数字娱乐设计系列教材

丛书主编　付志勇

三维游戏设计

Three Dimensional Game Design by Virtools Dev 4.0

付志勇　高鸣　编著

清华大学出版社

北京

内 容 简 介

本书主要面向三维游戏的设计与制作，通过一个基于 Virtools 软件的三维游戏原型开发实例详细讲解相关的程序与方法，同时也介绍了三维游戏与交互娱乐技术的相关知识与理念。本书介绍的方法和技能不仅适用于快速开发游戏原型，也可以满足制作在线的三维交互内容、三维界面设计以及三维虚拟展示设计的需要。本书是清华大学美术学院艺术设计（数字娱乐设计方向）校内第二学士学位的教材，适合数字游戏设计、交互设计、数字媒体艺术和艺术设计等专业的本科生、研究生学习，也可作为游戏设计爱好者的自学用书。

图书在版编目（CIP）数据

三维游戏设计/付志勇，高鸣编著. —北京：清华大学出版社，2008.10
（数字娱乐设计系列教材）
ISBN 978-7-302-18394-5

Ⅰ. 三…　Ⅱ. ①付…②高…　Ⅲ. 三维-游戏-软件开发-教材　Ⅳ. TP311.5

中国版本图书馆 CIP 数据核字（2008）第 123094 号

责任编辑：廿　莉
责任校对：王凤芝
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
社　总　机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张：**14　**字　数：**275 千字
版　　次：2008 年 10 月第 1 版　　**印　　次：**2008 年 10 月第 1 次印刷
印　　数：1～5000
定　　价：33.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：028002-01

前　言

随着文化创意产业的发展，数字娱乐设计正在成为一个新兴的专业方向。数字娱乐设计是以大众的娱乐和休闲方式为主要研究对象，基于数字化和网络化的平台，通过多媒体的交互手段，创造具有参与性、互动性和娱乐性的产品或环境。具体的设计内容以数字游戏设计为主，同时也与移动内容设计、网络艺术设计、数字影音设计、数字动画及周边产品设计、虚拟现实技术应用、主题娱乐公园体验设计等领域有着密切的关联。

数字娱乐设计是信息时代的媒体艺术、设计、影视、音乐与数字技术融合产生的新兴交叉学科领域，相关的教学和研究在国内还处于起步阶段。为了更好地探索该领域的人才培养模式，清华大学美术学院在2006年设立了艺术设计（数字娱乐设计方向）校内第二学士学位，同时也在信息艺术设计专业方向开展了相关的数字娱乐设计教学实践。《数字娱乐设计》系列丛书的推出正是为了满足教学实践的需要，在总结现有教学经验的基础上，进一步规范和推动数字娱乐设计教学的发展。在内容编排上，本丛书以培养复合型数字娱乐和游戏设计人才为目标，既注重培养学生的数字游戏设计创意和评价能力，同时也强调培养学生在游戏开发与制作表现方面的实践技能。

本书内容包括数字游戏的理念、三维游戏的基础知识与代表作品、三维游戏引擎的介绍、数字游戏的设计流程、三维游戏原型制作的技巧、游戏作品的评价与发布、虚拟现实和交互技术在游戏中的应用等。在教学方面偏重于对游戏设计和制作流程的讲解，即如何利用一款三维游戏引擎以及各种游戏制作素材（三维模型、二维贴图等）实现游戏的核心玩法（Core Gameplay）。本书在游戏原型制作方面选择了“达索”Virtools Dev 4.0软件，它具有“可视化编程”以及“实时运行调试”的功能，使学习者能够回避复杂的游戏程序问题，而将更多的精力放在三维游戏的关卡设计与实现上。

本书强调知识点的连贯和循序渐进，主要通过一个三维游戏原型的开发实例来讲解三维游戏设计与制作的知识和技能，以游戏研发团队中“关卡设计师”的工作为主要侧重，便于学习者充分理解和把握三维游戏设计的流程和方法，并为今后使用其他游戏引擎制作三维游戏奠定基础。本书介绍的理论方法和制作技能不仅适用于快速开发游戏原型，也可以满足制作在线的三维交互

内容、三维界面设计以及三维虚拟展示设计的需要。本书适合数字游戏设计、数字媒体艺术和艺术设计各专业的本科生、研究生学习，也可作为游戏设计爱好者的自学用书。

作者于清华园

2008 年 5 月

目 录

数字娱乐设计系列教材
Textbook Series of Digital Entertainment Design

数字娱乐设计史
数字游戏策划
游戏美术设计
游戏动画设计
数字游戏设计基础
三维游戏设计
数字游戏编程基础

ISBN 978-7-302-18394-5
9 787302 183945 >
定价：33.00元(含光盘)

第 1 章　三维游戏设计概论

三维游戏在视觉表现上具有深度感，能够在数字化空间中营造出逼真的幻想世界，产生电影感的视觉效果；同时三维的角色行为和场景表现更符合游戏者真实的认知习惯，结合更自然的交互方式，能够创造出更为引人入胜的游戏。本章主要介绍三维游戏的基本知识和理念、三维游戏的类型和范例、三维游戏的基本构成要素，并对三维游戏设计的程序和方法进行初步的描述。

1.1　三维游戏的基础知识

1.1.1　数字游戏的概念与特性

游戏设计师安德鲁·罗琳斯（Andrew Rollings）认为，游戏是一种参与或交互的娱乐形式，它具有参与（Participation）、互动（Interactive）、娱乐（Entertainment）的特性。游戏是以娱乐为主的行为，也是过程性的行为，游戏的可玩性（Gameplay）是这个过程的重点。游戏可玩性源于游戏规则，它体现了游戏的目标性、变化性和竞争性，是游戏规则在游戏进程中的具体表现。

数字游戏包括了三个主要的部分，即核心机制（Core Mechanics）、交互性（Interactivity）、讲故事和叙事（Storytelling and Narrative），如图 1.1 所示。核心机制可以理解为游戏的玩法和规则，它决定了游戏的可玩性。大多数游戏具有复杂的关卡设计和游戏规则，但是也有很多类似俄罗斯方块（Tetris）的游戏，提供简单但是持久的快乐。叙事的部分为游戏者提供了一个背景故事，引导游戏者在悬念和历险中体验到游戏的世界观与内涵。交互性是人与机器之间的沟通和反馈，通过软件和硬件的顺畅配合来创造良好的可用性和可玩性。

数字游戏设计不仅提供可供欣赏的画面和可玩的过程与体验，也需要构建一个可以展开娱乐的交互平台，以及依照规则行事的操控方式。游戏设计是一种需要协作的艺术形式，既含有艺术要素，也含有功能因素。游戏的艺术性一方面体现在总体构思和美术设计中，同时也体现在游戏规则的定义和提炼过程中，游戏设计既需要创造力，也需要精心的规划。

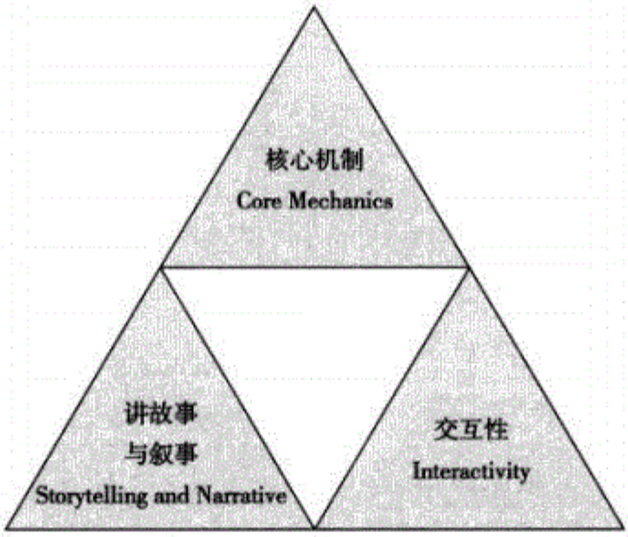


图 1.1　游戏设计的三个领域

信息时代文化、艺术与科学的交融和互动，给人们认识数字游戏产品和设计提供了新的维度。数字游戏是文化、艺术与科技交叉与融合的产物。数字游戏的设计主要包含策划、艺术和制作几个方面。策划包含游戏心理研究、游戏脚本编写、游戏运营等，它反映的是当代文化与商业手段的融合。艺术不仅包含游戏角色、道具与场景的设定、游戏动画、音频等多媒体表现方面，也包括角色性格塑造、游戏氛围渲染等情感化的方面，它是多媒体形式与认知体验的整合。制作则包含游戏编程、游戏关卡制作、游戏引擎和交互平台的运用，它是现代科技前沿的综合体现。

1.1.2　三维游戏的概念

三维游戏（Three Dimensional Game）是相对于二维游戏（Two Dimensional Game）而言的。三维游戏因其采用了立体空间的概念，其中物体的位置由三个坐标决定，所以更显真实，而且对空间操作的随意性也较强，更容易吸引游戏者。三维游戏中，角色在立体空间中进行运动，可以实现角色的前进、后退、旋转、跳跃等行为，动作更真实和更连贯。角色的运动也有了更广阔的场景空间，游戏者的视角可以平滑地转换，可以从各种视角观察场景，如图 1.2 所示。

图 1.2　《文明 4》和《Doom 3》游戏截图

三维游戏再现的是更为真实的世界，人类生活中的认知习惯可以复制在虚拟空间中，因此也可以采用更自然的人机交互控制形式，如虚拟现实、增强现实等交互技术方法，将游戏者带入到一个更逼真的虚拟空间或场所。在三维空间中，传统游戏的交互性和电影的视觉冲击与叙事结合在一起，能够营造一个虚拟的幻想世界，来满足人类的精神需求，从而创造出更持久的游戏体验。

1.1.3　二维与三维游戏的比较

早期的数字游戏基本都是二维游戏，所有图形元素，如角色、装备、地图、建筑等，都是由平面元素构成的，制作较为容易。《马里奥》（Mario）系列、《创世纪》（Ultima）系列、《暗黑》（Diablo）、《盟军敢死队》（Commandos）、《星际争霸》（Starcraft）、《冰风谷 2》（Ice Wind）、《博德之门》（Baldur's Gate）等都是比较经典的二维游戏，如图 1.3 所示。二维游戏易于操控、全局性好。早期的二维游戏受到技术的限制，角色、场景画面以像素画为主，色彩也非常有限。随着图形显示技术和游戏引擎的进步，二维游戏也具有了非常丰富的视觉效果。二维游戏中的动画是以帧的形式预先制作的，然后再调入游戏中，因此会影响角色动作的连续性。

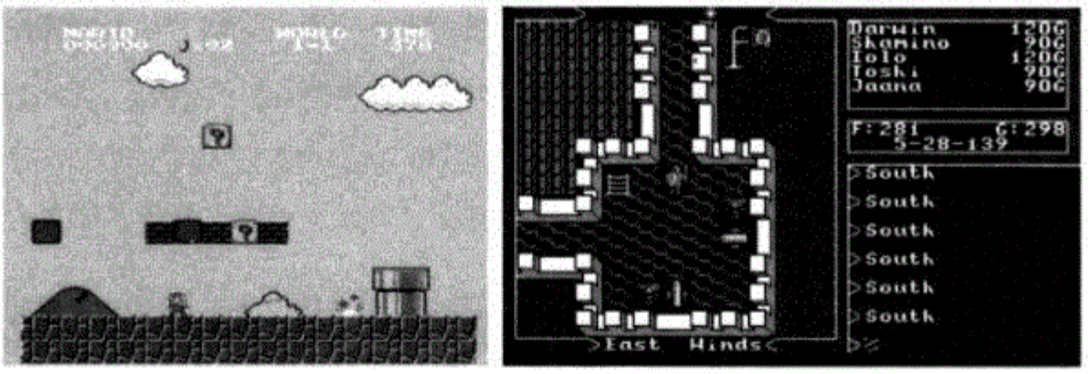


图 1.3　《超级马里奥兄弟》和《创世纪》游戏界面

在二维中也可以创造出空间效果，如可以通过位置、遮挡、大小比例变化、疏密、色彩、虚实、阴影等方法来表现空间感。例如，1979 年的角色扮演游戏《阿卡拉贝斯》（Akalabeth）采用了透视原理和角色图像的大小变化，利用空间线条绘制，给游戏者带来立体环境的感受，其续作就是上文提到的《创世纪》（Ultima）系列，此外，“Star Wars Arcade”游戏也采用了类似的视觉表现手法，如图 1.4 所示。

在二维界面中，依照成角透视或正等测的原理能够创造出模拟的立体效果，有时称作 2.5D 界面。具有 45 度视角的 2.5D 是在二维与三维界面之间的一个特殊维度，它可以提供给游戏者带有固定角度的宏观视图，使游戏者能快捷地了解界面中各个角落发生的事情，因此大部分的策略游戏、网络游戏使用 2.5D 的表现形式。但相对于

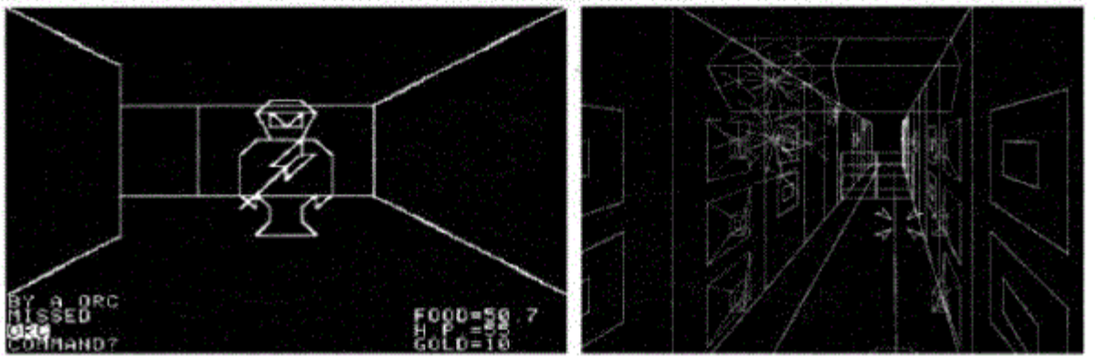


图 1.4 《阿卡拉贝斯》与“Star Wars Arcade”游戏截图

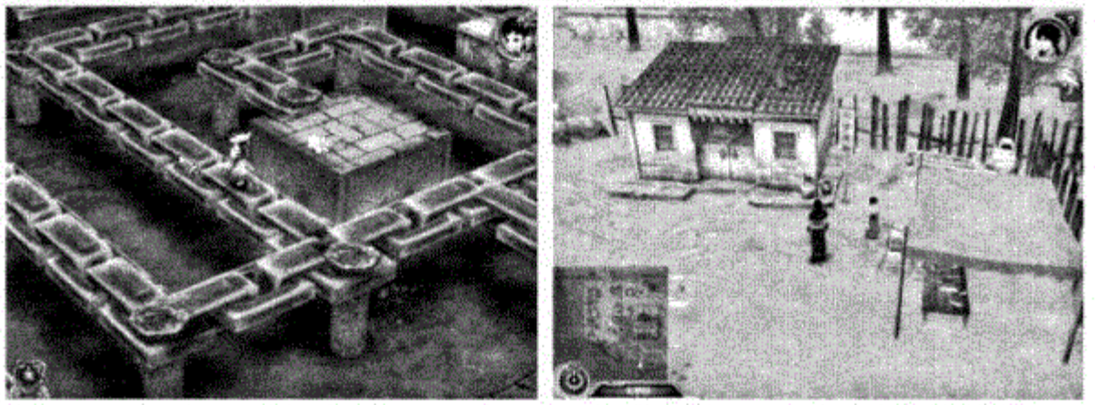


图 1.5 《仙剑奇侠传》系列游戏截图

三维来说，2.5D 界面没有更多维度的交互操作，在一定程度上削弱了游戏者的沉浸感。如图 1.5 所示的《仙剑奇侠传》系列游戏截图，显示了介乎二维和三维之间的宏观视图效果。

三维游戏具有真实的空间感、全视角的角色外观、更丰富的动作、更快的游戏节奏等特点。游戏中的动作可以进行连续的实时计算，因此运用三维技术可以为动作类、冒险类游戏营造出更好的现场气氛。三维方式给游戏者提供了更大的空间，如图 1.6 所示，用户可以在三维游戏场景中自由地活动，这给用户带来更加强烈的现实体验和沉浸感。同时，现实生活中的方向感、路径记忆等也可以应用于操作三维界面的过程中。三维游戏对显卡的运算速度和内存容量比二维游戏有更高的要求，三维游戏中，角色和场景是由多边形构成的，大量多边形的绘制会降低游戏速度，但是多边形不足时，近观三维角色时会感觉不真实。

在视觉表现上，也有将三维效果转变为二维的卡通渲染（Cartoon Rendering /Toon Shading/Cel Shading）风格，它是通过在三维模型上贴上卡通的纹理贴图，使表现形式变为二维效果，但保留了三维的运动与操控方式。如图 1.7 所示，任天堂 Game Cube 上的《塞尔达传说：风之杖》（The Legend of Zelda：The Wind Waker）游戏采用了卡通渲染风格，具有明显的阴影、较少的高光以及物体边缘的轮廓。

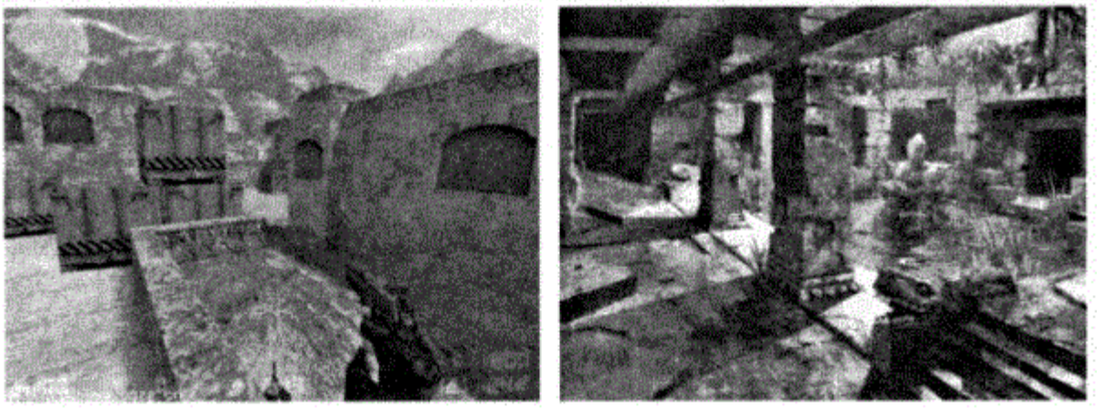


图 1.6　《反恐精英》与《孤岛惊魂 2》游戏界面

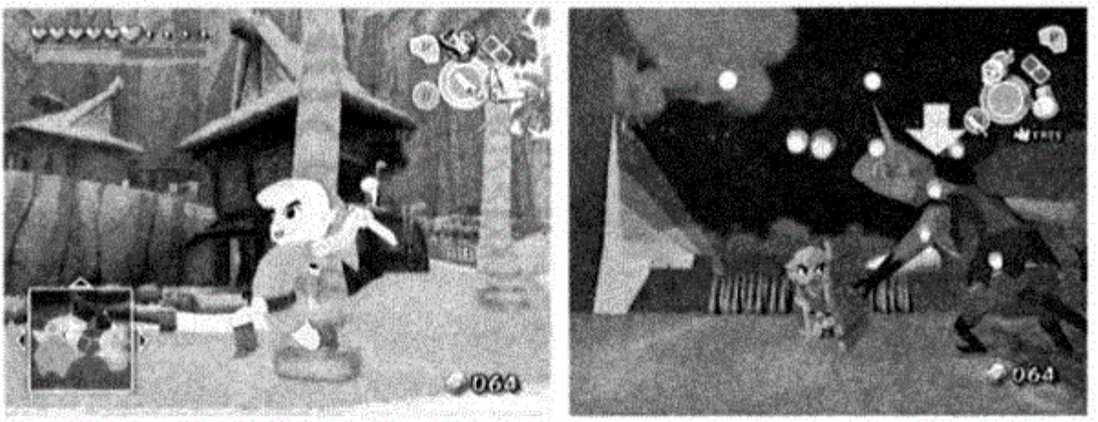


图 1.7　《塞尔达传说：风之杖》游戏截图

1.1.4　游戏中的时空元素

时间和空间元素在三维游戏中有着非常重要的作用。在数字世界中，时间与空间是两个密不可分的概念。空间是一种具有无形、无相认知特征的客观存在形式。在中国传统的时间观念中，时间与空间构成了宇宙的概念。例如，《文子・自然》篇中曾记载："老子曰：……往古来今谓之宙，四方上下谓之宇。"西方的传统科学中，最重要的空间概念就是欧几里得的三维空间，现代物理学则在三维空间的基础上增加了时间维度形成了四维概念。爱因斯坦认为，每一瞬间三维空间中的所有实物再占有一定的位置就是四维的。

大部分的三维游戏作品都在模拟现实空间，技术上大量使用仿真渲染。但也有诸多的另类空间概念可以运用在游戏设计中，如埃舍尔的非现实的矛盾空间、海市蜃楼一般超越自然力的空间、古代山水画的散点透视空间，以及其他感情、记忆交集的抽象空间等。利用三维游戏的虚拟现实特性，可以设计出更多超越自然阻力，现实中不能实现的事物以及状态，创造出超乎意料的作品。空间和时间的概念常被用作游戏的元素，通过空间变换、时间延迟、暂停或倒流等方式来创造新的游戏体验。

动作类游戏中经常会运用时间限制的方式，在游戏进行中通过一个倒数的计时器

来显示时间，当计时器为零时，将会导致任务失败或角色死亡。时间限制通常有三种方法：第一种是回合制的时间控制，在规定时间内完成任务；第二种是运用时间限制作为重大灾难事件的倒数计时器；第三种是运用时间计时器的方式限制某些物品的作用时间。

游戏中的时间也可以被人为地操控，如《大神》游戏利用神笔在天空画太阳和月亮来让时间改变，《红侠乔伊》游戏中的主角是利用自身加速来放慢敌人时间，《鬼泣3》及《恶魔城》游戏中使用了时间停止技能。微软的 XBOX 游戏《霹雳酷炫猫：时空清洗者》（Blinx：The Time Sweeper）也是以时间操控为主题。游戏者扮演的主角是时间工厂的员工，负责清除“游戏时间”，能够用类似吸尘器的“时间抽取机”把时间抽出来。游戏中会借助视频播放的形式，通过播放、暂停、快放等方式产生出新的游戏玩法。其续作《霹雳酷炫猫 2：时空大师》（Blinx 2：Masters of Time & Space）游戏中，主角拥有可以随意控制游戏动作加快或倒转的能力，并宣称“将是世界上第一款 5D 游戏，能够让玩家控制时间和空间”。《波斯王子：武者之心》（Prince of Persia：Warrior Within）游戏中“时间”技能同第一人称射击游戏相结合，游戏者能体验到“时间变慢”和“时间倒流”等技能。又如《波斯王子：时之砂》（Prince of Persia：The Sands of Time）游戏则通过获得“时之砂”神秘的力量，时光倒流、时间延迟、时间停止、时间加速、预知未来等，来拥有自由扭转时间的能力，如图 1.8 所示。

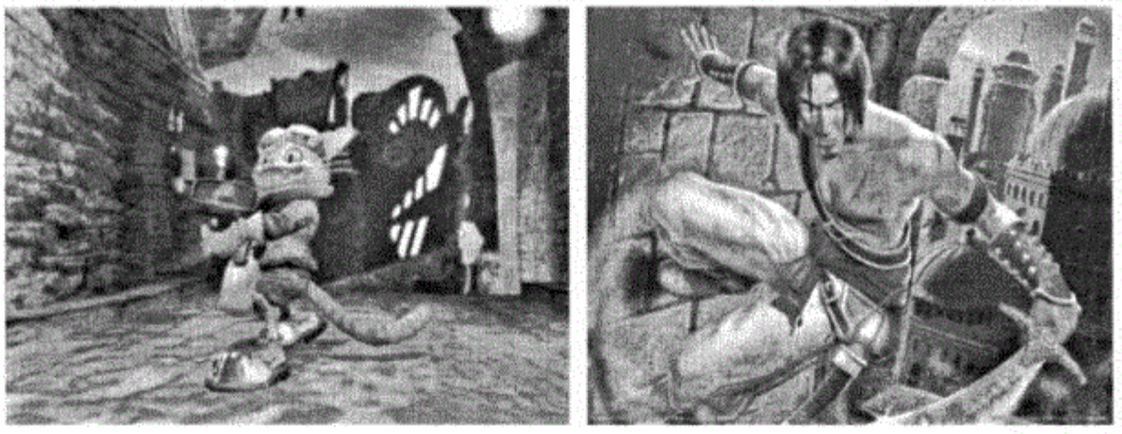


图 1.8 《霹雳酷炫猫：时空清洗者》与《波斯王子：时之砂》游戏截图

2005 年的科幻第一人称射击游戏《时空飞梭》（Timeshift），如图 1.9 所示。其游戏内容与“时间”密切相关，游戏引入了“四维空间”的概念。游戏中具有“时空转换”的功能，游戏者能够随心所欲地改变时间的进程。游戏中配备了一套特殊的装备，可以使主角免受时间进程变化的损伤。《时空飞梭》游戏改变了传统的第一人称射击游戏方式，让游戏者使用“时间元素”来展开游戏。游戏中“时间”技能侧重点在于如何生存、逃生及战斗。利用“时光”技能扭转当前不利于游戏者的境况，譬如使用“时间暂停”，瞬间夺走敌人的枪支弹药，恢复时间后迅速消灭敌人或逃生；

