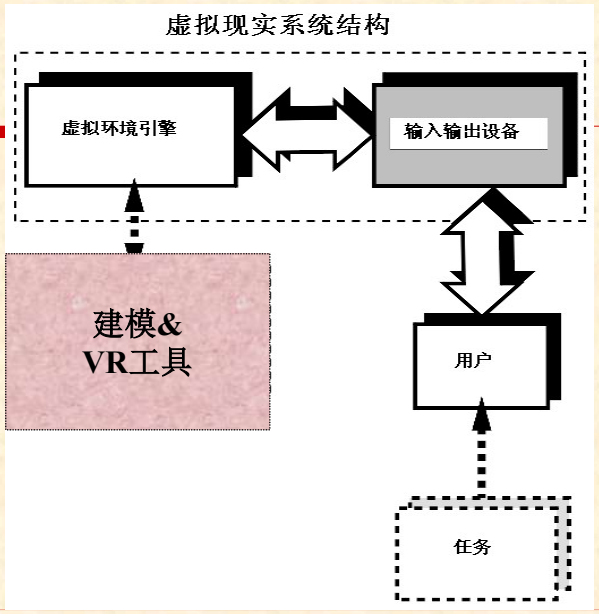
虚拟现实技术

第 10 章 虚拟现实的编程技术

本章主要内容

- VR 软件工具包
- 场景图实例
- VR建模步骤



虚拟现实编程工具包特点

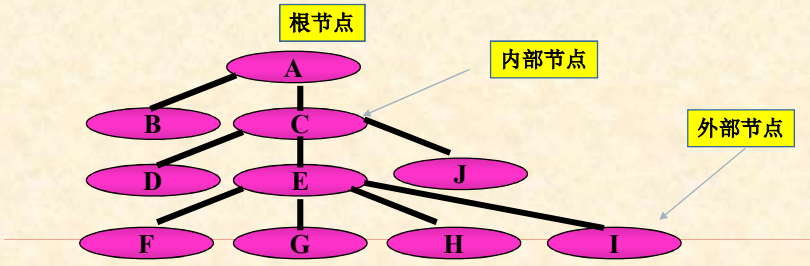
- 面向对象的建模软件包，有利于用户开发
- 支持各种VR中常见的I/O设备，开发者不用写驱动程序
- 允许输入CAD 模型 ，节省时间
- 具有内置网络功能，用于多用户交互

本章讨论的VR 软件工具包

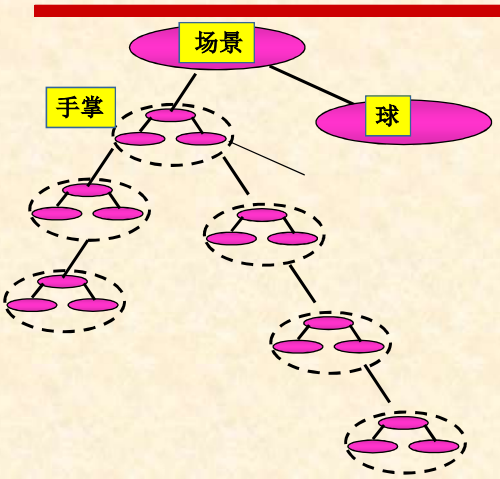
名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包, 275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

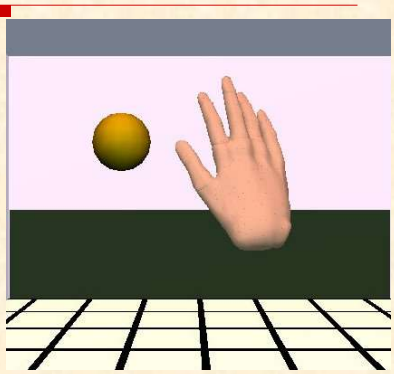
场景图实例

- 是在虚拟世界中物体的一个分层组织
- 场景图用树状结构表示，节点用分枝连接
- 可见物体用外部结点表示，称为叶子
- 内部节点表示转换



场景图不断变化

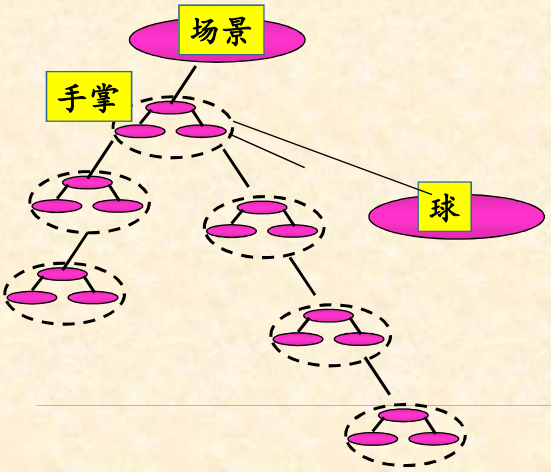


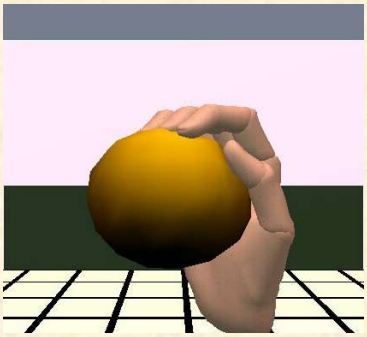


球是场景的子结点

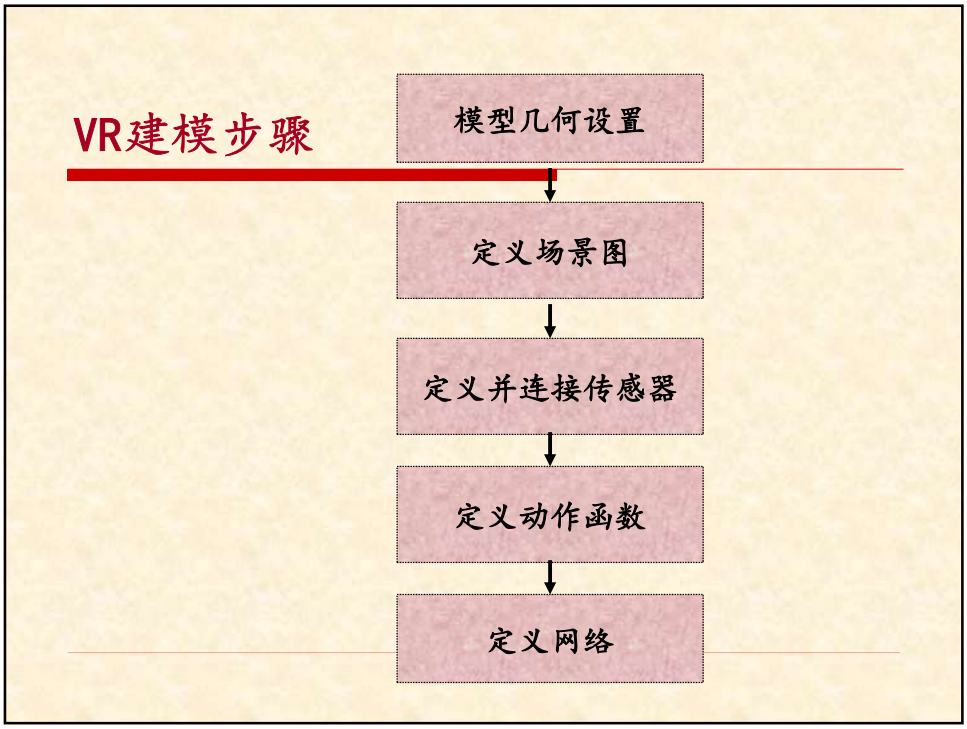
场景图不断变化(续)

