3f attenuation)*

Realismo, Interação e Animação
Iluminação

- Classe *SpotLight*
  - Especifica uma fonte de luz em um ponto fixo que espalha raios de luz em uma direção a partir da fonte
  - Mesmos atributos de *Light* com a adição dos seguintes parâmetros
    - Direção: eixo do cone de luz
    - Ângulo de expansão (entre o eixo de direção e a aresta do cone de luz)
    - Concentração: o quanto a intensidade da luz é atenuada em função do ângulo de expansão

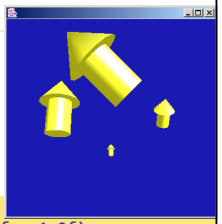
Realismo, Interação e Animação
Iluminação

- Classe *SpotLight*
  - Contribui para as reflexões difusa e especular
  - Construtores
    - ***SpotLight*** *()*
    - ***SpotLight*** *(boolean lightOn, Color3f color, Point3f position, Point3f attenuation, Vector3f direction, float spreadAngle, float concentration)*
    - ***SpotLight*** *(Color3f color, Point3f position, Point3f attenuation, Vector3f direction, float spreadAngle, float concentration)*

Realismo, Interação e Animação

Iluminação

- Exemplo 1

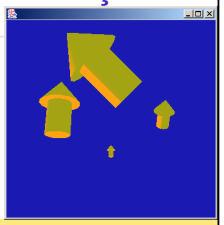


```
Color3f corLuz = new Color3f(0.9f, 0.9f, 0.9f);
Vector3f direcaoLuz  = new Vector3f(-1.0f, -1.0f, -1.0f);
Color3f corAmb = new Color3f(0.2f, 0.2f, 0.2f);
AmbientLight luzAmb = new AmbientLight(corAmb);
luzAmb.setInfluencingBounds(bounds);
DirectionalLight luzDir = new DirectionalLight(corLuz, direcaoLuz);
luzDir.setInfluencingBounds(bounds);
objRaiz.addChild(luzAmb);
objRaiz.addChild(luzDir);

Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
          new Color3f(0.0f,0.0f,0.0f), new Color3f(0.8f,0.8f,0.1f),
          new Color3f(1.0f,1.0f,1.0f), 100.0f);
```

Realismo, Interação e Animação

Iluminação

- Exemplo 2

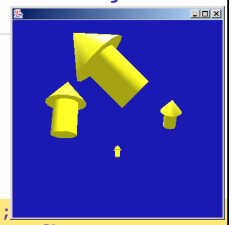


```
Color3f corLuz = new Color3f(0.9f, 0.0f, 0.0f); // Alterado
Vector3f direcaoLuz  = new Vector3f(1.0f, 1.0f, 1.0f); // Alterado
Color3f corAmb = new Color3f(0.2f, 0.2f, 0.2f);
AmbientLight luzAmb = new AmbientLight(corAmb);
luzAmb.setInfluencingBounds(bounds);
DirectionalLight luzDir = new DirectionalLight(corLuz, direcaoLuz);
luzDir.setInfluencingBounds(bounds);
objRaiz.addChild(luzAmb);
objRaiz.addChild(luzDir);
Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
          new Color3f(0.0f,0.0f,0.0f), new Color3f(0.8f,0.8f,0.1f),
          new Color3f(1.0f,1.0f,1.0f), 100.0f);
```


Realismo, Interação e Animação

Iluminação

- Exemplo 3

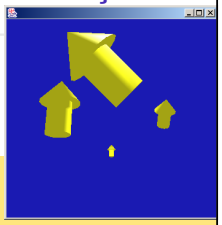


```
Color3f corLuz = new Color3f(0.9f, 0.9f, 0.9f);
Point3f posicaoLuz   = new Point3f(0.0f, 2.0f, 0.0f);
Point3f atenuacaoLuz = new Point3f(0.2f, 0.2f, 0.2f);
Color3f corAmb = new Color3f(0.2f, 0.2f, 0.2f);
AmbientLight luzAmb = new AmbientLight(corAmb);
luzAmb.setInfluencingBounds(bounds);
objRaiz.addChild(luzAmb);
PointLight luzPont = new PointLight(corLuz,posicaoLuz,atenuacaoLuz);
luzPont.setInfluencingBounds(bounds);
objRaiz.addChild(luzPont);
Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
          new Color3f(0.0f,0.0f,0.0f), new Color3f(0.8f,0.8f,0.1f),
          new Color3f(1.0f,1.0f,1.0f), 100.0f);
```