也可利用“时间迟缓”等技能分散敌人的注意力。“时间”技能的运用并不会影响武器的威力或使用，“时间”技能真正影响到的是敌人的行为。

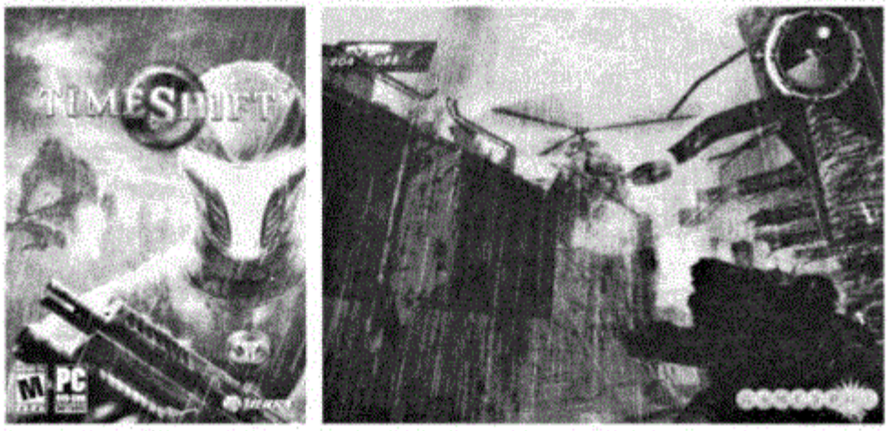


图 1.9　《时空飞梭》游戏截图

游戏当中，空间元素的应用有多种方式，如通过瞬时移动，角色可以迅速转移到游戏画面中其他位置，这在早期的街机游戏中比较常见。2007 年 Valve 公司出品的《洞穴》（Portal）游戏是一款以“HL2”引擎制作的第一人称射击游戏，在游戏中利用空间传送方式打开空间入口，完成各种谜题。游戏中游戏者拥有的道具有类似《半条命 2》中的重力枪与两个传送门，利用传送门，游戏者可以制造两个超时空通道，并在两个传送门之间穿行。两个传送门分别为橙色和蓝色，并没有功能上的区别，游戏者可以随便进入其中一个就可以从另外一个出来。其非常规的空间感，对游戏者们的逻辑思维能力是一个挑战，如图 1.10 所示。

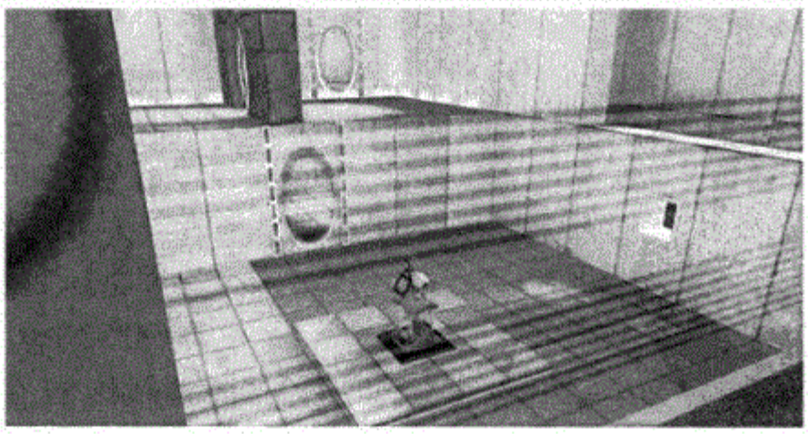


图 1.10　《洞穴》游戏截图

1.2　三维游戏的类型与范例

游戏开发是一个创造性的事业，关于游戏的类型已有很多的分类方式，如动作类、冒险类、角色扮演类、迷宫与解谜类、模拟类、体育类、策略类等，但仍会有交

叉的分类和不全面之处，这就是创造性产品的特点，创造性产品的第一规则就是没有规则。

1.2.1 动作类游戏

动作类游戏（ACT）包括几种类型，通常以游戏者替身的视角进行游戏，这一类游戏最常见的是第一人称射击游戏（FPS）。在三维游戏中，代表作品有《虚幻》（Unreal）系列、《半条命》（Half Life）系列、《毁灭战士》（Doom）系列（如图 1.11 所示）、《雷神之锤》（Quake）等。动作游戏也有多人在线的版本，那里的敌对方是由真实的人来控制的，而不是计算机。在 FPS 游戏中，需要快速的反应、良好的手眼协调能力和对武器能力的精通。也有一些游戏是第三人称视角的，游戏者能够看到自己的游戏角色或替身，也可以看到替身所处的虚拟世界的其他地方。

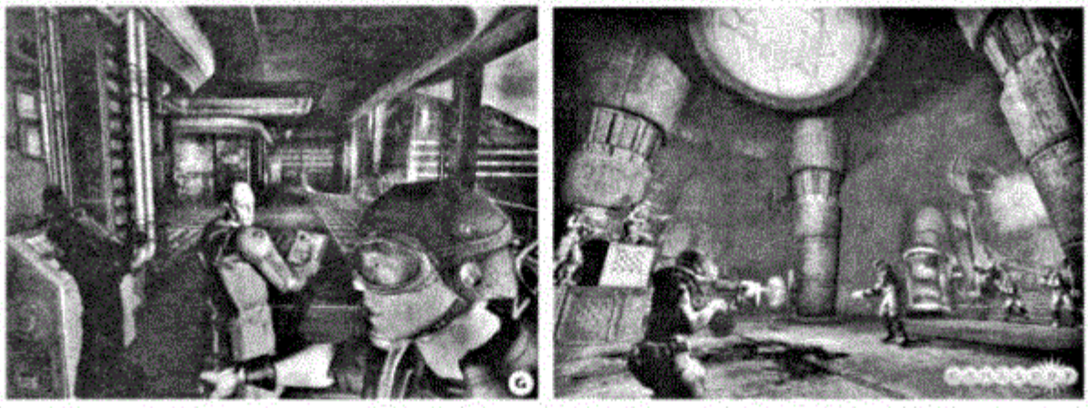


图 1.11 《毁灭战士》游戏截图

1.2.2 冒险类游戏

冒险类游戏（AVG）主要是关于探险的。游戏角色会去寻求、发现物品以及解决难题。早期的冒险游戏是基于文字的，需要游戏者输入运动命令。进入一个新的地方或房间，则需要给出一个所处位置简短的描述，例如“你在一个有着扭曲过道的迷宫中，周围一切都非常相似”。最好的冒险游戏就像交互书籍或故事，你作为游戏者决定下一步发生什么，以及在何种程度下发生。一般来说，指令与解谜是冒险游戏的两大要素。文字冒险游戏逐渐进化，开始通过静态图像来给游戏者关于环境的概念。随着三维技术的发展，游戏者可以处在第一或第三人称视角的环境中来体验游戏。冒险游戏需要依赖故事，并且通常是线性的，游戏者需要逐个任务地寻找解决办法，随着故事的发展，游戏者会逐渐掌握预测游戏进程的能力，其成功取决于预测以及做出最好选择的能力。

单纯的冒险游戏需要游戏者在剧情进行到关键时刻进行选择，以决定故事的走向。早期的冒险类游戏主要是通过文字的叙述以及图片的展示来进行的，如《亚特兰

蒂斯》（Atlantis）系列、《猴岛小英雄》（Monkey Island）系列等。此外，还有包含了动作元素的冒险游戏（Action Adventure），如 CAPCOM 的《生化危机》（Biohazard）系列、PS2 的《零》（Zero）系列、《鬼武者》系列、《古墓丽影》（Tomb Raider）系列和《波斯王子》（Prince of Persia）系列等。如图 1.12 所示，《古墓丽影》通过三维图形引擎，实时生成人物和场景，展示了怪石嶙峋的山洞、神秘幽深的森林和飞瀑直下的深潭，这些都发挥了三维技术的优势。360 度全方位地观察视角和强烈的方位感与纵深感的音效，创造了类似交互式电影的效果，烘托出惊险恐怖、扣人心弦的游戏故事情节。

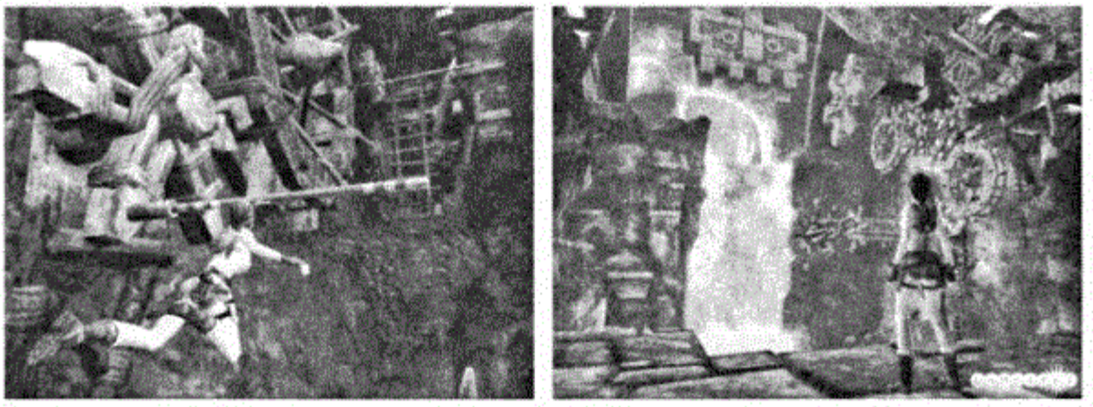


图 1.12　《古墓丽影》游戏截图

1.2.3　角色扮演类游戏

角色扮演游戏（RPG）的流行可能是受到了儿时游戏的影响。六七岁的小孩经常会受到故事书和玩具的启发，设想或从事一些激动人心的冒险。和战略类游戏一样，这类游戏也起源于桌面游戏，如《龙与地下城》（Dungeons & Dragons），这些游戏移入到计算机后，计算机承担了游戏者大量的数据处理任务。角色扮演类游戏的核心在于扮演与培养，通常会向游戏者提供诸多的世界观设定以及故事剧本。这类游戏通常是科幻的和想象的题材，在三维平台上，能够更好地在幻想的世界表现出建筑、妖怪和生物。游戏者负责开发游戏角色的技能、体型外观、忠诚度和其他属性。此外，也有一些历史题材的角色扮演游戏。角色扮演游戏中如果加入大量的动作成分，则称为动作角色扮演游戏（ARPG），如果加入战略成分，则称为战略角色扮演游戏（SRPG）。

日本的 RPG 游戏，在 20 世纪 90 年代有诸多引起强烈反响的系列作品，如《最终幻想》、《勇者斗恶龙》、《星之海洋》、《火焰之纹章》、《萨尔达传说》、FALCOM 公司的《英雄传说》等。1997 年，Bullfrog 制作的《地下城守护者》具有独特的即时策略和动作、巧妙的关卡设计和与众不同的三维立体界面。《地下城守护者 2》（Dungeon Keeper 2）充满了魔法和怪物，并在游戏的奇幻世界中展示了别具一格的画面，如图 1.13 所示。

图 1.13 《地下城守护者 2》游戏截图

多人在线角色扮演游戏（Massive Muti-player Online RPG，MMORPG），如《魔兽世界》、《传奇》、《天堂》等大多数网络游戏，为众多的用户创造了可以互相交流的虚拟世界，游戏者扮演不同特点的角色来体验生活。游戏本身是持续发展的，游戏者通过即时信息方式互相沟通。在这类游戏中，经验值和等级通常非常重要，需要投入大量的时间和精力来达到更高的级别，也才能体会到游戏带来的快感。

早期的多人在线游戏一般都是基于文本的，比如著名的《泥巴》（Multi-Player Under Dungeon，MUD）。直到 1997 年，电子艺界（EA）推出了基于图形界面的《网络创世纪》（Ultima Online），这款游戏的出现极大地影响了多人在线游戏市场。在 1999 年，索尼在线娱乐公司发布了《无尽的任务》（Everquest），这是一款全 3D 在线角色扮演游戏。2002 年由神话娱乐公司发布的《亚瑟王宫的阴影》也是运作得很成功的多人在线游戏。最近流行的《模拟人生在线》以及其他一些新兴的虚拟社区，如“There. com”和《第二人生》（Second Life）等，也都拥有一大批新的用户群，如图 1.14 所示。

图 1.14 《第二人生》游戏截图

1.2.4 解谜类游戏

迷宫游戏和解谜游戏（Maze and Puzzle Game）有些相近。在迷宫游戏中，游戏者需要在一个迷宫中探索出路，路线是由墙和障碍物限定出来的。早期的迷宫游戏是二

维顶视图的形式，最近则演变为近似于三维冒险和第一人称射击的游戏。解谜游戏遇到的不是越过物理障碍和探路，而是需要解决的难题。解谜游戏有时会要求游戏者用更间接的解决问题方式，只有完成一系列的操作后，才能触发进一步的行动。许多解谜游戏利用直接解决模式，谜题以视觉方式呈现，游戏者需要处理屏幕上的图标或以正确的流程进行操控来解决问题。最好的解谜游戏是通过逻辑推理来解决问题的，以试错的方式解谜容易让游戏者感觉乏味。

《神秘岛》（Myst）是 Cyan 公司在 1993 年推出的解谜类游戏，有时也被列入冒险解谜类型。有着奇幻的游戏设定、逼真的视觉表现以及环状全场景画面，如图 1.15 所示。1997 年梦工厂出品的《黏土世界》（Neverhood）也是一款富创意和可玩性的游戏，采用了独创的黏土动画、具有幽默曲折的剧情、个性鲜明的人物、设计精巧的谜题。《犯罪现场调查 4：确凿的证据》（CSI 4：Hard Evidence）的游戏者在解谜时需要使用现实的调查手法和高科技设备、综合运用各种现代化侦察技巧，破解隐藏在犯罪现场的蛛丝马迹，才能最终找出隐藏的真相，如图 1.15 所示。

图 1.15　《神秘岛》与《犯罪现场调查》游戏截图

1.2.5　模拟类游戏

模拟类游戏（Simulation Game）的目标是重建一个尽可能真实的场景。衡量模拟精确性的标准就是逼真度。大部分模拟游戏非常强调游戏的视觉外观、声音和物理学方面。这类游戏最大化地映射了真实的探险体验，如图 1.16 所示的《极品飞车》游戏、如图 1.17 所示的微软《模拟飞行》和《模拟火车》等游戏。此类游戏强调游戏环境的总体沉浸性，要使游戏者能够感觉处于其中，如同自己正在驾驶飞机或火车。模拟类游戏通常需要特殊的输入设备和控制器，如飞行操纵杆和方向舵踏板，或者建立一个真实的模拟舱环境来增强沉浸感。经典的作品如《模拟城市》（Sim City）、《模拟人生》（The Sims）系列游戏，如图 1.18 所示。但是因为模拟中也包含着经营的成分，有时《模拟城市》也会被当成策略模拟游戏。其他代表性的游戏还有《捍卫雄鹰 4.0》（Falcon 4）、《格兰披治传奇》（Grand Prix Legends，1998）等。

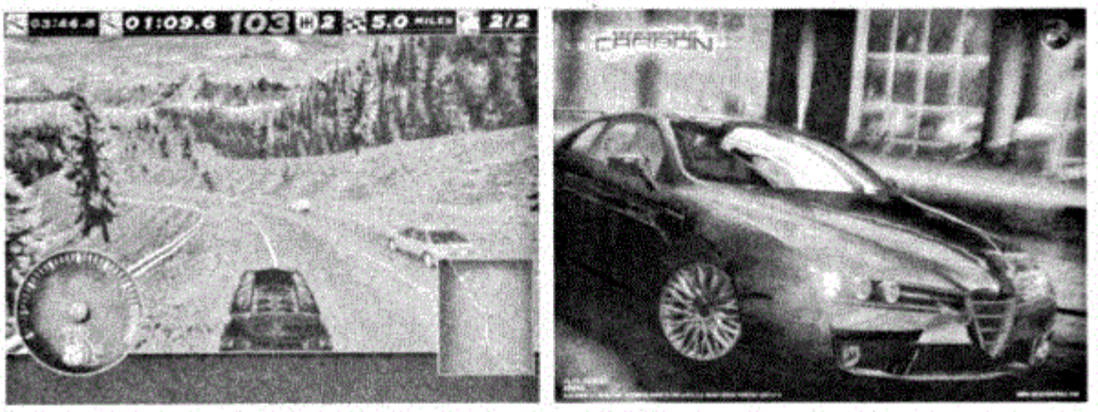


图 1.16 《极品飞车》游戏截图

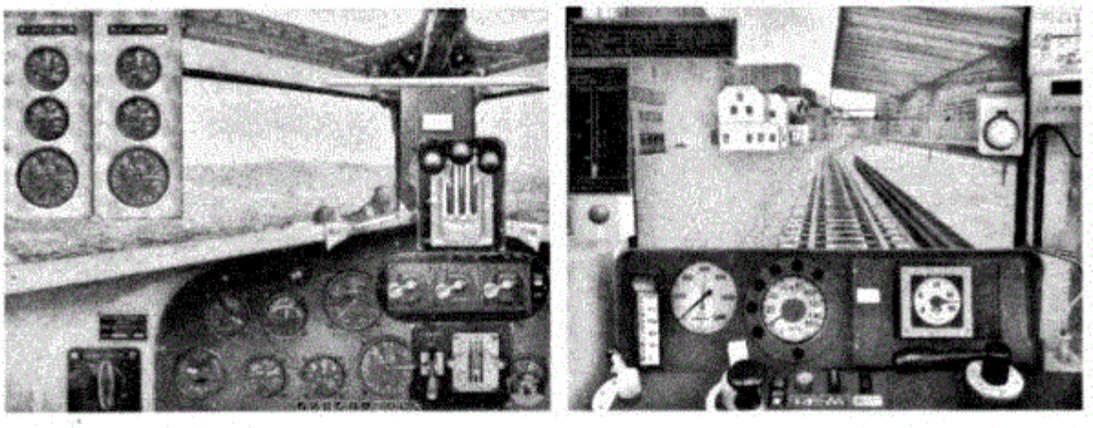


图 1.17 《模拟飞行》与《模拟火车》游戏截图

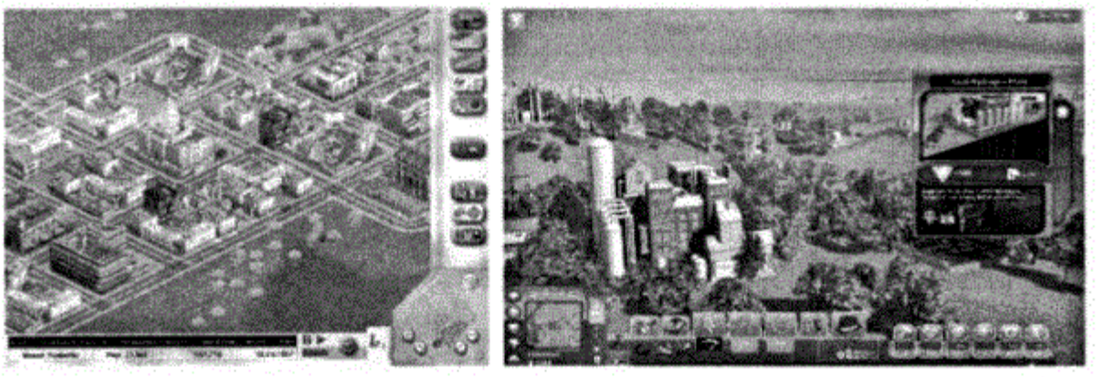


图 1.18 《模拟城市》游戏截图

1.2.6 体育类游戏

体育游戏（Sports Game，SPG）的目标就是尽可能精确地再现比赛。游戏者可以参加各个级别的体育比赛，或通过逼真的三维环境来观看竞赛。不像动作类战争游戏或驾驶模拟游戏，体育类游戏都有经理人或赛季角逐的设计。当然，游戏者也可以选择教练、领队的角色，可以像职业棒球协会那样挑选、交换和推荐新的队员。在现代的体育模拟游戏中，还可以管理预算和安排年赛的时间表，以及在不同的体育场举办

比赛或在不同的跑道上竞赛，如图 1.19 所示。

图 1.19　NBA LIVE 2008 与 FIFA 2008 游戏截图

1.2.7　策略类游戏

策略类游戏（Strategy Game）起源于已有几个世纪历史的笔纸类（Pen-and-Paper）桌面游戏，例如战争游戏。随着计算机的引入，计算机界面和随机生成器取代了传统的查找地图和扔骰子方式。卡片标记或者模压军事模型的桌面战场（或者沙盘战场）也被搬进了计算机。早期的桌面游戏的玩法通常是轮流做出选择和对部队发出命令，然后扔骰子来测定命令的结果，随后游戏者要根据结果来修改战场，并要观察新的战场形态，策划下一步的行动，游戏随之反复延续。

计算机策略游戏将实时（Real Time）的概念带到了游戏战争的前线，计算机决定移动的结果，并进一步构筑相应的战场，由此诞生了即时战略游戏（Real-Time Strategy，RTS），它会按照时间进度来表现行动。有时计算机也会压缩时间进度，有时候会按照实时进行，使游戏中的一分钟行动等于现实生活中的一分钟。三维的战略游戏使游戏者可以从各种角度和透视来观察战场，以便谋划下一步的行动。这类游戏多半是以经营为主要的游戏目标，有些游戏会加入一些战斗成分来强化游戏的耐玩度，像《恺撒 4》（Caesar）以及《文明》（Civilization）等，如图 1.20 所示。

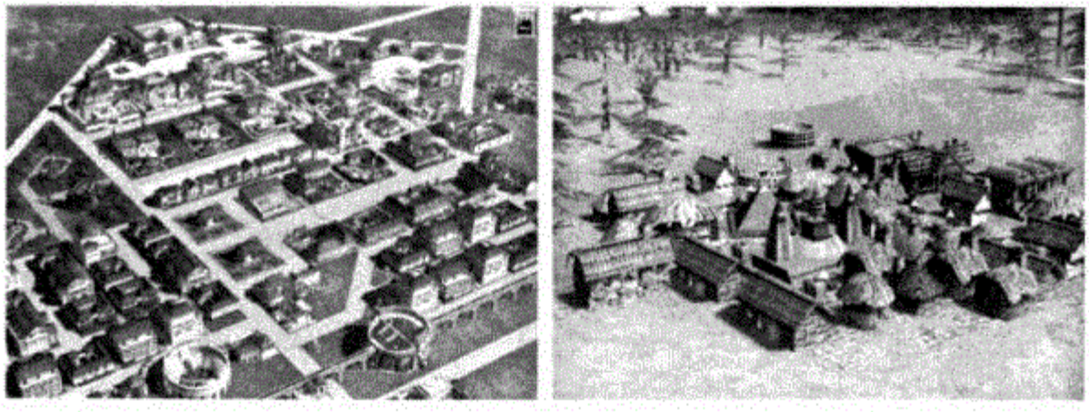


图 1.20　《恺撒 4》与《文明》游戏截图

战略游戏和策略游戏有很多的相似之处，像《三国志》这类的战略游戏中也包含了策略游戏中的经营成分。战略游戏（WG）又分为两个不同的模式，一是实时的战略游戏，主要是表现出游戏进行的持续性，游戏者需要反应快速，不太容易做深层次的思考；另一个是回合制，双方交替进攻，使游戏者可以去思考其作战策略，两者的结合产生了实时回合混合制的游戏，如《英仑霸主》游戏。

1.3 三维游戏的基本构成要素

尽管三维游戏有诸多的类型，但是从基本的结构看，三维游戏主要包括引擎、脚本、图形用户界面、模型、材质、音频、支撑结构等几个构成要素。

1.3.1 游戏引擎

游戏引擎塑造了游戏环境的大部分重要特征，如三维场景渲染、网络、图形和脚本等元素，并在后台进行整合，使之有序地工作，游戏引擎的构成要素如图 1.21 所示。1994 年，肯·西尔弗曼为 3D Realms 公司开发了 Build 引擎，在该引擎基础上开发了《毁灭公爵》游戏。1996 年以后，3Dfx 公司推出了 Voodoo 图形加速卡，各种特效技术的应用得以展现。随后，id Software 的 DOOM 3 引擎、Epic 的虚幻 3.0 引擎（Unreal Engine 3.0）等又成为三维游戏的基础引擎。

图 1.21 游戏引擎的构成要素

引擎是“用于控制所有游戏功能的主程序，从计算碰撞、物理系统和物体的相对位置，到接受游戏者的输入，以及按照正确的音量输出声音，等等”。游戏引擎考虑到对复杂游戏环境的渲染，每个游戏都使用不同的体系来塑造游戏的视觉表现，三维游戏有着丰富的材质和造型。贴图多边形渲染（Textured Polygon Rendering）是第一人称射击游戏最普遍的渲染模式，通过创造一致性的图形环境，以及根据特定的物理规律来增加环境中的物体，游戏引擎可以根据游戏进程发展出越来越多的叙事情节，使遵循游戏规则的角色能给人更真实的感觉，从而减少游戏者的怀疑并更沉浸于游戏。

通过引用物理公式，游戏可以真实地描述物体的移动、跌落和粒子运动。例如，当角色跳起的时候，系统内定的重力值将决定他能跳多高，以及他下落的速度有多快，子

弹的飞行轨迹、车辆的颠簸方式也都是由物理系统决定的。检测碰撞是物理系统的核心部分，它可以检测游戏中各物体的物理边缘。当两个三维物体撞在一起的时候，这种技术可以防止它们相互穿过，也会根据物体之间的特性确定两者的位置和相互的作用关系。经过不断的改进，游戏引擎已经发展为包含多个子系统的复杂系统，从建模、动画到光影、粒子特效，从物理系统、碰撞检测到文件管理、网络特性，还有专业的编辑工具和插件，通过引擎，游戏开发者可以直接设计游戏世界，并能塑造出更沉浸的游戏环境。

1.3.2　图形用户界面

图形用户界面（GUI）是图形和脚本的整合，它传递游戏的视觉外观并接受用户的控制输入。GUI 包含了主要的启动菜单、设定和选项菜单、对话框以及各种游戏消息系统。以《模拟人生 2》（Sim 2）为例，游戏者可以建立虚拟角色和家庭，给角色附加上特有的属性，属性的设定可以细致到头发的颜色和样式等。游戏者可以自由地转换游戏视角，从整个城区的布局、街道、房间里的家具，可以真实地模拟现实生活中的各种情景。游戏中的操作和现实生活中一样，例如，虚拟角色要操控电视，提示的选项中就会出现看电视、转换电视频道等信息。用户可以很自然地在三维画面和二维面板之间进行自由的切换，三维画面用来展示人与人、人与物的交互，而二维面板用来进行系统设置和虚拟角色的状态调整，如图 1.22 所示。