- 场景已经被修改，球成为手掌的孩子

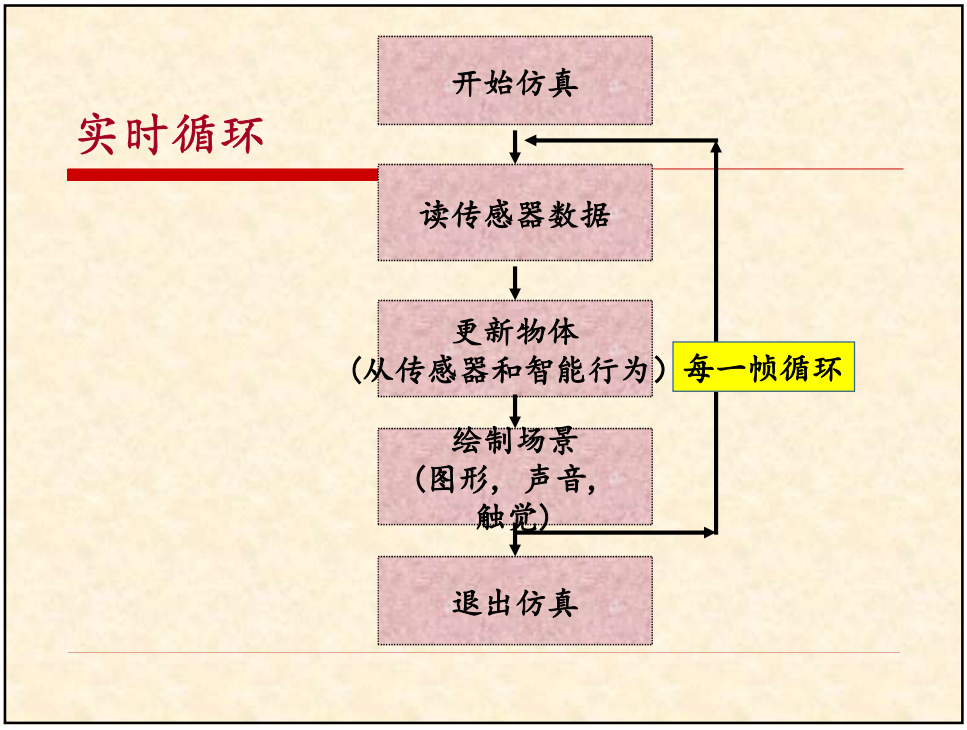




VR建模步骤


模型几何设置
定义场景图
定义并连接传感器
定义动作函数
定义网络

实时循环

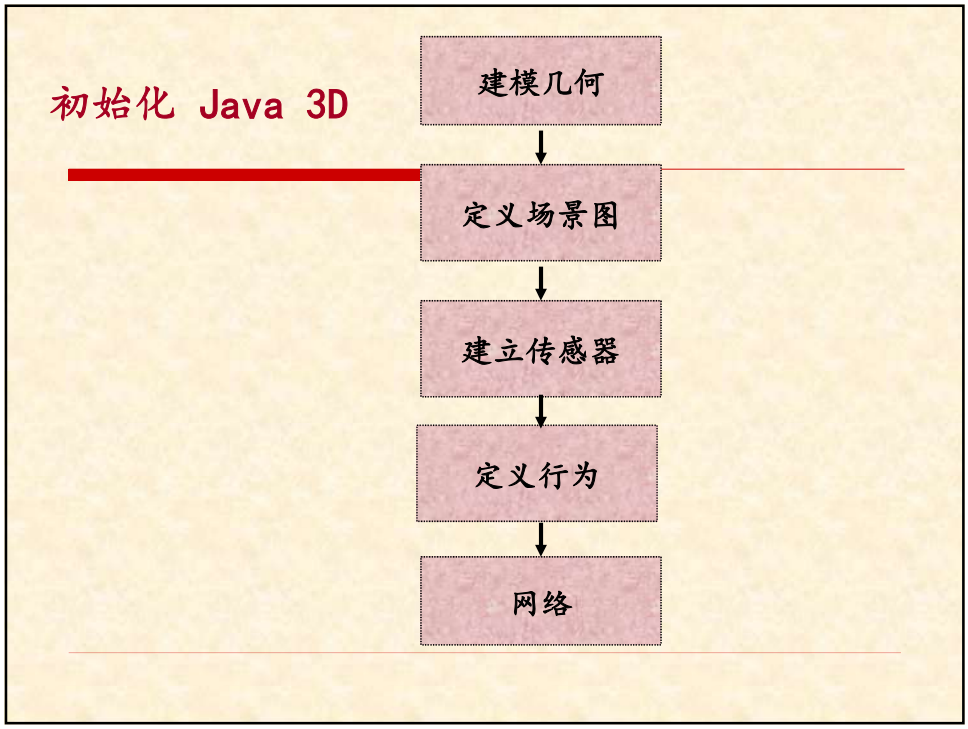

开始仿真
读传感器数据
更新物体
(从传感器和智能行为)
每一帧循环
绘制场景
(图形，声音，
触觉)
退出仿真

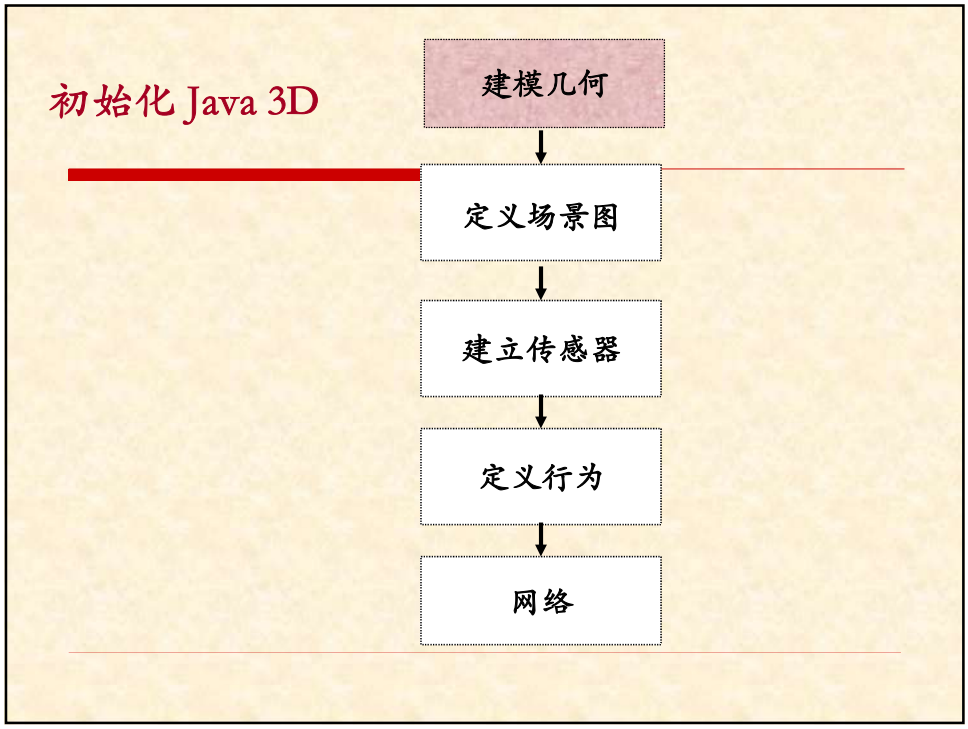
本章讨论的VR 软件工具包

名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包，275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

Java 及 Java 3D

- *Java*
  - 面向对象的程序设计语言
  - 用于网络应用开发
  - 平台独立
  - 比 C/C++慢
- *Java 3D*
  - Java 类的层次，作为3D图形绘制和声音绘制系统的接口
  - 完美地集成了Java
  - 强大的面向对象的体系结构
  - 有利的3D 图形应用程接口（ API）

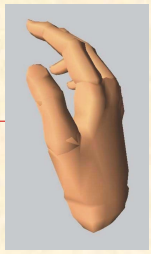

初始化 Java 3D
建模几何
定义场景图
建立传感器
定义行为
网络


初始化 Java 3D
建模几何
定义场景图
建立传感器
定义行为
网络

Java 3D 几何

- 几何可以从各种文件格式导入 (如. 3DS, DXF, LWS, NFF, OBJ, VRT, VTK, WRL)

```
导入几何
loader.load(“Hand.wrl")
```

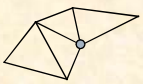


- 能够作为图元产生 (例如. sphere, cone, cylinder, …)

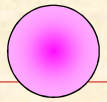
```
几何图元:
new Sphere(radius)
```

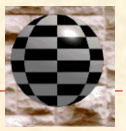
- 用特定的类指定顶点、边、法向量、纹理坐标产生定制几何

```
定制几何:
new GeometryArray(…)
new LineArray(…)
new QuadArray(…)
new TriangleArray(…)
```



Java 3D 物体外观

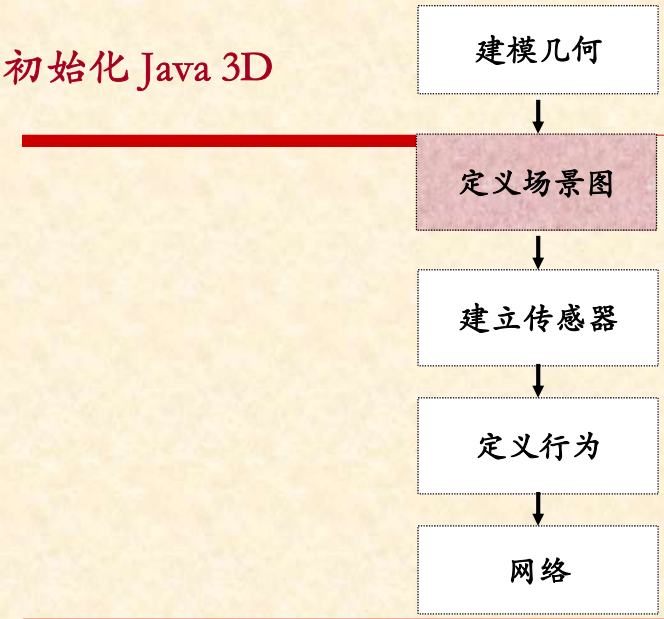




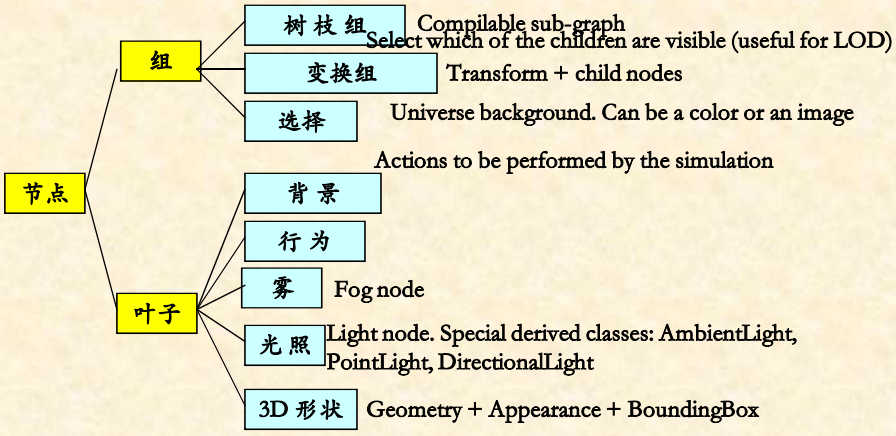
- 几何的外观用外观对象给定
- 一个外观类对象存储有关材料（漫反射、镜面反射 …. diffuse, specular, shininess, opacity ）和纹理

```
Mat = new Material();
Mat.setDiffuseColor(r, g, b);
Mat.setAmbientColor(r, g, b);
Mat.setSpecularColor(r, g, b);

TexLd = new
TextureLoader(“checkered.jpg”, ...);
Tex = TexLd.getTexture();

Appr = new Appearance();
Appr.setMaterial(Mat);
Appr.setTexture(Text);

Geom.setAppearance(Appr)
```

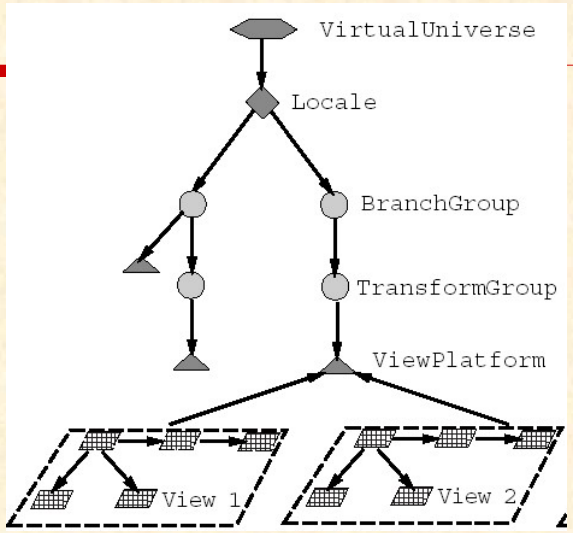
初始化 Java 3D



Java3D 节点类型



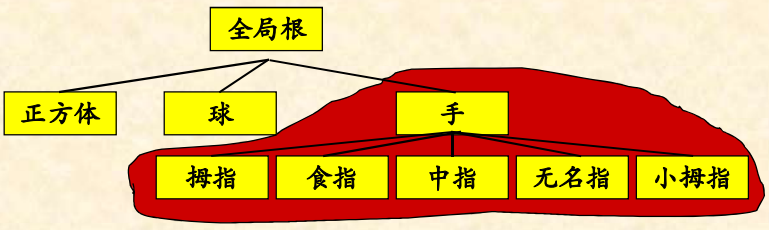
Java3D 场景图



节点

从文件导入物体

- Java3D 默认支持Lightwave 和 Wavefront 模型文件
- 载入器把读入文件的内容加到场景中，作为一个独立的物体.



