土木与建筑工程CAE

第六章 虚拟现实技术在CAE中的应用

清华大学土木工程系
胡振中
huzhenzhong@tsinghua.edu.cn

Tsinghua University

第六章 虚拟现实技术在CAE中的应用

6.1 虚拟现实技术

- 1989年，美国VPL Research公司创始人Jaron Lanier提出了“Virtual Reality”（虚拟现实）的概念(VISC-1986)。“Virtual Reality”（虚拟现实）的另一个名称是“Virtual Environment”（虚拟环境）。

- **定义**
  - “Virtual”-虚拟，人工构造，存在于计算机内部。
  - 虚拟现实技术是直接把计算机合成的信息提供给人的感觉器官，生成逼真的视、听、说、触、动和嗅等感觉的虚拟环境，操作者借助必要设备可以自然方式与虚拟环境及物体进行交互，从而产生身临其境的感受和体验。

虚拟现实的3个特性

- **沉浸** (Immersion)

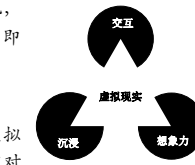
  借助交互设备和自身的感觉、知觉系统，产生置身于相应真实环境中的虚幻感，即身临其境的感觉。

- **交互** (Interaction)

  用户可借助交互设备以自然的方式与虚拟现实系统进行交互，虚拟现实系统可以对用户的动作或命令作出响应。

- **人的想象** (Imagination)

  想象力让人感知到虚拟环境，从中得到理性和感性认识。人的想象力决定了虚拟现实技术的应用。



虚拟现实的3I特性

虚拟现实系统中的虚拟环境

- **模仿真实世界中的环境**
  - 这种真实环境，是已经存在的，如建筑物，武器系统或战场环境。为了逼真地模仿真实世界中的环境，要求逼真地建立几何模型和物理模型，环境的动态应符合物理规律。这一类虚拟现实系统的功能，实际是**系统仿真**。

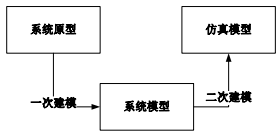
- **人类主观构造的环境**
  - 这种环境是虚构的，几何模型和物理模型就可以完全虚构。例如，用于影视制作或电子游戏的三维动画。系统的**动画技术**。

虚拟现实系统中的虚拟环境

- **模仿真实世界中的人类不可见的环境**
  - 模拟的是真实环境，是客观存在的，但是人类的视觉和听觉不能感觉到。这一类虚拟现实系统的功能，实际是**科学可视化**。
  - 例如，分子的结构，空气中速度、温度、压力的分布等。对于分子结构这类微观环境，进行放大尺度的模仿，使人能看到。对于空气中速度这类不可见的物理量，可以用流线显示速度（流线方向表示速度方向，流线密度表示速度大小）。

系统仿真技术

- **系统仿真的概念**
  - 计算机仿真技术是以数学理论、相似理论、信息技术、系统技术及其应用领域有关的专业技术为基础，以计算机和各种物理效应设备为工具，利用系统模型(数学模型)或部分实物，对具有不确定因素的实际或设想系统进行试验研究的一门综合技术。



系统原型与模型的关系

系统仿真技术

- **计算机仿真系统的分类**
  - **物理仿真**
    - 即全实物仿真，指按照实际系统的物理性质构造系统的物理模型，并在物理模型上进行实验。它倾向于“模拟”，是一般的物理性实验。
  - **数学仿真**
    - 指对实际系统进行抽象，构造数学模型，并在数学模型上进行试验。数学仿真故而又称计算机仿真。

系统仿真技术

- **软件在回路中仿真**
  - 又称半实物仿真，指将实际系统的一部分用数学模型加以描述，转变为仿真模型并在计算机上运行；将系统的另一部分以实物或物理模型引入到仿真回路中。
  - 这主要是因为真实系统某些部分很难或根本无法建立准确的数学模型，使得数学仿真的实现很困难。
- **硬件在回路中仿真**
  - 是指将系统计算机与仿真计算机通过接口连起来，一起参与系统实验。目前控制系统、导航系统、制导系统等高精度系统广泛采用数字计算机，通过专用接口将这些系统和仿真系统连接，可以进行控制、导航、制导的系统仿真。

系统仿真技术

- **人在回路中仿真**
  - 指操作人员在系统中进行操作的仿真实验。
  - 这种仿真将对象实体的动态性质通过数学模型转换为仿真模型在计算机上运行，通过模拟人的感觉的各种物理效应设备，模拟生成人的感觉的物理环境。

虚拟仿真技术

- **虚拟仿真的概念**
  - 将虚拟现实与系统仿真相结合成一种有机整体，利用虚拟现实技术来对系统仿真结果进行后处理就是虚拟仿真，或称虚拟现实仿真。
  - 系统仿真和虚拟现实是目标与表现方法的关系。系统仿真是虚拟仿真技术的基础。虚拟现实技术是虚拟仿真技术的核心。

虚拟仿真技术

- **虚拟仿真的分类**
  - **可视化仿真**

    包括仿真结果可视化(仿真可视化)和仿真计算过程可视化，它是将仿真技术与可视化技术、动画技术相结合，把仿真过程和仿真结果描述得更形象、直观，更富有真实感。
  - **多媒体仿真**

    将图形、图像、声音、感觉等多媒体信息融入仿真模型中，进行仿真技术，并将结果用多媒体形式表示出来的仿真技术。它使人的感官和思维“进入”回路，即“思维在回路”和“感觉在回路”的仿真。

虚拟仿真技术

- **虚拟仿真的分类**
  - **视景仿真**

    采用图形图像技术，根据仿真目的构造仿真对象的三维模型并再现真实环境，达到非常逼真的仿真效果，它使用户产生身临其境的交互式仿真环境，实现用户与该环境直接交互。
  - **虚拟现实仿真**

    即狭义的虚拟仿真，综合了虚拟现实技术和仿真技术的各自优点。

虚拟现实技术的作用

- **用狭小的空间代替广阔的空间**
  - 宇宙飞船控制台；
  - 战斗机驾驶舱.
- **可以体验到那些由于危险、经济代价高或费时等原因而不易到达的地方**
  - 火星、太阳等宇宙旅行；
  - 核反应堆、海底等探险；
  - 遥控手术.
- **虚拟体验因大小关系而无法体验的事情**
  - 在人的血管、体内探险.

虚拟现实技术的作用

- **检验设计的合理性**
  - 建筑物建成后的外观等；
  - 车辆扬弯分析.
- **用于虚拟空间辅助决策系统**
  - 虚拟产品的性能评价和模拟体验；
  - 住宅环境的设计、模拟体验等.
- **用于训练**
  - 模拟喷气机训练培养飞行员；
  - 医生用其进行手术训练.

6.2 虚拟现实技术在CAE中的应用

- **土木与建筑领域**
  - 虚拟设计
    - 在产品和工程CAD中，VR技术的应用，可使设计者的设计思想和意图，迅速变成相应的实物模型或虚拟样品，经修改完善，以达到更高的设计水平。
  - 虚拟建造（制造）
    - 实现产品的设计、工艺规划、加工制造、性能分析、质量检验。
  - 虚拟环境中的灾害模拟和防灾减灾
    - 实现虚拟环境下的灾害模拟、多灾种灾变行为的仿真模拟、灾害救援模拟和决策支持等。

虚拟设计

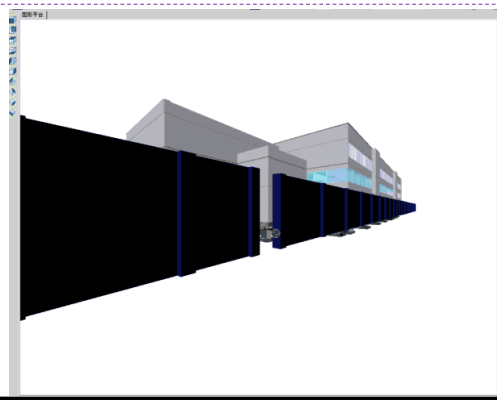
- **增强可视化的CAD系统：** 利用现有的CAD系统进行建模，对数据格式进行适当转换后**输入虚拟环境系统**，在“真实”的环境中对模型进行交互与分析。这种系统可集成现有的CAD系统，开发强度低，成本少，是目前主要的虚拟设计环境的开发形式。
- **虚拟现实的CAD系统：** 完全将CAD系统建立在虚拟环境上，使设计师**通过各种交互设备与虚拟环境交互，直接进行三维设计**。这种系统在设计阶段就可以逼真的看到产品的外观，可以提高设计的效率和质量，但开发及硬件成本较高，目前的应用较少。

虚拟设计

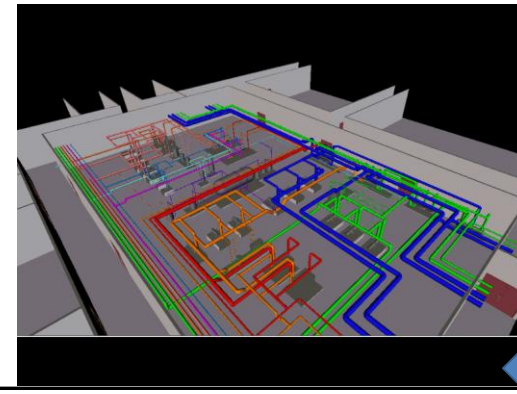
- **三维建筑场景模拟**
  - 帮助房地产开发商通过三维场景模拟，展示房地产项目的设计规划方案，以满足各参与方的要求。
  - 通过对建筑场景模型进行虚拟漫游，使投资方、预期客户和审批部门可以看到项目完成后在现实环境中的整体效果和周边环境。包括建筑、道路、景观等。
  - 三维建筑场景模拟可以帮助开发商减少项目的风险，通过CD、DVD和网络的方式发布项目的三维场景模拟，提高项目的可信性，促进资金筹集、审批和销售。

RAPIDsite Visualization for this proposed SunCrest Planned community.

虚拟设计



虚拟设计

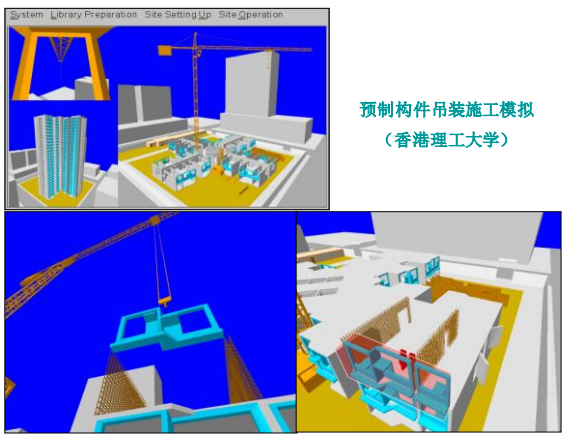


虚拟建造

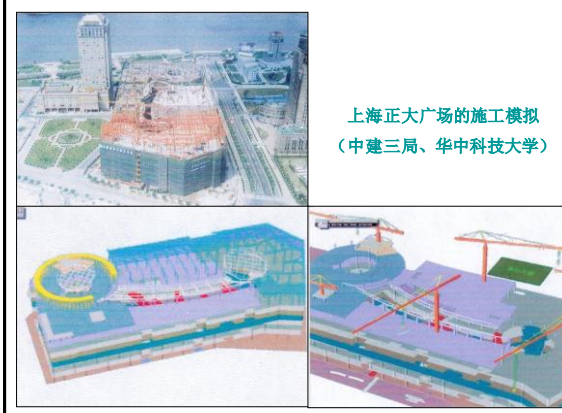
- **虚拟建造（制造）**
  - 采用计算机建模与仿真技术，在高性能计算机与高速网络的支持下，在计算机上群组协同工作，通过动画或虚拟现实，实现产品的设计、工艺规划、加工制造、性能分析、质量检验。
  - 施工方案的演示1
  - 施工方案的演示2
  - 施工方案的演示3

虚拟建造

- **施工方法实验**
  - 虚拟施工实验室（VCL）系统
  - 针对施工作业计划，为施工管理者提供接近实际的虚拟施工环境
  - 以类似于现实施工的方式安排施工作业计划，并评估其有效性
- **施工方案优化**
  - 施工前在计算机上完成多种吊装方案的实验和优化