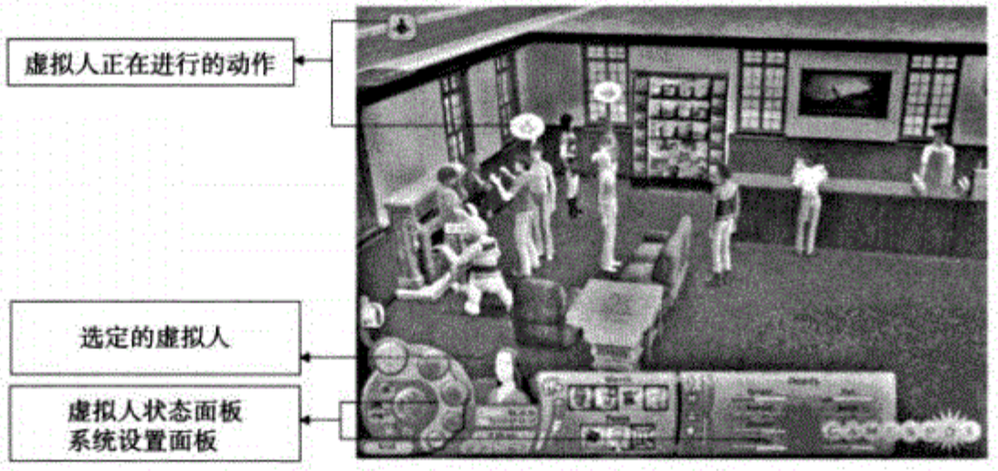


图 1.22　《模拟人生 2》游戏操作界面分析

1.3.3　模型

三维模型是三维游戏的灵魂，在游戏屏幕上的视觉元素，除了图形用户界面（GUI）的元素外，通常都是某种模型。游戏者的角色替身是模型，角色所站的地方是被称为地形（Terrain）的模型，所有的建筑、树木、街灯和交通工具等也都是模型。数字化模型有许多的表现方法，如使用线框图、三角形表面的多边形建模，使用

实体通过布尔运算来建模，使用样条曲线（Spline）和 NURBS 曲线建模等。三维游戏当中主要采用的是多边形建模技术。游戏建模中有许多成熟的技巧，在满足形体表达的情况下，结构简单、面数少的模型更符合三维游戏的要求，能够提供更好的运行效率，如图 1.23 所示。更为真实的细节部分，通常要用材质贴图来实现。

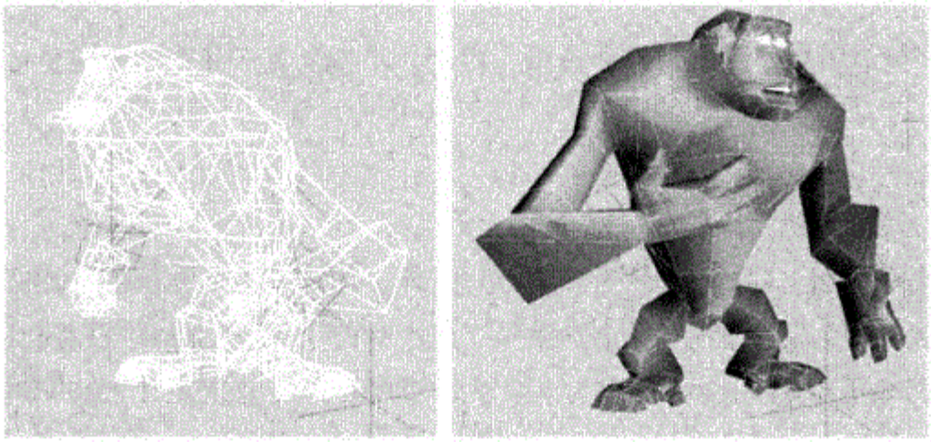


图 1.23　一个角色的三维线框图模型

三维物体都是由一系列的点构成的，这些点被称为顶点。这些顶点构成了游戏人物的骨架，这些骨架被贴上相应的材质后，就是平时所看到的游戏画面。随着 GPU（图形处理器）和 CPU（中央处理器）的不断增强，游戏开发者可以在游戏中应用更为复杂的模型，使得游戏中的人物造型更为细腻和真实。

1.3.4　材质与渲染

在三维游戏中，材质是模型渲染的一个重要部分。材质在某些特定的场合被称作皮肤（Skins），如图 1.24 所示，它定义了三维游戏中所有模型的视觉渲染外观。在三维模型中适当地使用材质，不仅可以增强模型的视觉效果，也能够帮助减少模型的复杂度，加快模型绘制的速度，从而提高游戏的性能。

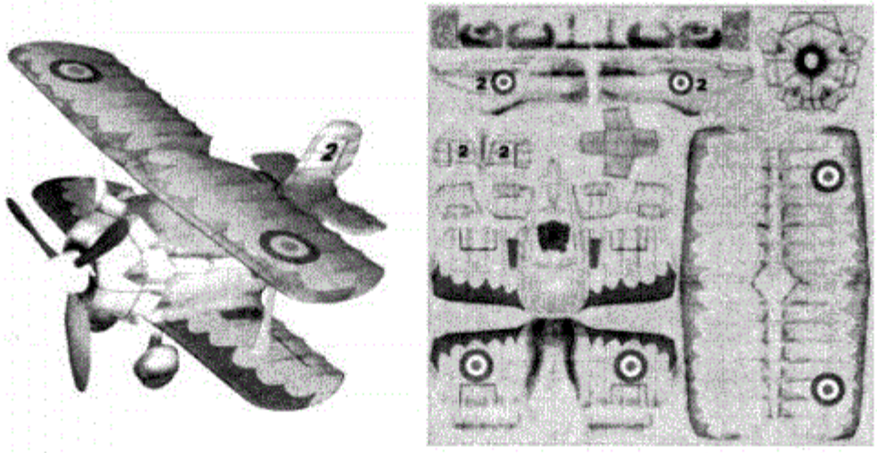


图 1.24　飞机模型与材质蒙皮

渲染模块是游戏引擎中最复杂，也是最能体现游戏效果的模块纹理映射技术，它能增强场景绘制的真实感并能提高场景的渲染速度。模型设定主要涉及阴影特征、表面特性，如色彩、材质、位置、姿势、比例等。表面特性如色彩和材质是由多种因素来决定的——基本的颜色、光泽和它的漫反射、高光、反射、亮度、透明度、折射等，并综合表现出物体的特征。

三维模型制作完毕之后，游戏制作人员会按照不同的面把材质贴图赋予模型，这相当于为骨骼蒙上皮肤，最后再通过渲染引擎实时计算模型、动画、光影、特效等效果并展示在屏幕上。渲染引擎直接决定着最终的输出质量。

受到电影和仿真的影响，计算机三维图形学力图重现真实的物理世界，尽管这种方法在如仿真训练、沉浸感游戏中是重要的，但也可能是缺乏效率的。游戏与电影不同。游戏的画面必须是实时生成的，不可能像电影一样，花大量的时间建立场景，然后由计算机花几天时间生成最终的一帧图像。因此，有时候“非真实感”的渲染，在传达交互状态方面也会很有效，同时从技术角度而言，也能够加快渲染的速度。“非真实感”绘制的另外一个优点是能在三维环境中创建一种风格和氛围，突出环境特性而不是精确的渲染，如“There. com”的游戏画面，采用的是更具卡通特点的风格，因为游戏者更关心人际交流而非画面的真实，如图 1. 25 所示。

图 1. 25　角色扮演游戏：There. com 场景的非真实感渲染

在第一人称射击这类的游戏中，真实感则是游戏制作者追求的目标。如图 1. 26 所示的《孤岛危机》（Crysis）游戏，生成逼真的丛林有助于增强游戏的沉浸感，为了生成数英里布满浓密植被的游戏场景，需要为不计其数的多面体添加光影效果。而且为了实时计算，又必须把运算量维持在处理器能承受的范围内。通过 HDR（高动态范围渲染）技术，可以大幅提升画面的表现力。HDR 的核心在于模拟人眼在光线变化时的瞳孔收缩，从而动态改变光照强度，基于 DirectX 10 技术的动态模糊会使场景更真实。

图 1.26 《孤岛危机》游戏截图

1.3.5 动画

游戏所采用的动画系统可以分为两种：骨骼动画系统和模型动画系统。前者用内置的骨骼带动物体产生运动，后者则是在模型的基础上直接进行变形。引擎把这两种动画系统预先植入游戏，动画师可以方便地为角色设计丰富的动作造型。软件开发商 Natural Motion 开发的 Euphoria 技术，能赋予游戏角色虚拟的人体结构，如神经、肌肉、骨骼及重量等，能够实现游戏角色真实肌肉运动和生物力学的快速模拟。通常在游戏中，预设的动画会使人物动作显得呆板，而动态运动合成（Dynamic Motion Synthesis）免除了动画的使用，不仅会节约内存，也会增强动作的真实性，如图 1.27 所示。

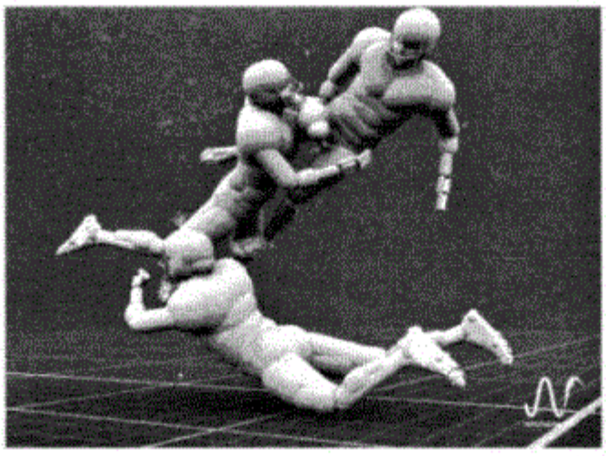


图 1.27 Euphoria 技术模拟动态运动

一般的三维数字游戏制作，需要通过动作捕捉设备来获得人类的真实动作。如果捕捉过程中某些捕捉点的信息缺失，计算机就不能很好地再现人体的动作，必须人工调节来重建缺失的部分。Organic Motion 公司的 Stage 系统不需要演员穿戴特殊的服装，它是通过 10～14 个摄像机生成的图像合成三维数据点云，将相邻的数据点三角化后形成人体的轮廓与运动，从而可以更好地记录每一位演员的动作过程，如图 1.28 所示。

分析和重现人类最熟悉同时也是最微妙的面部表情是非常具有挑战性的。心理学家保罗·艾克曼（Paul Ekman）和沃勒斯·佛莱森（Wallace V. Friesen）在研究解剖

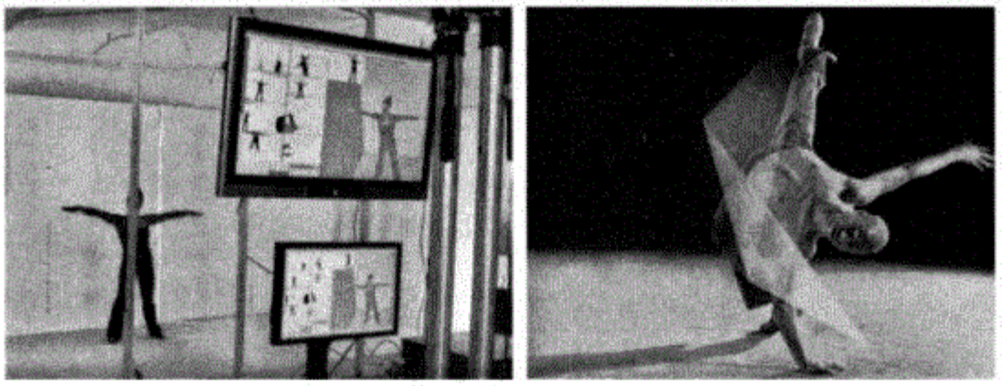


图 1.28　Stage 系统的动作捕捉模式

学和人类交流时定义了人类面部活动类型，分析概括了 60 种基本的面部活动。如图 1.29 所示，Valve Software 公司的开发人员以此为基础来构建人类的面部表情系统，增加人脸表情的多样性，赋予不同部位的面部皮肤以不同的光泽，并通过恰当的眼部活动增强人脸的逼真性。如图 1.30 所示，PS3 的动作冒险游戏《如龙见参》，也应用了世嘉（SEGA）最新的面部表情技术，增强了游戏角色的真实感。

图 1.29　游戏人物的面部表情 Faceposer staticexpressionexample. jpg

图 1.30　《如龙见参》游戏人物的面部表情

1.3.6 脚本、特效与人工智能

引擎借助脚本来处理所有的复杂任务、完成图像渲染、人工智能等设定，并使这些能力组织在一起。没有脚本很难创造出具有复杂而功能齐全的游戏。脚本可以整合引擎的不同部分，提供游戏玩法的功能，使游戏世界的规则生效。具体如计分、管理游戏者、定义游戏者和工具的行为以及控制图形用户界面等。以游戏者饮用啤酒时的脚本为例，代码主要跟踪啤酒被消费的量，并且在每喝一口后的五秒钟给予游戏者一点儿提神的能量。它给游戏者的客户端屏幕发送信息，显示其所做的行为——吮吸或是一饮而尽，同时也播放音效，发出明显的呼气声和每次喝完的满足声。

游戏程序员追求的是运算速度与丰富的功能，更精美的画面和更好的游戏体验。高水平的程序设计技巧能提高图形运算速度，可以表现出复杂、华丽的图形效果。逼真的游戏特效一直是三维游戏设计者追求的目标。

在三维游戏场景中，除了构建具有真实外观的建筑物，还需要实时模拟建筑受损和倒塌的状况。现有的方法用事先设计好的大楼残片表现倒塌的大楼，或者播放已经录制好的大楼倒塌动画。新的物理引擎将建筑物转化为数百个微小的四面体，然后根据需要模拟物体粉碎、破裂、扭曲和撕裂的效果。在模拟石块撞击时，射向建筑物的石块在碰撞点替代了那里的四面体，接着引擎根据材料的密度、强度和质量来演算碰撞点周围的四面体应如何相互碰撞、移位，制造出受损部位的连锁反应效果。所有使用数字粒子引擎表现的物体都必须遵守一系列物理规律，如物体的硬度、弹性等。数字粒子引擎还决定了物体的内力如何聚集，在超过阈值之后使物体在最脆弱的位置破裂，如图 1.31 所示。

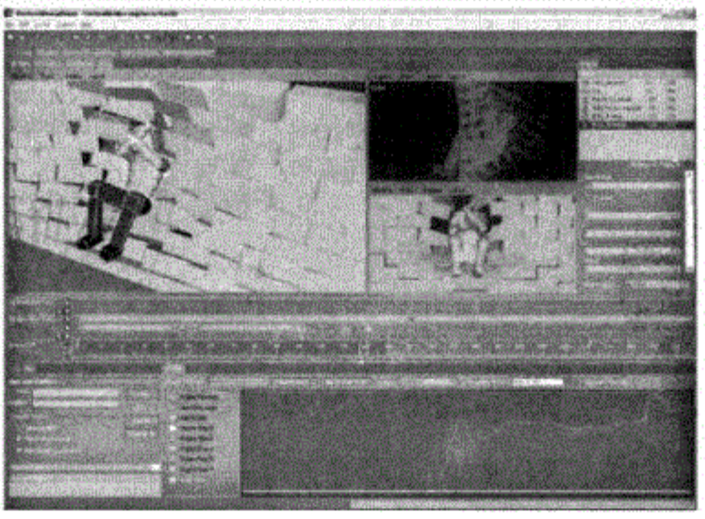


图 1.31 Euphoria 模拟的墙体倒塌特效

在水的表现方面，数学方法已经能计算最精细的液体流动，整合所有这些细节就能模拟出海洋的效果，但只有超级计算机才能满足其运算需要。采用物理加速卡可以

实现更为逼真的物理特效。新近被 NVIDIA 收购的 Ageia Technologies（拥有 PhysX 物理加速卡）以及 Intel 收购的 Havok 公司都具有完整的三维图形物理加速解决方案。游戏开发者也在通过粒子系统来实现水的效果，所有粒子都以特定的规则对外界做出反应，使用湍流模型能够绘制出更加真实的液体飞溅、气泡和波浪效果，如图 1.32 所示。

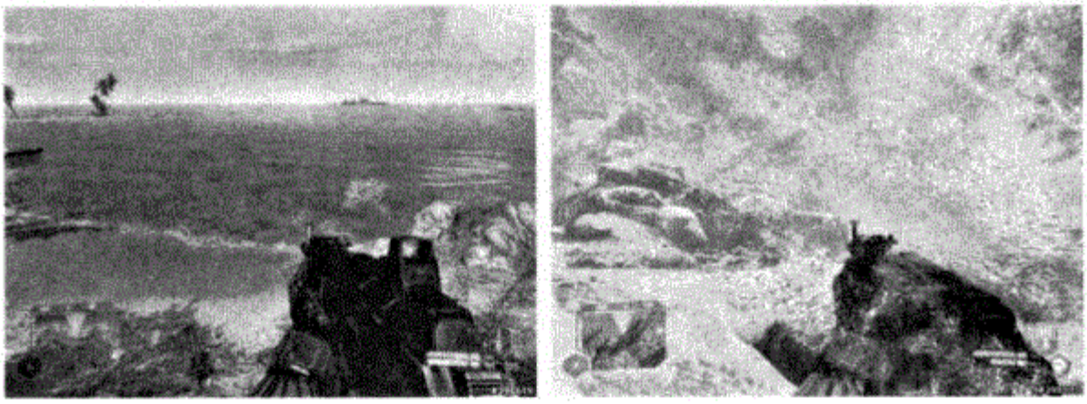


图 1.32　《孤岛危机》游戏中水和结冰的特效

在游戏中表现火焰的行为跟水很像，区别在于火焰移动更快，也更复杂。过去的游戏用实景动画来表现火焰，现代游戏则着重于通过火焰的变形和黏性来表现动态的烟雾和焚烧效果，育碧公司蒙特利尔工作室制作的《孤岛惊魂 2》（Far Cry 2）采用了加强版的 CryENGINE 1.5 引擎，其火焰特效可以模拟从燃烧到化成灰烬的各种效果，如图 1.33 所示。

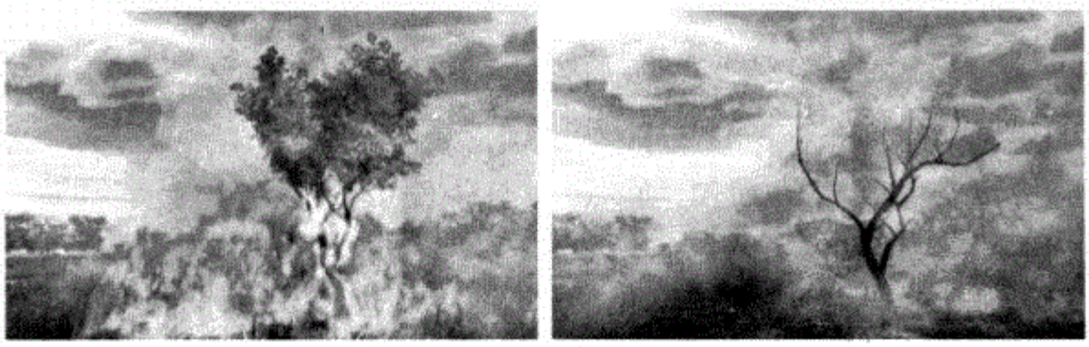


图 1.33　《孤岛惊魂 2》的火焰特效

游戏者也希望非玩家角色（NPC）有更高的智能，增强游戏中敌人的智能可以提高游戏的可玩性，下一代的人工智能将能够独立地与游戏中的角色进行动态交互。在《孤岛危机》游戏中，外星人具有相当高的 AI，他们能使用热成像来找出玩家躲藏的位置，也可通过地面所留下的痕迹来进行查询。而在《刺客信条》（Assassin's Creed）这样的游戏里，追踪的敌人会形成小队来爬楼和越过屋顶。如图 1.34 所示。

图 1.34 《刺客信条》游戏的截图

1.3.7 声音与音乐

三维游戏中的声音为游戏进程提供了调剂。它能够为一个事件提供声音暗示，背景音响也能够暗示出环境和脉络，以及游戏者位置的线索。恰当地使用音效是制作良好三维游戏的基础。一些游戏，特别是多人游戏很少使用音乐，而像单人冒险游戏，音乐则是一个基本的工具，可以用来营造故事线的氛围和为游戏者提供前后关系的线索。在游戏中创造某种情绪是十分具有挑战性的，而加入一段合适的音乐可能正好符合创造情绪的需要。

1.3.8 基础支持

持久稳固的网络支持对于多人在线游戏比单机游戏更重要，在考虑游戏基础支持时，需要考虑记录游戏者得分和技能的数据库、自动升级工具、网站和支持论坛以及游戏管理和游戏者管理工具等。

网站提供了解游戏新闻的途径，可以发现重要的或有趣的信息，下载补丁和修补游戏，网站为游戏提供一个平台，如同店面的作用一样。自动升级的程序装载在游戏者的系统中，更新程序会在游戏开始时经过互联网链接到指定的网站，寻找更新的文件、补丁或用户上次运行程序后已经变化的数据。而后使用这些升级信息在程序运行前下载适当的文件。像《三角洲特种部队黑鹰坠落》（Delta Force：Blackhawk Down）、《网络第二次世界大战》（World War Ⅱ Online）和《无尽的任务》（Everquest）游戏在登录时都有自动升级的设定，需要的话可以自动将文件传到客户端，也有一些游戏是下载到本地之后再安装的。

社区论坛或公告牌是开发者提供的一个有价值的工具，论坛是一个充满活力的，游戏者谈论游戏、特点、竞赛或者进行游戏比较的场所，也是一个客户支持的反馈机制。如果是长期的在线游戏，在网站获取工具来创造和删除游戏者账号、改变密码和管理相关事情是很重要的，这些可以通过建立交互表格和网页来实现。因为安全原因，游戏者的长期得分、造诣和存储的设定等这些需要保护的数据不能存储于本地机，则需要

建立后台的数据库。通常数据库中有创建用户记录的管理工具，它能和游戏服务器和数据库进行通信来鉴别用户、提取和存储得分，并能存储和调出游戏设定和配置。

1.4　常用的商业游戏引擎

对于商业游戏开发者，竞争日趋激烈的游戏产业迫使他们使用游戏引擎来降低游戏研发成本以及复杂程度，并满足市场对于商业游戏在图像等方面日益增长的需求。最典型的一个例子就是游戏续集的开发，一个游戏的续集可能并不等于一个更为优秀的游戏，但是它往往可以为公司带来更多的商业利润，对游戏引擎的复用可以大大节约续作研发的成本，从而将一款游戏的商业价值发挥到极致。

对于独立游戏制作者，往往无须顾虑市场的需求以及发售日期等制约因素，而更专注于对游戏性的探索和实验。无投资，小预算，以个人或工作室为单位进行研发是他们的特点。使用游戏引擎可以让独立游戏制作者更专注于游戏的设计，并使小团队甚至个人完成整个游戏的制作成为可能。大多数独立游戏制作团队没有能力编写游戏引擎，但这并不妨碍制作出精彩的游戏，因为现在有大量高性价比以及免费开源的游戏引擎可供使用。

几乎每一个成功的商业游戏背后，都对应着一个成熟的游戏引擎，而商业游戏引擎在技术上也始终走在行业的最前端，以下主要介绍一些世界级的商业游戏引擎以及使用它们所研发出的次世代游戏。

1.4.1　id Tech 4 引擎

id Tech 4 引擎又称为 Doom 3 engine，由 id Software 公司研发并首次应用在 PC 游戏《毁灭战士 3》中。该引擎由《毁灭战士》和《雷神之锤》的程序员兼设计师约翰·卡马克（John Carmack）开发，在游戏业界处于领先地位。使用 id Tech 4 开发的著名游戏有《毁灭战士 3》（见图 1.35）、《雷神之锤 4》（见图 1.36）、《掠食》（Prey）（见图 1.37）等，该引擎在三维图形方面的应用程序接口（API）采用的是 Open GL（Open Graphics Library）。

图 1.35　《毁灭战士 3》游戏截图

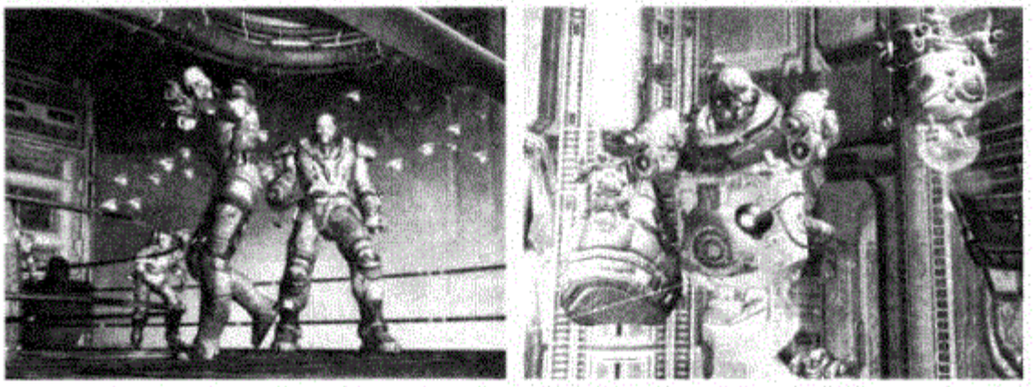


图 1.36 《雷神之锤 4》游戏截图

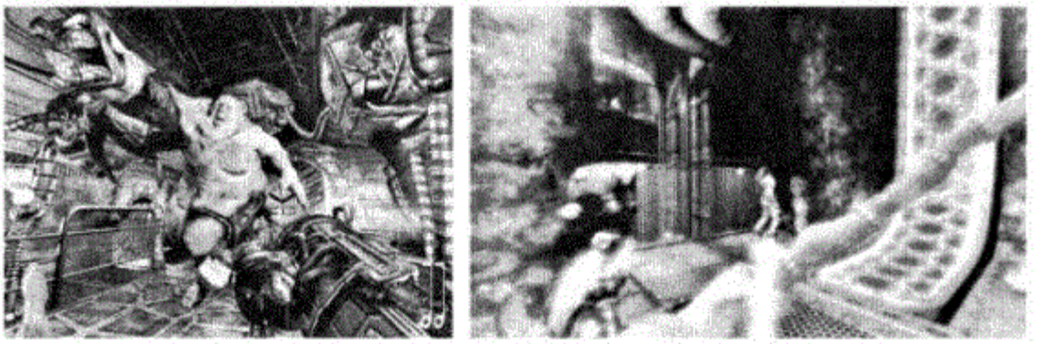


图 1.37 《掠食》游戏截图

1.4.2 CryENGINE 2 引擎

Crytek 公司的 CryENGINE 是《孤岛惊魂》（Far Cry）采用的游戏引擎，CryENGINE 2 则是在其基础上改进的全新游戏引擎，使用它研发的第一人称射击游戏《孤岛危机》（Crysis）在画面和物理模拟上都达到了很高水平，如图 1.38 所示。