Java3D 模型导入

```
Scene Sc = loader.load(“Hand.wrl”);
BranchGroup Bg = Sc.getSceneGroup();
RootNode.addChild(Bg);
```

把模型加入场景的图中

```
Scene Sc = loader.load(“Hand.wrl”);
BranchGroup Bg = Sc.getSceneGroup();
Thumb = Bg.getChild(0);
Index = Bg.getChild(1);
Middle = Bg.getChild(2);
Ring = Bg.getChild(3);
Small = Bg.getChild(4);
```

访问导入模型的子零件部分

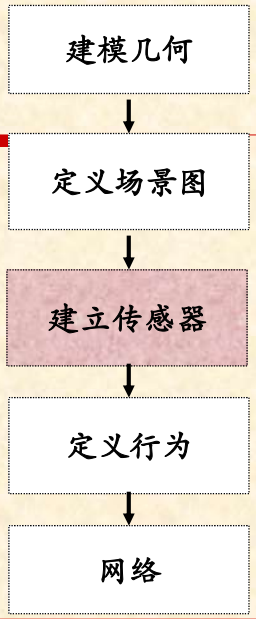
Java3D 虚拟手的导入实例

```
Palm = loader.load("Palm.wrl").getSceneGroup();
ThumbProximal = loader.load("ThumbProximal.wrl").getSceneGroup();
ThumbDistal = loader.load("ThumbDistal.wrl").getSceneGroup();
IndexProximal = loader.load("IndexProximal.wrl").getSceneGroup();
IndexMiddle = loader.load("IndexMiddle.wrl").getSceneGroup();
IndexDistal = loader.load("IndexDistal.wrl").getSceneGroup();
MiddleProximal = loader.load("MiddleProximal.wrl").getSceneGroup();
MiddleMiddle = loader.load("MiddleMiddle.wrl").getSceneGroup();
MiddleDistal = loader.load("MiddleDistal.wrl").getSceneGroup();
RingProximal = loader.load("RingProximal.wrl").getSceneGroup();
RingMiddle = loader.load("RingMiddle.wrl").getSceneGroup();
RingDistal = loader.load("RingDistal.wrl").getSceneGroup();
SmallProximal = loader.load("SmallProximal.wrl").getSceneGroup();
SmallMiddle = loader.load("SmallMiddle.wrl").getSceneGroup();
SmallDistal = loader.load("SmallDistal.wrl").getSceneGroup();
```

Java3D 虚拟手的层次

```
Palm.addchild(ThumbProximal );
ThumbProximal .addchild(ThumbDistal );

Palm.addchild(IndexProximal );
IndexProximal .addchild(IndexMiddle );
IndexMiddle .addchild(IndexDistal );

Palm.addchild(MiddleProximal );
MiddleProximal .addchild(MiddleMiddle );
MiddleMiddle .addchild(MiddleDistal );

Palm.addchild(RingProximal );
RingProximal .addchild(RingMiddle );
RingMiddle .addchild(RingDistal );

Palm.addchild(SmallProximal );
SmallProximal .addchild(SmallMiddle );
SmallMiddle .addchild(SmallDistal );
```

初始化 Java 3D

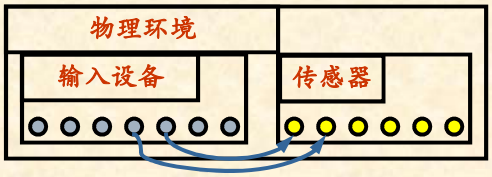


Java3D 中的输入设备

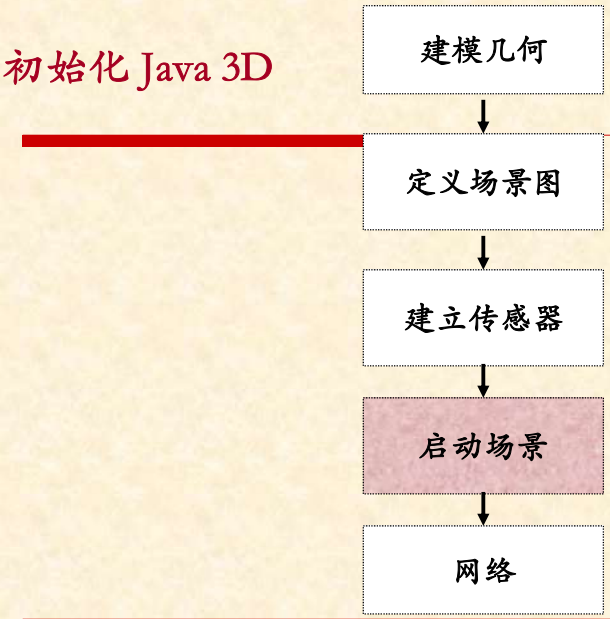
- Java3D 中默认的输入设备是鼠标和键盘
- 目前VR应用中集成其它的输入设备(位置传感器，跟踪球，游戏杆等)完全由开发者的需要而决定
- 通常驱动程用C/C++写， 使用人员可以使用Java重写驱动程序，也可以使用JNI (Java Native Interface)来调用 C/C++ 版本的驱动程
- Java3D 提供很好的通用输入设备接口，可以用于集成传感器.

Java3D 通用传感器接口

- class PhysicalEnvironment - 存储全部输入设备和仿真中的传感器的信息
- class InputDevice – 输入设备驱动程序接口



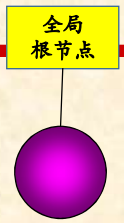
初始化 Java 3D



Java3D – 启动仿真

- Java3D 提供控制仿真的Behavior对象
- 一个Behavior对象包含一些列仿真中接受的动作
- 激活由WakeupCondition对象发出
- 一些激活的类:
  - WakeupOnCollisionEntry
  - WakeupOnCollisionExit
  - WakeupOnCollisionMovement
  - WakeupOnElapsedFrames
  - WakeupOnElapsedTime
  - WakeupOnSensorEntry
  - WakeupOnSensorExit
  - WakeupOnViewPlatformEntry
  - WakeupOnViewPlatformExit

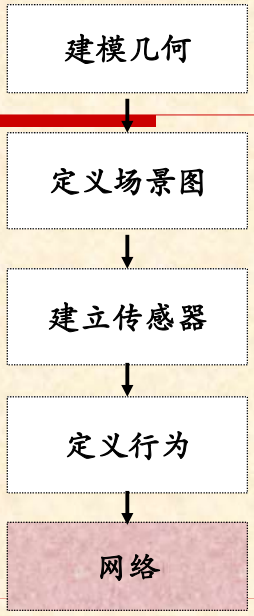
Java3D – 行为（Behavior） 的使用



- 定义一个行为 Bhv，它按照1度旋转球；
- 每一帧都被调用，以便球能够转动

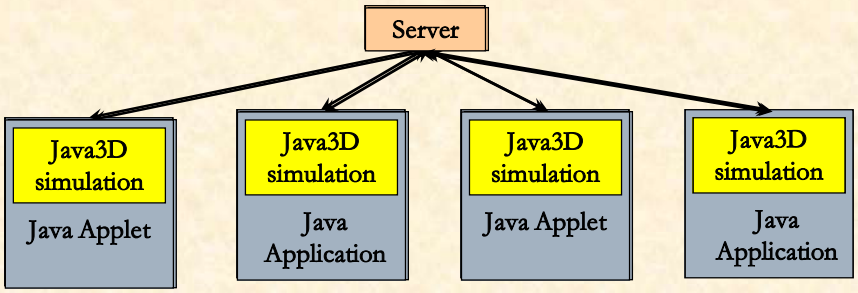
```
WakeupOnElapsedFrames Wup = new WakeupOnElapsedFrames(0);
Bhv.wakeupOn(Wup);
```