预制构件吊装施工模拟
（香港理工大学）



上海正大广场的施工模拟
（中建三局、华中科技大学）

虚拟建造

- **过程模拟(仿真)**
  - **模拟：** 设计一个现实或虚拟系统的模型，并在此模型上进行试验的过程。
  - **目的：** 了解系统的行为或者评估系统运行的策略。
  - **模式：**
    - 如果模式关系比较简单，则整个系统往往可以直接利用数学分析方法，诸如代数、微积分或概率理论求得最终的答案，这种过程称为**解析法**。
    - 实际上各种系统组成相当复杂，无法利用简单的解析法直接获得结果，需要在计算机上建立起这些系统的数理逻辑模型，通过仿真的过程，以**实验的方法**来获得答案。

虚拟建造

- **模拟分类**
  - **连续模拟：** 系统的状态变量随时间连续地变化，通常表现为一个有时间变量的函数；
  - **离散事件模拟：** 系统的状态变量在有限的时间离散点上产生瞬间变化。这种状态参数发生的突然变化被称为**事件**。
    - **施工过程的模拟：** 其中每一个工序的开始、中断、恢复和完成，都伴随着系统的某些参数和状态的变化，如施工对象的变化，施工资源的占用与释放等，都可以看成是一个事件。
    - 离散事件模拟在计算机上的实现，只需要按照时间顺序推算在各个时点系统的状态变化，以模拟实际施工过程。
    - 离散事件模拟在施工管理中的应用较为广泛。

虚拟建造

- **SDESA（简化的离散事件模拟方法）**
  - 香港理工大学鹿明教授提出SDESA(Simplified Discrete-Event Simulation Approach) （简化的离散事件模拟方法）
  - 模型及模拟方法的改进
    - 模拟计算流程的优化
    - 场地布置信息和资源移动的模拟
    - 复杂施工过程控制
    - 施工过程动画演示

虚拟建造

- **SDESA模型**

  一个完整的SDESA模拟模型由施工过程定义、场地布置信息定义、控制变量和资源属性三部分组成：
  - **施工过程定义**
    - **流动实体**（Flow Entity）：施工过程中处于流动状态物体，使用一次后即不再可用，如待加工的零构件等。
    - **工序实体**（Activity）：代表施工过程中的具体施工工序，其中包括工序的工期、工序的优先级、完成工序所需要的资源等信息。
    - **资源实体**（Resource）：表示施工过程中所涉及的人员、材料、机械等资源，包括资源的种类，数量，可使用的时间等，使用完毕后该资源将被释放，可再次使用。

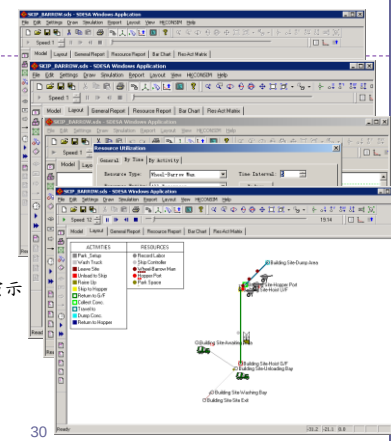
虚拟建造

- **场地布置信息定义**
  - **施工工地和关键地点**（Site & Location）：一个模型可包含多个场地，每个场地上又可定义一系列的关键地点，如工地上的塔吊、升降机、搅拌站、堆料区等，都可以定义为一个关键地点
  - **工序起止地点定义：** 工序分为生产工序和运输性工序。
  - **资源实体初始位置：** 模拟的过程中，资源实体的位置会发生移动，因此，需要给每个资源实体都设置初始的地点位置
  - **资源移动时间：** 指定每个资源从一个地点移动到另一个地点所需要的移动时间，以考虑由于资源移动所消耗的时间
- **控制变量和资源属性**
  - **控制变量：** 描述系统状态的全局变量
  - **资源属性：** 描述单个资源实体状态的变量

虚拟建造

SDESA系统

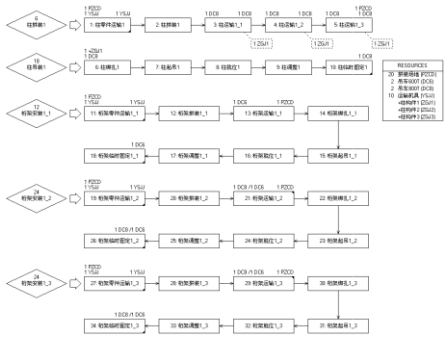
- 模型的建立
- 施工场地布置
- 施工过程模拟
- 结果统计分析
- 施工过程动画演示



30

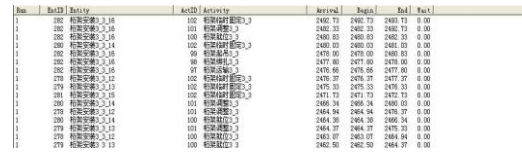
虚拟建造

- 国家体育场鸟巢钢结构拼装模拟与分析

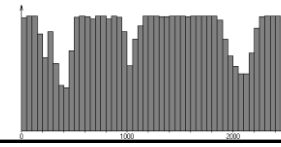


虚拟建造

- 国家体育场鸟巢钢结构拼装模拟与分析
  - ✓ 完成时间

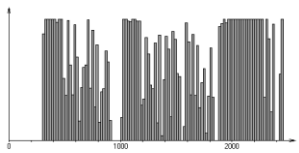


  - ✓ 拼装场地使用情况

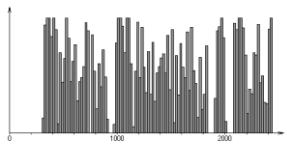


虚拟建造

- 国家体育场鸟巢钢结构拼装模拟与分析
  - ✓ 800T吊车使用情况



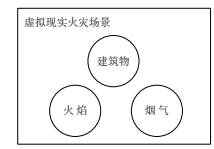
  - ✓ 600T使用情况



虚拟环境中的灾害模拟和防灾减灾

- 虚拟环境下的火灾模拟
  - 建筑火灾场景模拟

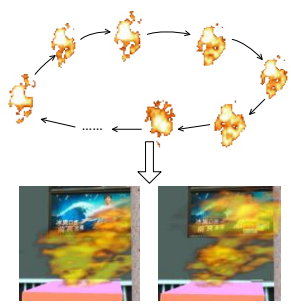
虚拟现实火灾场景的分解



火焰与烟气的模拟是建筑火场的关键技术

虚拟环境中的灾害模拟和防灾减灾

- 火焰的可视化技术
  - 火焰效果通过纹理循环渲染方法实现。
  - 通过一段火场视频获取多帧静态火场图片，利用图像处理软件将火场图片中非火焰的部分去除后得到多幅静态的火焰图片，将这些静态的火焰图片在虚拟场景中循环渲染，就可以营造出动态和逼真的火焰效果。

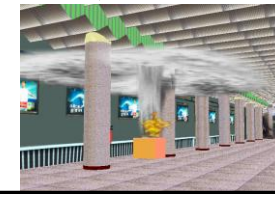


纹理循环渲染

虚拟环境中的灾害模拟和防灾减灾

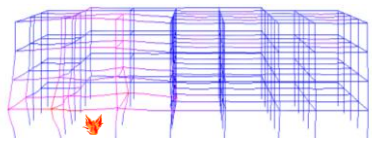
- 烟雾的可视化技术
  - 烟雾效果是基于粒子系统的原理实现的。通过每个粒子自身的烟雾纹理循环渲染，可营造动态的效果。
  - 根据FDS火灾场模拟的结果，提取建筑物中烟气分布和变化的规律，以此为基础，控制粒子系统的生命周期、粒子数目、粒子源、限定框、粒子大小、粒子贴图等各种属性的变化，可以得到很好的烟气分布和扩散可视化效果。

烟气羽流
顶棚射流



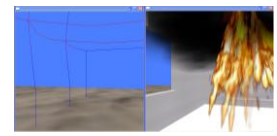
虚拟环境中的灾害模拟和防灾减灾

- **建筑结构反应模拟**
  - 基于相关结构火灾反应计算结果，以线段代表结构杆件，模拟随火灾发展过程中建筑结构变形情况。
  - 结构反应模拟需要建筑结构杆件等实体在虚拟环境中发生形变，这是区别火场模拟的最大特征。

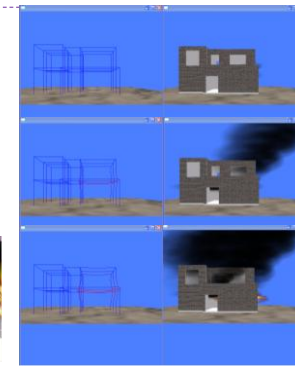


虚拟环境中的灾害模拟和防灾减灾

- **建筑火灾场景与结构火灾反应同步模拟实现**



火场与结构场景中同步漫游



火场与结构同步动态模拟

虚拟环境中的灾害模拟和防灾减灾

- **城市建筑地震波动模拟**

城市仿真系统

从可视化的角度出发，通过VR、GIS等技术构建出可自由漫游的城市模型，但这个模型是静态的，而不是基于地震计算的。

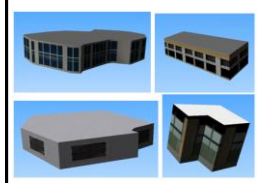
城市地震计算

通过合理简化的方法，计算出地震下城市房屋楼层的位移和破坏程度，这种计算是基于数值的，而庞大的计算结果并不能产生直观的印象。

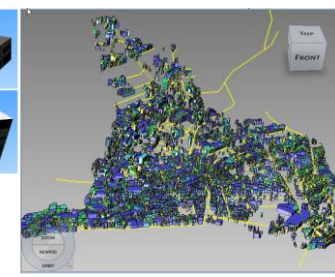
城市仿真与建筑物地震分析相结合

虚拟环境中的灾害模拟和防灾减灾

带颜色的建筑模型



带纹理的建筑模型



用不同颜色表示建筑破坏程度

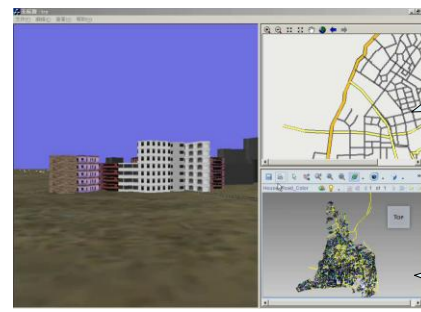
虚拟环境中的灾害模拟和防灾减灾

- **虚拟现实环境下楼层振动模拟**

结合地震结构响应模型的计算结果，在Vega的虚拟现实环境中表现建筑楼层振动

虚拟环境中的灾害模拟和防灾减灾

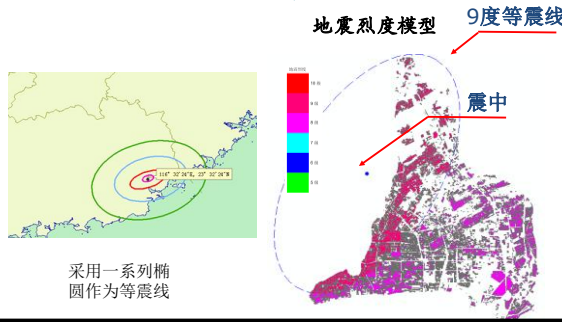
- **CAD/GIS/VR视图集成**



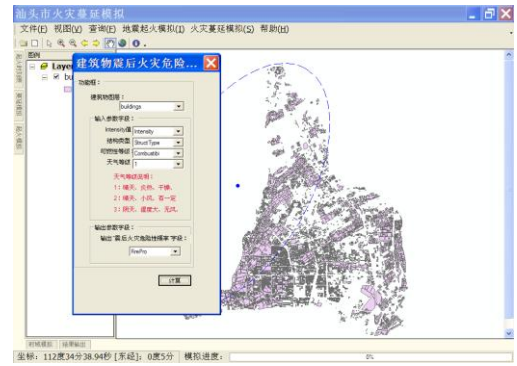
GIS视图
通过鼠标双击定位VR视图当前观察点，并提供查询功能

CAD视图
通过不同颜色描述建筑物的破坏程度

- 地震次生火灾蔓延模拟
  - 建筑物震后火灾危险概率计算



采用一系列椭圆作为等震线

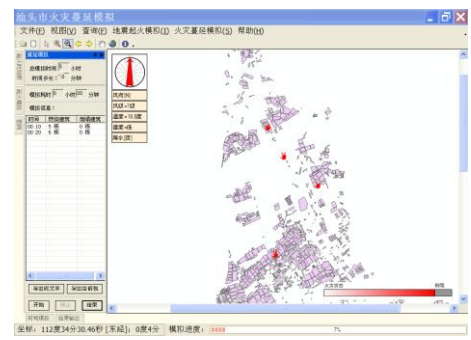


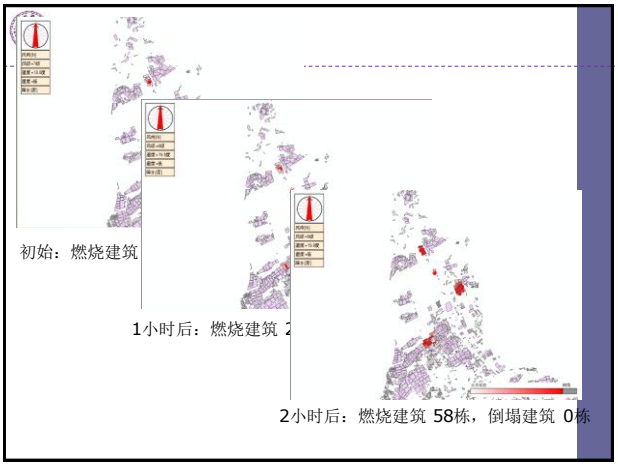
- 内部火灾发展和建筑间火灾蔓延建模
  - 建筑内部火灾发展的简化建模
    - 将个体起火建筑近似为独立的点火源
    - 5个阶段：起火（Ignition）、轰燃（Flashover）、火灾充分发展（Full-development）、倒塌（Collapse）和熄灭（Extinguishment）
    - 3个独立变量：时间量，温度和热释放率
  - 建筑间火灾蔓延建模
    - **热辐射**：
      - 热辐射强度包括：室内热烟气和火焰通过外墙开口发射的热辐射强度，加热后外墙发射的热辐射强度以及从窗口喷出火焰发射的热辐射强度