Realismo, Interação e Animação

Iluminação

- Exemplo 4



```
Color3f corAmb = new Color3f(0.2f, 0.2f, 0.2f);
AmbientLight luzAmb = new AmbientLight(corAmb);
luzAmb.setInfluencingBounds(bounds);
objRaiz.addChild(luzAmb);
Color3f corLuz       = new Color3f(1.0f, 1.0f, 1.0f);
Point3f posicaoLuz   = new Point3f(0.0f, 2.0f, 0.0f);
Point3f atenuacaoLuz = new Point3f(0.1f, 0.1f, 0.1f);
Vector3f direcaoLuz  = new Vector3f(0.0f, -1.0f, 0.0f);
SpotLight luzSpot = new SpotLight(corLuz, posicaoLuz, atenuacaoLuz,
                  direcaoLuz, (float)Math.toRadians(12), 60.0f);
luzSpot.setInfluencingBounds(bounds);
objRaiz.addChild(luzSpot);
Material material = new Material(new Color3f(0.8f,0.8f,0.1f),
          new Color3f(0.0f,0.0f,0.0f), new Color3f(0.8f,0.8f,0.1f),
          new Color3f(1.0f,1.0f,1.0f), 100.0f);
```

Exercício 4

Objetivo: testar as diferentes fontes de luz existentes na API Java 3D

Realismo, Interação e Animação

- Iluminação
- **Textura**
- Interação
- Animação

Realismo, Interação e Animação
Textura

- Cada tipo de material tem características próprias, que permitem sua identificação visual ou táctil
  - Microestruturas que produzem rugosidade na superfície dos objetos
  - Exemplos: plástico, madeira, areia e mármore
- Em Computação Gráfica, estes detalhes da superfície de um objeto são chamados de textura

Realismo, Interação e Animação
Textura

- Textura mapeada
  - Uma técnica de mapeamento de texturas
  - Consiste simplesmente no mapeamento de uma imagem (mapa de textura/padrão de textura) para a superfície do objeto

Realismo, Interação e Animação
Textura

- Textura de polígonos em Java 3D
  - Criação da aparência apropriada
  - Armazenamento da imagem de textura, incluindo o seu posicionamento na geometria e a especificação dos atributos
- "Receita" para a especificação [Sun 2003]
  1. Preparar a imagem de textura
  2. Carregar a textura
  3. Associar a textura com a aparência
  4. Determinar as coordenadas de textura da geometria

Realismo, Interação e Animação
Textura

- Preparar a imagem de textura
  - Consiste na criação e edição da textura
  - Externo ao programa Java 3D
  - Essencial:
    - Imagem deve estar num formato compatível para leitura
      - JPG, GIF, entre outros
    - O seu tamanho deve ser múltiplo de 2 em cada dimensão
      - 2, 4, 8, 16, 128, 256, ...

Realismo, Interação e Animação
Textura

- Carregar a textura
  - A imagem pode estar em um arquivo local ou em uma URL
  - Para facilitar, utiliza-se um objeto *TextureLoader*

```
// Applet
java.net.URL texImage = null;
try { texImage = new java.net.URL(getCodeBase().toString()
                                  +"deserto.jpg");
}
catch (java.net.MalformedURLException ex) {
     System.out.println(ex.getMessage());
     System.exit(1);
}
TextureLoader loader = new TextureLoader(texImage, this);
:
```

Realismo, Interação e Animação
Textura

- Definir a textura na aparência
  - Colocar a textura no objeto *Appearance*

```
:
TextureLoader loader = new TextureLoader(texImage, this);
app.setTexture(loader.getTexture());
Material material = new Material(new Color3f(0.2f,0.2f,0.2f),
                                 new Color3f(0.0f,0.0f,0.0f),
                                 new Color3f(1.0f,1.0f,1.0f),
                                 new Color3f(0.5f,0.5f,0.5f), 100.0f);
app.setMaterial(material);
:
```