初始化 Java 3D



Java3D – 网络

- Java3D 不提供内置的网络虚拟环境的解决方案
- 允许开发者集成使用Java 提供的强大网络功能
- Java3D应用程序能够像标准的应用程序一样运行。

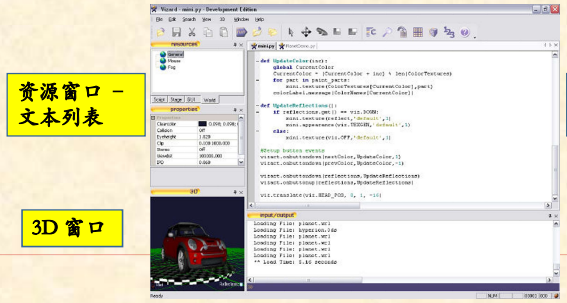


本章讨论的VR 软件工具包

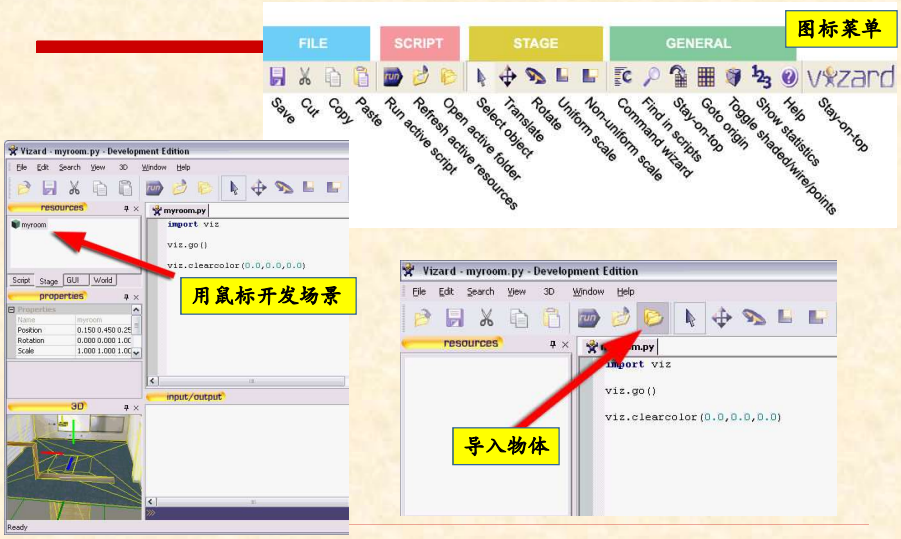
名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包, 275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

Vizard 特征

- ✓ 使用Python，它是可升级和跨平台的；
- ✓ 它是面向对象的，很容易集成C/C++
- ✓ 运行于Unix, Windows, Mac 和其它平台;
- ✓ 使用一个4-window 界面，允许变成人员写并执行代码，查看3D models, 拖放对象, 发出命令同时绘制场景.



工具的使用



Vizard 虚拟手

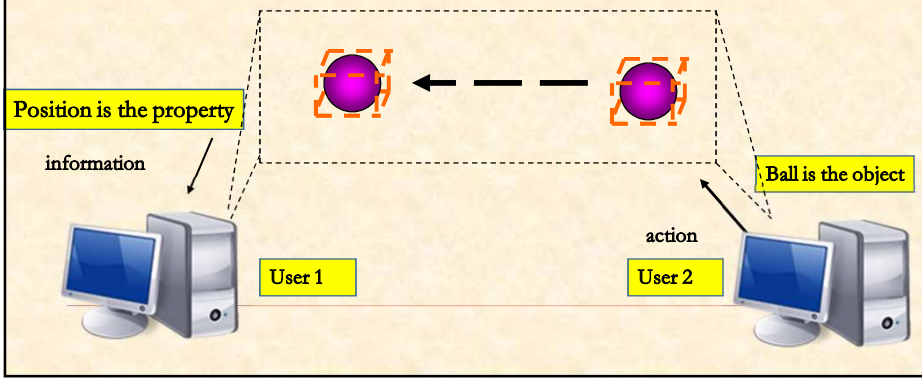
```
import viz
import hand

viz.go()
#Identify the 5DT glove's port.
PORT_5DT_USB = 0

#Add the 5DT sensor
sensor = viz.add('5dt.dls')

#Create a hand object from the data glove
glove = hand.add(sensor,hand.GLOVE_5DT)

#Place the hand in front of the user
glove.translate(0,1,5)
glove.rotate(180,-90,180)
# now when you run the script the glove should be moving
```

Vizard 多纹理

```
import viz
viz.go()

logo = viz.add('logo.wrl') #add vizard logo and place it in front of user
logo.translate(0,2,4)

tex1 = viz.add('gb_noise.jpg') #add two textures that will then be
applied to the logo
#tex2 = viz.add('brick.jpg')
logo.texture(tex1) #applies the first texture
logo.texture(tex2,'',1) #applies the second texture to the logo
blend = viz.add('multitexblend.fp') #indicate how to blend the two
textures
logo.apply(blend)
```

Vizard 仿真服务器

- 多用户存在同一个环境中，每个用户需要运行 Vizard. 当虚拟世界建立后, 每个用户必须建立两个 “mail boxes”.
- 一个接收来自其它用户的信息。



Vizard 网络举例

```
Import viz
Viz.go()
Ball=viz.add('ball.wrl') #create a Ball object that is controlled by the other user
  #add the world that will be displayed on your computer
  #Use a prompt to ask the other user the network name of his computer.
target_machine = viz.input('Enter the name of the other machine'). upper()

#Add a mailbox from which to send messages to the other user. This is your
outbox.
target_mailbox = viz.add(viz.NETWORK, target_machine)
#Add an id for the timer.
BROADCAST = 1

#Add the timer.
def mytimer(num):
  if num == BROADCAST:
    #Retrieve your current position.
       position = viz.get(viz.HEAD_POS)
    #Send the data to the target mailbox. All the recipient will get  your yaw, x and z
coordinates.
    target_mailbox.send(position[0], position[1], position[2])
```

Vizard网络举例

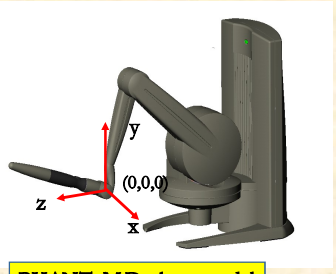
```
#This function will deal with incoming messages.
def mynetwork(message):
  #message[0] is who sent the message, message[1] is a description of what
he
  #sent and message[2] and greater are the messages themselves.
  x = message[2]
  y = message[3]
  z = message[4]
  ball.translate(x,y,z)

# Callback the network function to await incoming messages.
viz.callback(viz.NETWORK_EVENT, mynetwork)
# Callback the timer.
viz.callback(viz.TIMER_EVENT, mytimer)
# Start the timer.
viz.starttimer(BROADCAST, 0.01, -1)
```

本章讨论的VR 软件工具包

名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包, 275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

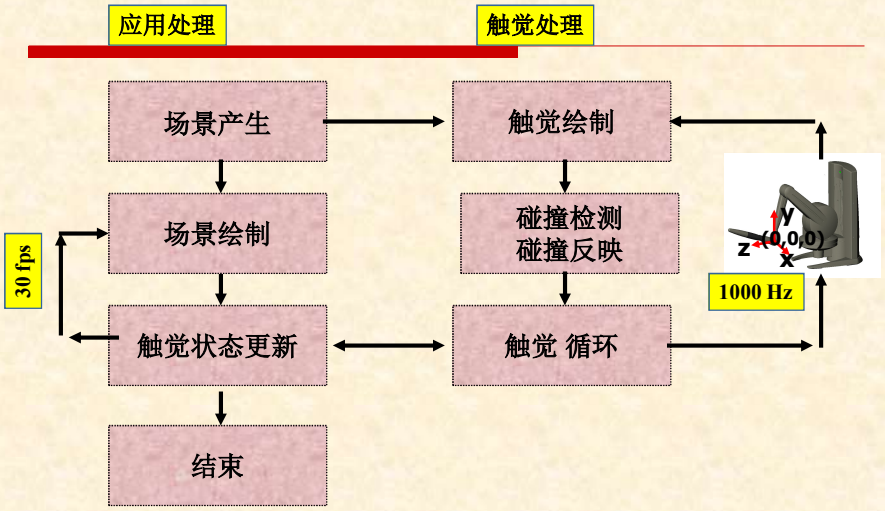
PHANToM 的GHOST 软件包

- 提供真实感的触觉交互；
- 提供并使触觉交互更直观
- 提供触觉交互的场景图，利用3D图形APIs
- 提供触觉交互技术的扩展环境；
- 与PHANTOM点触觉交互；
- 产生相应的运动