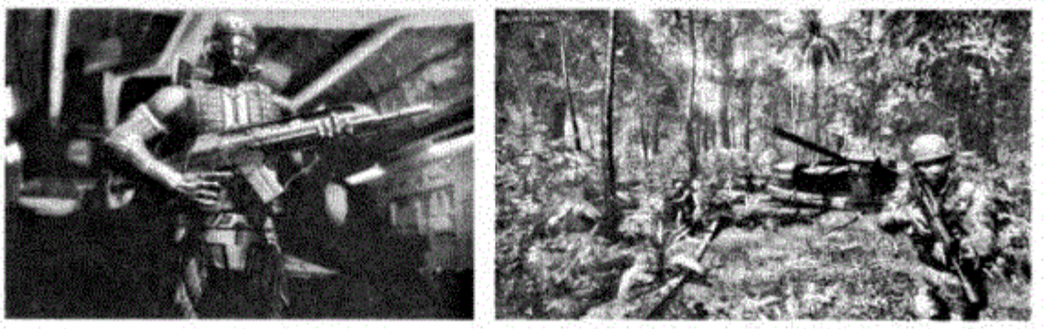


图 1.38 《孤岛危机》游戏截图

1.4.3 RAGE 引擎

RAGE 是 Rockstar Advanced Game Engine 的缩写，由游戏制造商 Rockstar 研发。它是客户定制的视觉引擎，长处在于渲染精细的动画，特别是衣服纹理。RAGE 引擎的第一款游戏是获得了很高评价的《乒乓》，它具有高解析度的画面和尖端的动作系统，如图 1.39 所

示。此外，动作冒险游戏《侠盗猎车手 4》（GTA 4）也是由“RAGE”引擎进行研发的。

图 1.39　《乒乓》游戏截图

1.4.4　弯刀引擎

弯刀（Scimitar）引擎是由育碧（Ubisoft）研发用于《刺客信条》（Assassin's Creed）游戏的一款高端三维引擎。它改变了传统动作游戏路径有限、敌人反应固定的过关模式。使用该引擎开发的游戏，除了能灵活地展现各种动作外，还具有开放式的互动场景与完全自由的路线选择，并能为非玩家角色（NPC）赋予更为真实的人工智能，如图 1.40 所示。

图 1.40　《刺客信条》游戏截图

1.4.5　起源引擎

起源引擎（Source Engine）由 Valve Corporation 研发，拥有高度集成的特性，基于 Shader 的渲染引擎具有口型同步和标签系统，以及功能强大而高效的物理模拟引擎，如图 1.41 所示，应用的作品有第一人称射击游戏《半条命 2》（Half Life 2）、解谜射击游戏《洞穴》（Portal）等。

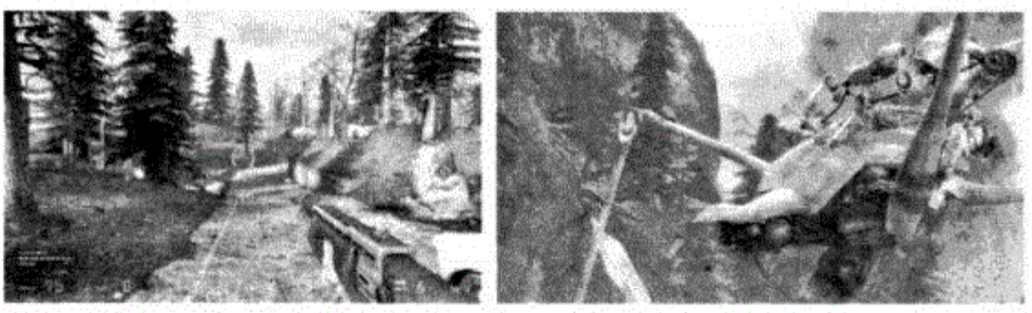


图 1.41　《半条命 2》游戏截图

1.4.6 虚幻引擎

虚幻引擎（Unreal Engine）由 Epic 游戏公司研发，是在游戏业界广泛使用的商业游戏引擎之一。如图 1.42 所示的第一人称射击游戏《虚幻竞技场 3》（Unreal Tournament 3）、如图 1.43 所示的《生化奇兵》（BioShock）、如图 1.44 所示的《战争机器》（Gears of War）、动作类游戏《细胞分裂》（Splinter Cell）系列和大型多人在线角色扮演游戏（MMORPG）《天堂 2》（Lineage Ⅱ）等都采用了该引擎。

图 1.42 《虚幻竞技场 3》游戏截图

图 1.43 《生化奇兵》游戏截图

图 1.44 《战争机器》游戏截图

虚幻 3 游戏引擎（Unreal Engine 3）是一套为次世代游戏主机和装配有 DirectX 9 的计算机定制的游戏开发框架，提供了大量核心技术、内容创建工具，并且为高端游戏开发者提供了强有力的底层构架。虚幻 3 作为面向次世代游戏开发的专业游戏引擎，属于高端的商业引擎，其功能列表如表 1.1 所示。

表 1.1　Unreal Engine 3 引擎的功能列表

图像 API	OpenGL、DirectX	操作系统	Windows、Linux、MacOS、Xbox360、PS3
编程语言	C/C + +	说明文档	完备
特点			
总体特点	面向对象的设计，插件系统，Save/Load 系统		
程序编写	• Unreal Script——用于编写游戏互动的脚本语言，并自动支持元数据（Metadata）；支持各种灵活的文件格式；支持为关卡设计师在 UnrealEd 中显示程序中定义的属性值；基于 GUI 的脚本编译器；原语言支持多种在游戏编程中非常重要的概念，例如动态范围自动机（Dynamically scoped state machine）、定时脚本触发等 • 可视的 AI 编程工具帮助设计者完成游戏内部复杂的剧情交互设计、游戏事件触发器、可交互剧情动画等		
内建编辑器	• UnrealEd 自带可视物理建模工具，支持为模型和骨骼动画网格创建优化的碰撞检测体；可交互的物理模拟和内嵌调试器 • 关卡设计师可以在 UnrealEd 中看到 AI 路径并手动进行调整和修改 • “UnrealMatinee”——基于时间轴的剧情动画制作工具，设计师可以使用它通过排列动画帧，移动包括摄像机在内的各种物体来制作游戏内可交互或不可交互的剧情动画，还可以触发游戏内容和 AI 事件 • UnrealEd 内嵌的可视化声音编辑工具使设计师可以完全掌控音效的处理。音效的参数被放置于与程序代码相独立的另一块区域之中，设计师可以通过它控制包括游戏、剧情动画、动画帧在内的各种音效		
物理模拟	基础物理、碰撞检测、刚体、交通工具物理模拟		
光照	Per-vertex（逐顶点）、Per-pixel（逐像素）、Volumetric（体积光照）、Light mapping（光照贴图）、Gloss maps（高光贴图）、Anisotropic		
阴影	Shadow Mapping（阴影贴图）、Projected planar（平面投射阴影）、Shadow Volume（体积阴影）		
贴图	Basic、Multi-texturing（复合贴图）、凹凸贴图，Mipmapping、Volumetric、Projected，Procedural；支持当前所有的贴图技术		
Shaders	顶点、像素、高级		
场景管理	普通、BSP、Portals、LOD		
动画	反动力学、关键帧动画、骨骼动画、面部动画、动画混合		
网格	Mesh Loading、Skinning、Progressive、Tessellation、Deformation		
特效	环境贴图、光晕、广告板、粒子系统、景深、动态模糊、天空、水、火、爆炸、贴花、雾、天气、镜面		
地形	渲染、CLOD、Splatting		
网络系统	用户服务器模型、点对点模型		
声音视频	2D 音效、3D 音效、流媒体		
人工智能	寻路算法、决策选择、有限自动机、可编程逻辑		

1.5　三维游戏设计的基本流程

1.5.1　游戏制作的基本流程

游戏设计的程序通常包括构思、提案、原型、制作、整合、测试等几个阶段，如

图 1.45 所示。不同的创作团队面对不同类型的游戏也会有所调整。本节以育碧（上海）公司的游戏开发流程为例，具体解析三维游戏设计的过程。

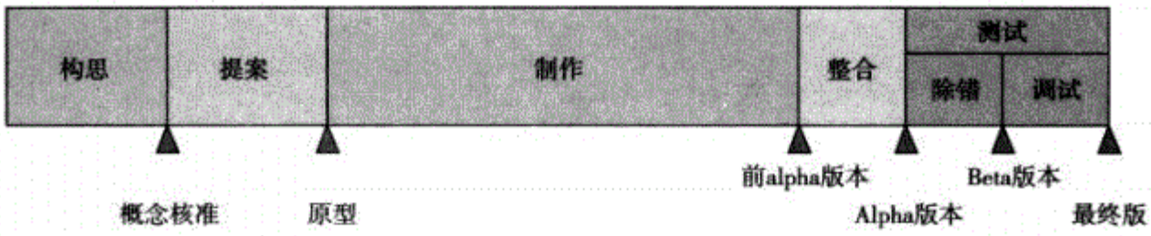


图 1.45　游戏设计的基本流程

游戏构思（High Concept）一般是来自公司的总编室，这是一个由各方面专家组成的团队，包括程序方面的专家、游戏分析方面的专家（Game Analysis）等，专家分工较为详细。一个小组大约在十几个人左右，是对游戏开发过程中最高层面上的质量控制。他们结合市场的调查以及对现有游戏的一些分析就会形成一些概念，概念的来源可以说是多方面的，但最后立项是由总编室决定的。育碧公司每年都有一个经理会（Managing Directing Meeting），各分部的总经理聚集在总部一起讨论项目的安排。总部会综合考虑各分部人员的素质，包括人员的计划、安排，最后决定项目的分派。

项目安排确立之后，所在的分部就要选择制作人和核心小组（Core Team）。这个核心小组包括主程序、创意总监和艺术总监三个核心人物。而后这个核心小组要去总部准备开题会（Kick Off）。他们首先要跟总编室的团队一起进行一个研讨会，主旨就是去确认这个概念构思的可行性，大家一起经历头脑风暴，研讨支持这样一个构想的具体实现方案，并确定哪些是可行的，哪些在技术上有难度的，然后把这些东西全都罗列出来进行筛选，最后确定出核心的游戏制作思路，如图 1.46 所示。当游戏制作思路确定以后就会进行开题会，在会上进一步确认项目的时间进度、项目的质量要求、项目核心成员的团队组织方式等。

图 1.46　概念（Concept）

开题会之后，核心组成员返回本部，制作一个演示（Demo），称之为 F. PP 阶段（First Playable Prototype），也就是游戏的一个原型，通常就是游戏中的一个关卡。这个关卡要求达到一个比较高的质量，包括美工、引擎等，在这个原型里都要比较完善地体现。在这个过程中，此间总编室会不断审核工作进度，不断提出改进意见，甚至提出方针上的一些调整。同时，会逐步把制作团队组织起来。在制作 Demo 的过程中，核心小组还会对一些新人或者经验不够丰富的人员做一些必要的训练。当 F. PP 完成以后，核心小组需要再次到总部演示 Demo 版本的游戏。最后当一切就绪的时候，总部会下达正式的决定开始制作，如图 1.47 所示。

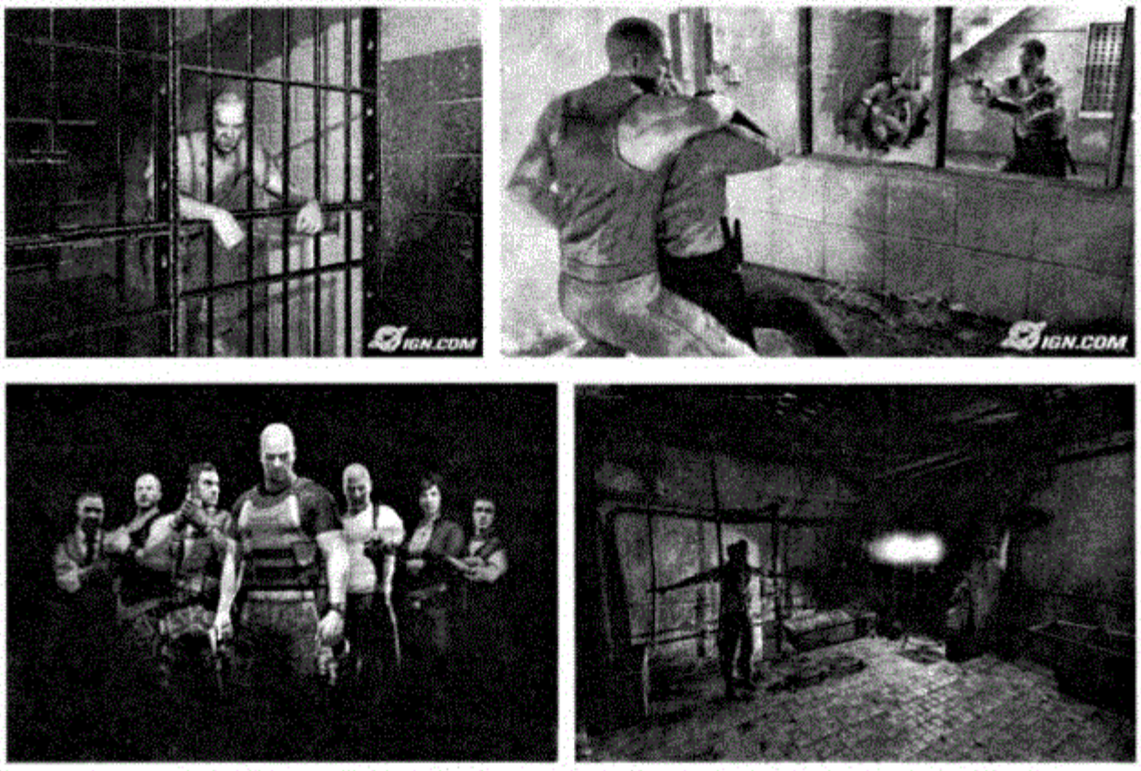


图 1.47　首个可玩的原型（F. PP）截图

与软件行业一样，原型阶段之后是 Alpha 阶段。Alpha 阶段要整合所有的内容，尽管有些东西还没有被润色过，但是整个游戏要从头到尾都能够玩通。当所有的设计内容都制作出来后，如图 1.48 所示。就从 Alpha 阶段进到 Beta 阶段。这时整个游戏的内容已基本上稳定，不会再有什么改动。Alpha 阶段之后，工作的主要重点就集中于找错与除错（Debug），如图 1.49 所示。

在后期的工作中，测试部门开始大规模介入，测试地图和关卡。如果在哪个地方被卡住，在哪个地方有缺陷，测试人员就会提出来。设计团队在最后的调整和润色的同时，根据测试报告进行修改。全部修改调整之后，送给总部审批，如图 1.50 所示。

图 1.48　游戏的 Alpha 版本截图

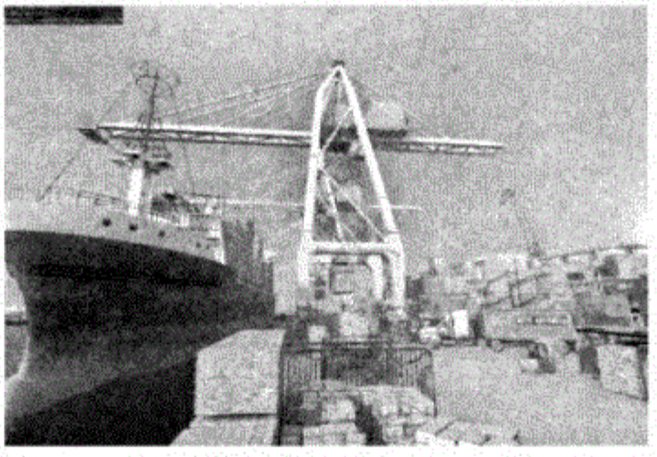


图 1.49　游戏的 Beta 版本截图

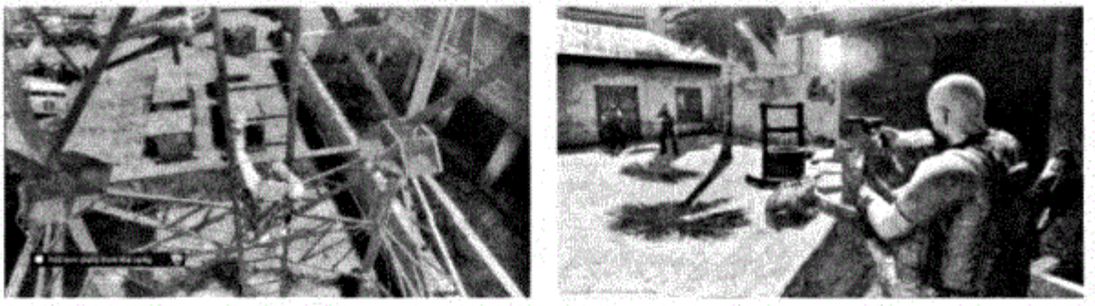


图 1.50　最终版（Final）截图

时间安排上会根据项目的性质有所不同，如果原创游戏是一个续集，那么开发的时间是不一样的。但是一般来说，F. PP 之前的构思阶段是最长的。很多原创的游戏项目在这个阶段会持续一年左右，甚至一年以上。从 F. PP 到 Alpha 阶段是一个正式制作的过程，一般来说是 4 ~ 6 个月。从 Alpha 到 Beta 阶段的时间比较短，一般是 2 ~ 3 个月。然后 Alpha 到 master 也是 2 个月，这是一个比较理想化的安排。

1.5.2　游戏美工部门的具体分工

美工部门中，二维方面有原画师（Concept Artist）与美术总监一起合作绘制效果

图和故事板（Story Board），角色美工（Character Artist）专门制作人物和非现实角色的建模。与角色动画部门相关的建模，都是由角色美工来完成的。

场景的搭建由关卡美工（Level Artist）专门负责。关卡设计组决定整个场景搭建的风格和地域，然后由美工组搜集相应的素材。关卡美工基于关卡设计的一些构想和游戏设计的一些指导性要求来搭建场景，搭建的时候不仅要考虑建筑学上的合理性，也要考虑进一步的光影渲染与氛围创造等问题。例如，构想一个在比较喧闹场所进行的游戏活动，关卡美工就需要提出这个喧闹的地方具体在哪里，然后与关卡设计师确定这个地方的具体设计，以及更细节的部分，如这条路上的建筑里会发生哪些具体事情，之后才会开始场景的构建。

建模的素材在不同的游戏间相互使用会降低成本，从次世代游戏的制作流程上来说，重复使用是最省资源的，也会加快项目的进度。但是过度地重复利用也会引起似曾相识的感觉。视觉效果方面的控制要由艺术总监来完成，使之在视觉上基本感觉不到很明显的相似性。另外，每完成一个项目，这个项目里使用的贴图、声音、动画、建模要备份到数据库，美术总监有权限去查阅这些素材，如果制作进程中需要，就可以选取并做适当的修改，这样能够节省一些工作量。

随后的流程由专门负责贴图的美工（Texture Artist）完成。在做基本模型时，其材质已基本完成，但在精度方面、真实性方面不是很高。贴图美工主要就是完成更进一步的修改和完善。渲染美工（Shade Artist）则更偏重于美工与程序的结合，负责表现木头、金属、玻璃等具体的物质材质。

次世代游戏对贴图有特别要求，因此渲染美工在渲染表现时，需要考虑材质在各个角度看上去都能体现出同样的感觉，渲染美工的工作不是绘画，主要是基于引擎参数方面的调整。此外，还需要用户界面美工（UI Artist）设计游戏中的菜单界面。

美工部门是最复杂的一个团队，其中还包括特效部门（Special Effect），如场景里的爆炸、光影或是基于游戏过程中的一些特效、基于场景的特效，特效制作人员会协助关卡美工去实现。

育碧公司的游戏《细胞分裂 4》（Splinter Cell 4）通过大量的凹凸贴图、法线贴图等手段创造出了生动的环境。光照、水、冰、影子以及人物面部的刻画都达到了新的水平，不同难度的关卡和各种多人游戏模式也增强了游戏可玩性。在团队人数安排方面，《细胞分裂 4》项目美工部门，包括二维和三维接近 60 人。其中动画部门大约是 10 人，关卡设计部门大约是 15 人，游戏设计部门大约是 5 人。程序组 20 人左右，音效部门是 3 ~ 5 人，还有一些外包的人员。质保部门（Quality Assurance）是 5 人左右。测试部门的人员比较多，至少有 50 人。编剧的人员较少，只有几个人。还有一些协助制作团队工作的人员，如语言师（Linguistic）。育碧公司的游戏发行在欧洲是 5 种语言、美国是 3 种语言，因此需要有一个专门的部门去完成相关翻译、整合、录音工作。

1.6 本章回顾

本章首先介绍三维游戏的基本知识，包括三维游戏的概念与特征，与二维游戏的特点比较以及三维游戏中的时间和空间观念等；接下来讲解了三维游戏的类型和相关的范例，不同类型的游戏有着不同的特点，通过对这些特征的分析和研讨，将对进一步的三维游戏设计提供全面的知识和理念。

为了深入剖析三维游戏的构成形式，本章也讲解了三维游戏的基本构成要素和主要商业引擎的特点，通过对游戏引擎进行图示说明，使读者能够更加深入地了解游戏引擎的各要素。本章最后结合对著名游戏公司的访谈，归纳了三维游戏的设计程序和方法，为随后进一步学习与制作奠定了基础。

1.7 课后思考

1. 三维游戏的特点是什么？包括哪些基本要素？

2. 结合三维游戏的主要类型做进一步的调研，在互联网上搜索有代表性的游戏作品视频和截图，对三维游戏的演进和发展进行深入的了解和思考。

3. 浏览“DevMaster. net”网站（http://www. devmaster. net/engines/）的引擎数据库，查询当前最流行的游戏引擎，了解你所熟悉的游戏都是使用哪些游戏引擎开发的。

第2章 Virtools使用入门

学习三维游戏制作可以选取多种软件，本书采用的是Virtools软件。它具有“可视化编程”以及“实时运行调试”的功能，使学习者能够回避复杂的游戏程序问题，而将更多的精力放在三维游戏的关卡设计与实现上，适合编程经验较少的艺术设计学生使用；同时它也广泛用于虚拟现实和仿真的领域，具有较好的扩展性。本章介绍Virtools的基本知识和开发流程，并且通过一个简单的案例来介绍制作三维角色互动漫游的方法。

2.1 Virtools Dev 4.0 简介

法国达索系统公司（Dassault）的Virtools是整合三维互动技术的软件，它可以应用于游戏研发、互动展示、模拟仿真以及虚拟现实等领域，其产品可以发布到计算机、游戏主机、局域网及互联网等平台上，基于Virtools研发的交互场景和游戏如图2.1和图2.2所示。Virtools作为游戏开发工具，可以完成包括多人游戏、计算机游戏、主机游戏、游戏模型、网络游戏等方面的设计制作任务。在工业应用方面，可以满足设计评估、任务规划、训练、虚拟仿真、销售配置等方面的需求，使用户具有体验沉浸式虚拟现实的能力。在互联网方面，Virtools能以三维的形式，构建广告、营销、多媒体或者数字化学习（E-learning）的在线体验。[①]

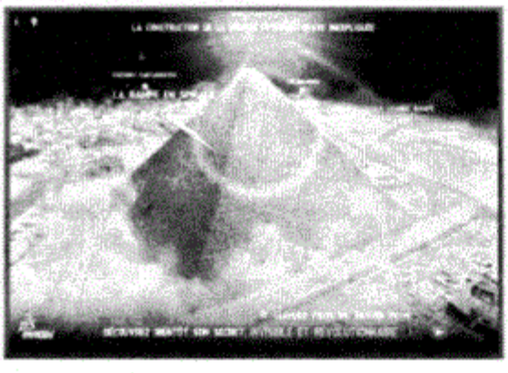


图2.1 金字塔网络交互场景截图

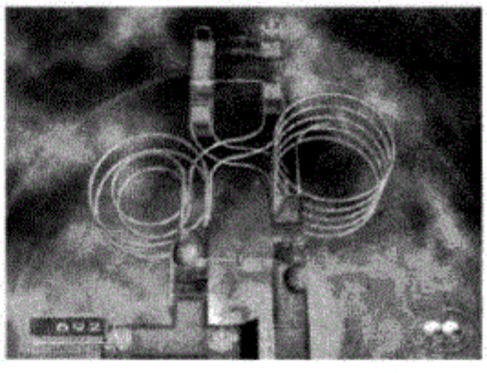


图2.2 创意游戏《Balance》截图

① 参见 http://www.virtools.com.cn/about/200705/215.html.

2.1.1 Virtools 的开发流程

在 Virtools 的开发流程结构中，通过组合各种模块，可以扩展数字内容开发的应用范围，如图 2.3 所示。Virtools 的开发流程主要包括三个阶段：捕获、创作、部署与体验。在捕获阶段要进行模型的导入。三维内容捕获（Virtools 3D Content Capture）可以满足多种类型的三维数据的交互应用。Virtools 支持多种三维数据，也能够直接输出和转换 3D XML 档，适用于 CAD 或者其他数字内容创造工具，开发者可以利用 Virtools 平台选择合适的数据模型来进行实时工作。在内容开发创作阶段，Virtools 开发平台通过图形界面，以直观的形式提供各种复杂的程序，还附带有行为脚本模块（包括物理属性、人工智能属性、虚拟现实、多用户服务器等）能够满足交互式三维创作的需要。在部署与体验阶段，Virtools 解决方案能够满足在互联网、公司内部网、其他设备以及大范围的虚拟现实环境中发布产品的需求，使用户可以获得更真实的体验。

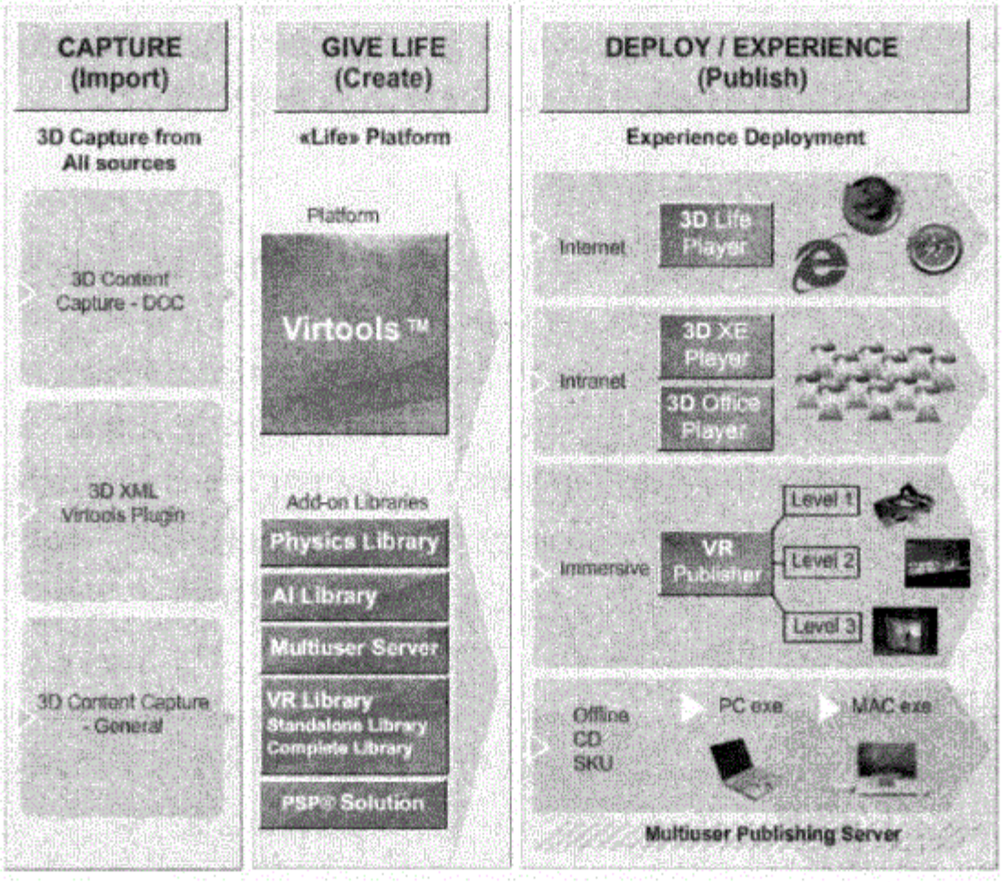


图 2.3 Virtools 开发流程架构图①

2.1.2 使用 Virtools 开发游戏

数字游戏的制作阶段可以粗略地划分为两部分：制作游戏中使用的各种资源（包

① 参见 Virtools 开发流程架构图，http://www.virtools.com.cn/solutions/.