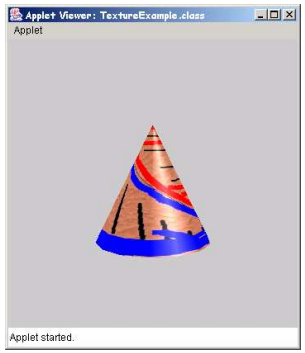
Realismo, Interação e Animação
Textura

- Especificar as coordenadas de textura da geometria
  - O programador determina a posição da textura em uma geometria através de coordenadas de textura
    - Definidas por vértice
    - Determinam um ponto de textura a ser aplicado no vértice
    - Dependendo da especificação dos pontos, a imagem será rotacionada, "esticada", duplicada, etc

```
:
Cone cone = new Cone(0.4f, 0.8f, Cone.GENERATE_NORMALS |
                 Cone.GENERATE_TEXTURE_COORDS, 14, 14, app);
cone.setAppearance(app);
:
```

Realismo, Interação e Animação
Textura

- Resultado



Exercício 5

Objetivo: testar a aplicação de texturas em objetos

Realismo, Interação e Animação

- Iluminação
- Textura
- **Interação**
- Animação

Realismo, Interação e Animação
Interação

- Interação
  - Consiste na ocorrência de alterações em resposta a ações do usuário
  - Objetivo: alterar o grafo de cena, ou os seus objetos, em resposta a estímulos do usuário
    - Pressionar uma tecla, mover o mouse...
- A interação e animação são especificadas com subclasses da **classe abstrata *Behavior***

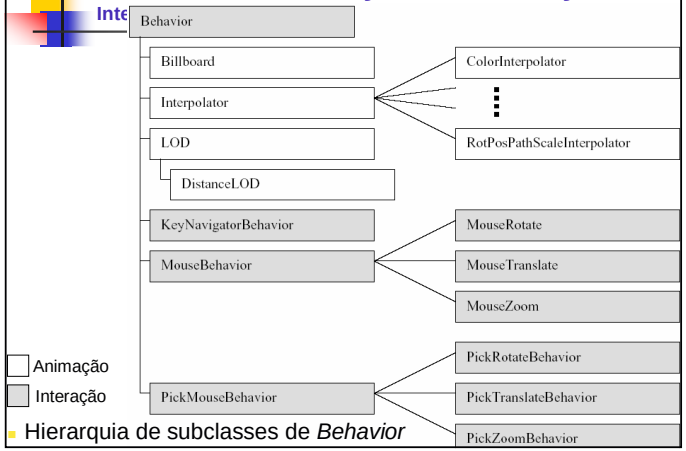
Realismo, Interação e Animação
Interação

- Classe abstrata *Behavior*
  - Fornece mecanismos para incluir o código necessário para alteração
    - Remover objetos do grafo de cena, trocar seus atributos...
  - Esta classe e seus descendentes consistem em *links* para código do usuário
    - Altera a parte gráfica e/ou os sons do universo virtual

Realismo, Interação e Animação
Interação

- *Behavior*
  - Elo entre um estímulo e uma ação
  - Um estímulo pode resultar em uma ou mais alterações

Realismo, Interação e Animação
Inte



- Hierarquia de subclasses de *Behavior*

Realismo, Interação e Animação
Interação

- "Receita" para criar subclasse de *Behavior* [Sun 2003]
  - Implementar (pelo menos um) construtor
    - Armazenar uma referência para o objeto que será alterado
  - Sobrescrever o método *initialization()*
    - Especificar um critério para começar a interação
  - Sobrescrever o método *processStimulus()*
    - Executa a ação
- Exemplo [Sun 2003]...

Realismo, Interação e Animação
Interação

```
private class SimpleBehavior extends Behavior {
     private TransformGroup targetTG;
     private Transform3D rotation = new Transform3D();
     private double angle = 0.0;
    // create SimpleBehavior - set TG object of change
    SimpleBehavior(TransformGroup targetTG) {
       this.targetTG = targetTG;
    }
    // initialize the Behavior set initial wakeup condition
    // called when behavior becomes live
    public void initialize() {  // set initial wakeup condition
       this.wakeupOn(new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED));
    }
    // called by Java 3D when appropriate stimulus occurs
    public void processStimulus(Enumeration criteria) {
        // do what is necessary in response to stimulus
        angle += 0.1;
        rotation.rotY(angle);
        targetTG.setTransform(rotation);
        this.wakeupOn(new WakeupOnAWTEvent(KeyEvent.KEY_PRESSED));
    }
 } // end of class SimpleBehavior [Sun 2003]
```


Realismo, Interação e Animação
Interação

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

    TransformGroup objRotate = new TransformGroup();
    objRotate.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
    objRoot.addChild(objRotate);
    objRotate.addChild(new ColorCube(0.4));