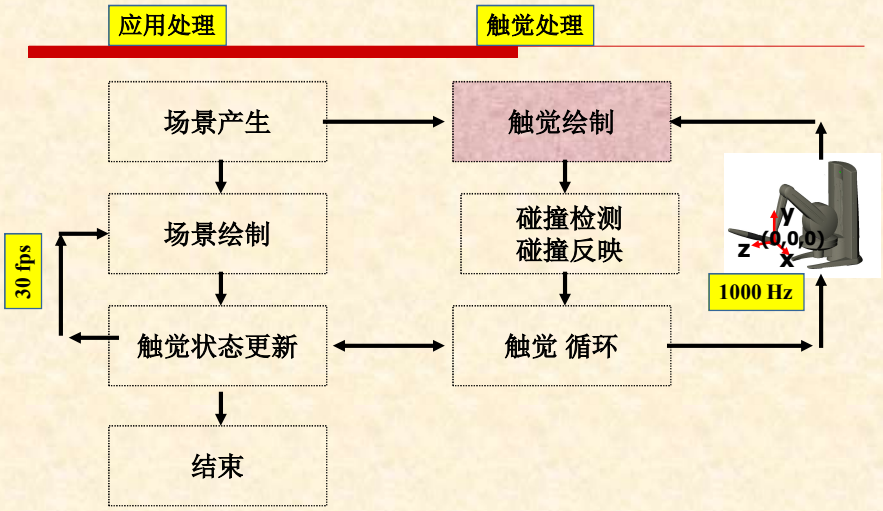


PHANToM Desktop model

GHOST – 应用交互



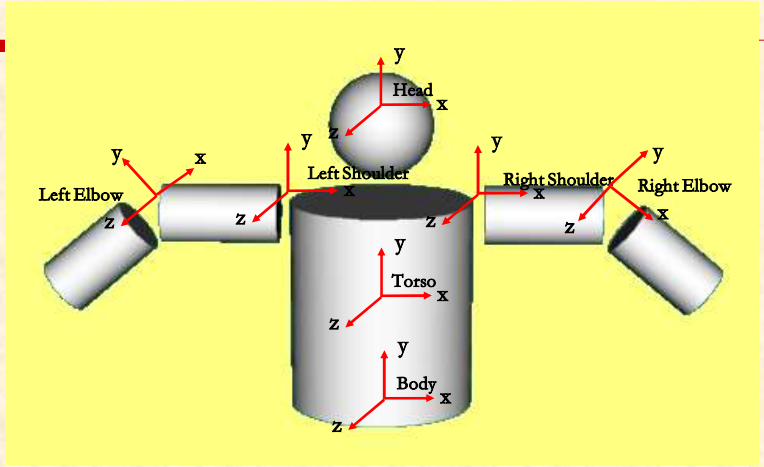
GHOST 触觉场景图



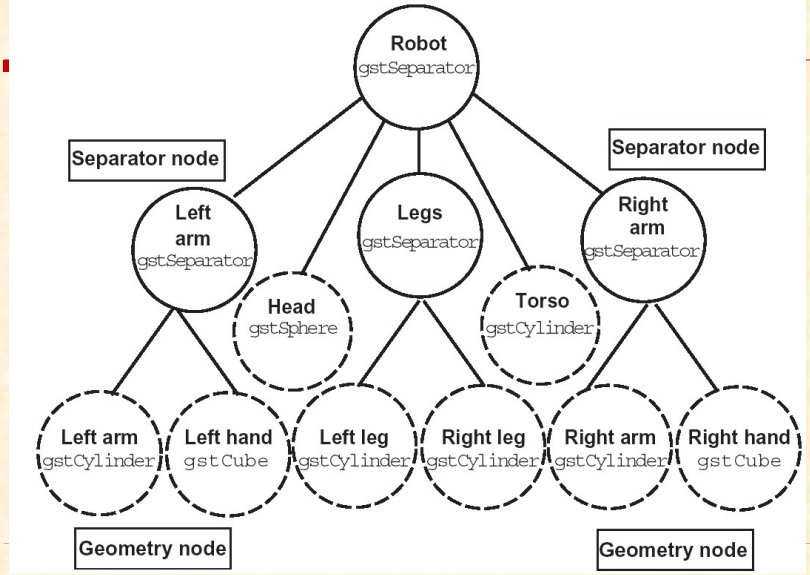
触觉场景图

- 提供一种构造触觉场景的结构化方法，包括几何和运动
- 自顶向下遍历
- 每个节点只被遍历一次（每个子节点只有一个双亲节点）

场景图举例


Head
Left Shoulder
Right Shoulder
Left Elbow
Right Elbow
Torso
Body

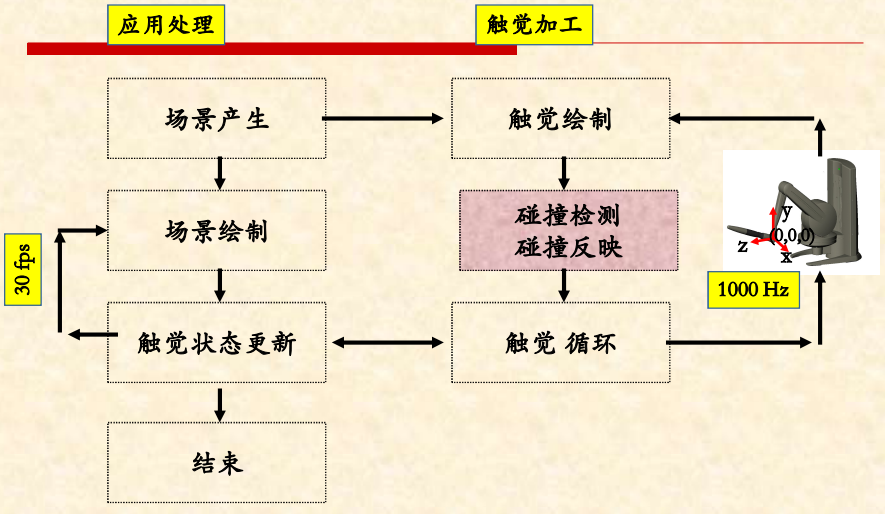
静态场景图


Robot
gstSeparator
Separator node
Separator node
Left arm
gstSeparator
Legs
gstSeparator
Right arm
gstSeparator
Head
gstSphere
Torso
gstCylinder
Left arm
gstCylinder
Left hand
gstCube
Left leg
gstCylinder
Right leg
gstCylinder
Right arm
gstCylinder
Right hand
gstCube
Geometry node
Geometry node

GHOST 代码举例

```
#include <stdlib.h>
#include <gstBasic.h>
#include <gstSphere.h>
#include <gstPHANToM.h>
#include <gstSeparator.h>
#include <gstScene.h>

Main()
gstScene *scene = new gstScene;
gstSeparator *root = new gstSeparator;
gstSphere *sphere = new gstSphere;

Sptere -> setRadius(20);
gstPHANToM *phantom = new gstPHANToM (``PHANToM name``);
Root -> addChild(phantom);
Root-> addChild(sphere);
Scene-> setRoot(root);
Scene -> startServoLoop();
While(!scene -> getDoneServoLoop())
                    // end application by calling scene -> stopServoLoop ();
```

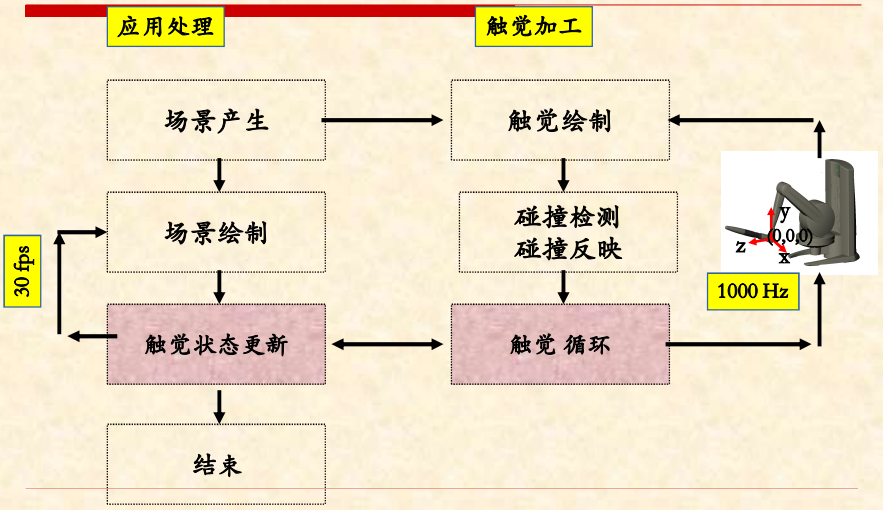
力计算和动力学



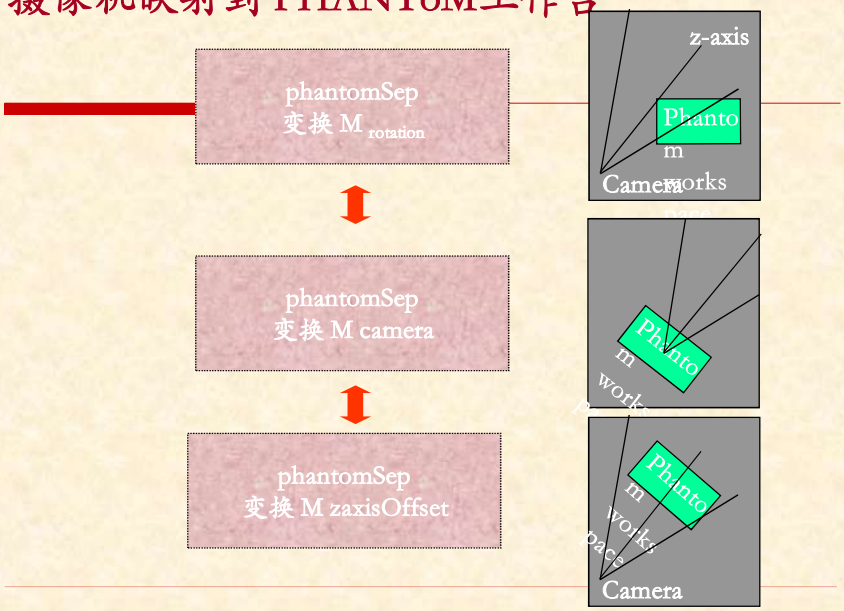
碰撞检测与反应

- 场景图包含至少一个通过*gstPHANToM*表示的触觉接口的节点（一个触觉场景中有四个触觉接口）
- 碰撞检测用*gstShape*节点进行
- 当碰撞检测存在时，*gstPHANToM_SCP*被加入到场景图中，这个节点应该加入到相同双亲的下面，作为*gstPHANToM*节点。

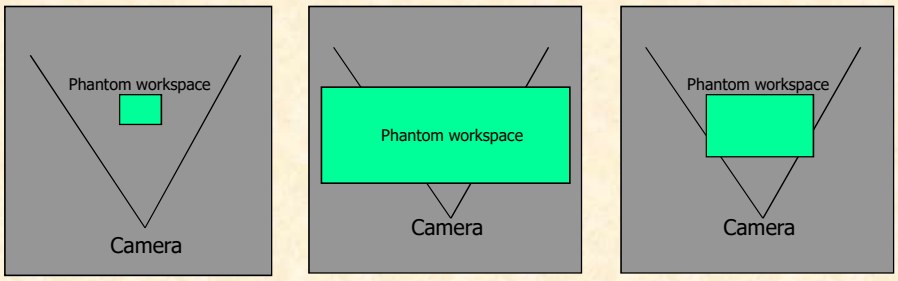
更新应用



摄像机映射到 PHANToM工作台



缩放摄像机和 PHANToM 工作空间



摄像机工作空间太大

摄像机工作空间太小

摄像机工作空间适当

本章讨论的VR 软件工具包

名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包, 275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

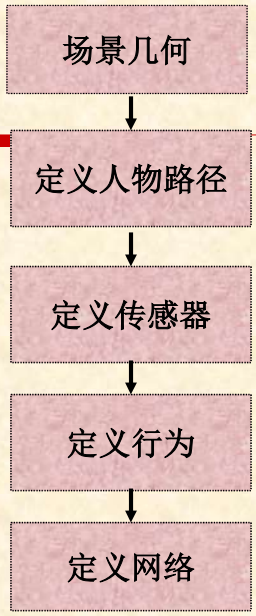
BDI PeopleShop 特点

- 提供了一种在实时场景中模拟人物的使用方法
- 是一种任务层次的变成环境，结合基于菜单的图形用户接口（ GUI）
- 非常适合分布式交互仿真（ DIS ）。因为低带宽需求和预测;
- 起初为军事而设计，现在用于民用，例如，事件回放、建筑漫游、仿真驱动、警察训练。

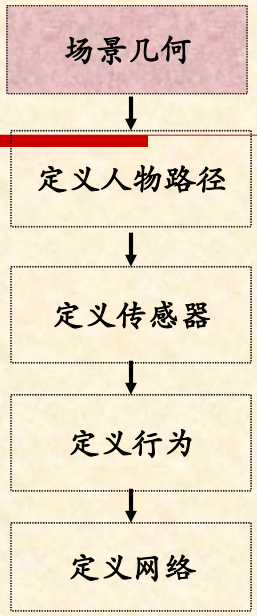
BDI PeopleShop特点

- 连接对象库，可以运行于 SGI，Intel PCs，和其它平台
- 库具有实时动作引擎模块、图形显示模块、运动数据、3D图形模型、纹理和用于DIS的网络接口
- 运行于OpenGL，Direct3D，Sense8 WTK，Mak Stealth和其它软件包
- 建议的硬件是 Intel > 200MHz，64 MB RAM，和图形加速器、（Open GL，OpenGVS or Direct3D).