括美术资源、声音资源、文字资源等）和使用脚本语言实现游戏的逻辑（也就是编写游戏中的各种互动效果）。在制作游戏资源的时候，需要遵循一系列的原则，而最重要的原则就是“不能超过游戏引擎的极限”。如果只考虑美术素材的制作（如人物和场景），即时渲染的画面质量、场景和人物的多边形面数、是否支持高级显示技术（如 Shader）等因素都受制于引擎。在这一点上，尽管 Virtools 不是渲染效果最强、支持高端视效技术最多的游戏引擎，达不到 Quake 或 Half Life 的高超视觉效果，但在一般效果的制作上，Virtools 在模型面数和材质贴图大小限制较小，使设计师可以根据硬件条件充分展示自己的创意，并且能够支持游戏业界广泛采用的视效技术，也扩大了设计师的发挥空间。另外，Virtools 对于诸多游戏业界常用的三维软件（如 3DS MAX、MAYA）都配有专用的导出插件，不仅可以保留完整的模型，也可以将动画信息导出应用于游戏之中。

在实际的商业游戏制作中，Virtools 也具有“快速研发和低研发开支”的特点，因此通常被用于游戏原型制作（快速研发）或掌机游戏的开发（低研发开支）。Virtools 可视化编程语言适合艺术设计类的学生学习与使用，预设 Building Block 资源可以缩短开发的流程，不仅适合于游戏原型的开发，同样也适合游戏教学与研究中的短期应用，使学生能在短期内做出具有一定水准的游戏作品。Virtools 随软件附带了教学文档，包括技术演示实例、脚本函数速查手册、SDK 开发手册等文档，可以方便用户掌握 Virtools 的操作。

2.2 Virtools 使用初步

本节中将主要介绍 Virtools Dev 4.0 的操作界面，了解一些常用的操作方法，为后面的学习打下基础。

2.2.1 软件界面概览

如图 2.4 所示的 Virtools 界面，大致可以分为编辑、资源和设定三个部分。界面的左上部为编辑部分，包括 3D Layout 和工具栏。在三维预览窗口（Perspective View）中可以预览三维世界，并对其中显示的各种物体进行操作，而其左侧的工具栏中的各种功能也是针对 3D Layout 的，包括了物体创建修改、新建物体、调整视角等功能。

界面右上角是资源部分，包括 Data Resources（资源）、Building Blocks（脚本模块）、Hierarchy（继承）等面板。资源面板可以用来管理在游戏制作中需要用到的各种外部资源，Building Blocks（简称 BB）中分门别类存放所有 Virtools 自带的程序模块，使用它们可以完成绝大部分交互程序脚本的书写。Hierarchy 使用层级目录管理所有工程文件内的物体继承关系，通过设置物体间的继承关系，可以更方便地对它们整

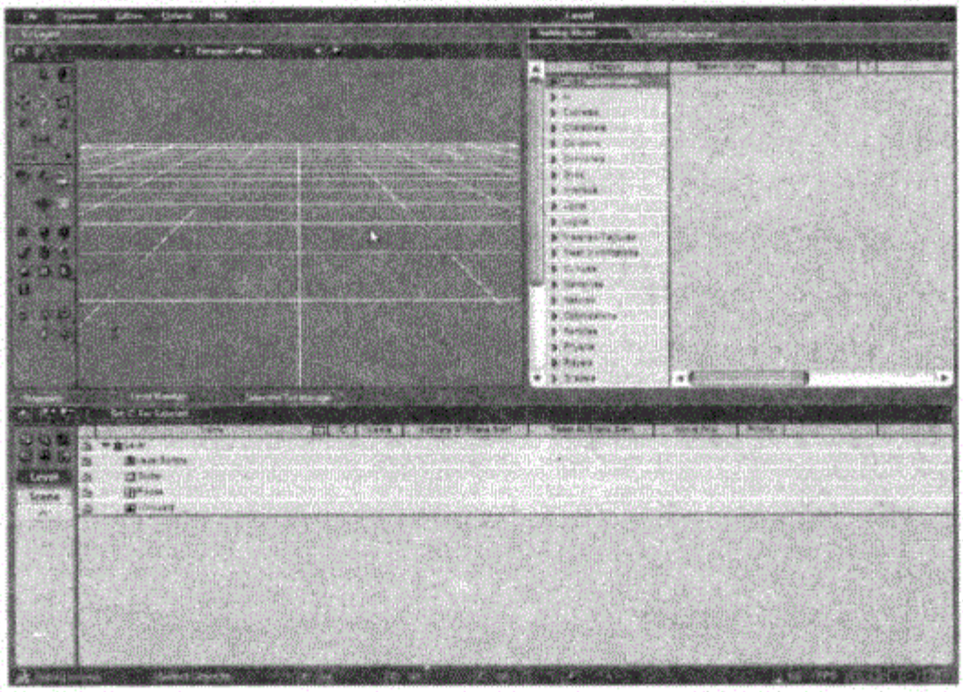


图 2.4 Virtools 界面

体进行各种操作和设置，在默认的界面设置下，Hierarchy 面板是不显示的，需要在 Editors 菜单中将其打开。

界面下方是设定部分，包括 Level Manager、脚本书写（Building Blocks）、VSL（Virtools Scripting Language）代码书写、属性面板和所有物体的设置面板。Level Manager 是 Virtools 程序的资源浏览器，脚本书写和 VSL 代码编程也都是在其下方的相应面板中，Attributes 面板管理所有程序中用到的属性，对所有物体，如 3D Object、Material、Texture、Light、Camera 等的设置面板，也都集中以标签页的形式显示在下方。

2.2.2 3D Layout 窗口

3D Layout 是三维世界的预览窗口，如图 2.5 所示。通过左侧工具栏可以操作其中的物体，由于 Virtools 采用“所见即所得”的开发模式，在这个窗口中看到的就是最终的画面效果，也就是“实时渲染”。三维建模软件则通常是“预渲染”，如 3DS Max 或 Maya，最终的渲染图像质量要比预览窗口图像质量高很多。3D Layout 中可以设置不同的预览视角，程序自带四个视角：Perspective view（透视图）、Top view（顶视图）、Front view（前视图）、Right view（左视图）、Orthographic view（无透视视图）以及所有在程序中创建的 Camera（摄像机）视角，通过 3D Layout 上方的下拉菜单就可以进行切换。

Virtools 的四个视角可以根据制作需要灵活切换，但无法作为游戏运行时的视角，也无法通过程序对它们进行任何设置和控制，如果没有添加任何自定义 Camera（摄像机），那么在游戏运行的时候将默认使用 Perspective view 视角。

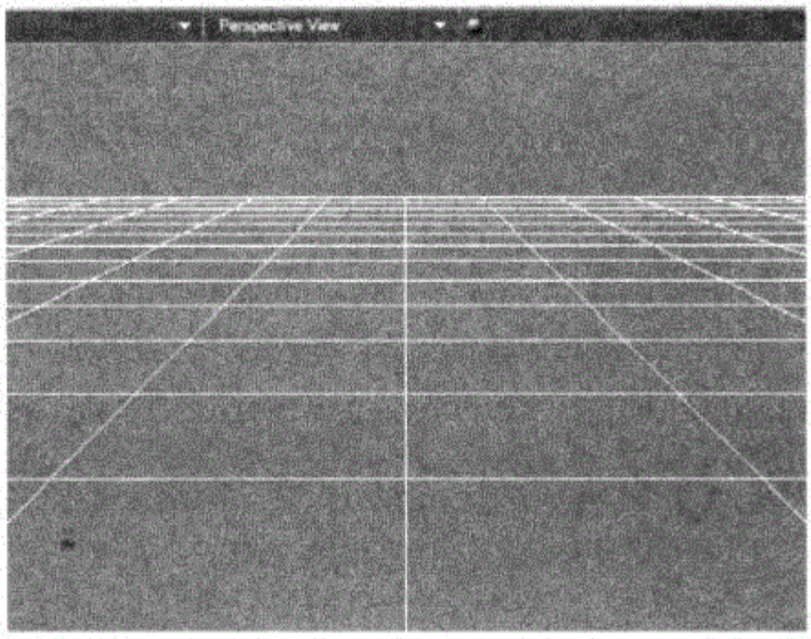


图 2.5　3D Layout 窗口

2.2.3　工具面板

工具面板共分为四个部分，最上一个部分为“选择操作工具组”，第二部分为“移动变形操作工具组”，第三部分为“创建物体工具组”，最下方为“视角操作工具组”，下面将逐一对其进行介绍。

2.2.3.1　选择操作工具组（如图 2.6 所示）

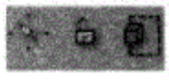


图 2.6　选择操作工具组

最上方的三个按钮与 Select（选择）相关。左边的箭头状按钮用来激活或关闭“选择物体”，当选择一个或多个 3D Layout 中的物体时，需要首先激活该按钮。在测试游戏的时候，往往不希望鼠标误选其他物体或者出现误操作，此时最好关闭此按钮。

中间锁状的按钮是 Lock（锁定）功能按钮，在选中物体后，锁定物体（显示为关闭的锁）可以防止选中其他物体，当需要解除锁定的时候，只需要再单击一下该按钮就可以（显示为打开的锁）。

右边的图标是一个虚线框加一个立方体，它是“矩形框选择”按钮，在同时选择多个物体的时候，可以使用该按钮，即按住鼠标左键并拖动鼠标，对多个物体进行框选。通过该按钮可以在“框住部分即选中”和“框住全部即选中”两种模式之间进行切换。

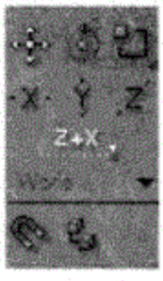


图 2.7　移动变形操作工具组

2.2.3.2　移动变形操作工具组

移动变形操作工具组，如图 2.7 所示，实际上只包含移动、旋转、缩放三种工具，但是设置选项较多，占据了 5 行的位置。

第二行标有 X、Y、Z 的三个图标，如图 2.8 所示，表示当前的操

作被限制在坐标轴的哪一个轴上，如果在移动操作时激活了 X 按钮，物体将只能沿着 X 轴进行移动。

第三行和第二行类似，是将操作限定在一个 2D 平面之上，有“X + Y”、“X + Z”、“Y + Z”三个选项，如图 2.9 所示。

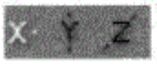


图 2.8 X、Y、Z 轴操作限定

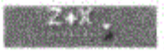


图 2.9 平面操作限定

第四行没有按钮，是一个下拉菜单，可以设置坐标系为 World（世界）坐标系或者 Local（本地）坐标系，如图 2.10 所示。

最后一行包括三个简单实用的工具，如图 2.11 所示。

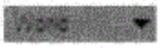


图 2.10 坐标系选项

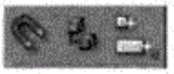


图 2.11 辅助工具组

左侧磁铁形状的工具为 Snap Tool（吸附工具），若激活 Snap（单击变为白色），在 3D Layout 中接近其他物体，达到一定范围时，会自动吸附到该物体表面。

中间的图标表示操作是否会影响具有继承属性的子物体。例如，将四个车轮设置为车身的子物体（Children），如果是图 2.12 所示的选项状态，当移动车身的时候，车轮也会随着移动；如果是图 2.13 所示的选项状态，当移动车身的时候，车轮就不会受到影响。

图 2.12 影响继承子物体

图 2.13 不影响继承子物体

图 2.11 的最右侧图标为轴坐标选项，在进行旋转和缩放操作的时候，绕自身的中心旋转和绕两个物体的中间位置旋转，其结果是不同的。

2.2.3.3 创建物体工具组

如图 2.14 所示，该工具组用来创建 3D Layout 中的物体，从左到右从上到下依次为 camera（摄像机）、point light（点光源）、3D Frame（三维形体）、Curve（曲线）、Grid（网格）、2D Frame（二维形体）、Material（材质）、Texture（贴图）、Portal（传送门）、Video（视频）。

图 2.14 创建物体工具组

2.2.4 脚本模块资源库

脚本模块资源库（Building Blocks Resource）面板包含了在编写交互脚本时要用到的全部 Building Block（简称 BB），除了 Virtools 自带的 BB，如果有自己编写的 BB，

也会显示在其中，如图 2.15 所示。

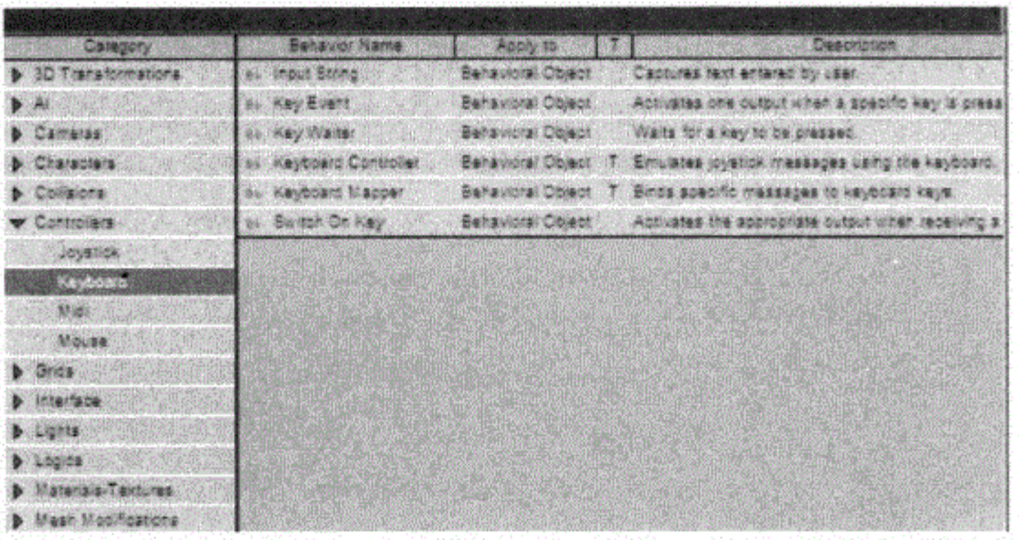


图 2.15　Building Blocks 面板

所有的 BB 都被分门别类放在相应的目录之中，可以方便地通过分类进行检索，每一个 BB 都包含有几项基本信息，如图 2.16 所示。

图 2.16　Building Block 的基本信息

其中，Category（分类）表示 BB 所属的分类或次级分类；Behavior Name（行为名称）表示 BB 的名字；Apply to（应用于）表示该 BB 可以被应用于哪种类型的物体；T 表示该 BB 可以指定目标（Target）；Description（描述）表示该 BB 功能的简要描述；Version（版本）表示该 BB 的版本信息；Author（作者）表示该 BB 的作者，所有 Virtools 自带 BB 的作者都是 Virtools。

将 BB 拖曳到 3D Layout 或 Level Manager 中的物体上，可以为该物体添加脚本模块；按 Shift 或 Ctrl 键 + 鼠标单击可以选中多个 BB；将 BB 直接拖曳到物体上会在该物体的第一个脚本中添加 BB，若该物体还没有脚本，则会自动创建一个新的脚本。如果该 BB 有需要设置的参数则会自动弹出设置窗口，也可以直接将 BB 拖曳到脚本流程图（Schematic）面板中。

2.2.5　资源面板

Resources（资源）面板管理“外部资源”，程序内资源则由层级管理器（Level Manager）管理。将外部资源导入到程序内，可以使用菜单中的 Import 选项，但是 Import 是直接将资源加入到工程文件内而不能赋给某一个物体，也不便于管理和浏览。Resources 面板则可以分门别类地管理各种可能使用到的资源类型，从三维模型、纹理

贴图到脚本程序，并且提供了预览用的缩略图，支持“.nmo”格式文件（Virtools 导出资源文件格式）的预览，还支持文件的拖曳功能。

默认情况下，界面中会显示 Virtools 随软件自带的资源库面板——VirtoolsResources，如图 2.17 所示。通过浏览该资源库可以了解各个分类对应的资源类型和文件格式，同时其中的很多资源也可以为游戏制作带来很多便利，如 Primitive（基本三维形体）、Texture（纹理贴图）等。

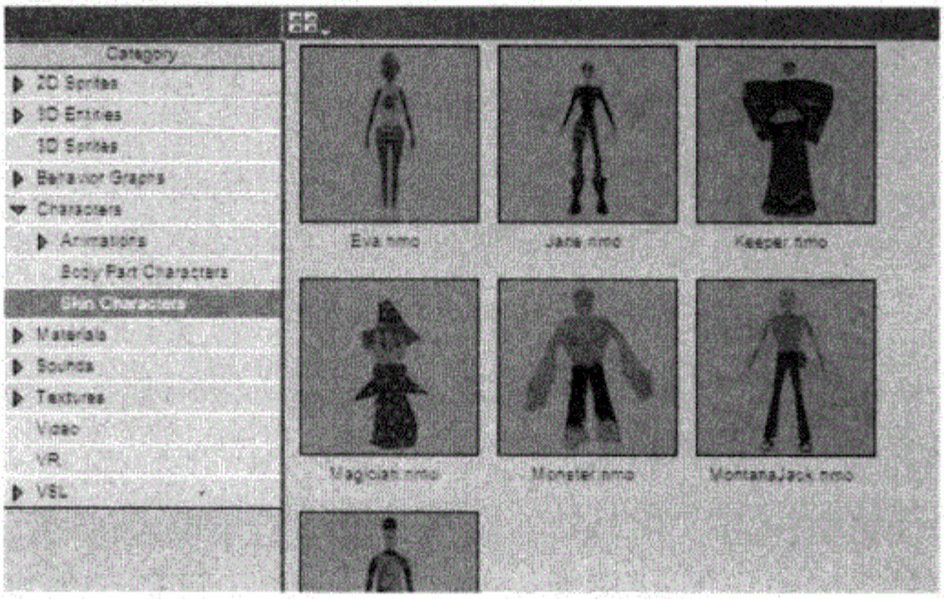


图 2.17　VirtoolsResources 资源库

Virtools 自带的 Shader 也都被放置在 Materials/Shader Materials 目录下，如图 2.18 所示。由于 VirtoolsResources 面板包含了软件自带的一些功能和使用这些功能所需要的资源，所以建议不要关闭它。

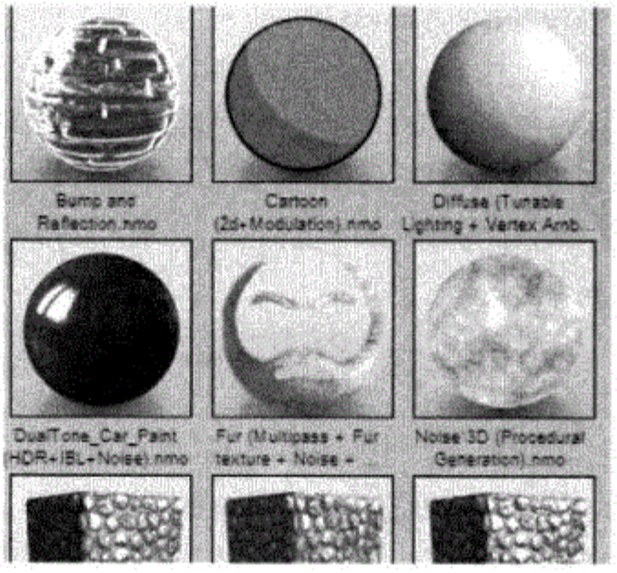


图 2.18　Virtools 自带的 Shader 资源

除了 Virtools 自带的资源库，用户也可以创建属于自己的资源库。执行 Resource | Create New Data Resource 菜单命令，如图 2.19 所示，并指定资源库的路径和名称即可完成创建。

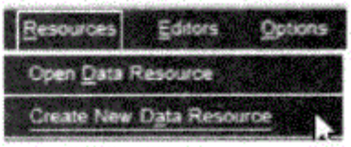


图 2.19 Resources 菜单

新建的资源库包括资源库文件（资源库名称 . rsc）以及与资源库同名的根文件夹，在其下细分有多个目录用以放置不同的资源，若需要添加资源，只需要复制资源文件到相应文件夹即可，若放错了文件夹，文件将不能在 Resource 面板中正常进行预览和显示。同时打开多个资源库时，可以通过执行 Resource | Open Data Resource 菜单命令。图 2.20 展示了一个自定义的资源库。

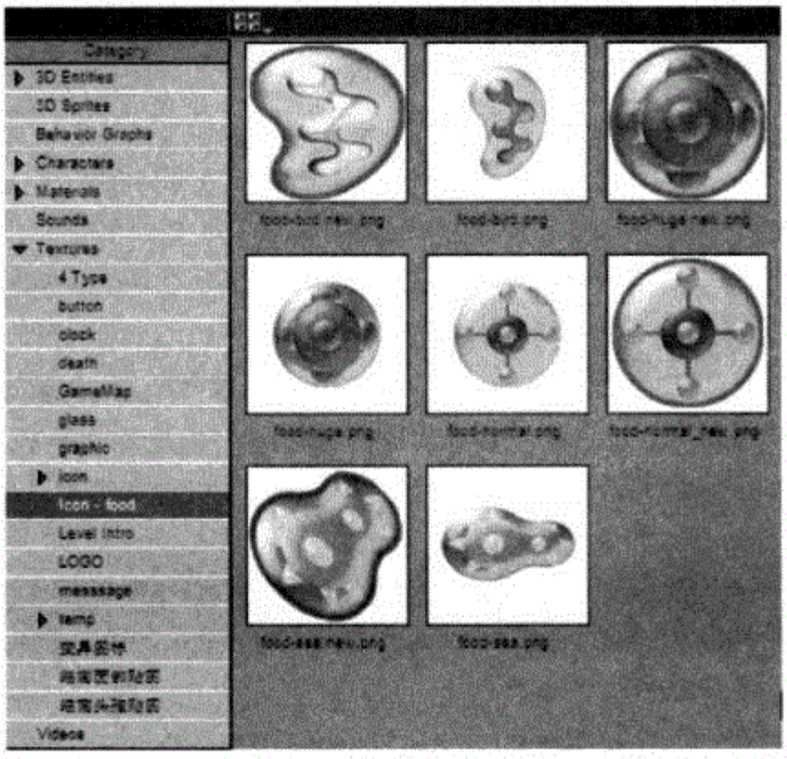


图 2.20 用户自定义的资源库范例

2.2.6 状态栏

状态栏（Status Bar）位于屏幕最下方，如图 2.21 所示。在状态信息中，Selection Name（选择物体名称）表示当前选中的物体名称，Action（动作）表示当前使用的工具名称，X Y Z Coordinates（坐标）表示 3D Layout 中被选物体的三维坐标，Event Log Information 表示事件日志信息，Restore Initial Conditions（恢复初始状态）表示停止播放并恢复物体的初始状态（此前需要进行相关的设置），Play/Pause（播放/暂停）表示播放或暂停当前程序。右键单击可以进行时间设置，Step One Frame（前进一帧）表示前进一帧并暂停。

图 2.21 状态栏

2.2.7 层级管理器

层级管理器（Level Manager）用来显示和管理所有使用到的物体和资源，一个工程文件包含一个 Level 项，但可能会包含几个 Scene（场景）项，如图 2.22 所示。

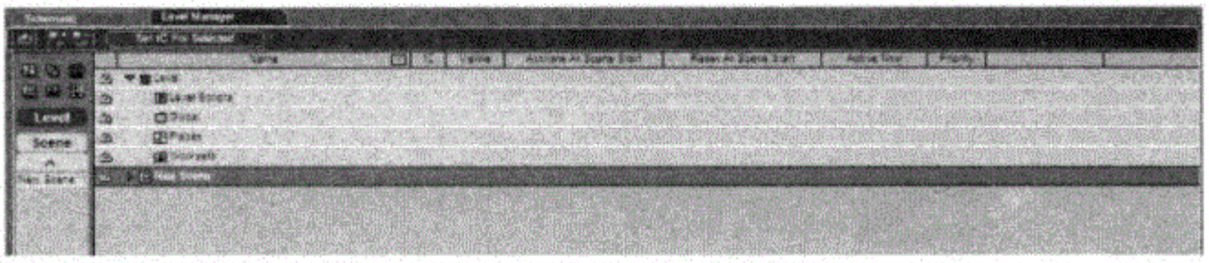


图 2.22 Level Manager

创建工具组主要用来创建无法被渲染的、逻辑上存在的物体，如图 2.23 所示。从左到右从上到下依次为 Place（地点）、Group（组）、Array（数组）、Scene（场景）、Workset（工作组）和 Script（脚本）。

如图 2.24 所示，Level 按钮用来激活并进入 Level 模式，Scene 按钮用来激活并进入场景模式。下方列表显示所有的场景，单击任何一个场景的名字会自动进入 Scene 模式并进入该场景。

图 2.23 创建工具组

图 2.24 场景选择

如图 2.25 所示的物体基本信息，具体包括如下内容：

图 2.25 物体基本信息

- Name 是物体分类名称列表。
- IC（Initial Conditions）表示是否设置了初始状态，若该物体被设置了初始状态，这里会显示叉号图标。
- Visible 表示该物体的可视状态。可以在下面三种状态中切换：可视、不可视，