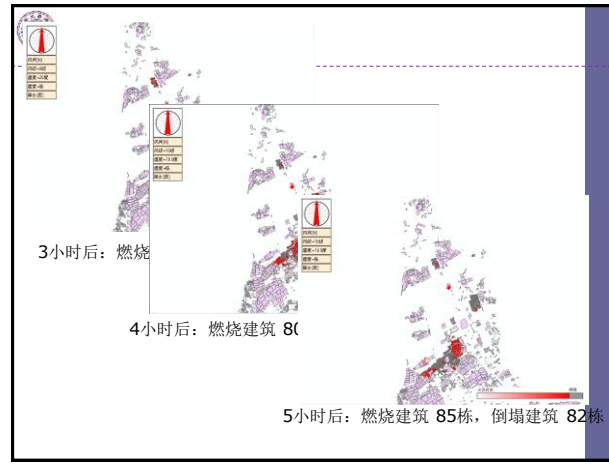
- **热羽流：**
  建筑物燃烧后产生的热烟气，由于其温度很高，密度很小，在空气中自然对流形成热羽流，导致流通路径上建筑物起火，造成蔓延。
- **飞火**
  除热辐射和热羽流外，如有大风存在，可燃物被大风带走，飞行一段距离之后落在其它可燃物上，并将其它可燃物点燃而形成飞火。飞火可跨越阻火要素（水体，公园，停车场等）而远距离点燃可燃物。
- **起火判断**
  - **外墙为木制结构：**外墙起火先于室内，当外墙接收的热辐射强度超出起火极限强度即判断起火。
  - **外墙不可燃：**起火出现在室内，当窗口接收的热辐射强度超出起火的极限热辐射强度时，即判断起火。

- **气象条件的影响**
  - **气温：**一天中的温度变化可以利用余弦曲线来表示
  - **湿度：**划分成“非常高”（>80%）、“高”（60%~80%）、“中等”（40%~60%）、“低”（20%~40%）和“非常低”（<20%）五个级别
  - **降水：**降水天气下模型认为火灾局限于建筑内部
  - **风：**风向控制室外火灾蔓延的方向，风力决定火灾蔓延的速度，考虑16个风向方位

基于GIS的地震次生火灾蔓延模拟系统







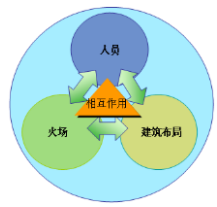
虚拟环境中的灾害模拟和防灾减灾

- 虚拟环境中火灾场景下人员疏散模拟

研究对象

火灾场景下的人员疏散

抽象模型

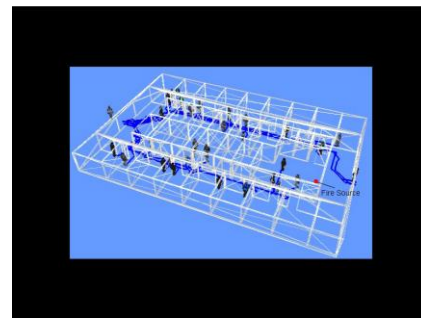


准确描述要素个体及个体间相互作用

虚拟环境中的灾害模拟和防灾减灾

- 灾害救援模拟及智能决策

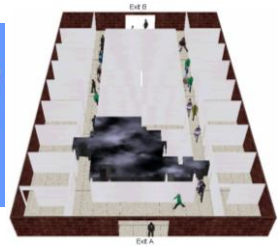
佩戴偏振眼镜观察火场人员疏散的虚拟仿真

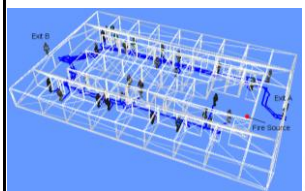


演示了通过模型计算求得的人员疏散轨迹、相应的火源位置、着火房间内人员的疏散情况。

虚拟环境中的灾害模拟和防灾减灾

全局视图：建筑环境、火场、人员的综合表现





轮廓视图：人员疏散轨迹与火源位置

虚拟环境中的灾害模拟和防灾减灾

着火房间内人员疏散

受困火场情形

虚拟环境中的灾害模拟和防灾减灾

走廊内的人员疏散

跟从行为

虚拟环境中的灾害模拟和防灾减灾

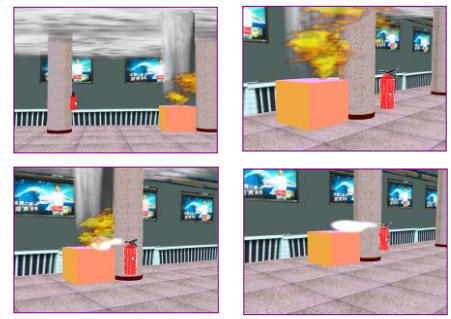
虚拟火灾疏散演习

虚拟环境中的灾害模拟和防灾减灾

基于数值结果的疏散模拟

虚拟环境中的灾害模拟和防灾减灾

- 虚拟环境中火灾扑救



虚拟火灾扑救演练

6.3 虚拟现实硬件和软件

- 虚拟现实设备
  - 高性能计算机；
  - 外部设备：数据手套；
  - 立体头盔；
  - 立体投影系统。

虚拟现实设备

立体头盔

立体投影系统

三维鼠标

数据手套

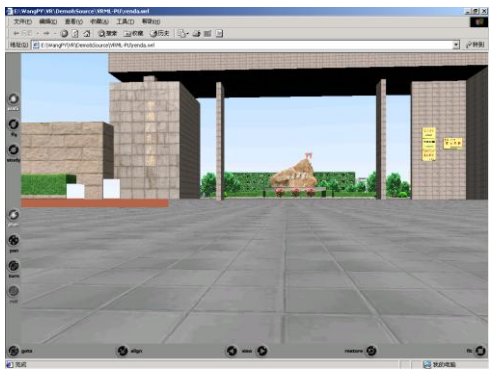
虚拟现实软件

- **OpenGL**
  - OpenGL是近些年发展起来的三维图形标准。在OpenGL之前，不同厂商的显示硬件使用不同的与硬件相关的图形函数库，这使得软件实现跨平台扩展成本很高。针对这种情况，在SGI等世界闻名的计算机公司的倡导下，以SGI的IRIS GLTM图形函数库为基础制定了一个通用的、开放式的、独立于操作系统和硬件环境的三维图形标准。
  - 目前大部分主流的显卡硬件都支持OpenGL，包括Microsoft，SGI、IBM、DEC、SUN、HP等公司都采用OpenGL作为三维图形标准，许多软件厂商也纷纷以OpenGL作为基础开发出自己的产品。

虚拟现实软件

- **VRML (Virtual Reality Modeling Language)**
  - VRML是一种模型标记语言，需要一个渲染引擎把模型描述转换成人们所见到的图象。
  - VRML作为一个WEB应用的标准，其浏览器插件有：
    - Microsoft VRML2.0浏览器，Windows自带的VRML插件；
    - 经典VRML插件：Cosmo;
    - blaxxun公司的blaxxunContact;
    - ParalleGraphics公司的CORTVRML

虚拟现实软件



虚拟现实软件

- **WTK**
  - WTK(World Tool Kit)由Sense8公司出品，是在OpenGL函数库的基础上开发的一种简洁的跨平台软件开发系统，包含1000个以上的C语言函数，应用于需要3D建模、实时浏览的仿真系统和虚拟现实系统。
  - WTK在其结构上采用了面向对象的方法，命名采用了面向对象的原则，把OpenGL中的大部分函数分类组合在一起。

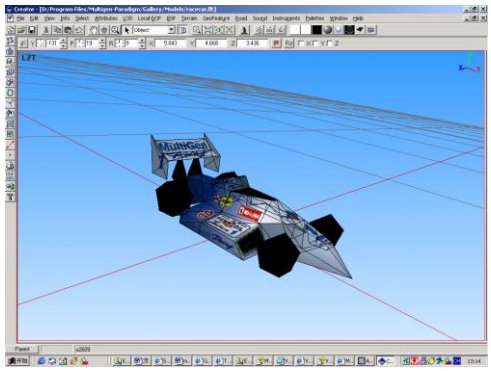
虚拟现实软件

- WTK支持常见的三维实体数据文件格式：
  - Autodesk DXF 格式;
  - Wavefront OBJ格式;
  - Autodesk 3DS格式;
  - MultiGen Flight格式;
  - WTK NFF格式;
  - VRML (*.WRL)格式.

虚拟现实软件

- **Vega & MultiGen Creator**
  - Vega是一套模块化C++函数库，用于创建实时可视化和音频仿真、虚拟现实系统。Vega中还包含LynX，一个可视化的编辑环境，可以在图形界面中修改视点、观察者、特殊效果、模型、数据库等信息。
  - MultiGen Creator是一个图形化的三维建模工具，用于视景仿真、城市仿真等应用领域，支持多种文件格式，包括3D Studio File(*.3DS)，AutoCAD(*.DXF)，STL(*.STL)，以及OpenFlight（*.flt）等多种文件格式。
  - 利用Creator可以在图形界面中进行多边形建模、纹理贴图、雾化处理、LOD(Level Of Detail)分层等操作。同时它提供了一套地形表面生成工具，可以根据标准的数据格式USGS、NIMA等来生成地形表面。

虚拟现实软件



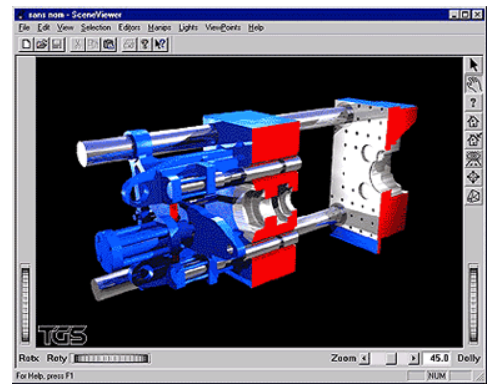
虚拟现实软件

- **Open Inventor**
  - Open Inventor 3D Toolkit是TGS公司的一个面向对象的软件工具箱，用C++和Java语言对Open GL模块进行了封装，是一个跨平台（包括Windows，UNIX，Linux）、面向对象的3D图形函数库，内含450个以上的类，从底层的实体如球、材质、灯光等到高层应用的实体材质编辑器等。
  - Open Inventor 是用C＋＋语言编写的面向对象的函数库，允许程序员用自定义的新实体来扩展函数库。一些Open Inventor的程序员在基本函数库的基础上添加了Bezier曲面、CSG实体、动画实体等实体。

虚拟现实软件

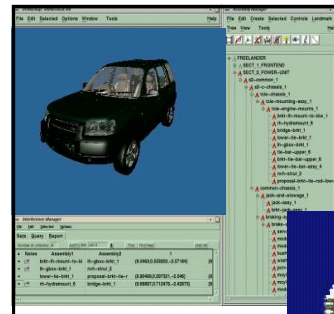
- 针对Mircosoft windows操作系统，TGS公司开发了一个MFC的3D扩展，叫IVF(Interactive Visual Framework™)，已经包括在Open Inventor中。利用IVF，在VC的WIZARD中添加自己的向导，可以方便的得到一个Open Inventor应用程序，在这个基础上在添加自己的功能代码也很方便。