PeopleShop初始化



PeopleShop
初始化



PeopleShop 人物

- 具有54自由度（DOF），11 连接点（links）的关节多边形结构；
- 车也作为人物；
- 不同人物具有不同的动作；
- 每一种类型人物具有用户选择的外观（例如，人物的车可以是坦克或警车）
- 人物使用纹理增加真实感，并减少多边形的数量。

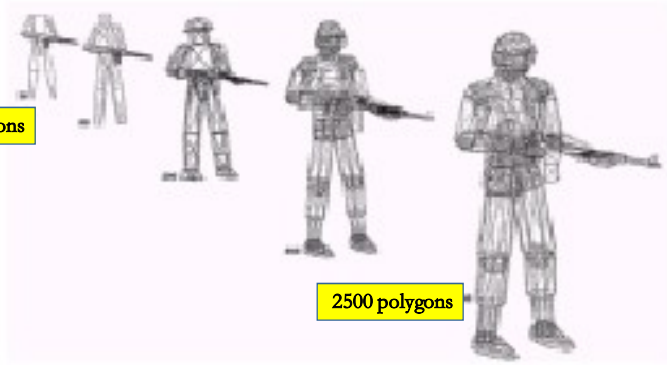
人物选择

人物类型决定可以接受的动作 (可以选择的菜单)

外观选择

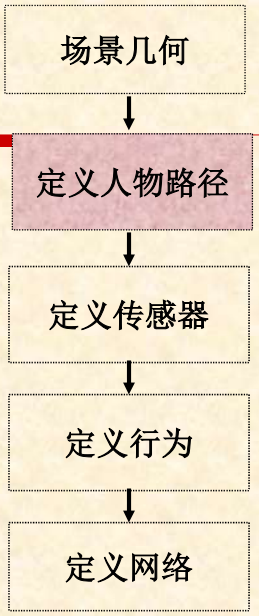
人物外观 (可选的菜单)