    SimpleBehavior myRotationBehavior=new SimpleBehavior(objRotate);
    myRotationBehavior.setSchedulingBounds(new BoundingSphere());
    objRoot.addChild(myRotationBehavior);

    // Let Java 3D perform optimizations on this scene graph.
    objRoot.compile();
       return objRoot;
}
```

Realismo, Interação e Animação
Interação

- Classe *Behavior*
  - Fornece um conjunto de funcionalidades para incluir ações definidas pelo usuário no grafo de cena
  - Especifica um critério para começar a interação
  - Métodos
    - *abstract void initialize ()*
      - É chamado apenas uma vez para inicializar o *behavior*
    - *abstract void processStimulus (java.util.Enumeration criteria)*
      - Processa uma ação em resposta a um estímulo
    - *protected void wakeupOn (WakeupCondition criteria)*
      - Define o *wakeup* critério

Realismo, Interação e Animação
Interação

- Classe *WakeupOnAWTEvent*
  - Indica que uma interação deve ser realizada quando um determinado evento AWT ocorre
  - Construtores e métodos
    - *WakeupOnAWTEvent (int AWTId)*
    - *WakeupOnAWTEvent (long eventMask)*
    - *java.awt.AWTEvent [] getAWTEvent ()*
      - Retorna o vetor de eventos AWT consecutivos que foram desencadeados neste *wakeup*

Realismo, Interação e Animação
Interação

- Outra maneira de interagir com os objetos é através da classe *OrbitBehavior*
  - Move a *View* em torno do ponto de interesse quando o mouse é arrastado com o botão pressionado
  - Inclui ações
    - Rotação: arrastar o mouse com o botão esquerdo pressionado
    - Translação: arrastar o mouse com o botão direito pressionado
    - *Zoom*: arrastar o mouse com o botão do meio pressionado (ou alt+botão esquerdo)
  - Exemplo...

Realismo, Interação e Animação
Interação

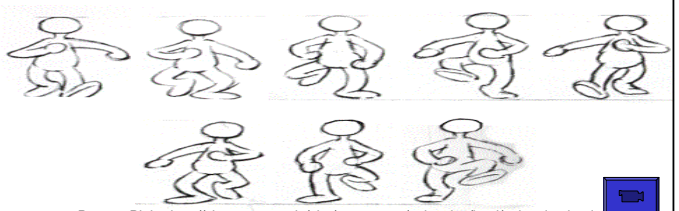
```
public Behavior() {
    :
    BranchGroup scene = criaGrafoDeCena();
    universe = new SimpleUniverse(canvas);

    // O código abaixo faz com que a ViewPlatform seja movida um
    // pouco para trás, para que os objetos possam ser visualizados
    ViewingPlatform viewingPlatform = universe.getViewingPlatform();
    viewingPlatform.setNominalViewingTransform();