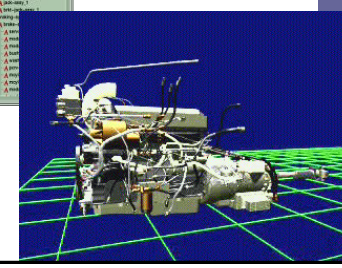
虚拟现实软件



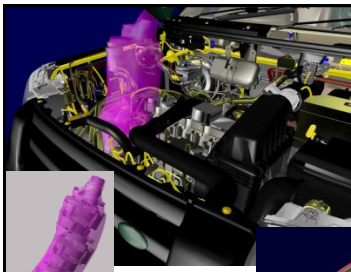
虚拟现实软件

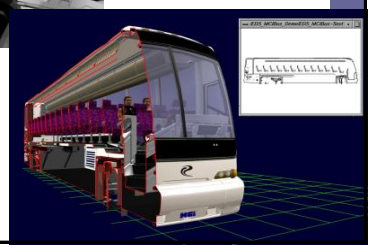
- **dV/Reality**
  - 大型产品模型实时浏览
  - 浏览整个产品的数字模型及渲染效果
  - 获取并改进产品设计思维
  - 减少实物模装次数，节约成本
- **dV/MockUp**
  - 获取完整的产品数字模型，进行预装配
  - 操作仿真，如安装或拆除
  - 利用人体模型进行人机工程设计，提高安全性
  - 作为产品操作及维护的训练手段





- **干涉检查**
  - 静态干涉检查
- **干涉及间隙检查**
  - 动态干涉检查





• 剖切检查
• 测量

• 设定安装及拆卸路径
可计算包络体

6.4 虚拟现实构造语言

- VRML (Virtual Reality Modeling Language)
  虚拟现实构造语言
- 被广泛地应用于Internet上创建虚拟的三维空间，VRML可以创建虚拟的建筑物、城市、山脉、飞船、星球等，在虚拟世界中添加声音、动画，还可以和浏览者进行交互的虚拟空间。
- VRML文件特征是由一个公开的VRML规范文件所规定的，在 http://tecfa.unige.ch/guides/ vrml/vrml97/spec 中可以找到这个规范文件，文件中叙述了VRML所有的技术细节。

6.4 虚拟现实构造语言

- **VRML文件**
  - VRML文件是虚拟三维空间的文本描述，可以通过任何文本编辑器(写字板,Word)编写，然后保存为后缀为wrl的文件
  - 常用的浏览器如Microsoft的Internet Explorer 通过自身集成VRML浏览插件，可以直接浏览带有VRML的网页

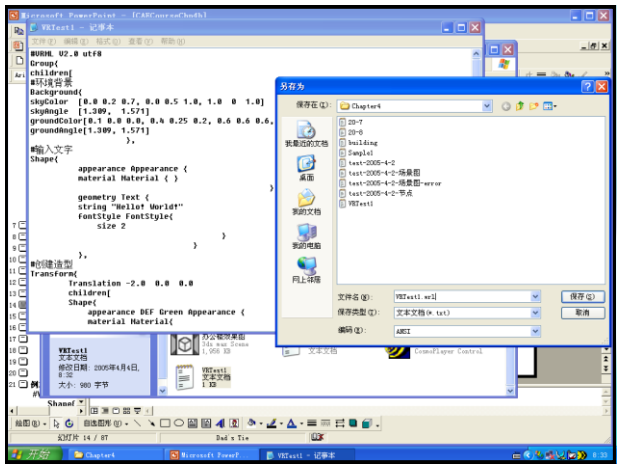
例1： VRML文件举例:
打开文本编辑器输入以下文本:(VRTest1)

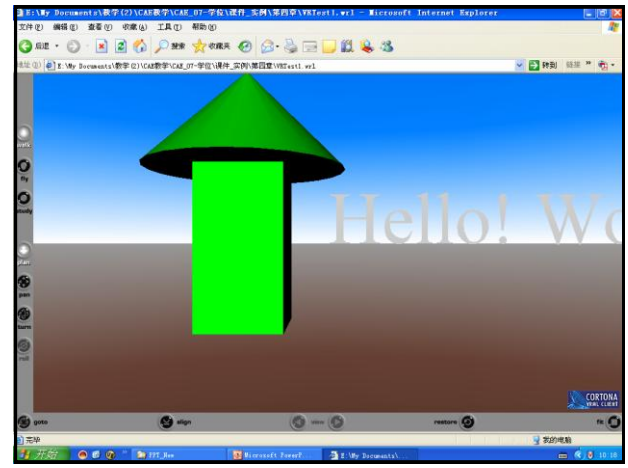
```
#VRML V2.0 utf8
Group{
children[
#环境背景
Background{
skyColor  [0.0 0.2 0.7, 0.0 0.5 1.0, 1.0  0  1.0]
skyAngle  [1.309,  1.571]
groundColor[0.1 0.0 0.0, 0.4 0.25 0.2, 0.6 0.6 0.6,]
groundAngle[1.309, 1.571]
        },
```

```
#輸入文字
Shape{
        appearance Appearance {
        material Material { }
                                }
        geometry Text {
        string "Hello! World!"
        fontStyle FontStyle{
           size 2
                                }
                        }
    },
```

```
#创建造型
Transform{
        Translation -2.0  0.0  0.0
        children[
        Shape{
          appearance DEF Green Appearance {
          material Material{
          diffuseColor 0.0  1.0  0.0
                                }
                                                }
  geometry Box{
    size 2.0  4.0  2.0
                }
              },

  Transform{
    translation 0.0  3.0  0.0
```

```
children Shape{
        appearance USE Green
        geometry Cone{
         height 2.4
         bottomRadius 2.4
                                }
                        }
                }
              ]
    }
  ]
}
```





VRML文件的基本结构

- **VRML文件是后缀名为“wrl”的文件，主要包括：**
  - VRML文件头(Header)、
  - 造型（场景图-Scene graph)、
  - 原型(Prototypes)、
  - 路由(Route)。

VRML文件的基本结构

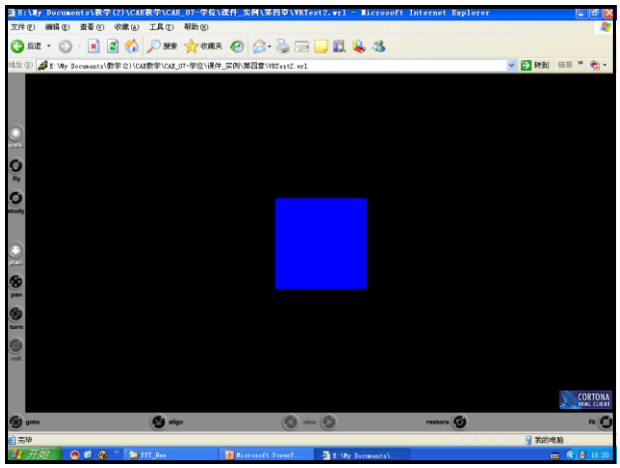
- **文件头(Header)**
  - 每一个VRML文件都必需，放在文件的第1行。
  - VRML文件头的语法形式：VRML V2.0 utf8
  - VRML文件头包括三部分：
    - VRML：告诉打开该文件的浏览器该文件是一个VRML文件；
    - V2.0：该VRML文件遵循的VRML规范是2.0版本；
    - utf8：该文件使用的字符集是国际UTF-8字符集。
      - UTF-8是USC Transform Format的缩写，其中USC是Universal Character Set的缩写，UTF－8是一个广泛支持多种语言的字符集，由国际标准化组织ISO 10646-1：1993标准所定义。

造型(场景图-Scene graph)

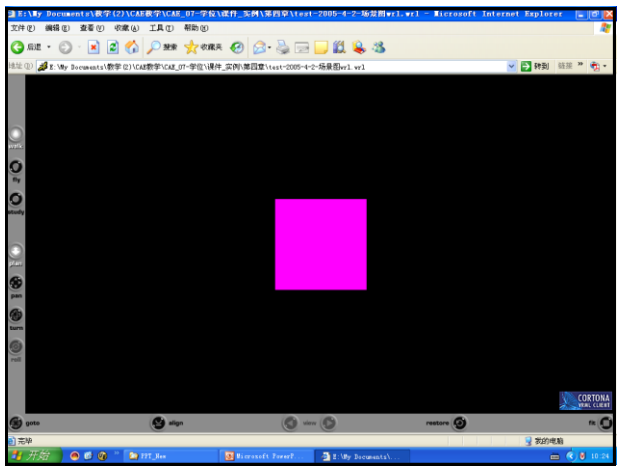
- 造型是VRML文件所描述的场景图，是实体的集合，并定义这些实体的顺序。
- 在一个场景中，排在前面的实体（结点）可以影响排在后面的实体。如实体的旋转（Rotation）或是材质（Material）会影响在它之后的实体，使它以某个角度旋转或是显示某种材质；
- 实体的分离（Separator）可以用来限制影响的范围。

例2 场景图 (VRTest2):

```
#VRML V2.0 utf8
    Shape{
      appearance Appearance{
        material Material{
            diffuseColor 0.0  0.0  1
                        }
                            }
      geometry Box {
                    size 2.0 2.0 2.0
                   }
       }
```



```
#VRML V2.0 utf8
    Shape{
      appearance Appearance{
        material Material{
            diffuseColor 1.0  0.0  1
                        }
                            }
      geometry Box {
                    size 2.0 2.0 2.0
                   }
}
```



VRML原型(Prototype)

- 原型是对节点内容的定义，这些节点可以是ISO/IEC组织中定义的,也可以是在VRML文件的原型部分中自行定义的。
- 原型包括:
  - 原型名;
  - 节点接口;
  - 节点体.
- PROTO Cube [ ] { Box { } }

例3:原型的使用(VRTest3)

```
#VRML V2.0 utf8
PROTO Pcone [
Field SFColor coneColor 1.0 0.0 0.0
]
{
Shape{
    appearance Appearance{
    material Material {
        diffuseColor IS coneColor
        specularColor 1.0 1.0 1.0
                }
            }
    geometry Cone {
        height 3.0
            bottomRadius 0.8
                }
    }
}
Pcone { }
```

定义原型的关键字

原型名

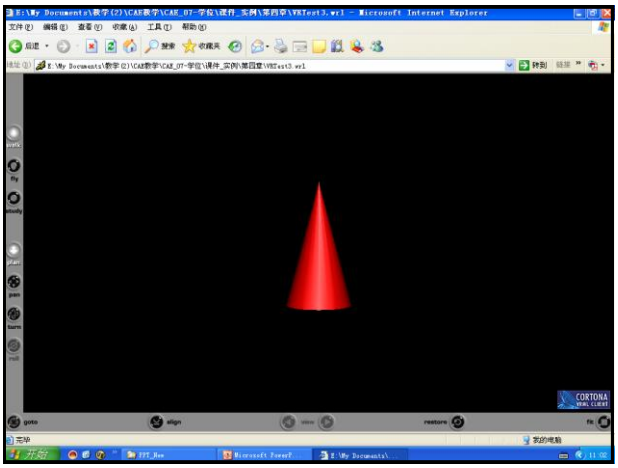
节点接口

原型中域的作用范围

读原型的节点体的具体内容

VRML原型(Prototype)

- **PROTO:** 定义原型的关键字;
- **Pcone:** 原型名;
- **[]:** 原型中域的作用范围,里面的内容为节点接口;
- **Field:** 说明节点域的关键字;
- **SFColor:** 节点域的类型;
- **coneColor:** 节点域的名称,即节点接口名称;
- **最外层{}:** 节点体的作用范围;
- **Shape:** 该原型的节点体的具体内容;
- **diffuseColor IS coneColor:** 使用节点接口的值.