BDI 软件包

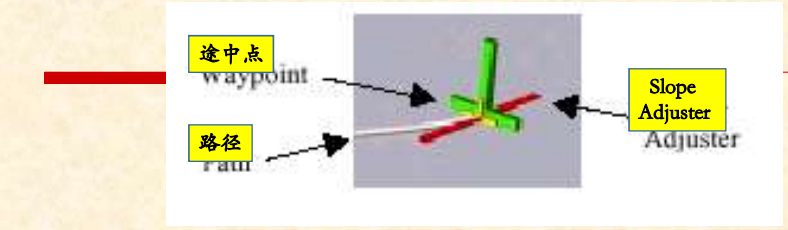


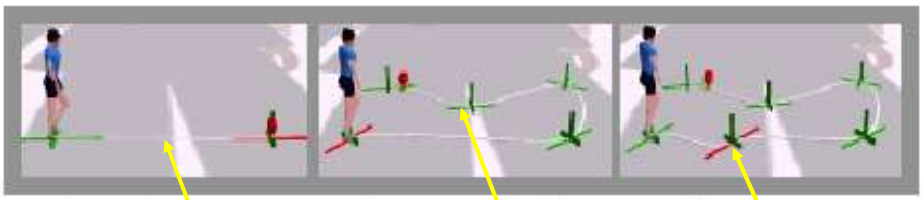
根据距离虚拟相机的距离决定人物LOD
一个场景最多100个人

PeopleShop 初始化

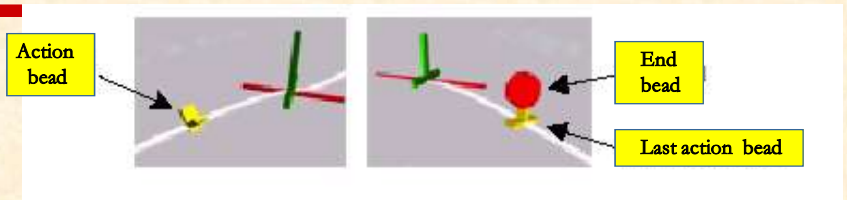


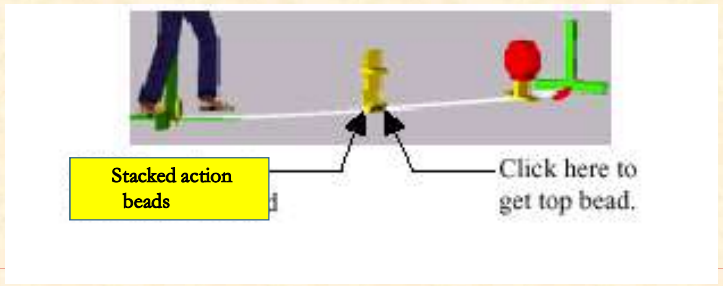
PeopleShop 指定路径



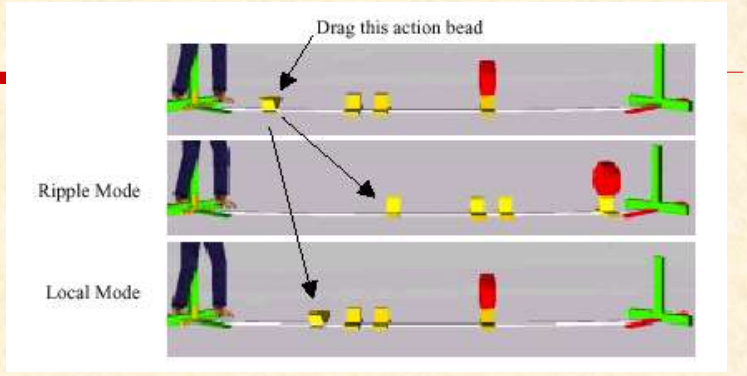


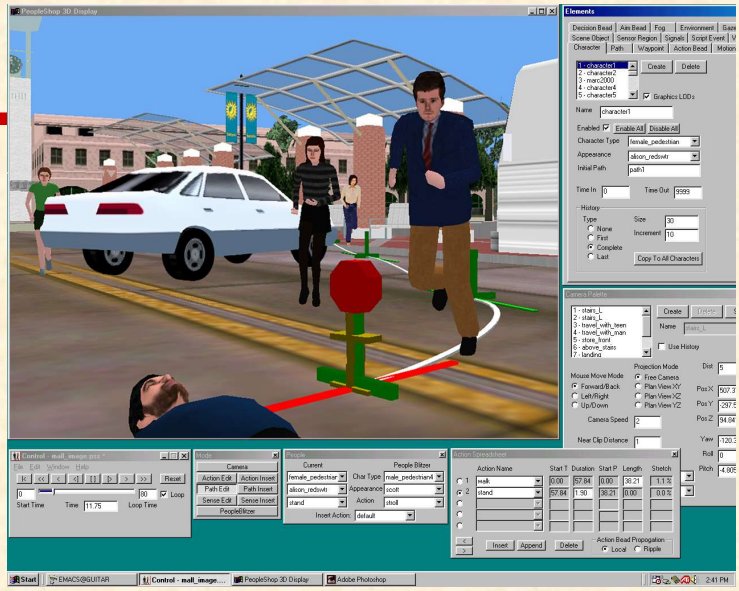
BDI 软件包





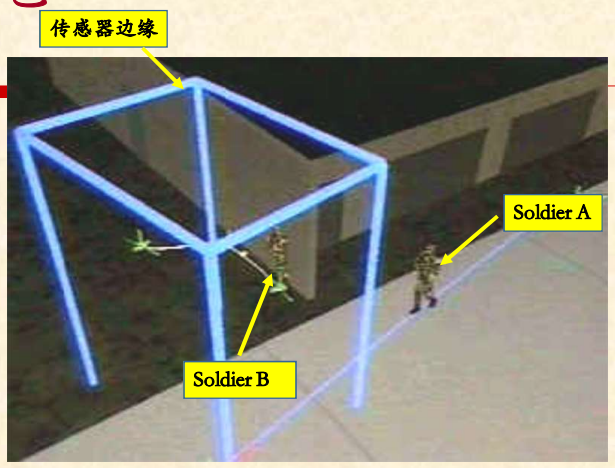
编辑行为





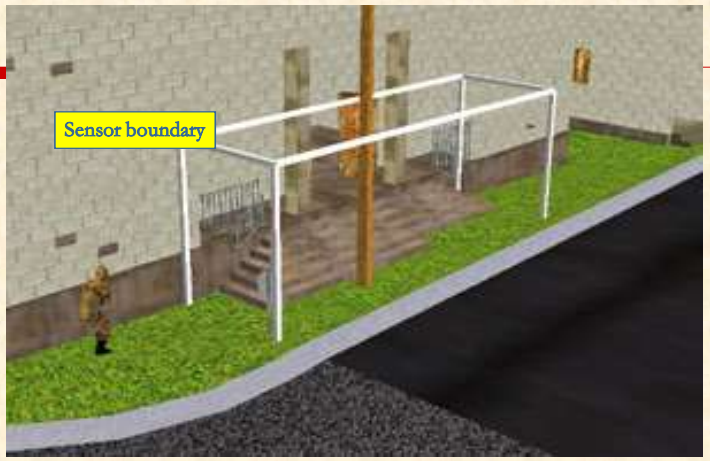
PeopleShop
初始化

BDI 软件包



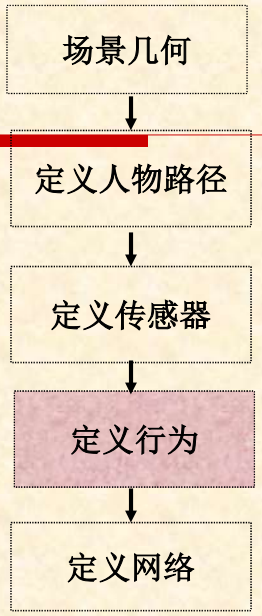
当士兵进入到传感器区域时，系统会发现察觉-这会触发士兵B向A开枪

PeopleShop 传感器



传感器是用户定义的对象，当人进入时被发觉

PeopleShop 初始化

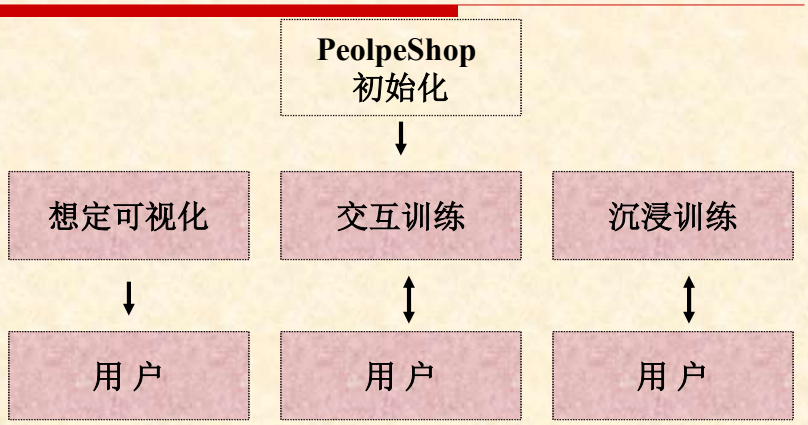


BDI 软件包

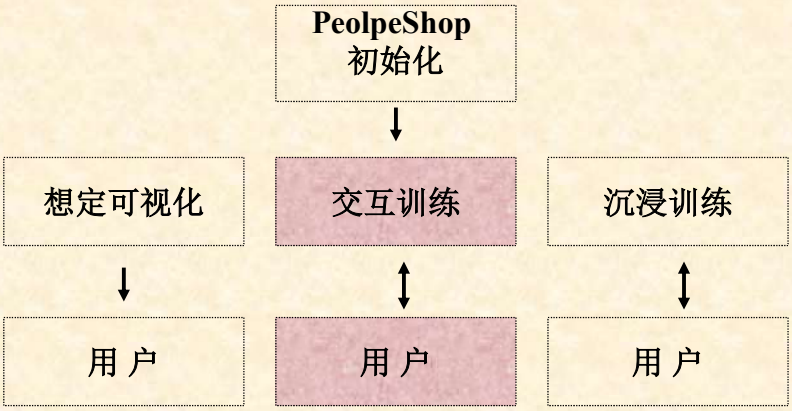
决策语句 (IF/THEN/ELSE)

If	Test to see if event has occurred
sense <character name> inside <region name>	Test to see if a given character has entered a given sensor region
signal <event>	Test to see if a given event has occurred
True	Always perform Then action. Useful when testing Then statement actions
False	Always perform Else action. Useful when testing Else statement actions
Then	Action to take if result of test is TRUE. More than one action can be specified, separating actions with comma.
Else	Action to take if result of test is FALSE. More than one action can be specified, separating actions with comma.
kill <character_name>	Given character dies, whereever he is on his path.
push_path <path_name>	Proceed to specified path
signal <event>	Future tests for specified event will produce a result of TRUE.

PeopleShop 实时性



PeopleShop实时性

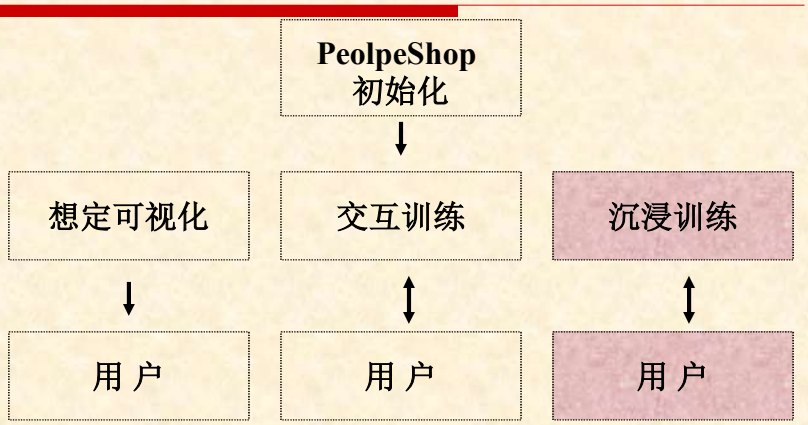


BDI PeopleShop 软件包

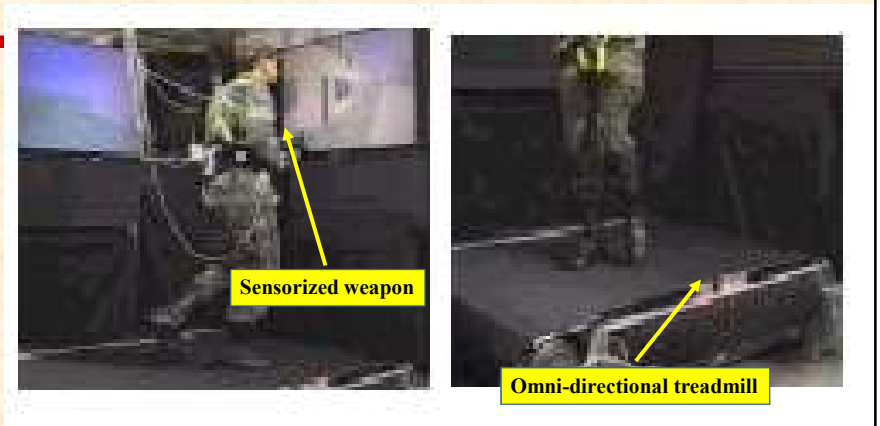
Action	Natural Speed
Crawl	0.38 m/s
Walk	0.67 m/s
Walk low	0.50 m/s
Walk backward	-0.45 m/s
Walk low backward	-0.61 m/s
Jog	1.87 m/s
All others	0.0

用户使用游戏杆或鼠标与仿真实时交互，菜单用于控制

PeopleShop 实时性

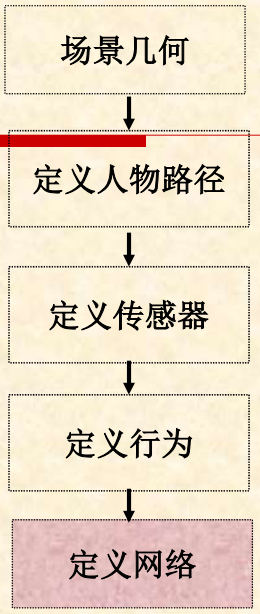


BDI PeopleShop – 实时模式



用户使用跟踪器和传感器与仿真实时交互

PeopleShop
初始化

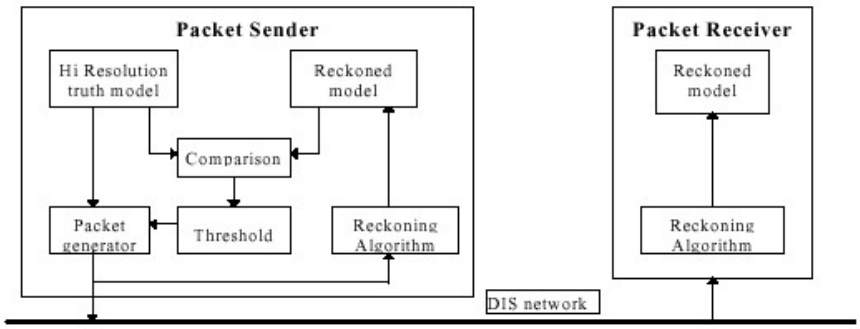


PeopleShop 网络

- 在DIS 环境中修改人物的外形需要更高的带宽；
- 车具有较小的自由度，而具有40个关节的人物以20 Hz 更新就需要800 包/秒；
- 人们仅仅在任务级（动作、位置、速度）上更新，而不更新每个节点。这样产生平滑仿真需要大约2 包/秒；
- 对于参与者数量多的仿真效果很好，例如军事训练；
- 使用实况定位（ live reckoning ）和推算定位（ dead reckoning ）

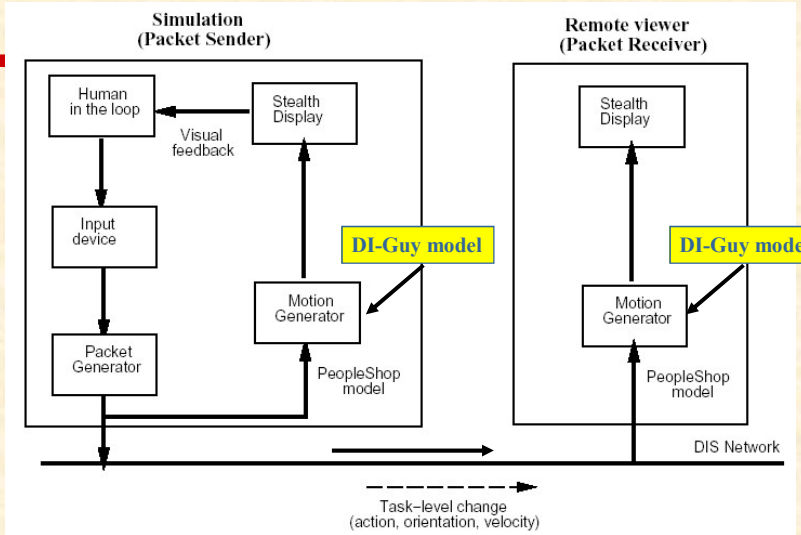
BDI 软件包

用推算预测的DIS



BDI 软件包

使用实况定位的DIS 人在循环中



任务级的改变
(动作, 方向, 速度)

PeopleShop “Top Gun”
(courtesy Boston Dynamics Inc.)