该物体以及其子物体均不可视。

- Activate At Scene Start 表示是否在场景开始的时候激活。
- Reset At Scene Start 表示是否在场景开始的时候重设初始状态。
- Active Now 显示当前激活状态。
- Priority 为优先级。

2.2.8 脚本流程图

脚本流程图（Schematic）面板用来查看、编辑和 Debug 脚本，如图 2.26 所示。脚本是一个物体“行为”的图解化表示。一个脚本有两个部分，左侧是脚本头，显示了脚本名称，脚本所属的物体等信息；右侧是脚本的本体，由 Building Blocks 构成。关于脚本的书写将在后面的章节系统学习。

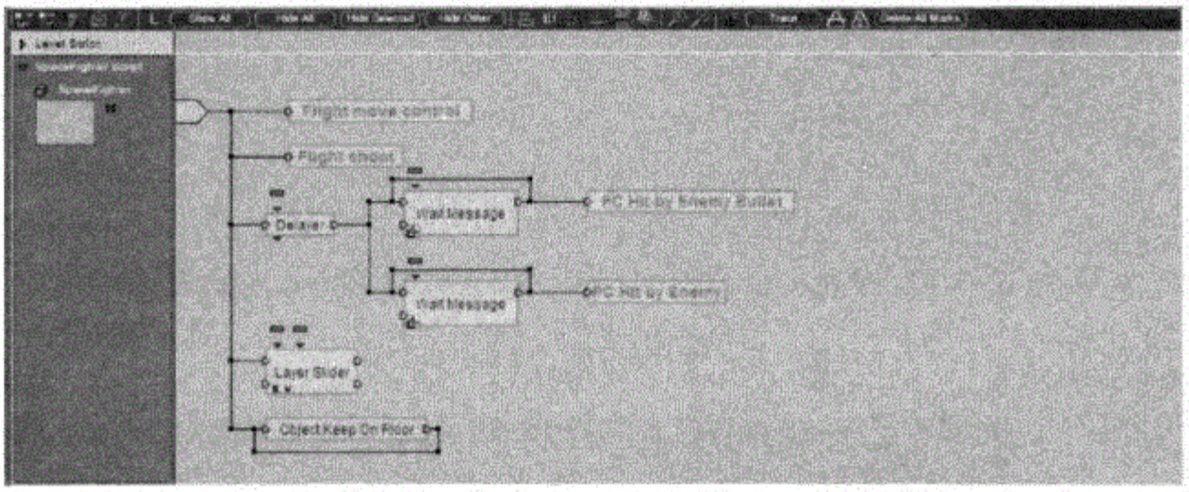


图 2.26 脚本范例

2.3 快速原型制作

Virtools 具有“拖曳式”的资源管理方式、Behavior Graph 脚本编写方式以及“即时调试”的测试方式，可以较快地建立游戏或其他三维互动程序的原型并进行“所见即所得”的测试。

本节将通过一个游戏原型来熟悉 Virtools 的基本操作。练习中不会使用外界的资源和编写代码，只是通过一些 Virtools 的常用功能，来建立一个简单的三维交互原型。

2.3.1 打开 Virtools Dev 4.0

Virtools 默认的界面布局如图 2.27 所示，根据分辨率的不同，各个窗口在尺寸上会略有区别。

如果此前已经使用过 Virtools，界面可能会有所变化，为了方便后面的教学，如图 2.28 所示，执行 Options | Reset Interface 命令，重设界面布局为默认值。

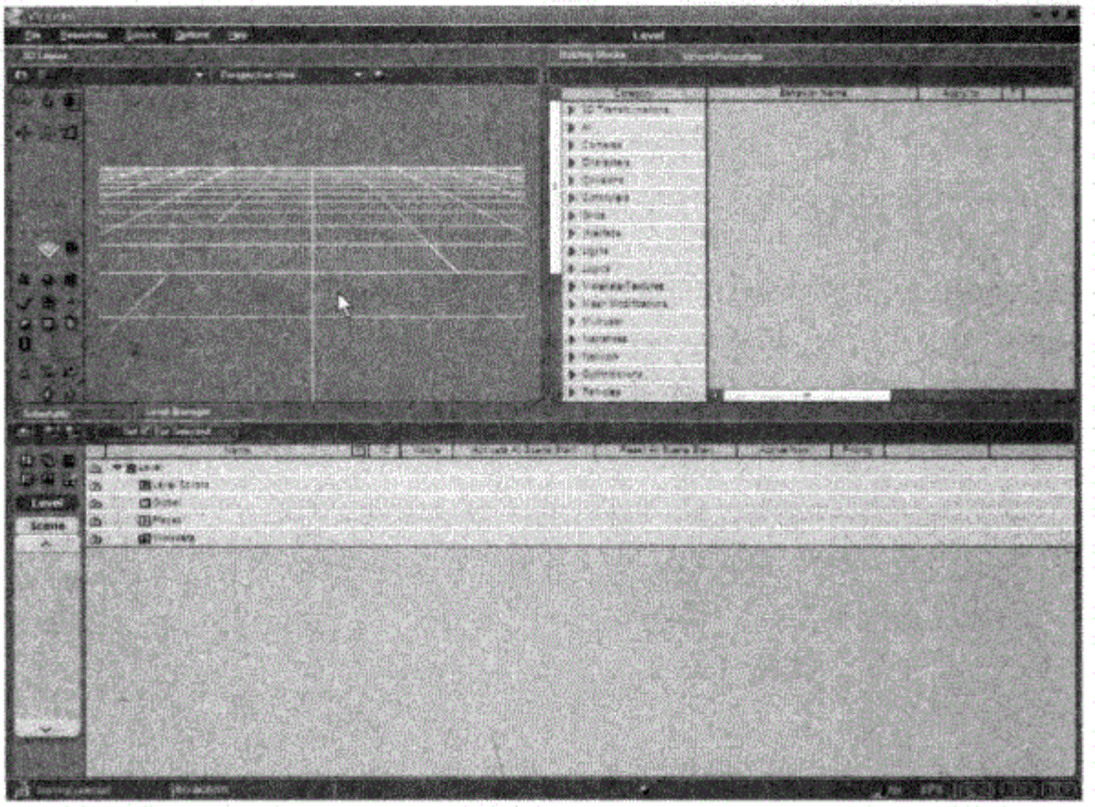


图 2.27 Virtools Dev 4 默认软件界面

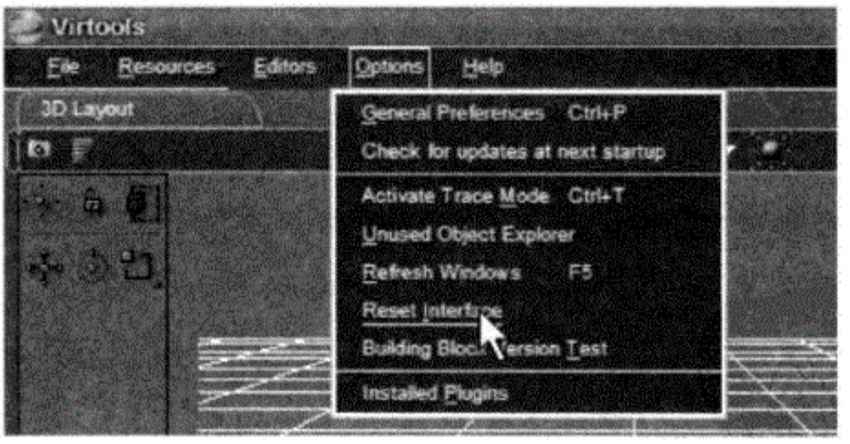


图 2.28 执行 Reset Interface 命令

2.3.2 放置地板和角色

进入 Virtools 软件，左上角是 3D Layout（三维预览）窗口，如图 2.29 所示。在所有三维建模软件，如 3DS Max、Maya 等，都会有类似的布满透视网格线的三维预览窗口。

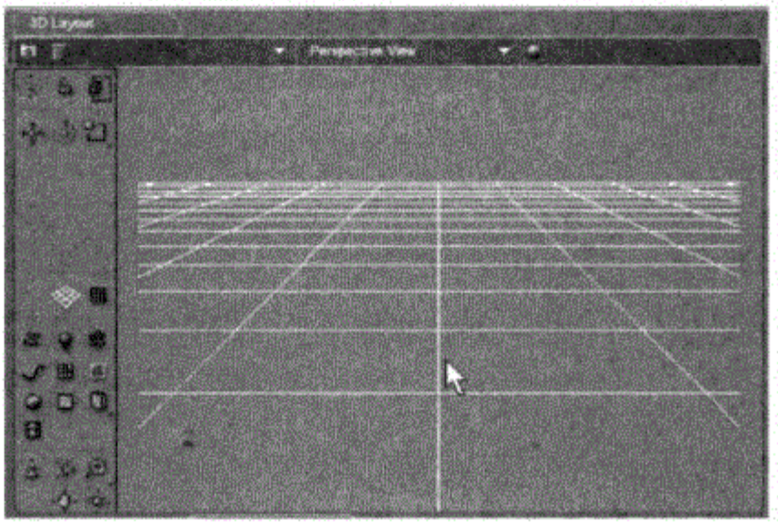


图 2.29　3D Layout 窗口

首先，在界面右上角选择 VirtoolsResources 资源标签，这里面包含了所有 Virtools Dev 4.0 自带的资源。在窗口左侧导航菜单（Category）内，选取 3D Entities | Primitives 选项，并在右侧选择“Plane 10 × 10. nmo”，如图 2.30 所示。

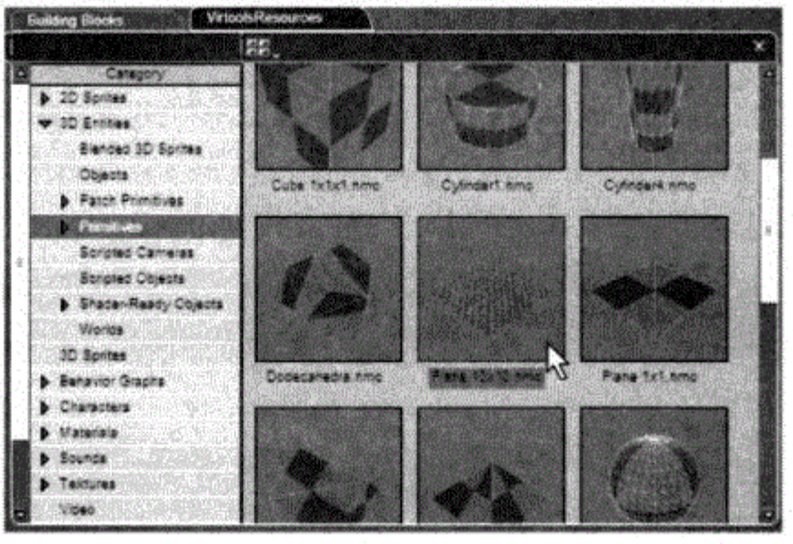


图 2.30　在 VirtoolsResources 面板选取资源

如果没有出现图 2.30 所示的缩略图，需单击资源预览窗口左上角的图标，在下拉菜单中选择 Thumbnails（缩略图）预览方式，如图 2.31 所示。

图 2.31　选择资源预览方式

用鼠标左键单击 Plane 10 × 10. nmo 的图标并拖曳到左侧的 3D Layout（三维预览窗口）中，松开左键后可以看到场景中出现了一个平面，如图 2. 32 所示。

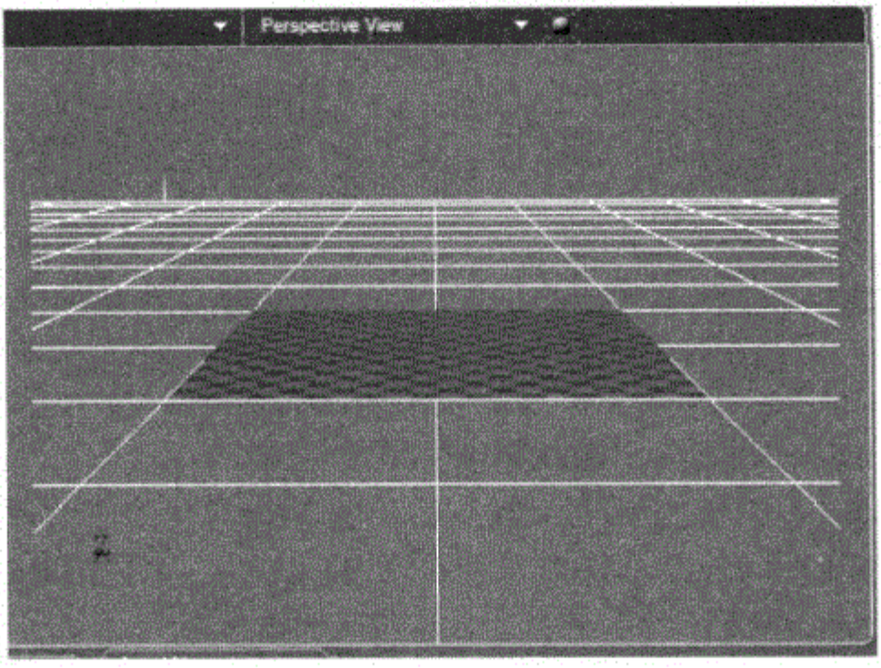


图 2. 32　在场景中加入平面物体

除了透视关系正确外，由于没有光，这个平面还没有三维立体感。通常 3DS Max、Maya 等建模软件都具有默认光源，在建模的时候可以看到较好的视觉效果。但即时渲染的三维引擎通常不会自动添加光源，因此需要手动添加。

在 3D Layout 左侧的工具面板中，单击灯泡形状的图标按钮（Create Light），如图 2. 33 所示，三维世界中出现了一个点光源，界面中白色的灯泡代表点光源在三维空间中的位置，如图 2. 34 所示。

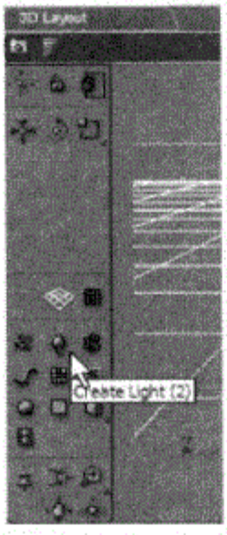


图 2. 33　选择 Create Light 工具

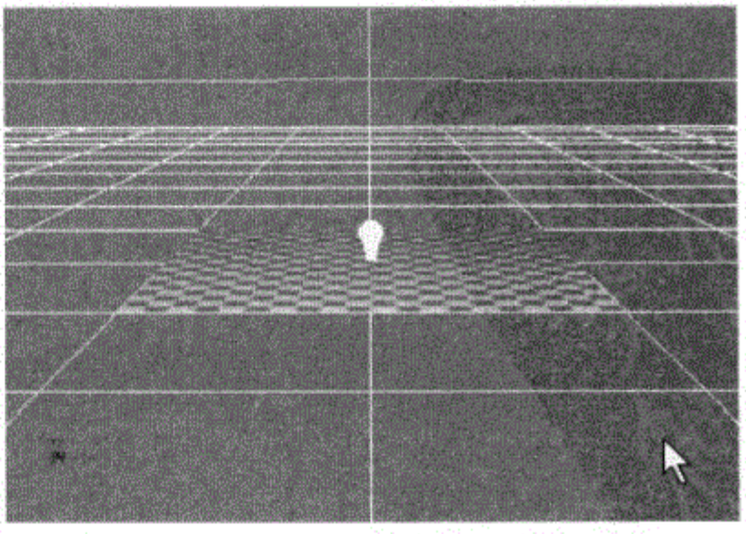


图 2. 34　增加光源后的效果

与此同时下方新出现一个名为 Point Light Setup 面板，该面板用于设置这个点光源的各种参数和属性，新加入的这个光源名字为 New Light（显示在最上方的 Name 栏）。点光源在实时渲染的时候并不会被渲染成一个灯泡形状，灯泡标识只是为了方便对光源进行放置、定位和设定。

由于该平面物体的材质（Material）已经设置了一定的发射光（Emissive Light），因此它可以在无光照的情况下显示出纹理（Texture）。此时需要为它赋予一个地面纹理（Texture），在 VirtoolsResources 面板中执行 Textures | Materials 命令，并选择右侧的 wood floor2 256 ×256. jpg 纹理图，如图 2. 35 所示。

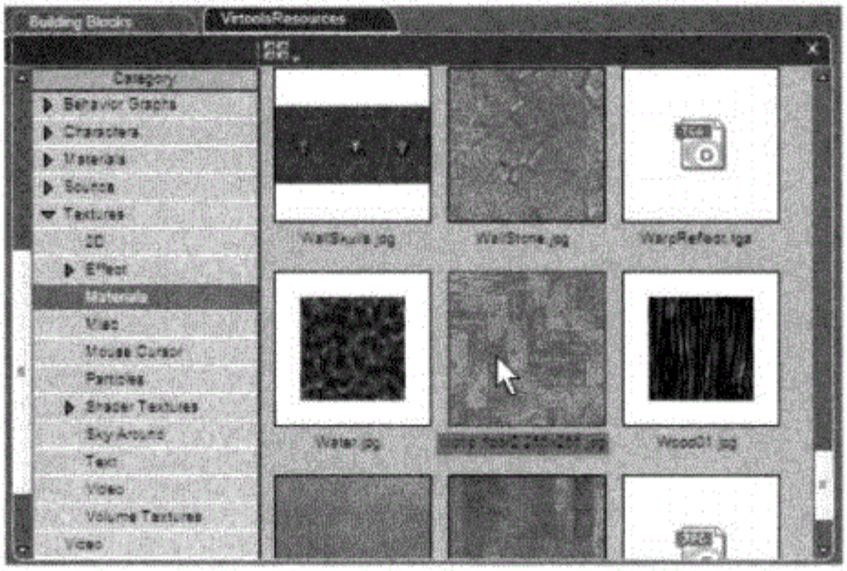


图 2. 35　选择 wood floor2 贴图文件

按住鼠标左键，将它拖曳到左侧三维预览窗口中的平面物体上，在平面显示出黄色外框的时候（表示平面为当前选择目标）松开鼠标，得到如图 2. 36 所示的效果。取消新弹出的 Texture Setup 面板，不做设置。

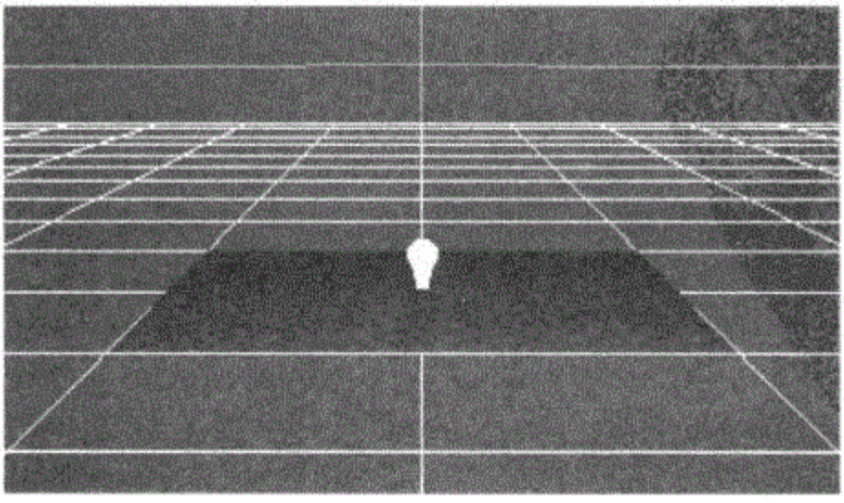


图 2. 36　增加木地板纹理后的效果

如果没有出现贴图，很可能是没有成功将材质拖曳到平面之上（一定要等到平面出现黄色外框后才松开鼠标左键）。没有被拖到平面上的纹理会被保存在当前工程的资源列表里，只是由于它没有被赋予任何材质（Material）因而没有被显示出来。

通过拖曳可以赋予材质，是因为木地板纹理（Texture）和地面存在如下从属关系：地面（3D Object）→Mesh→Material→木地板纹理（Texture）。

下面需要为三维世界添加一个角色，选择 VirtoolsResource | Skin Characters，选择右侧的 Jane. nmo 模型，如图 2.37 所示。

图 2.37　角色资源

将角色拖曳到 3D Layout 中，得到如图 2.38 所示的效果。

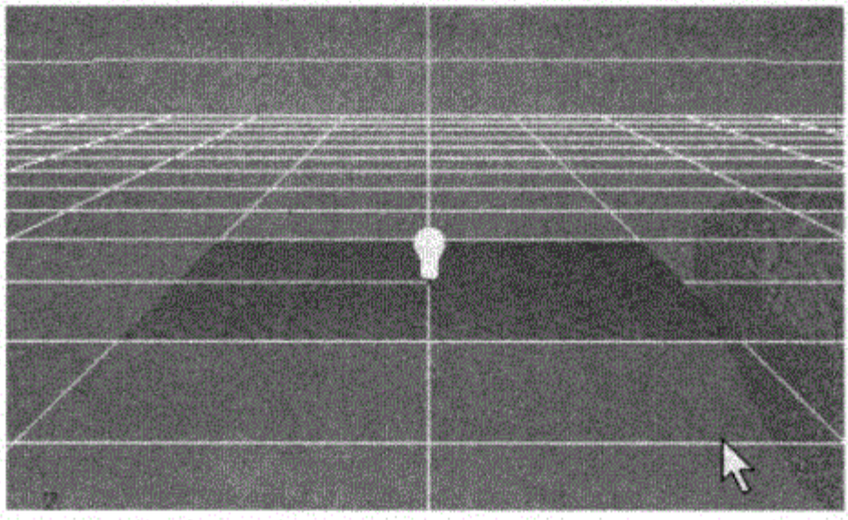


图 2.38　添加角色后的效果

此时角色 Jane 并没有出现在场景中，是因为灯泡挡住了角色 Jane。为了更清晰地看到刚刚添加的角色 Jane，需要调整一下视角。使用 3D Layout 左侧面板中的视角调

整面板，它位于面板的最下方两行。单击放大镜图标可以激活 Camera Zoom 按钮（激活后该图标将变为白色），如图 2.39 所示。

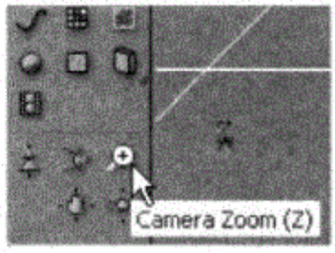


图 2.39　Camera Zoom

在三维预览窗口中按住鼠标左键进行上下拖动可以放大或缩小视角，直到能够清晰地看到角色 Jane，如图 2.40 所示。

由于还没有调整位置，角色 Jane 陷到了地板下方，使用移动工具可以将角色移动到地板上方，移动工具在工具面板的第二行第一个，为十字形图标，与放大镜工具使用方式相同，单击移动工具也可以激活它，如图 2.41 所示。

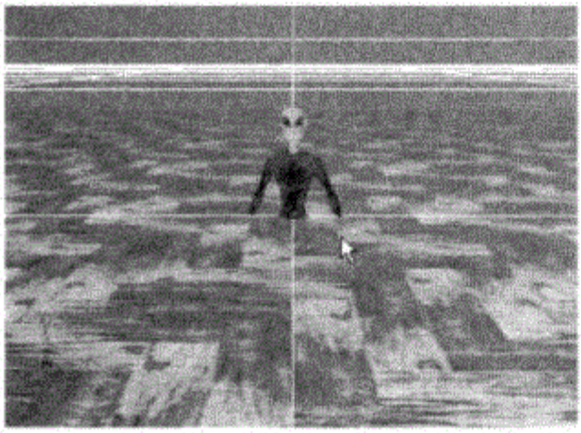


图 2.40　放大后的画面

图 2.41　Select and Translate

激活移动工具（Select and Translate）后，在三维预览窗口单击选中角色 Jane 并进行拖曳，会发现 Jane 是沿着 Z + X 平面移动的，这是由于当前的移动限制模式为 Z + X，移动的限制模式显示在移动工具的下方，此时 Z + X 选项被激活为白色，如图 2.42 所示。

单击上方的 Y 按钮以限制在 Y 轴移动，然后在三维窗口中向上拖曳角色 Jane 直到将其移动到地板上方，如图 2.43 所示。

图 2.42　Constrain Plane

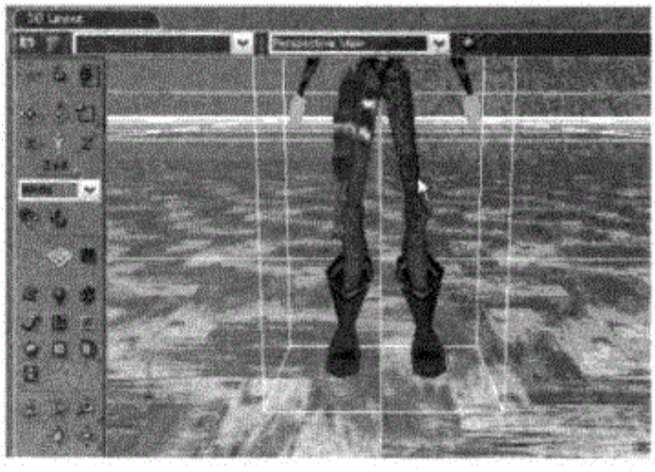


图 2.43　移动 Jane 至地板上方

此时角色 Jane 并没有被显示于画面中央，为了使其居中，需要使用视角调整功能——居中放大。用鼠标左键一直点按放大镜按钮，直到菜单被显示出来，如图 2.44 所示。

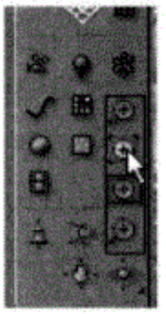


图 2.44 放大工具的其他选项

按住鼠标左键并移动到列表中央带有矩形外框的放大镜图标上，松开鼠标左键，这时画面将以角色 Jane 为中心自动居中，如图 2.45 所示。该按钮的功能是在 3D Layout 中居中放大当前选中的物体，如图 2.46 所示，若没有选中角色 Jane，画面就不会有变化。