VRML原型(Prototype): 节点

- 节点(Node)
  - VRML文件的最基本的组成部分是节点:
  - 一个VRML节点是一个对象，节点是各种现实物体或概念的抽象。
  - VRML场景中的各个对象都是特定类型的节点:
    - **简单的几何节点**（Geometry Node）：长方体（Box）、球体（Sphere）、圆柱体（Cylinder）等;
    - **复杂节点:** 包括域和事件，各种信息可以在节点之间通过路由-Route来传递。
    - **其他类型节点:**
      - ✓ **材料节点**（Material Node）：定义该节点之后所有几何节点的表面属性;
      - ✓ **纹理节点**（Texture Node）：定义该节点后的所有几何节点的纹理属性。
  - VRML 1.0定义了36个节点;VRML 2.0定义了54个节点。

VRML原型(Prototype): 节点

- 域(Field)
  - 当节点属于同一种类型时，VRML通常用参数来区分它们。即VRML节点的域。
  - 域定义了节点的属性，每一个节点设有一个或多个域，域可能包括各种不同的数据和一个或多个域值，每个域都有各自的缺省值。
  - 域通常可分为两类:
    - **单值类型的域:**
      - ✓名称以“SF”开始;
      - ✓只包含单值：可以是一个单独的数，也可以是定义一个向量或颜色的几个数，甚至可以是定义一幅图像的一组数。

VRML原型(Prototype): 节点

- **多值类型的域:**
  - 名称以“MF”开始。
  - 包含多个单值。
  - 在VRML文件中，表示多值域的方法:
    - ✓一系列用逗号和空格间隔开的单值，整个用方括号括起来。
    - ✓如果一个多值域，不包含任何值，则只标出方括号“[]”，其中不填任何数。
    - ✓如果一个多值域，恰好只包含一个数，可以不写括号，直接写该值。

VRML中的各种域值类型列表

域值类型	功能说明
SFBool	表示开关值。值为TRUE或者FALSE，常用于指定某一属性的打开或者关闭
SFFloat/MFFloat	表示浮点值。值为具有正负之分的实数，用于指定某一确定的数值属性
SFColor/MFColor	表示颜色值。值为由3个浮点值组成的数组，用于指定一个RGB规定的颜色
SFRotation/MFRotation	表示旋转值。值为由4个浮点值组成的数组，前3个数指定旋转的坐标轴，第4个数指定旋转的角度
SFString/MFString	表示字符串。值为由中括号中的一组字符，用于指定多个选择中的一个选项
SFVec2f/MFVec2f	表示二维浮点向量。值为两个浮点值，用于指定一个二维的位置
SFVec3f/MFVec3f	表示三维浮点向量。值为三个浮点值，用于指定一个三维的位置
SFInt32/MFInt32	表示32位整数。值为正整数
SFImage	表示图像值。用于指定描述图像颜色的一系列的数值，用于绘制造型表面的纹理图像
SFTime	表示时间值。值为一个从格林威治时间1970年1月1日零时整开始的、以秒计算的时间，用于指定一个动画开始和结束的时间
SFNode/MFNode	表示节点值。用于表明一个属性节点，控制造型节点创建造型

VRML原型(Prototype)：节点定义和引用

- 节点的定义及引用
  - 在VRML文件中为节点定义一个名称后，即可在文件的后面反复地引用该节点。
  - 被定义的节点称为原始节点；对被命名节点的引用称为实例。
  - 节点的域值在原始节点中应已设定，在实例中这些域值将被不能修改地完全相同地引用。当原始节点中的域值被修改，所有的实例也将自动同时修改。
  - **定义节点名称的基本语法：**

    DEF 节点名称{……
    ……}
  - **引用节点的语法：**

    USE 节点名称

```
geometry DEF light01 Sphere{radius 1.0 }
                        }
                    },
#红灯
        Transform{
          translation 0.0 2.5 0.0
              children Shape{
                  appearance Appearance{
                      material Material{
                          diffuseColor 1.0 0.0 0.0
                                                }
                                                    }
                  geometry USE light01
                        }
                    },
```

VRML原型(Prototype)：节点造型

- 使用节点造型创建虚拟场景
  - **Shape节点：** 创建虚拟世界中单个的几何造型；
    - 长方体（Box）、圆柱体（Cylinder）、圆锥体（Cone）、球体（Sphere）4种基本空间造型是由专门节点直接创建；
    - 其他复杂的空间造型则是由其他高级的造型方法来创建。
  - **Group节点：** 将各个单个的造型节点结合在一起；
  - **Appearance节点：** 外观域的域值。

VRML原型(Prototype)：节点造型

- 使用简单节点造型
  - **Shape节点：**
    - **将Appearance指定的材质和质感应用到geometry域的几何节点。**

基本语法：

```
Shape {
  appearance  SFNode exposedField NULL
  geometry    SFNode exposedField NULL
    }
```

域值说明：

- **appearance：** 包含一个Appearance节点。 Appearance节点定义造型外观的颜色和纹理，缺省值为NULL，表示白色、发光的造型外观。
- **geometry：** 包含一个几何节点（如：Box、 Cone、IndexedFaceSet、PointSet），缺省值为NULL，表示没有造型存在。

VRML原型(Prototype)：节点造型

- **Appearance节点**
  - 在Shape节点中的appearance域中出现。该节点中所有域值均可为NULL。
  - 如果material域是NULL，与Appearance相关的几何形体是不亮的，
  - 如果material域包含一个缺省的材料节点，即该几何形体被缺省的材料节点的值照亮。

基本语法：

```
Appearance {
    material SFNode exposedField    NULL
    texture  SFNode exposedField    NULL
    textureTransform SFNode exposedField  NULL
        }
```

VRML原型(Prototype)：节点造型

域值说明：

- **material：** 包含一个Material节点。
- **texture：** 包含一个ImageTexture、MovieTexture或者PixelTexture节点。
- **textureTransform：** 包含一个textureTransform节点，如果texture域为NULL，则textureTransform无效。

VRML原型(Prototype)：节点造型

- **Material节点**
  - 为相关的几何形体定义表面材料的特性。
  - Material节点的域决定物体表面反射光并产生颜色的方式。
  - 该节点的域值都是从0.0到1.0。
  - specularColor和shininess决定了镜面反射光。
  - 当光线到物体表面的角度接近于表面到观察者的角度时，specularColor将被加到漫反射颜色的计算中。低反光值产生柔和光，而高反光值产生边界清晰的较小的亮点。

基本语法：

```
Material {
   diffuseColor  SFColor      exposedField  0.8 0.8 0.8
   ambientIntensity SFFloatexposedField  0.2
   emissiveColor  SFColor            exposedField  0.0 0.0 0.0
   shininess SFFloat          exposedField  0.2
   specularcolor SFColor             exposedField  0.0 0.0 0.0
   transparency  SFFloat     exposedField  0.0
        }
```

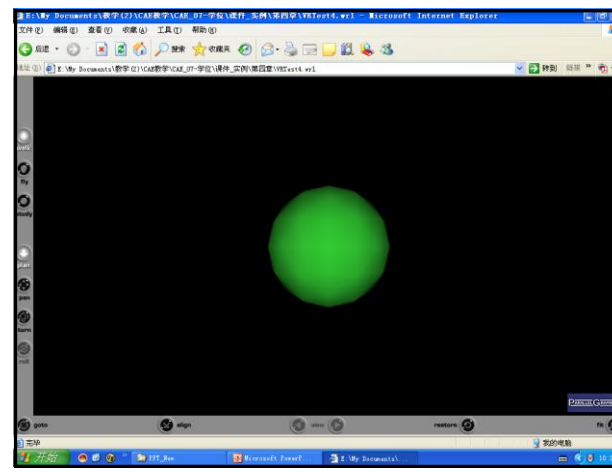
VRML原型(Prototype)：节点造型

域值说明：

- **DiffuseColor域：** 指定造型材料的颜色，采用3个0.0-1.0之间的有序实数表示某一种特定颜色，缺省域值为 0.8 0.8 0.8;
- **ambientIntensity域：** 指定空间的环境灯光对造型的影响程度，域值为0-1.0，缺省域值为0.2;
- **EmissiveColor域：** 指定造型发光的颜色，缺省域值为0.0 0.0 0.0（黑色），即造型不发光;
- **shininess 域：** 指定造型材料的亮度，域值较大时造型更有光泽，缺省域值为0.2；
- **specularColor域：** 指定造型有镜面反射的区域的颜色，缺省域值为 0.0 0.0 0.0;
- **Transparency域：** 指定造型的透明度，域值为0.0-1.0，缺省域值为 0.0，造型完全不透明。

例4： 节点(VRTest4)

```
#VRML V2.0 utf8
Shape {
        appearance Appearance {
            material Material{
            diffuseColor 0.2 0.8 0.2
            shininess 0.8
                        }
                                }
        geometry Sphere {
            radius 1.5
                        }
        }
```



例5：多个节点(VRTest5):

```
#VRML V2.0 utf8
Group{
    children[
        Shape{
            appearance Appearance{
                material Material{
                    diffuseColor 0.6 0.6 0.6
                                }
                                }
            geometry Box{
                size 3.0 10.0 1.0
                        }
            },
```

#绿灯

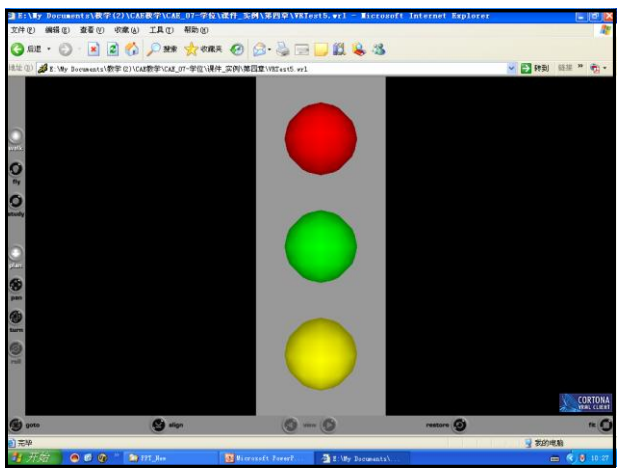
```
    Transform{
        translation 0.0 0.0 0.0
        children Shape{
         appearance Appearance{
              material Material{
                  diffuseColor 0.0 1.0 0.0
                                }
                                }
          geometry DEF light01 Sphere{radius 1.0 }
                        }
                },
```

#红灯

```
        Transform{
            translation 0.0 2.5 0.0
                children Shape{
                    appearance Appearance{
                        material Material{
                            diffuseColor 1.0 0.0 0.0
                                        }
                                        }
                    geometry USE light01
                            }
                },
```

#黄灯

```
        Transform{
          translation 0.0 -2.5 0.0
              children Shape{
                  appearance Appearance{
                      material Material{
                          diffuseColor 1.0 1.0 0.0
                                        }
                                        }
                  geometry USE light01
                            }
                    }
            ]
    }
```



VRML原型(Prototype)：基本造型节点

- **基本VRML造型节点**

  VRML文件中最基本的造型节点包括：Box节点、Cylinder节点、Cone节点和Sphere节点共4种。

  – **Box节点**

  - 描述长方体节点：长方体的中心位于局部坐标系原点，每个面均与坐标系平行，在缺省的情况下，长方体在x、y、z方向上的长度都是2。

  基本语法：

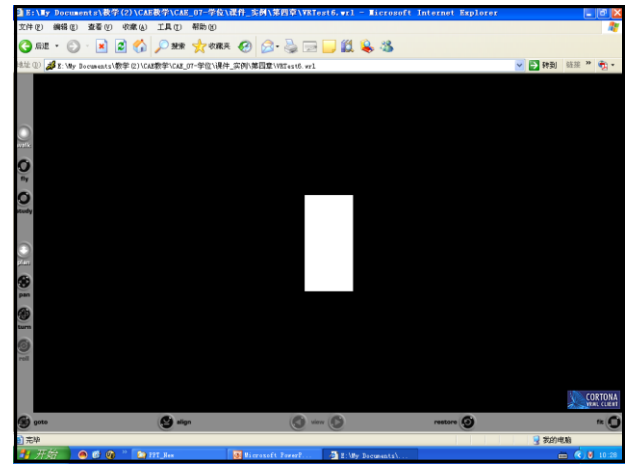
  ```
  Box {
        Size  MFFloat  2.0 2.0 2.0
      }
  ```

  域值说明：

  Size：相应于长方体宽、高和深度的x、y和z值。

例6：基本造型节点-Box (VRTest6)

```
#VRML V2.0 utf8
    Shape{
        appearance Appearance {
            material Material {
                diffuseColor 1.0 1.0 1.0
                            }
                                    }
        geometry Box {
                size 1.0 2.0 3.0
                    }
                }
```



VRML原型(Prototype)：基本造型节点

- Cylinder节点
  - 描述圆柱体的几何形体节点：以y轴为中心轴，其中心位置缺省值为（0，0，0），在三个坐标方向的长度缺省为2。可通过向radius和height域赋值创建不同大小的圆柱体。

基本语法

```
Cylinder {
    radius  SFFloat     field   1.0
    height  SFFloat     field   2.0
    side    SFBool      field   TRUE
    bottom  SFBool      field   TRUE
    top     SFBool      field   TRUE
        }
```