本章讨论的VR 软件工具包

名称	应用	专有	语言库规模
Java3D (Sun Microsystems)	一般应用	no	用C完成 用Java编程 19 工具包, 275 类
Vizard Toolkit and PeoplePak (WorldViz)	一般应用 扩展精灵	yes	OpenGL-based Python scripting language
GHOST (SensAble Technologies)	Phantom触觉	yes	C++
PeopleShop (Boston Dynamics)	军事/民用	yes	C/C++
3DGame Studio	游戏引擎	yes	C++

3D Game Studio 人物

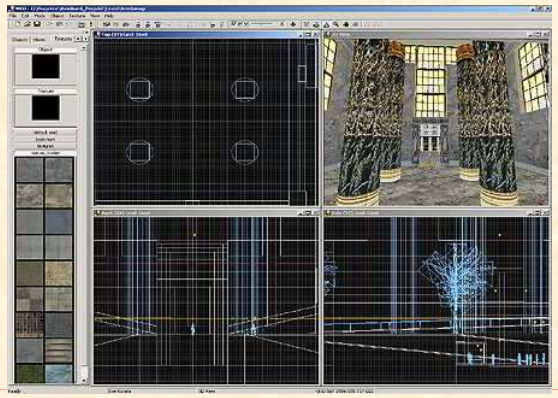
- 使用A6 游戏引擎，提供了简单的快速的编程构造虚拟世界的方法；
- 一种编程环境和基于菜单的图形接口(GUI)的结合，脚本编程更灵活；
- 适合多用户仿真通过局域网（LANs ）和广域网（ WAN ） (TCP/IP 及 UDP)；
- 最初为游戏设计，现在扩展为VR编程，性能好，价格可以接受（$500）

3D Game Studio 人物

- 3D 引擎允许多用户（多个虚拟像机），建立在DirectX 9 图形流水线；
- 具有剪切的优化 ；
- 深入的通过几何LOD和纹理细节的优化；
- 表面可以变形，物体可以是动态的
- 人物动画通过骨骼框架表示。

3D Game Studio 几何的产生

- 可以导入大的模型库，或用另外的建模软件（3D Studio Max）建好的模型
- 物体可以用 World Editor (WED) 3D editor产生

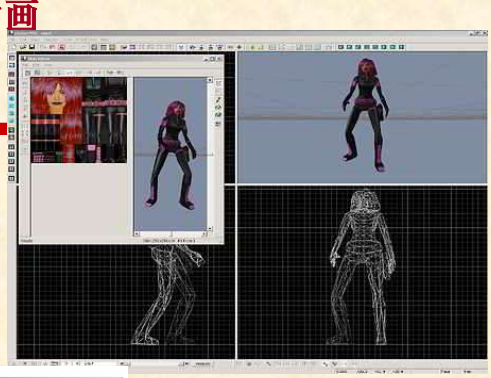


3D Game Studio 纹理

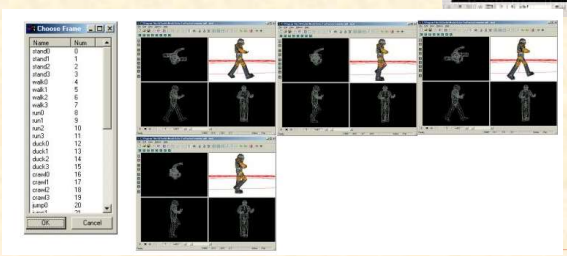
- 凸凹不平纹理

- 光照映射和多纹理

3D Game Studio人物动画



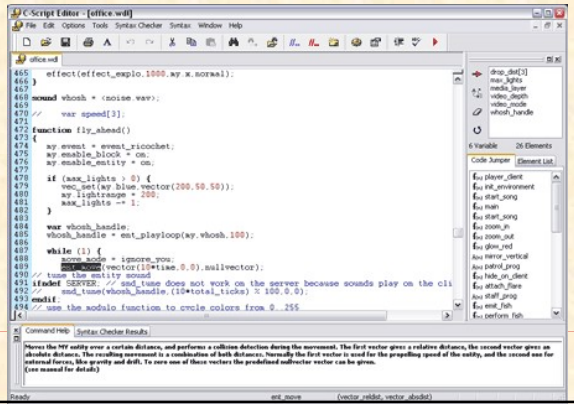
- 动画可以导入由 Maya产生的，或者人物可以用model editor创建



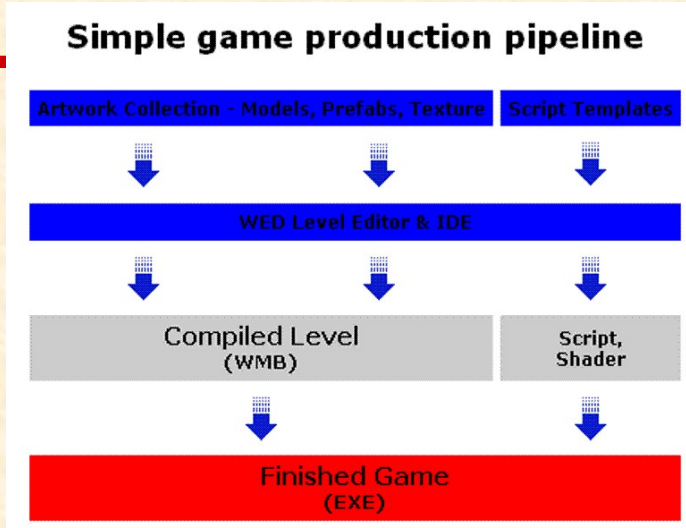
- 按照动作定义的帧

物理与声音引擎

- 具有碰撞检测功能，建立在多边形级别上，及物理行为重力、弹性和摩擦力
- *Cscript* – C++的简单版本，用于产生物理行为，同过函数进行I/O编程 (键盘、鼠标)
- 静态、动态 3D 声音资源，可编程



3D Game Studio 初始化

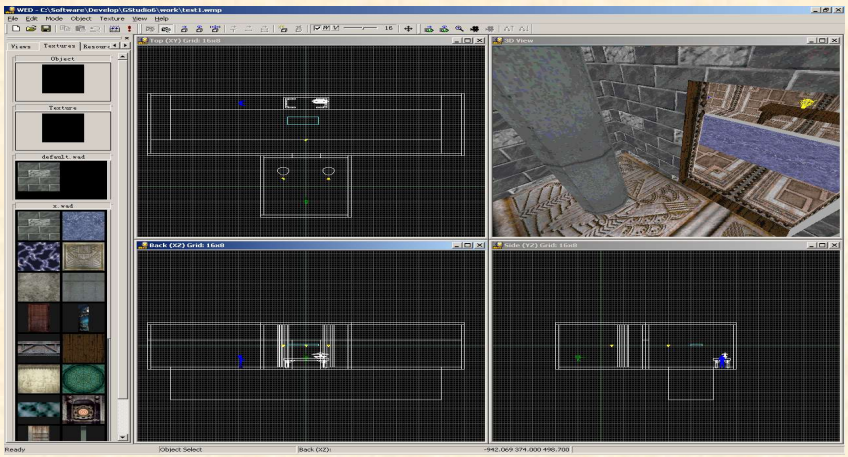


开发系统

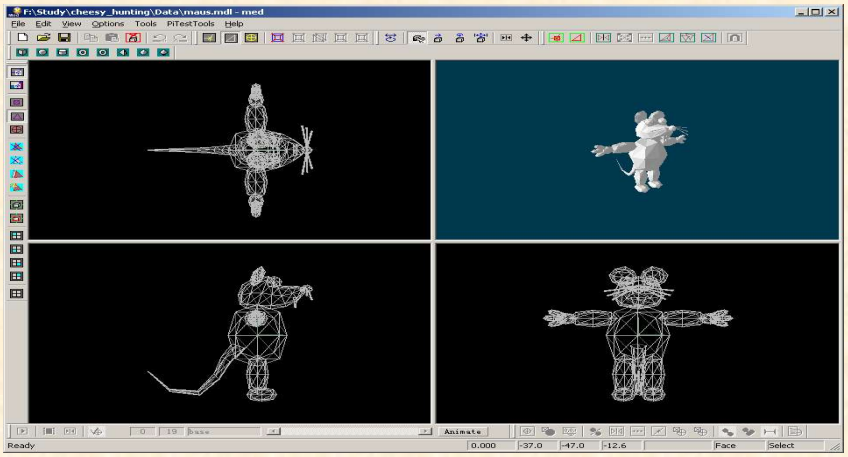
- 3D engine
- 2D engine
- Physics engine
- Level design
- Model and terrain editors
- C-Script programming language
- Huge libraries of 3D objects and artwork

用3D GameStudio开发的游戏

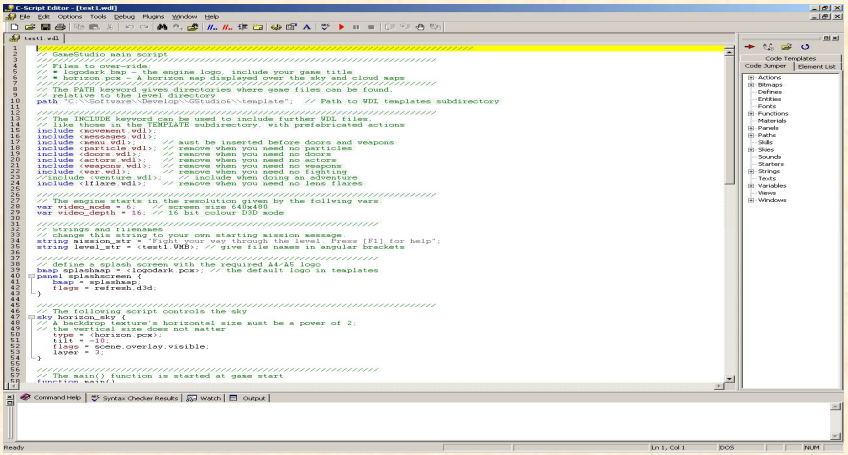
Level Editer - WED



Model Editer - MED



Script Editer - SED



Demo - techdemo.wmp