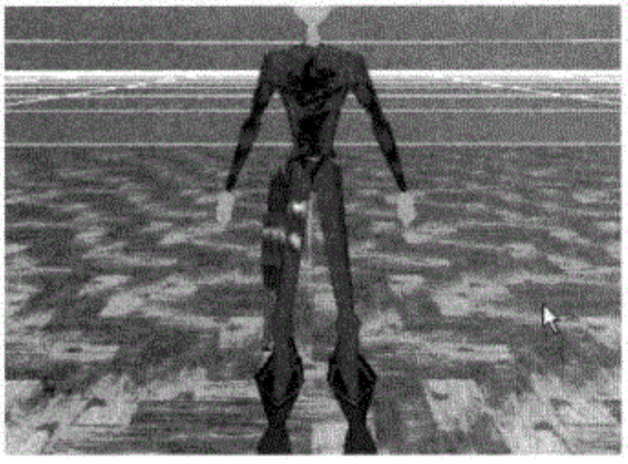


图 2.45 居中放大的效果

图 2.46 选择 Select 工具

单击 3D Layout 并上下滚动鼠标滚轮，这是一种快捷的放大缩小视角的方法，如图 2.47 所示，适当调整视角直到得到比较满意的效果。

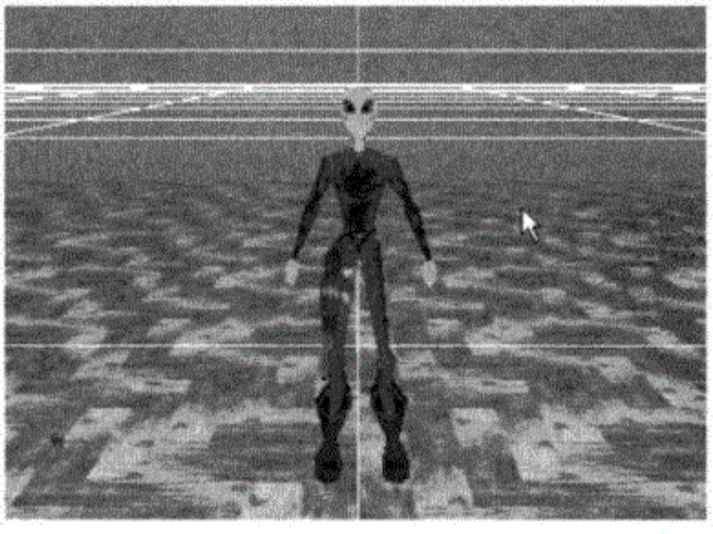


图 2.47 调整视角后的效果

截至到此的工程文件可以在资料盘中找到 QuickStart\QuickStart. cmo。

2.3.3　为角色添加简单的互动操作

放置地板与角色（Jane）后，需要为其添加简单的交互——让用户可以使用键盘控制角色 Jane 在地板上行走。继续使用此前的工程文件，或者在资料盘打开上一节结束时的工程文件 QuickStart/QuickStart. cmo。

为了让角色 Jane 行走，首先需要为其添加基本的行走和站立的动画，在右侧 VirtoolsResource 面板中选择 Characters/Animations/Skin Character Animation/Jane 选项，在其中找到 Stand. nmo、Walk. nmo 和 WalkBack. nmo，并将它们拖曳到角色 Jane 的身上。

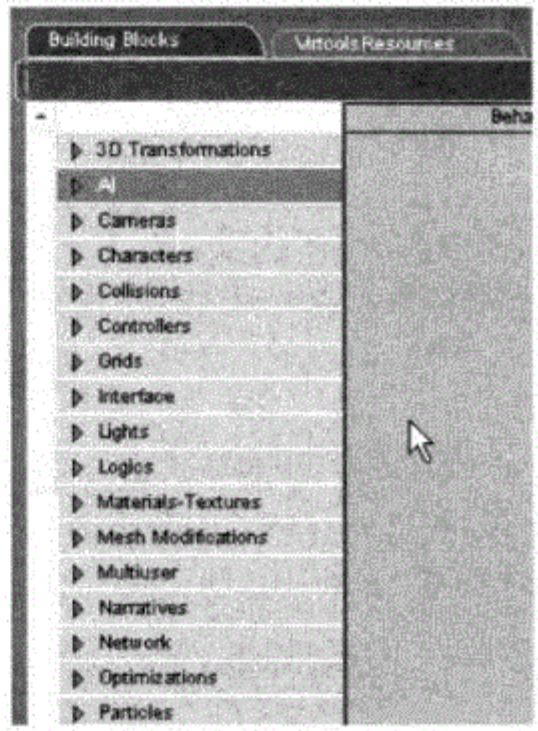


图 2.48　Building Blocks 面板

接下来要写一些简单的脚本来为程序添加互动，Virtools Dev 的脚本书写是通过组合各种 Building Blocks（简称为 BB）脚本模块来完成的。以下步骤主要学习拖曳添加脚本的方法，将不具体介绍每一个 BB 的用法和脚本逻辑。

Virtools 中，所有 BB 都被分门别类地放在了 Building Blocks 面板中，该面板位于软件界面的右侧，单击标签后将会看到如图 2.48 所示的画面。

在 Building Blocks 面板中选择 Characters | Movement 选项，然后选择右侧的 Unlimited Controller 脚本模块，如图 2.49 所示，并将其拖动到角色 Jane 身上。

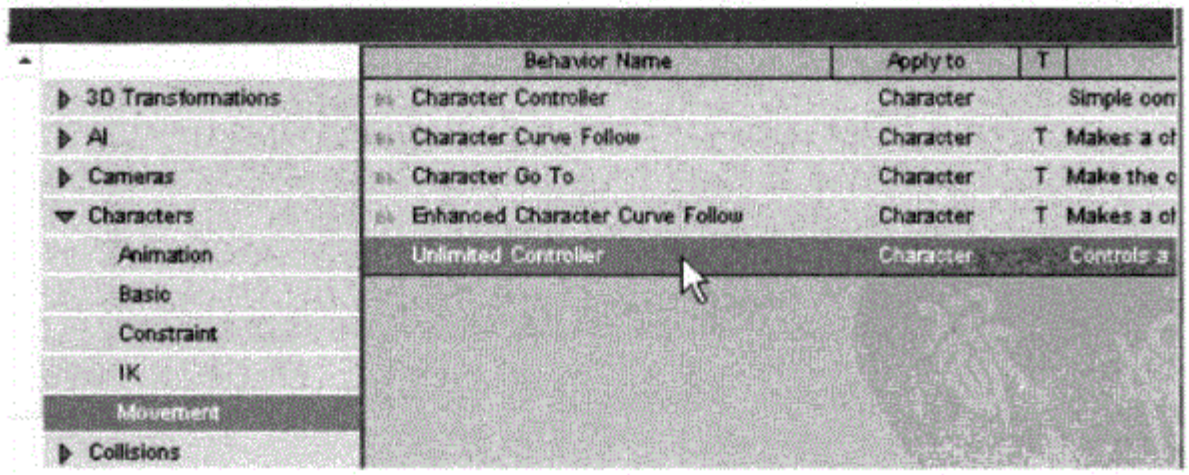


图 2.49　Unlimited Controller 行为脚本模块

在弹出的对话框的表格内，已经有三行数据。在 Message 一栏分别标示为 Joy_Up、Joy_Down 和空，双击任意一行的 Animation 栏可以在下拉列表中为其指定对应的动作，如图 2.50 所示。

图 2.50　Animation 菜单选项

如图 2.51 所示，将每一行的 Animation 设置好后，单击 OK 按钮完成。

图 2.51　Animation 设置结果

接下来在 BB 面板中选取 Controllers | Keyboard | Keyboard Mapper 选项，将其拖曳到 Jane 身上，在弹出的面板上方，单击 Message 选择 Joy_Up，单击 Key 并按下键盘上的 W 键，此时选框中出现了 W 字样，如图 2.52 所示。

图 2.52　输入 Key 按键信息

单击 Add 按钮，下方信息栏会出现一条新的信息，如图 2.53 所示。

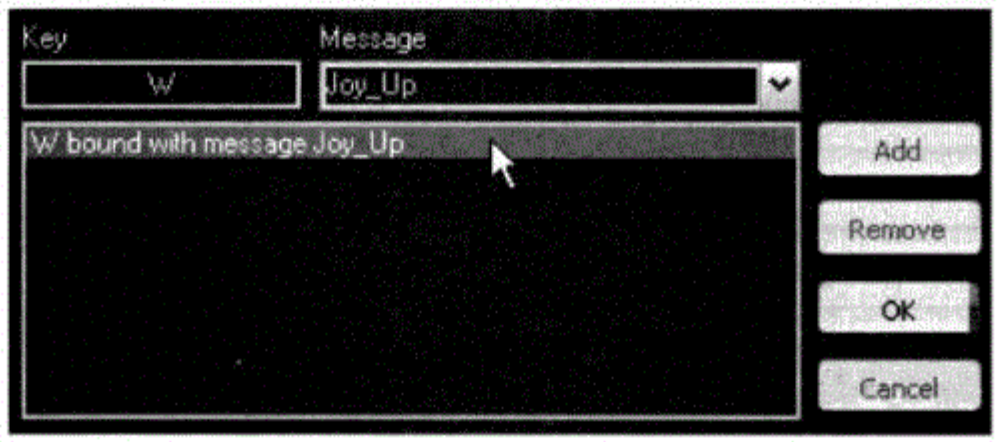


图 2.53　添加新的按键－消息对应关系

这表示已将键盘的 W 键（Key）和 Joy_Up 信息（Message）绑定在一起。按同样的方法将 S 键与 Joy_Down 绑定、A 键与 Joy_Left 绑定、D 键与 Joy_Right 绑定。最后得到如图 2.54 所示的结果（若绑定错误，可以通过 Remove 按钮来移除错误的绑定信息），单击 OK 按钮完成设置。

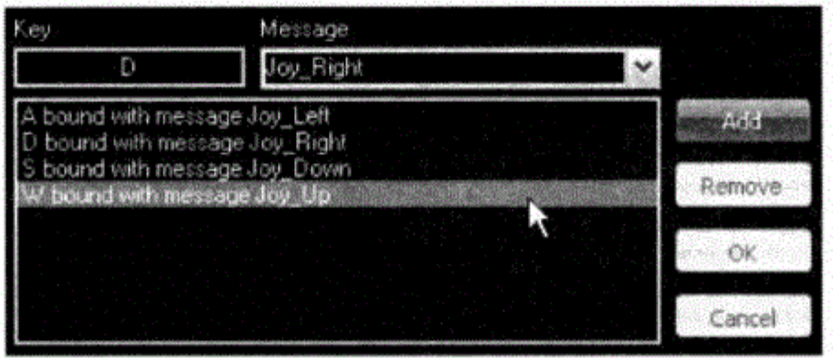


图 2.54　最终按键－信息对应关系图

2.3.4　测试

目前脚本已经写好，可以实际运行程序进行测试，如图 2.55 所示，单击软件右下角的播放按钮。

图 2.55　播放按钮

此时可以看到 3D Layout 中的白色网格线消失了，角色 Jane 的身体也开始左右晃动起来，这实际上就是在播放 Stand 动画，使用键盘上的 W、S、A、D 键可以控制 Jane 的前后移动和转向，如图 2.56 所示。

图 2.56　测试 Jane 行走效果

经过简单的拖曳已经完成本书的第一个三维交互作品。最终完成的工程文件可以在资料盘如下位置找到：CMOs\QuickStart\QuickStart_final. cmo。

2.4　本章小结

本章主要介绍 Virtools 的基本知识和开发流程，并且通过一个简单的案例来体会设置一个角色并进行场景漫游的方法。需要掌握的知识点包括：

（1）Virtools 界面布局和面板的功能；

（2）放置物体与角色的方法；

（3）添加脚本模块的方法。

2.5 课后练习

1. 登录 Virtools 官方网站（http://www.virtools.com）了解更多有关 Virtools 的信息，如它在游戏、网络、虚拟现实方面的应用以及版权条款等。

2. 进一步熟悉 Virtools 的界面，了解 Virtools Dev 4.0 的开发环境。

3. 按照第三节快速原型中介绍的方法，选取其他角色和物体进行交互作品的创作练习。

第3章 游戏原型制作

学习游戏设计，除了必要的游戏创意构思和理论知识外，在制作环节中，充分理解和把握三维游戏制作的方法，结合一定的逻辑与规律推进制作的流程，保持知识点的连贯性和循序渐进是非常重要的。不同的游戏公司使用的游戏引擎也是有差别的，只有掌握了共通性的创作与制作思路，举一反三，才能够为今后使用其他游戏引擎制作三维游戏打下良好的基础。

本章以一个简单的飞行射击游戏原型为例，在制作过程中逐步学习游戏要素的属性和行为的设定方法。

3.1 游戏原型制作初步

本节着重介绍如何从外部的建模软件导出资源到 Virtools 中，并对导入的资源进行基本的调整和设置，同时介绍最基本的 Virtools 脚本的书写。

3.1.1 导出 NMO 文件格式

为了将 3DS Max 中的模型导入 Virtools 软件，首先需要到 Virtools 官方网站（www. virtools. com）下载相应的导出插件。如 3DS Max 的插件为 VIRTOOLS 4.0 FOR MAX，下载后需要安装在 3DS Max 的 plugins 文件夹中。这里以 3DS Max8 为例，其他版本的 3DS Max 和其他软件（如 Maya）在方法上大同小异。

（1）在资料盘内找到如下文件（CMOs\Plane\Plane. max），使用 3DS Max 打开后，可以看到一个飞机的模型。

（2）可以使用快捷键 Ctrl + A 全选飞机，然后选择 File 菜单下的 Export Selected（导出选中物体），在弹出的对话框内，选择文件类型选项中的 Virtools Export（只有正确安装插件才会出现该选项），指定好保存地点和文件名后单击“保存”按钮，如图 3.1 所示。

（3）单击“保存”按钮后，将出现 Virtools Export 面板（如图 3.2 所示），保留其他设置为默认值，只对两项进行调整：

图 3.1 指定导出文件目录

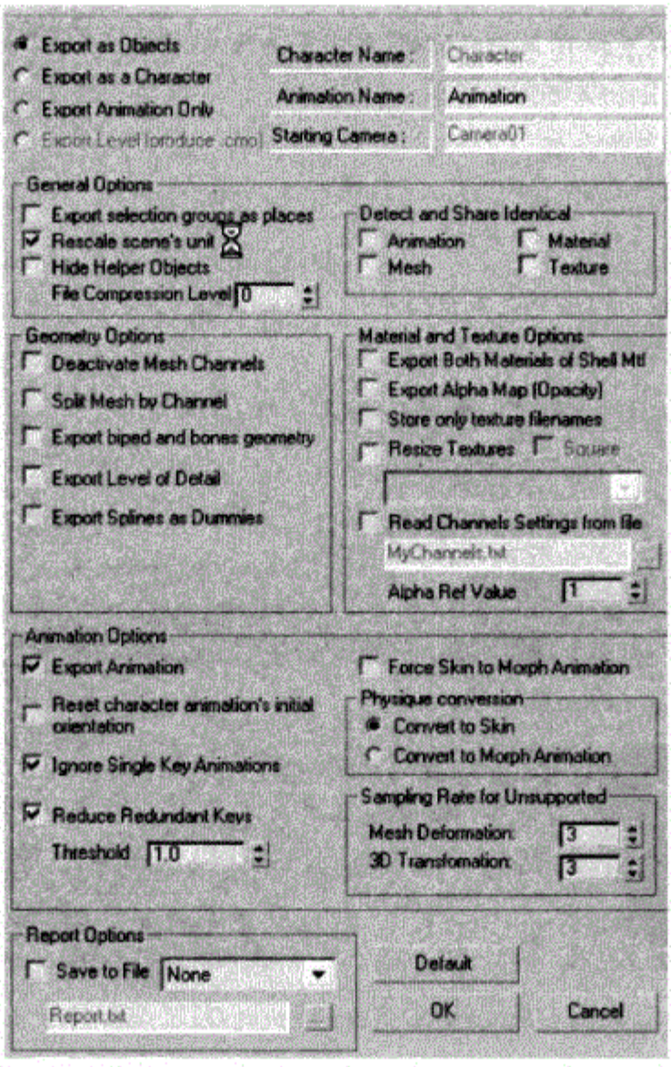


图 3.2 导出选项菜单

- Export as Objects（导出为物体）：由于飞机并不会作为一个 Character（角色）来使用，也没有骨骼动画，因此不将其导出为角色、动画。

- General Options 中的 Rescale scene's unit（重设场景尺寸）：不选此项，导出的物体将按照 3DS Max 里面的尺寸经过换算导入到 Virtools 中，由于这个飞机在建模时使用的单位是“米”，如果不进行重设，导出的模型在 Virtools 里会显得非常大。

图 3.3 导出得到的 NMO 文件

（4）单击 OK 按钮完成导出。导出成功后，便会在选定的目录下看到一个后缀为 NMO 的文件，如图 3.3 所示。

3.1.2 导入 Virtools 并进行调整

1. Virtools 导入 NMO 文件

打开 Virtools 软件，执行 Resources | Import File 命令，在对话框中选定之前导出的 Plane. NMO，然后单击打开，完成资源的导入。此时 3D Layout 中将出现一个黑色的飞机外形，这就是已导入的飞机模型，如图 3.4 所示。

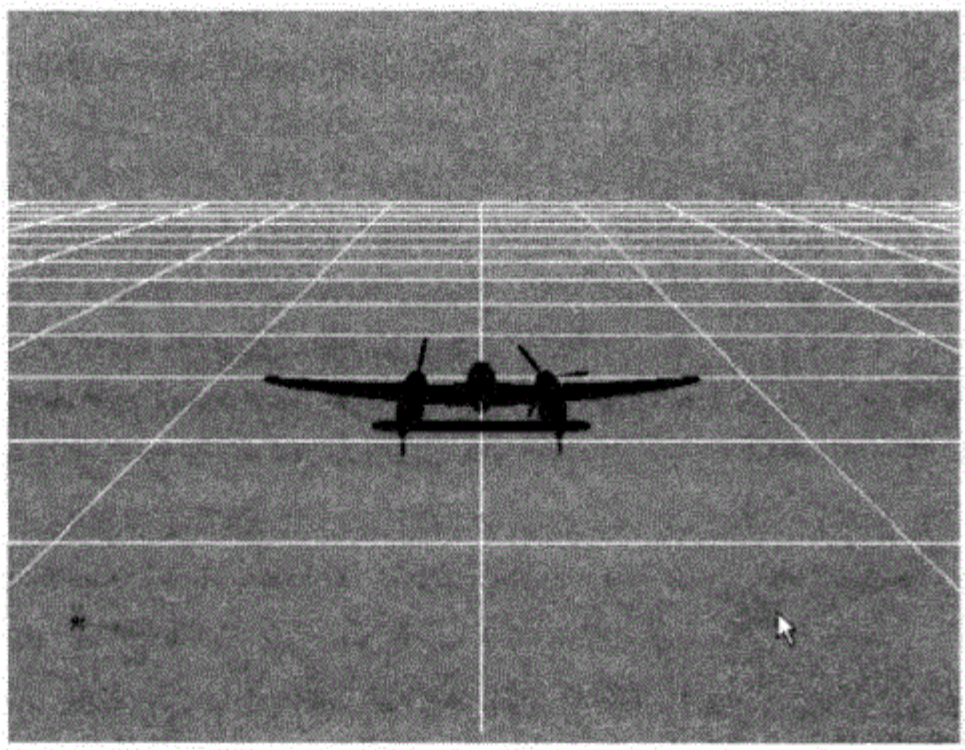


图 3.4 导入飞机模型

2. 添加光源

由于在 3DS Max 建模的过程中，没有设置自发光，在没有光源的情况下，飞机呈现全黑的状态，因此要为 3D 世界中增加一个光源让飞机正常显示。单击 3D Layout 左侧工具面板中的灯泡按钮 Create Light，创建一个新光源，如图 3.5 所示。

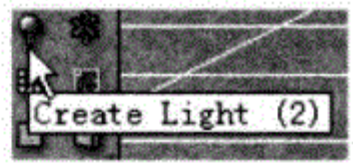


图 3.5 Create Light

新建一个光源后，飞机便显示出本来的色泽，如图 3.6 所示。

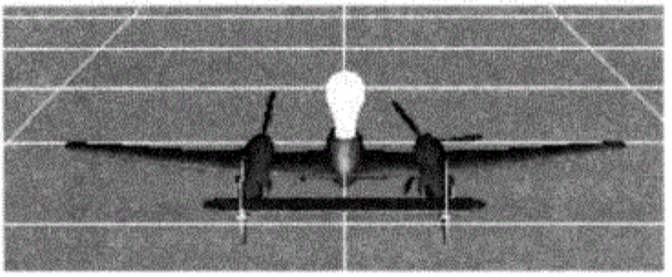


图 3.6 增加光源后的效果

利用视角调整工具和移动工具，将 Perspective 视角调整到飞机上方，并移动新建的光源到可以照亮飞机上部的位置，效果如图 3.7 所示。

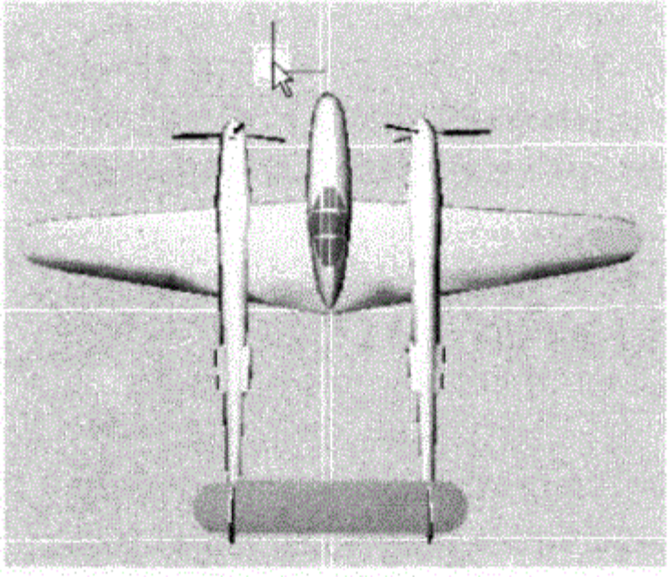


图 3.7 移动光源照亮飞机

进行到此步骤的工程文件可以在 CMOs\Plane\Plane_01. cmo 找到。

3. 光源的设置

从 3DS Max 里面导入到 Virtools 之后，模型机翼和尾翼上的贴图没有显示出来，这并不是 Virtools 不支持纹理贴图的导入，主要原因是 3DS Max 中并没有给材质（Material）的 Diffuse 通道指定 Bitmap 纹理贴图，而是使用了 Max 自带的 Checker 类型 Diffuse 贴图，这种纹理贴图（除了 Bitmap 外，所有由 Max 动态生成的纹理贴图）在导出的时候，并不会被自动生成为图像文件并随模型导出，因此在 Virtools 中不能显示，需要手动来指定贴图。

此外，在 3DS Max 中可以看到的金属高光，在这里也消失了。由于 Virtools 是实时渲染的引擎，3DS Max 中使用 Shader 实现的 Metal（金属）材质是不能被直接导入到 Virtools 中的，而只能保留一些基本的材质信息，虽然 Virtools 也有模拟金属效果的 Shader，但是现在可以用一种更简单的方式来模拟金属高光的效果，同时也是更高效

的模拟方式——使用 Specular 光。在 Point Light Setup 面板或右击灯泡选择 Point Light Setup（New Light）Return，进入刚刚创建的点光源设置面板，如图 3.8 所示。

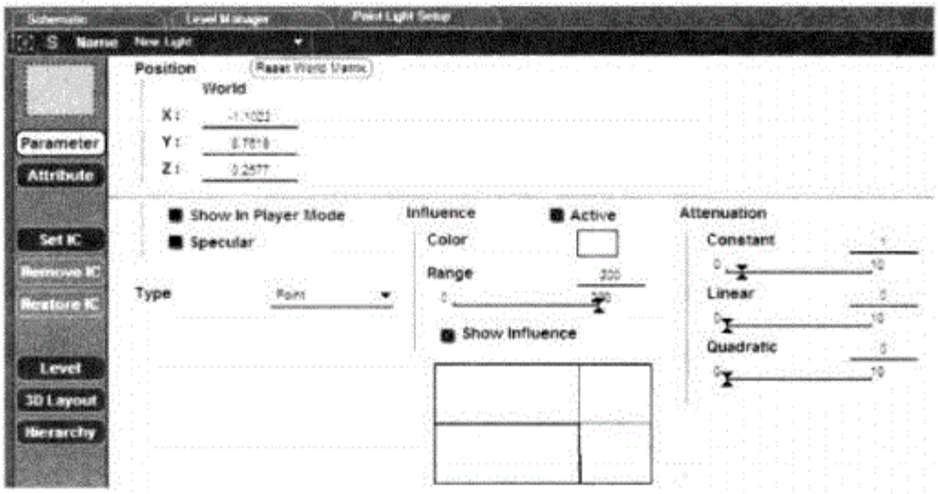


图 3.8 光源设置面板

首先要为每一个新建资源设置独特的名称，以方便后面的使用。单击左上方 Name 中的 New Light，然后按 F2 键重新设置这个光源的名字，在这里将其命名为 Global Point Light，如图 3.9 所示。

默认的光源是不支持 Specular 光属性的，需要手动打开它，如图 3.10 所示，选择 Specular 复选项。

图 3.9 重命名光源

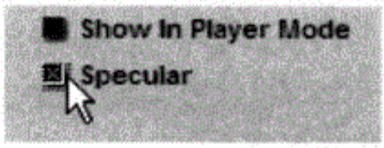


图 3.10 选择光源的 Specular 复选项

飞机的机身立刻出现了白色的高光，如图 3.11 所示。

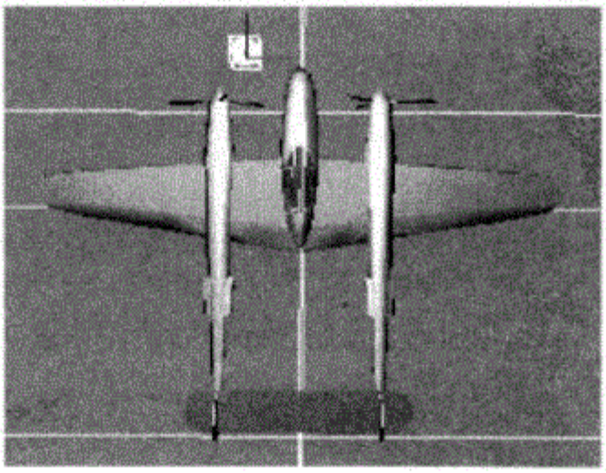


图 3.11 具有高光效果的飞机模型

反复开启该按键进行效果的对比。使用 Specular 属性来模拟简单的金属效果是一种常用的技法，此外还有很多方法可以尝试。

4. 文件的整理

通常情况下，从 3DS Max 导入 Virtools 的资源或多或少总是会有一些问题，如多余的资源文件，杂乱无章的命名（这种情况在多人合作的时候会经常出现），因此需要在导入新资源后对它们进行一番整理。