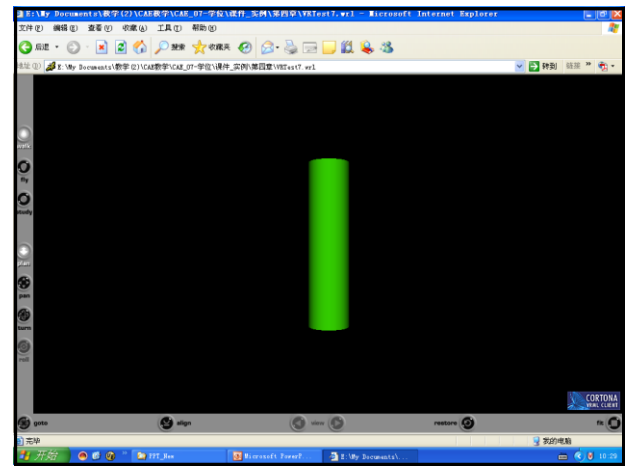
VRML原型(Prototype)：基本造型节点

域值说明：

- radius：指定圆柱体的半径
- height：指定圆柱体的高
- side：指明圆柱体的侧面是可见的（TRUE）或不可见的（FALSE）
- bottom：用来指明圆柱体的底是可见的（TRUE）或不可见的（FALSE）
- top：用来指明圆柱体的顶是可见的（TRUE）或不可见的（FALSE）

例7：基本节点—Cylinder (VRTest75)

```
#VRML V2.0 utf8
    Shape{
        appearance Appearance {
            material Material {
                diffuseColor 0.2 0.8 0.0
                            }
                                    }
        geometry Cylinder {
                radius 0.5
               height 4.0
                side TRUE
                bottom TRUE
                top TRUE
                            }
            }
```



VRML原型(Prototype)：基本造型节点

- Cone节点
  - 简单的锥体节点：其中轴是局部坐标系的y轴。缺省情况下，锥体的中心在点（0，0，0）处，并且在x、y、z三个方向上的尺寸都是从-1到1。

基本语法：

```
Cone   {
  bottomRadius  SFFloat     field    1.0
  height        SFFloat     field    2.0
  side          SFBool      field    TRUE
  bottom        SFBool      field    TRUE
      }
```

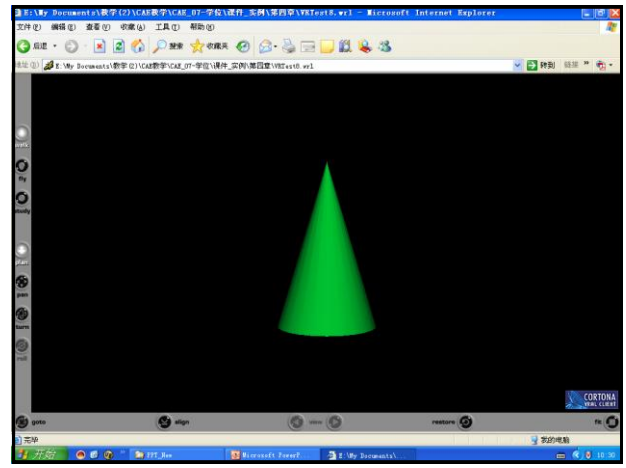
VRML原型(Prototype)：基本造型节点

域值说明：

- bottomRadius：用来指定锥体的底圆半径。
- height：用来指定锥体的高度。
- side：用来指明锥体的侧面是否是可见的（TRUE）或不可见的（FALSE）。
- bottom：用来指明锥体的底是否是可见的（TRUE）或不可见的（FALSE）

例8：基本节点—Cone (VRTest8)

```
#VRML V2.0 utf8
    Shape{
      appearance Appearance {
          material Material {
             diffuseColor 0.0 0.8 0.2
                        }
                           }
      geometry Cone {
          bottom FALSE
          bottomRadius 1.2
          height 4.0
           side TRUE
                        }
       }
```



VRML原型(Prototype)：基本造型节点

- Sphere节点
  - 球体节点：缺省时，球以原点为圆心，以1为半径。当一个纹理图应用到一个球上时，缺省的纹理将覆盖整个表面，从球体后面逆时针铺开。纹理在yz平面的后面缝合。

基本语法：

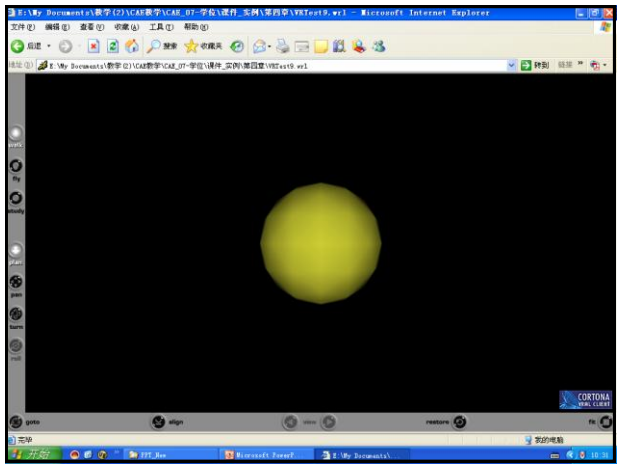
```
Sphere{
  radius  SFFloat   field   1.0
    }
```

域值说明：

- ✓radius：用来指定球的半径。

例9：基本节点—Sphere (VRTest8)

```
#VRML V2.0 utf8
    Shape{
      appearance Appearance {
          material Material {
             diffuseColor 0.8 0.8 0.2
                        }
                           }
      geometry Sphere {
          radius 1.5
                        }
       }
```



VRML原型(Prototype)：造型定位及变换

- **造型定位及变换**
  - 在VRML文件中通过对坐标系平移和旋转，形成新的处于不同位置和不同方向上的空间坐标系，然后在新的坐标系中创建空间造型。
  - 在VRML中创建新的空间坐标系都是通过将所要安排位置和方向的造型用Transform节点编组，一个Transform节点表示相对当前坐标系形成了一个新的坐标系，而在Transform节点编组中的空间造型都是相对于这个新创建的坐标系所创建的。

基本语法：

```
Transform {
    addChildren     MFNode      EventIn
    removeChildren  MFNode      EventIn
    children        FNode       exposedField  [ ]
    center          SFVect3f    exposedField  0 0 0
    rotation        SFRotation  exposedField  0 0 1 0
    scale           SFVect3f    exposedField  1 1 1
    scaleOrientation SFRotation exposedField  0 0 1 0
    translation     SFVect3f    exposedField  0 0 0
    bboxCenter      SFVect3f    field         0 0 0
    bboxSize        SFVect3f    field         -1 -1 -1
        }
```

域值说明：

- ✓**addChildren**：输入接口，将指定节点加入该组的子项列表中。
- ✓**removeChildren**：输入接口，将指定节点从该组子项列表中删除。
- ✓**children**：受该节点指定的变换影响的子节点。
- ✓**center**：指定缩放和旋转操作的原点。
- ✓**rotation**：给定旋转的轴和角度（以弧度为单位）。
- ✓**scale**：指定缩放比例，各轴向缩放比值可以不相等。
- ✓**scaleOrientation**：指定缩放和旋转操作的轴向。
- ✓**translation**：指定变换量。
- ✓**bboxCenter**：围绕该变换子项的包围盒的中心。
- ✓**bboxSize**：包围盒在x、y、z方向的值，缺省值是无包围盒。

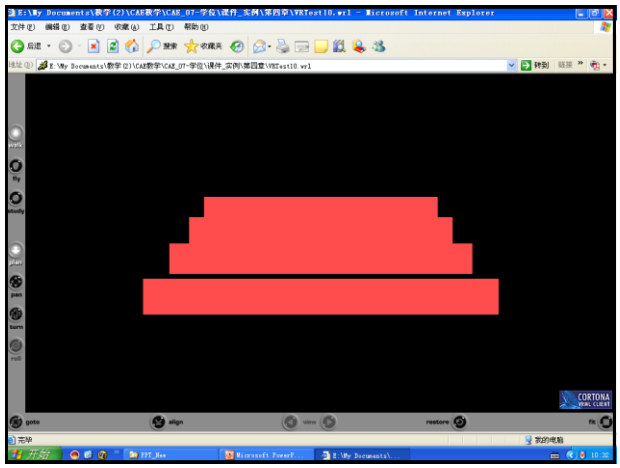
例10：造型变换(VRTest10)

```
#VRML V2.0 utf8
Transform {
    children [
        Shape{
            appearance DEF bRed Appearance {
                material
                    diffuseColor 1.0 0.3 0.3
                                                }
                geometry Box {
                size 5.0 2.0 2.5
                                }
            },
```

```
Transform {
    translation 0.0 -0.25 1
        children [
            Shape {
                appearance USE bRed
                    geometry Box {
                        size 5.0 1.5 2.5
                                }
            },
```

```
Transform {
    translation 0.0 -0.25 1
    children [
        Shape {
            appearance USE bRed
            geometry Box {
                size 5.0 1.5 2.5
                }
        },
```

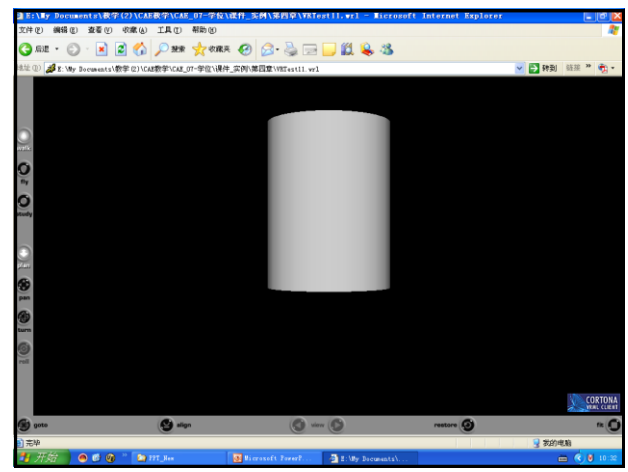
```
Transform {
    translation 0.0 -0.25 1
    children [
        Shape {
            appearance USE bRed
            geometry Box {
                size 5.0 0.5 2.5
                }
        },
    ]
  }
 ]
}
]
}
]
}
```



- **坐标系的缩放与旋转**
  - 利用Transform节点的scale域、scaleOrientation域和center域可以在空间的各个坐标轴方向上缩放当前坐标系以创建新的空间坐标系，然后在新的坐标系中创建造型。
  - **scale域：**
    - scale域值分别给出了在X,Y,Z三个坐标轴方向上的坐标缩系数(均为非零的正数)。各缩放系数的缺省值都为1。当缩放系数小于1大于0时，相应的坐标轴方向上将发生缩小；当缩放系数大于1时，相应的坐标轴方向上将发生放大。
    - 用scale域指定的空间坐标系缩放的默认缩放中心在当前坐标系的坐标原点，即在同一坐标轴上的缩放相对于坐标原点是对称的。
    - 缩放中心还可以通过Transform节点的center域指定。

例11：造型缩放(VRTest11)

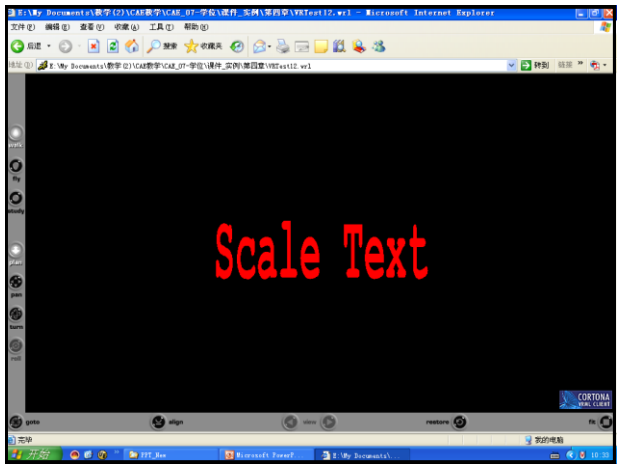
```
#VRML V2.0 utf8
Transform {
    scale 1.5 2.0 1.0
    center 0.0 -1.0 0.0
    children [
        Shape{
            appearance Appearance {
                material Material { }
                }
            geometry Cylinder {
                }
        }
    ]
}
```