    // Adiciona "mouse behaviors" à "viewingPlatform"
    // (equivale a trocar a posição do "observador virtual")
    OrbitBehavior orbit = new OrbitBehavior(canvas,
    OrbitBehavior.REVERSE_ALL);
    BoundingSphere bounds = new BoundingSphere(new
    Point3d(0.0,0.0,0.0), 100.0);
    orbit.setSchedulingBounds(bounds);
    viewingPlatform.setViewPlatformBehavior(orbit);
    universe.addBranchGraph(scene);
    :
}
```

Realismo, Interação e Animação

- Iluminação
- Textura
- Interação
- **Animação**

Realismo, Interação e Animação
Animação

- Animação
  - Exibição de imagens em seqüência
  - FPS – *Frames* (ou quadros) por segundo
    - Vídeo: 30 FPS



Preston Blair - http://viz.aset.psu.edu/gho/sem_notes/animation/html/animation.html

Realismo, Interação e Animação
Animação

- Animação aqui é tratada como:
  - Alterações que ocorrem no universo virtual independente das ações do usuário
- Assim como a interação, a animação é realizada através de objetos *Behavior*
- A API Java 3D fornece um grande número de classes úteis para criar animações

Realismo, Interação e Animação
Animação

- Um conjunto de classes de animação: *Interpolator*
  - Objetos *Interpolator* em conjunto com objetos *Alpha* manipulam alguns parâmetros de um grafo de cena para criar uma animação baseada no tempo
  - *Alpha* e *RotationInterpolation*
    - Descritos anteriormente

Realismo, Interação e Animação
Animação

- *PositionInterpolation*
  - Esta classe define um comportamento (*interpolator behavior*)
  - Modifica o componente de translação do *TransformGroup* através de uma interpolação linear entre um par de posições especificadas (usando o valor gerado por um objeto *Alpha*)
  - A posição interpolada é usada para gerar uma transformação de translação sobre o eixo X

Realismo, Interação e Animação
Animação

- *PositionInterpolation*
  - Construtores
    - *PositionInterpolator (Alpha a, TransformGroup target)*
      - Cria um *position interpolator* trivial com o *TransformGroup* especificado, com valores *default* para um eixo de transformação (identidade), posição inicial (0.0) e posição final (1.0)
    - *PositionInterpolator (Alpha a, TransformGroup target, Transform3D axisOfTransform, float startPosition, float endPosition)*
      - Cria um novo *position interpolator* que varia o componente de translação do nodo de transformação

Realismo, Interação e Animação
Animação

- *ScaleInterpolation*
  - Esta classe define um comportamento (*interpolator behavior*)
  - Modifica o componente de escala uniforme do *TransformGroup* através de uma interpolação linear entre um par de valores de escala especificados (usando o valor gerado por um objeto *Alpha*)
  - O valor de escala interpolado é usado para gerar uma transformação de escala sobre o sistema de coordenada do *interpolator*

Realismo, Interação e Animação
Animação

- *ScaleInterpolation*
  - Construtores
    - *ScaleInterpolator (Alpha a, TransformGroup target)*
      - Cria um *scale interpolator* trivial que varia o *TransformGroup* especificado entre dois valores *alpha*, usando o objeto *Alpha* especificado, a matriz identidade, um valor de escala mínimo (0.1) e um valor de escala máximo (1.0)
    - *ScaleInterpolator (Alpha a, TransformGroup target, Transform3D axisOfTransform, float minimumScale, float maximumScale)*
      - Cria um novo *scale interpolator* que varia o componente de escala do nodo de transformação

Realismo, Interação e Animação
Animação

- Outro conjunto de classes animam objetos em resposta a mudanças de visualização
  - *Billboard* e *Level of Detail* (LOD) *behaviors*
    - Mudam de acordo com a posição e orientação de visualização, e não de acordo com o tempo
    - *Billboard behavior*: usado para orientar de forma automática um polígono com textura para fique ortogonal à posição de observação
    - LOD *behavior*: objetos complexos são representados por múltiplos objetos com diferentes níveis de detalhe de acordo com a distância do observador virtual (quanto mais longe, menos detalhes)

Exercício 6

Objetivo: verificar como trabalhar com interação e animação em Java 3D

Referências

[Bicho 2002] A. L. Bicho, L. G. da Silveira Jr, A. J. A. da Cruz e A. B. Raposo. **Programação Gráfica 3D com OpenGL, Open Inventor e Java 3D**. REIC - Revista Eletrônica de Iniciação Científica. v. II, n. I, março, 2002. http://www.sbc.org.br/reic/edicoes/2002e1/tutoriais/ProgramacaoGrafica3DcomOpenGLOpenInventoreJava3D.pdf

[Barrilleaux 2001] J. Barrilleaux. **3D User Interfaces with Java 3D**. Manning Publications Co. 2001. 499 p.

[Brown 1999] K. Brown. **Ready-to-run Java 3D**. New York, NY: John Wiley & Sons, 1999. 400 p.

[Lathrop 1997] O. Lathrop. **The Way Computer Graphics Works**. Wiley Computer Publishing, 1997.

Referências

[Sowizral 1998] H. Sowizral, K. Rushforth, M. Deering. **The Java™ 3D API Specification**. 2nd Edition. Addison-Wesley. 1998. 482 p.

[Sun 2003] Sun Microsystems Java 3D Engineering Team. **Java 3D API Tutorial**. Disponível em http://developer.java.sun.com/developer/onlineTraining/java3d/. Acesso: setembro/2003.

[Tori 2002] R. Tori, R. Nakamura. **Desenvolvimento de Jogos para Aprendizagem de Java 3D: Um Estudo de Caso**. Workshop de Jogos, 2002.

[Walsh 2002] A. E. Walsh. **Java 3D: API Jump-start**. Upper Saddle River, NJ: Prentice Hall, 2002. 245 p.