例12：造型缩放(VRTest12)

```
#VRML V2.0 utf8
Transform {
    scale 1.0 2.5 1.0
        center 0.0 1.0 0.0
            children Shape {
                appearance Appearance {
                    material Material {
                        diffuseColor        1 0 0
                                        }
                                        }
```

```
geometry Text {
    string ["Scale Text", ]
        fontStyle FontStyle {
            family "TYPEWRITER"
            style "BOLD"
            horizontal TRUE
            leftToRight TRUE
            topToBottom TRUE
            justify ["MIDDLE", "END"]
                                }
                }
        }
}
```



路由

- 产生事件和接收事件的节点之间的连接。
- 路由的作用是将各个不同的节点绑定在一起以使虚拟空间具有动感和交互性。
- 一般情况下，路由的写法：
  ROUTE node.sourceFiled To node.sinkField
- 在两个节点之间存在着路由，事件将可以通过路由由这个节点传递到另外一个节点上。这样传递的事件通常可以改变相应节点的某些域值。
- 例如，在虚拟世界中分别创建了一盏灯和一个开关，通过合适的路由将两者绑定之后，可以通过鼠标单击开关来控制灯的亮灭，此时通过路由传输的事件就是灯的外观控制。

路由

- **动画效果**
  - **TimeSensor节点：在虚拟空间中创建一个驱动动画效果的时钟**

语法：

```
TimeSensor {
    enable SFBool field TRUE
    startTime SFTime field 0.0
    StopTime SFTime field 0.0
    cycleInterval SFTime field 1.0
    loop SFBool field FALSE
    isActive SFBool
    time SFTime
    cycleTime SFTime
    fraction_changed SFFloat
}
```

路由

域值说明：

- ✓ **Enable：** 控制该时间传感器是否打开，缺省值为TRUE
- ✓ **startTime：** 指定该时间传感器开始驱动的时间，缺省值为0.0
- ✓ **StopTime：** 指定该时间传感器停止驱动的时间，缺省值为0.0
- ✓ **cycleInterval：** 给出一个时间长度，指定该时间传感器从0.0时刻到1.0之间的周期间隔，单位为秒，缺省值为1.0
- ✓ **Loop：** 指定该时间传感器是否循环输出，缺省值为 FALSE
- ✓ **isActive：** 输出接口
- ✓ **Time：** 输出接口
- ✓ **cycleTime：** 输出接口
- ✓ **fraction_changed：** 输出接口，输出时间传感器的一些浮点时刻。

```
Example 20_7
#VRML V2.0 utf8
Group{
    children[
        Shape{
            appearance Appearance{
                material Material{
                    diffuseColor 0.6 0.6 0.6
                                }
                }
            geometry Box{
                size 3.0 10.0 1.0
                }
            },
#绿灯
        Transform{
            translation 0.0 0.0 0.0
            children Shape{
                appearance Appearance{
                    material DEF Bgreen Material{
                        diffuseColor 0.0 0.2 0.0
                                    }
                                }
                geometry DEF light01 Sphere{ }
                }
            },
```

```
#红灯
        Transform{
            translation 0.0 2.5 0.0
            children Shape{
                appearance Appearance{
                    material DEF Bred Material{
                        diffuseColor 0.2 0.0 0.0
                                    }
                                }
                geometry USE light01
                }
            },
#黄灯
        Transform{
            translation 0.0 -2.5 0.0
            children Shape{
                appearance Appearance{
                    material DEF Byellow Material{
                        diffuseColor 0.2 0.2 0.0
                                    }
                                }
                geometry USE light01
                }
            },
```

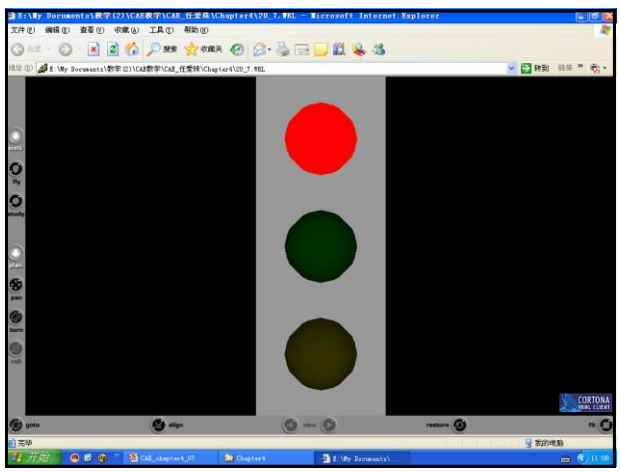
```
DEF Clock TimeSensor{
            cycleInterval 10.0
            loop TRUE
            },
    DEF Bgreenpath ColorInterpolator{
            key[ 0.0,0.49,0.5,0.99 ]
            keyValue[
                0.00 1.00 0.00,
                0.00 1.00 0.00,
                0.00 0.00 0.00,
                0.00 0.00 0.00,
                ]
            },
    DEF Bredpath ColorInterpolator{
            key[ 0.0,0.49,0.5,0.99]
            keyValue[
                0.00 0.00 0.00,
                0.00 0.00 0.00,
                1.00 0.00 0.00,
                1.00 0.00 0.00,
                ]
            },
```

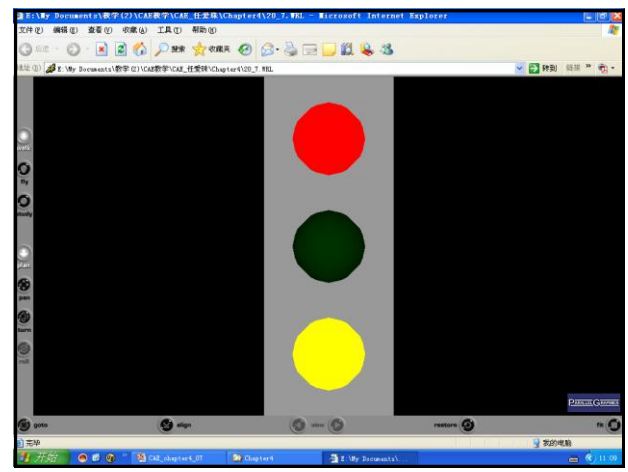
指定关键颜色值列表

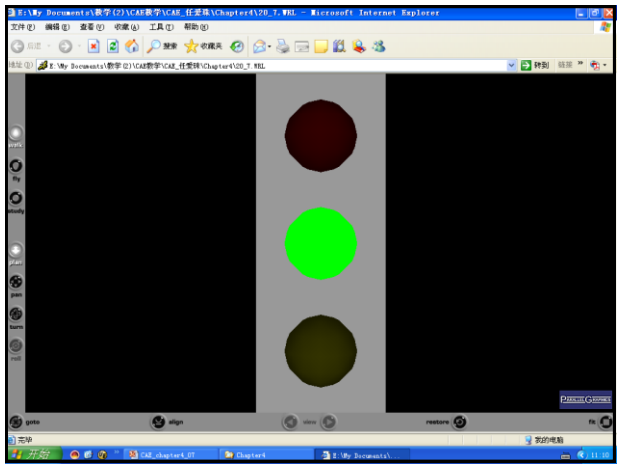
```
DEF Byellowpath ColorInterpolator{
            key[ 0.0,0.45,0.46,0.50,0.95,0.96 ]
            keyValue[
                0.00 0.00 0.00,
                0.00 0.00 0.00,
                1.00 1.00 0.00,
                0.00 0.00 0.00,
                0.00 0.00 0.00,
                1.00 1.00 0.00,
                ]
            }
    ]
}
ROUTE Clock.fraction_changed  TO Bgreenpath.set_fraction
ROUTE Clock.fraction_changed  TO Bredpath.set_fraction
ROUTE Clock.fraction_changed  TO Byellowpath.set_fraction

ROUTE Bgreenpath.value_changed TO Bgreen.set_emissiveColor
ROUTE Bredpath.value_changed TO Bred.set_emissiveColor
ROUTE Byellowpath.value_changed TO Byellow.set_emissiveColor
```

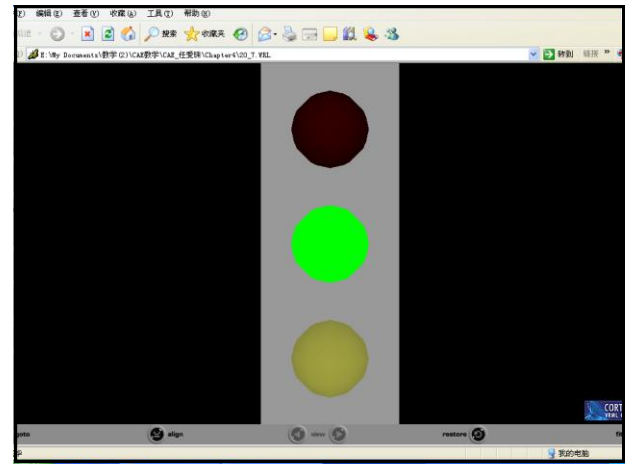
fraction_changed：输出时间传感器的一些浮点时刻

EmissiveColor域：指定造型发光的颜色，









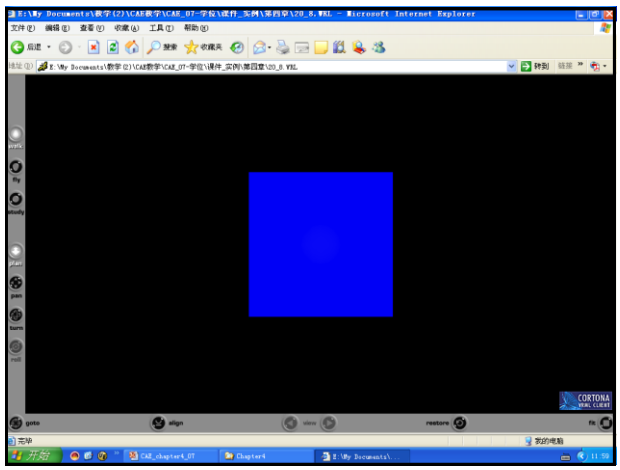
Example 20_8

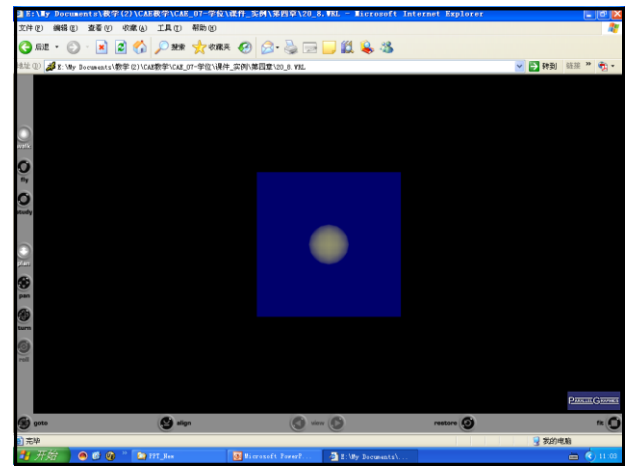
```
#VRML V2.0 utf8
Group{
    children[
      Shape{
        appearance Appearance{
            material DEF Bboxmeterial Material{
                                                diffuseColor 0.0 0.0 1.0
                                                }
                                    }
        geometry Box{
                            size 3.0 3.0 3.0
                        }
        },
    Shape{
        appearance Appearance{
            material Material{
                diffuseColor 1.0 1.0 0.0
                                }
                                    }
            geometry Sphere{
                                radius 0.5
                                }
        },
```

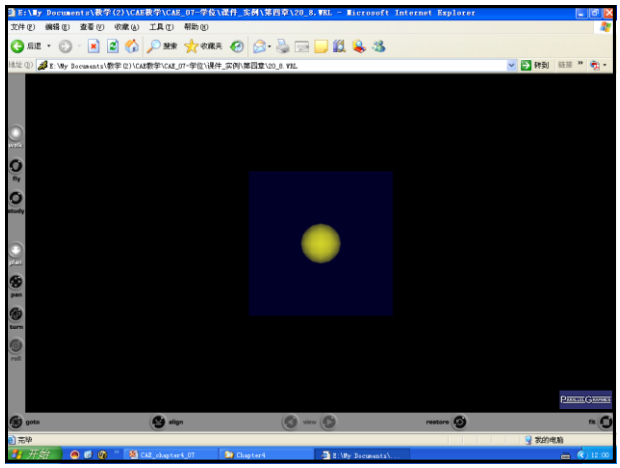
```
DEF Clock TimeSensor{
                    cycleInterval 10.0
                    loop TRUE
                            },
            DEF Btrancpath ScalarInterpolator{
                    key[ 0.0,0.5,1.0, ]
                    keyValue[0.0,1.0,0.0,]
                    }

            ]
    }
ROUTE Clock.fraction_changed TO Btrancpath.set_fraction
ROUTE Btrancpath.value_changed TO Bboxmeterial.set_transparency
```

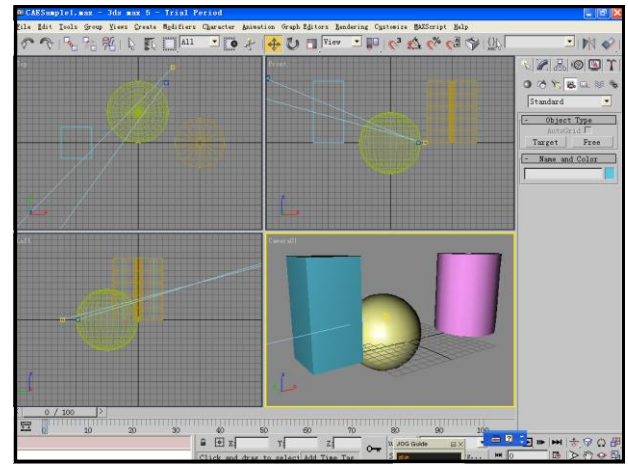
fraction_changed：输出时间传感器的一些浮点时刻

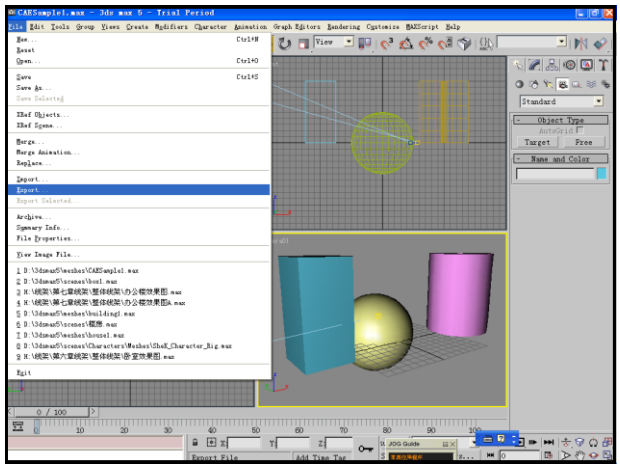
Transparency域：指定造型的透明度，

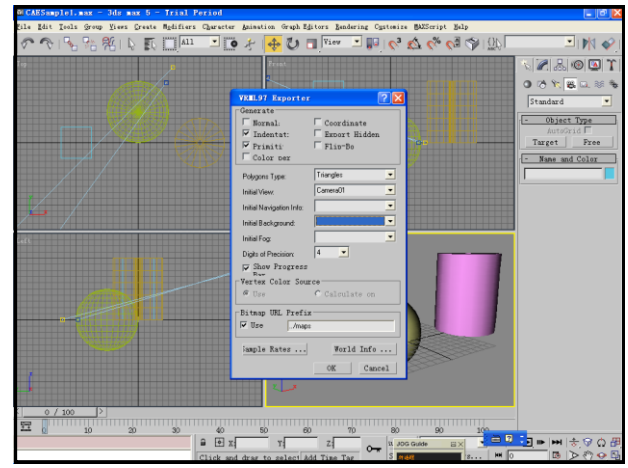


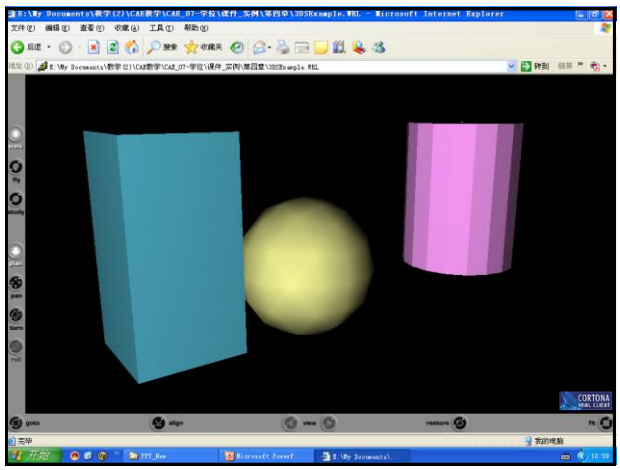


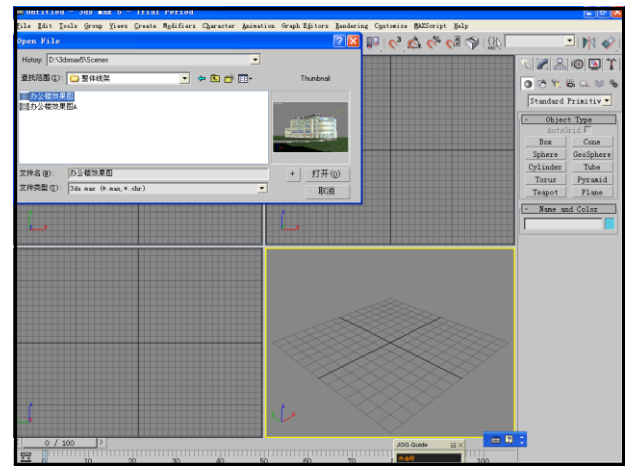


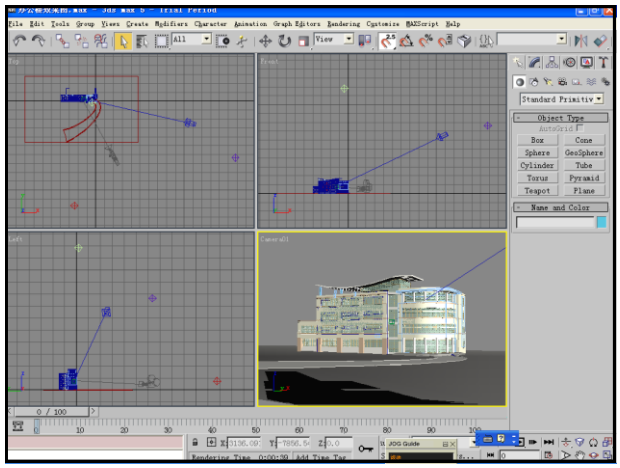


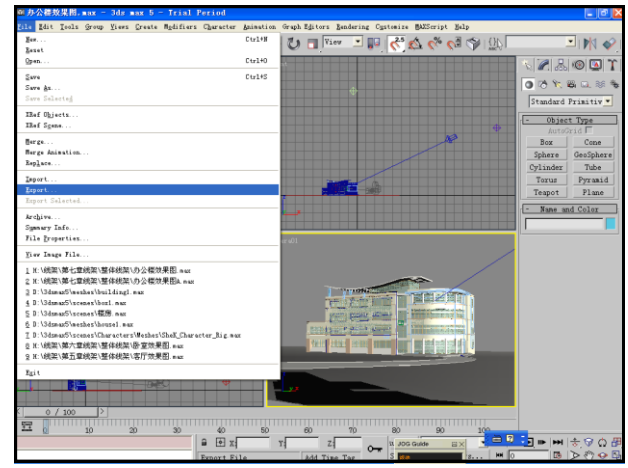


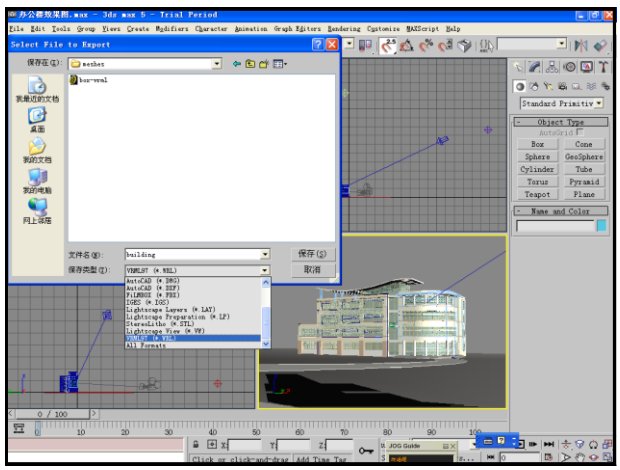


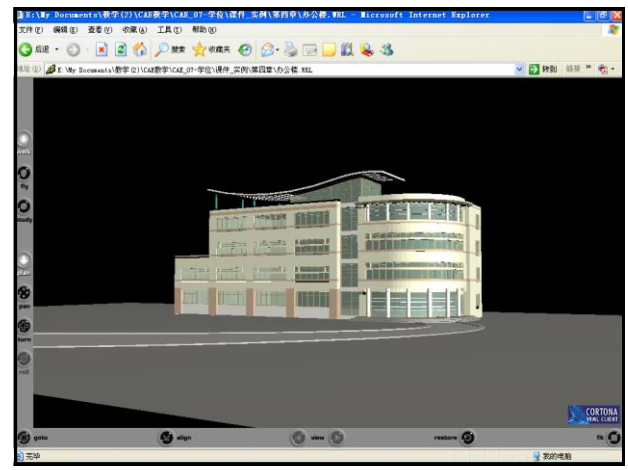






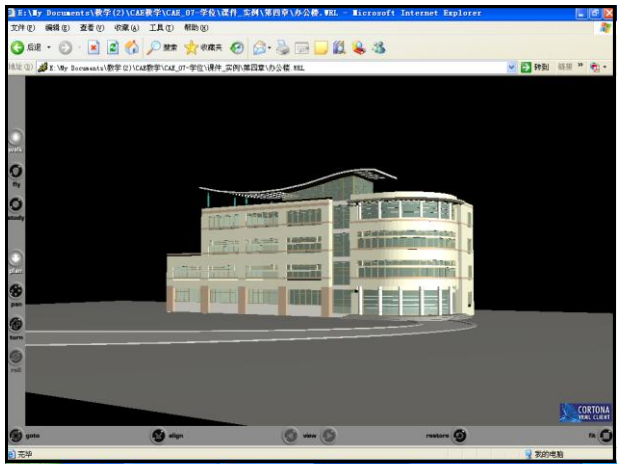






```
#VRML V2.0 utf8
    # Produced by 3D Studio MAX VRML97 exporter, Version 5, Revision 0.93
    # MAX File: 办公楼效果图.max, Date: Wed Apr 23 17:41:17 2003
    DEF Camera01 Viewpoint {
      position 2020 447.3 6219
      orientation -0.08768 -0.9959 0.02288 -0.5126
      fieldOfView 0.5656
      description "Camera01"
    }
    DEF Direct01 DirectionalLight {
      intensity 1
      color 1 0.9922 0.9529
      direction -0.8837 -0.4233 -0.1996
      on TRUE
    }
    DEF Direct01-TIMER TimeSensor { loop TRUE cycleInterval 3.333 },
    DEF Omni01 PointLight {
      intensity 0.494
      color 1 0.9922 0.9529
      location -2424 742.4 1.175e+004
      on TRUE
      radius 3.042e+004
    }
```

```
NavigationInfo { headlight FALSE }
DEF Rectangle0 Transform {
  translation -3120 1228 5.369e-005
  rotation -1 0 0 -1.571
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0.9529 0.9569 0.8745
          ambientIntensity 0.9056
          specularColor 0.0449 0.0449 0.0449
          shininess 0.0975
          transparency 0
        }
      }
      geometry DEF Rectangle0-FACES IndexedFaceSet {
        ccw TRUE
        solid TRUE
        coord DEF Rectangle0-COORD Coordinate { point [
          1781 0 -160, -226 0 -160, -226 0 160, 1781 0 160, 1781 20 -160,
```



本章学习重点

- 掌握虚拟现实、系统仿真、虚拟仿真的概念
  - 虚拟现实技术是直接把计算机合成的信息提供给人的感觉器官，生成逼真的视、听、说、触、动和嗅等感觉的虚拟环境，操作者借助必要设备可以自然方式与虚拟环境及物体进行交互，从而产生身临其境的感受和体验。
- 了解虚拟现实的意义
- 了解虚拟现实技术在土建工程中的应用
  - 虚拟设计
  - 虚拟建造
  - 虚拟防灾减灾
- 了解实现虚拟现实的软件和开发平台
- 熟悉VRML语言

提高内容参考

- 基于OpenGL开发一个能实现漫游的虚拟街道，街道两旁要有建筑、职务、行人和汽车等；
- 用VRML语言编写程序，做出一个复杂街区的虚拟现实场景；
- 用虚拟现实技术，实现钢结构吊装的虚拟施工；
- 用虚拟现实技术，实现“流沙”效果；

谢谢！

清华大学土木工程系

胡振中

huzhenzhong@tsinghua.edu.cn