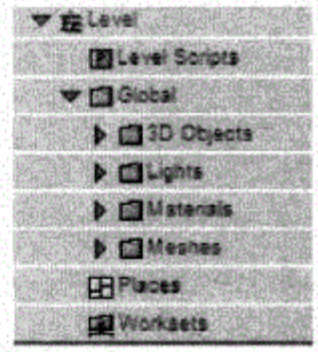


图 3.12 Level Manager 面板选项

首先进入下方的 Level Manager 面板，可以看到 Global 目录下已经出现了四个新的目录，如图 3.12 所示。

打开 3D Objects 目录可以看到五个文件名，它们分别对应飞机的五个三维形体（3D Object），单击任何一个物体，可以在 3D Layout 中选中该物体，以此能够得知每一个物体对应飞机的哪一个部分。这样的命名并不那么好记，所以需要根据物体的属性为它们重新进行命名。

右击选中 Rename 或按快捷键 F2，分别命名：gondola 为“飞机 机身”、port propeller 为“飞机 左侧螺旋桨”、port spinner 为“飞机 左侧螺旋桨轴”、starboard propeller 为“飞机 右侧螺旋桨”、starboard spinner 为“飞机 右侧螺旋桨轴”，重新命名后的效果如图 3.13 所示。

因为 3D Objects 拥有 Mesh，而 Mesh 又拥有 Material，所以 3D Objects 重命名后，为了方便以后的使用，还应为 Mesh 和 Material 命名。此时不是在 Level Manager 里面直接改名，而是要利用 3D Object→Mesh→Material 的层级关系，为每一个 3D Object 所属的 Mesh 和 Material 重命名。

首先以“飞机 右侧螺旋桨（3D Object）”为例，在 Level Manager 中双击该物体，进入它的设置页面，注意 Object Meshs 一栏，内有如图 3.14 所示的信息。

图 3.13 重命名后的效果

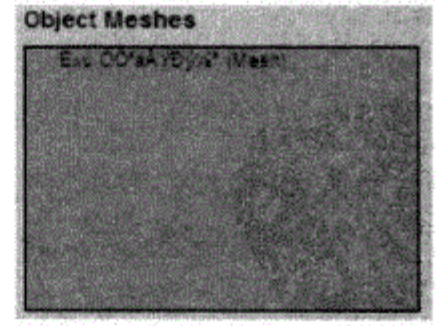


图 3.14 Object Meshs 选项

中文在 Virtools 中是“部分支持”的，此时会显示为乱码，不过这并不妨碍操作。Object Meshes 栏中列出的是这个物体所有的 Mesh（网格）。一个物体可以拥有多个 Mesh，但是每一个时刻，只能指定其中一个 Mesh 为当前激活状态，如图 3.15 所示，

右击 Mesh 的名字，然后在菜单中选择 Set As Current 为当前激活的 Mesh。

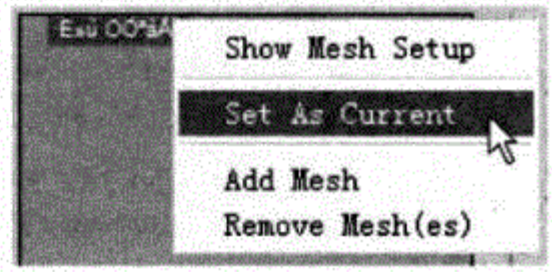


图 3.15 选取 Set As Current 命令

由于目前只有一个 Mesh，设置时不会有任何效果。“多 Mesh”的方式可以被灵活应用在很多种情况。例如模拟汽车逐步被毁坏时，可以为每一种毁坏状态的汽车制作单独的 Mesh，然后在需要的时候调用。

接下来对这个 3D Object 所拥有的 Mesh 进行设置，只需双击 Mesh 的名称（在这里是乱码）就可以打开 Mesh 设置面板，将其命名为与 3D Object 同样的名字以方便辨认。

在 Mesh 设置面板中，右侧的 Materials Used 栏列出了该 Mesh 用到的所有 Materials，如图 3.16 所示。

同理双击该 Material，进入材质设置面板，可以看到该材质已经被命名为“螺旋桨扇叶”，这个名字较好地表达了它的作用，因为它并不仅用作右侧扇叶的材质，还同样作为左侧扇叶的材质，可以在 Used By 栏中看到，它被三个 Mesh 所使用，如图 3.17 所示。

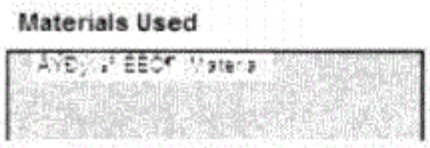


图 3.16 Materials Used 栏

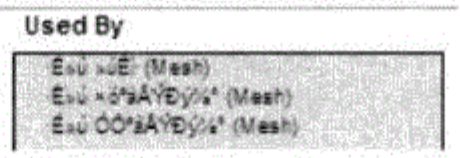


图 3.17 Used By 栏

通过双击 Mesh 名称可以进入相应 Mesh 的设置面板。利用 Used By 和... Used 列表就可以依照 3D Object→Mesh→Material 的层级关系来浏览相应资源，根据资源的所属关系，可以依照自己的习惯为它们命名，重命名后的效果如图 3.18 所示。

图 3.18 重命名后的效果

此时在 Materials 列表中有大量没有使用到的、名为“Material #..”的材质，这些材质可能是在建模过程中创建、但没有使用到的材质球，通过双击这些材质可以看到它们的 Used By 一栏是空的，如图 3.19 所示。

图 3.19 没有被引用的材质的 Used By 栏

删除没有被任何 Mesh 使用的材质，在 Level Manager 中按住 Shift 键将它们全部选中，按 Delete 键后弹出一个 Select Delete Options 窗口，如图 3.20 所示。

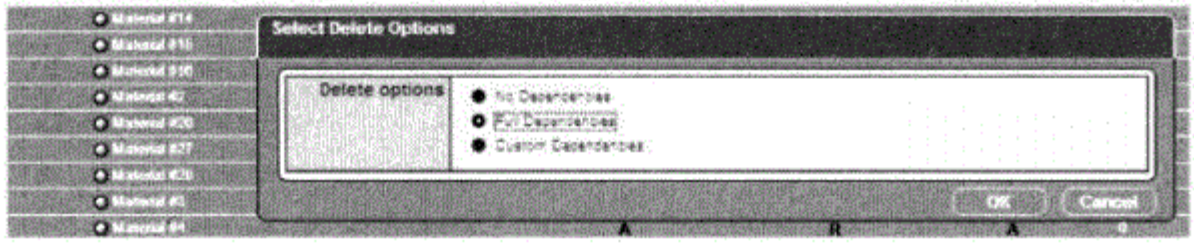


图 3.20 删除选项

在这里有三个选项，用来指定进行删除操作时如何处理该物体下层的继承对象：No Dependencies 为无继承（仅删除当前文件）、Full Dependencies 为全部继承（删除当前文件及其继承的所有子文件）、Custom Dependencies 为人工指定继承关系。

对于 Material 而言，它下层的继承对象只有一个 Texture（纹理贴图），而这些 Material 没有被赋予任何纹理，因此选择任何一个选项都是一样的，单击 OK 按钮完成删除，可以看到飞机并没有任何变化，这是因为这些材质并没有实际应用于物体之上。目前已完成了导入模型的整理工作，如图 3.21 所示。进行到此步骤的工程文件可以在 CMOs\Plane\Plane_02. cmo 找到。

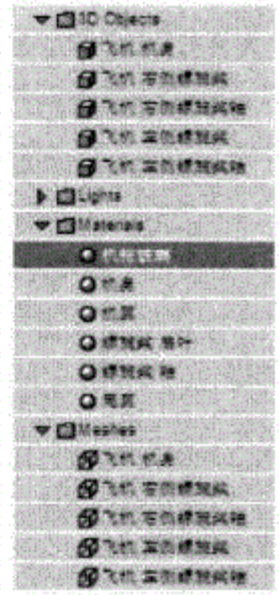


图 3.21 整理完毕的名称列表

5. 添加贴图

这一节中将为机翼和尾翼添加纹理贴图。首先需要创建一个资源库用来存放纹理贴图文件。在菜单中选择 Resource | Create New Data Resource 命令，然后指定创建资源库的目录以及资源库的名称，在这里将其命名为 MyResource，然后将 CMOs\Plane\目录下的两个图片（JPG 格式）文件 wing. jpg 和 wing tail. jpg 复制到“资源库所在目录/资源库文件夹/Textures”目录下。

返回 Virtools 中，在右侧的 MyResource 面板 Textures 目录下就可以预览到两张纹理，如图 3.22 所示。若找不到该面板，可能是由于面板过多导致无法一起显示，单击标签两侧的左右箭头可以显示其他面板。

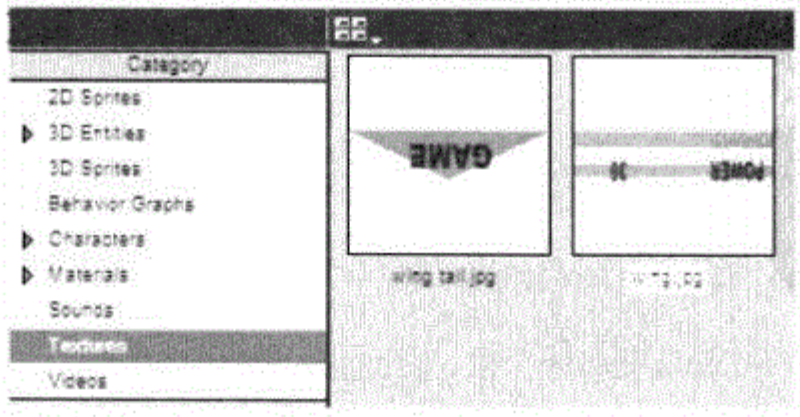


图 3.22 选择机翼纹理贴图

首先为机翼设置材质，这里使用拖曳的方式，从 Resource 面板中选中 wing. jpg 并拖曳到 3D Layout 中的机翼上释放鼠标，将该纹理赋给机翼材质，如图 3. 23 所示。

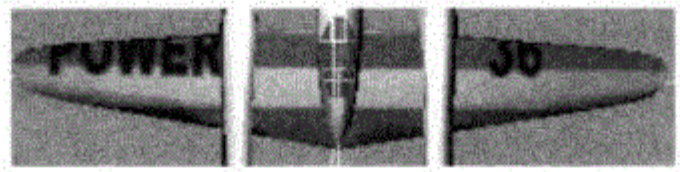


图 3. 23 增加贴图后的机翼

此时在弹出的材质设置面板中，“机翼” Material 中的 Texture 项已经被指定为 wing，单击 Texture 左侧的三角箭头可以进入 Texture 的设置面板，如图 3. 24 所示，保留 Texture 的设置不变。

图 3. 24 贴图设置面板

接下来为尾翼设置纹理贴图，并学习一种新的添加资源的方法——使用 Level Manager，这种方法虽然不如拖曳那样简便，但是在处理复杂工程的时候拖曳往往会造成诸多麻烦，比如不慎拖曳到错误的物体之上，而此种方法则不会。

首先打开 Level Manager，然后拖曳 wing tail. jpg 到该面板后释放鼠标，此时并没有什么变化，这是由于虽然将 Texture 添加到了工程中，却没有为它赋给任何材质。在 Level Manager 的 Texture 文件夹中可以找到 wing tail. jpg 的名称，如图 3. 25 所示。

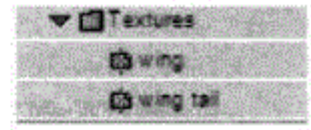


图 3. 25 Texture 目录

双击该 Texture 进入设置面板可以看到 Used By 一栏是空的，表示它没有被用于任何材质。

在 Level Manager 中双击打开尾翼（Material）的设置面板，也可以通过在 3D Layout 中单击右键选择尾翼，然后在弹出的对话框中选择 Material Setup 进入该设置面板。此时 Texture 一栏显示为“—None—”这表示没有使用材质，单击选择 wing tail，将纹理赋给该材质，效果如图 3. 26 所示。

图 3.26　添加贴图的尾翼

6. 设置材质颜色

现在已经成功添加了飞机的纹理，还需要重新调整整体的颜色。首先单击材质设置面板左上角的 Name 栏，在下拉菜单中选中需要进行设置的材质，进入机身的材质设置面板，如图 3.27 所示。

注意左侧的四个色彩设置选项，如图 3.28 所示。

图 3.27　材质设置面板 Name 下拉菜单

图 3.28　材质颜色通道设置选项

其中，Ambient 为环境色——是在阴影环境下表现出的色彩；Diffuse 为物体本来的颜色；Specular 为金属高光的颜色（Power 为光强，但并不是一个线形增加的函数，可以通过滑动条来测试其效果直到达到一个满意的数值）；Emissive 为发射光颜色，表示物体自身所发射出的光色。此时为纯黑色表示物体不发光，机身会有很强的明暗过渡，设置其为纯白色将发现机身变为白色，如图 3.29 所示，这是由于此时机身已经被视为发光体，无法通过外界光源来显示阴影效果。

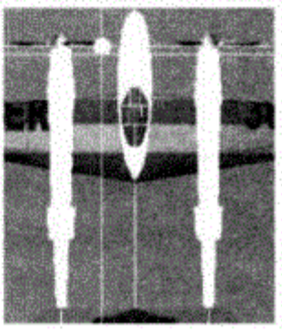


图 3.29　纯白发射光

回到原始状态，如图 3.30 所示。现在机身主要存在的问题是：机翼和尾翼的颜色不和谐，需要通过修改 Ambient 和 Diffuse 来调整；高光面积过大，可以通过 Specular 的 Power 项来调整；阴影过深，主要是由于点光源无法照到机身下方导致的。在真实情况下，由于漫反射的存在，物体的阴影一般不会显示为纯黑色，特别是翱翔于天空之上的飞机，需要为其指定一些发射光，即 Emissive。

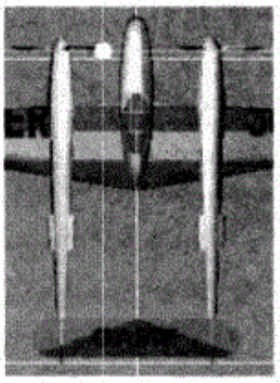


图 3.30　原始发射光

尝试调整相关选项，调整后的颜色通道如图 3.31 所示，最终效果如图 3.32 所示。

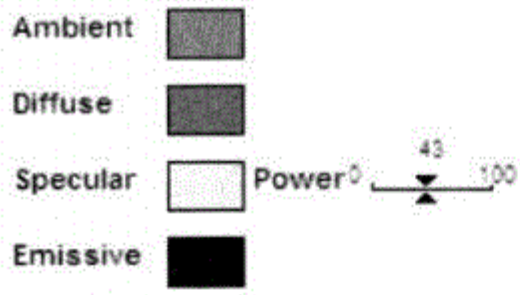


图 3.31 调整后的颜色通道

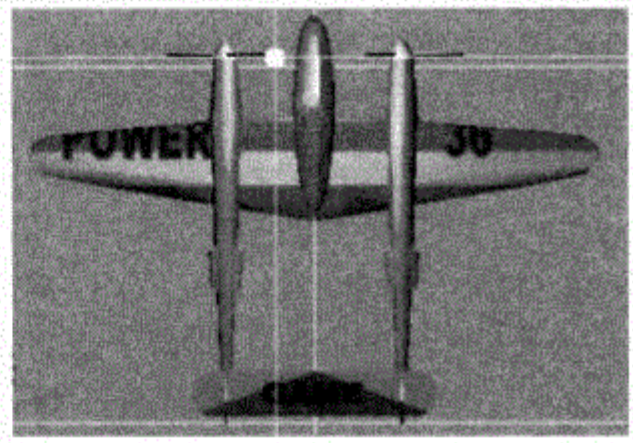


图 3.32 调整后的机身效果

此时飞机的背面可以看到发射光（Emissive）的效果，如图 3.33 所示。由于加入了深灰色的 Emissive，飞机的背面即使没有光照也不会显示为纯黑色。

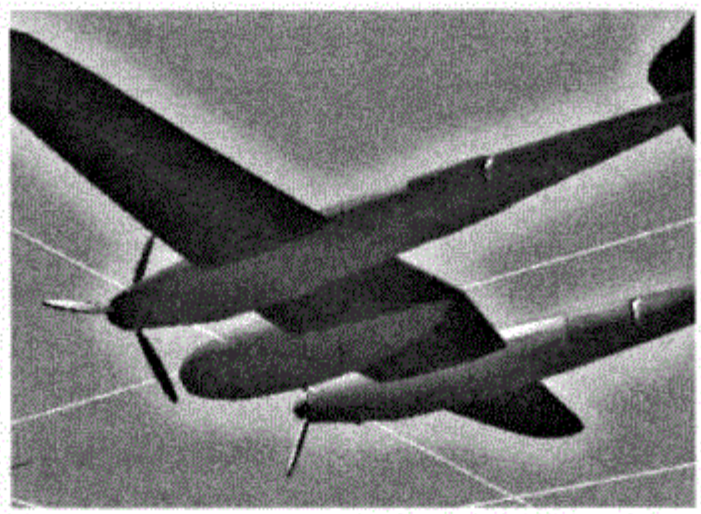


图 3.33 机身背面发射光效果

下面进入机翼 Material 的设置面板，注意到 Ambient 和 Diffuse 是灰色的，这是机翼颜色“发灰”的原因，将其改为纯白色可以清晰地看到机翼的色泽，如图 3.34 所示。

图 3.34 更改 Diffuse 颜色通道后的机翼

图 3.35 Specular 选项

现在机翼还没有金属光泽，这是由于 Specular 颜色被指定为黑色的缘故，如图 3.35所示，调整设置直到满意为止。

同理完成尾翼和螺旋桨的设置，并可以根据情况适当调整机身达到整体效果的和谐，机身最终效果如图 3.36 所示。

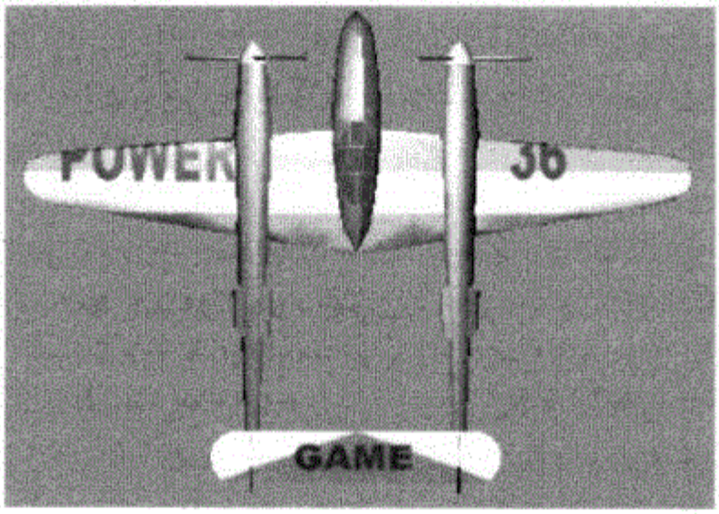


图 3.36 机身最终效果

最后设置机舱玻璃，注意 Diffuse 色旁边出现了 Alpha 滑动条，如图 3.37 所示。

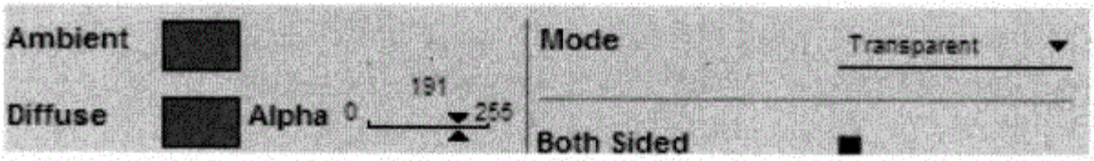


图 3.37 调整材质 Mode 为 Transparent

这是由于该材质的 Mode（模式）为 Transparent，表示该材质是透明的。通过调整 Diffuse 通道的 Alpha 值可以调整其透明度，如图 3.38 和图 3.39 所示。调整 Alpha 和其他颜色通道直到效果满意为止。

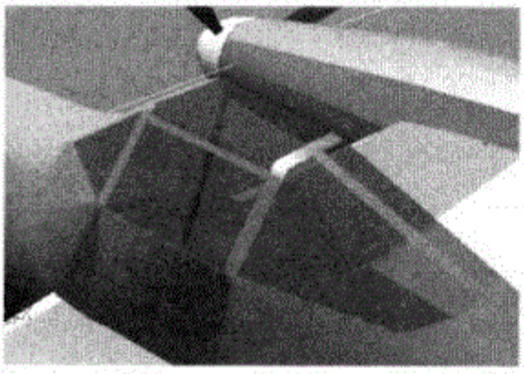


图 3.38 当 Alpha = 191 时的机舱玻璃

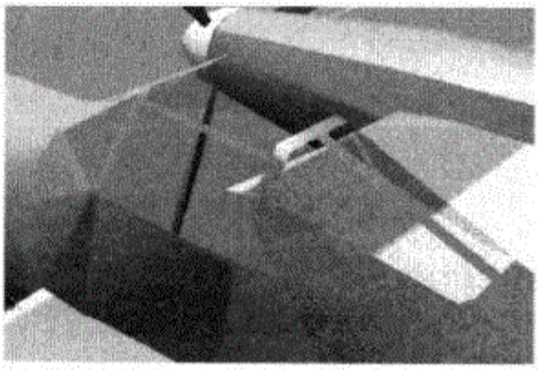


图 3.39 当 Alpha = 67 时的机舱玻璃

此时虽然玻璃是透明的，但是背面的玻璃和玻璃窗框并不能被正常显示，需要将机舱玻璃材质和机身玻璃窗框材质设置为 Both Sided（双面显示）模式，如图 3.40 所示。最终的效果如图 3.41 所示。

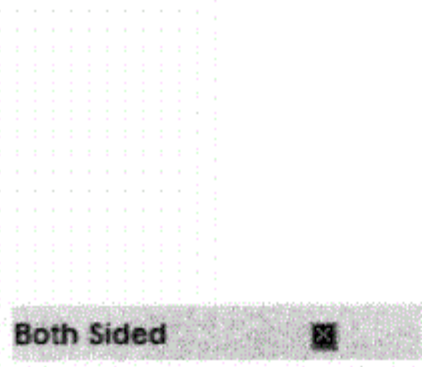


图 3.40 选择 Both Sided 选项

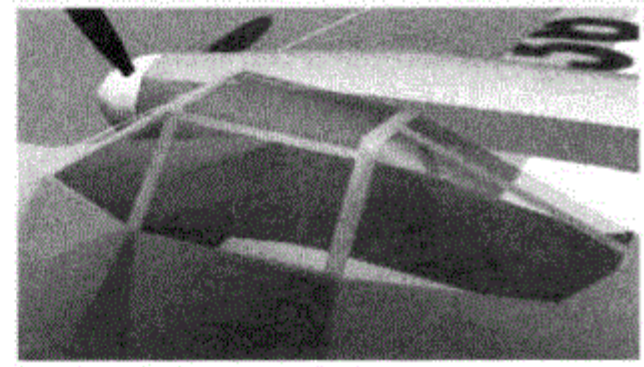


图 3.41 最终机舱玻璃效果

至此，通过调整各个色彩通道，利用 Specular 模拟金属光泽，使用 Transparent Mode 模拟玻璃透明效果，使用 Both Sided 选项双面显示材质，基本完成了飞机色彩的调整，最终效果如图 3.42 所示。

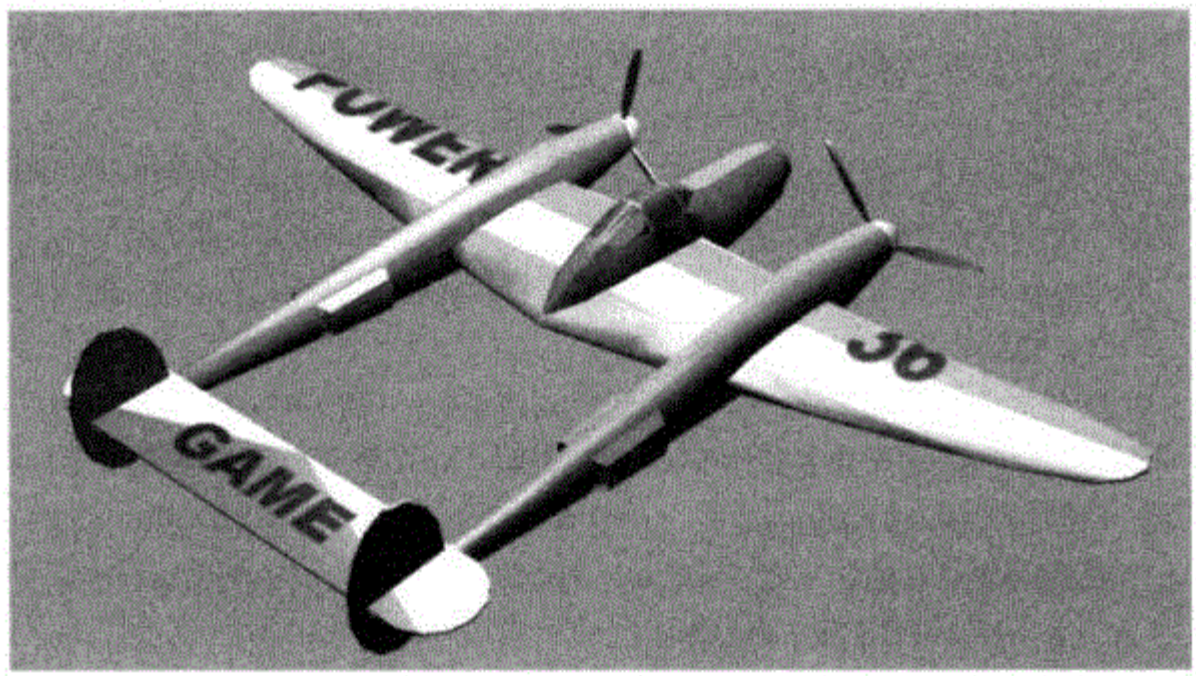


图 3.42 最终飞机模型效果

到目前为止的工程文件可以在 CMOs\Plane\Plane_03. cmo 找到。

3.2 Virtools 脚本初步

下面为飞机撰写第一个互动脚本，但此前需要做一些准备工作，以便脚本书写顺利进行。

3.2.1 设置直接的继承属性

目前的飞机实际上是由五个 3D Object 构成的，分别为机身和左右侧螺旋桨和螺旋桨轴，它们之间没有任何继承和从属关系，因此若移动机身，会发现螺旋桨和螺旋桨轴并没有跟随机身一起移动，如图 3.43 所示，按 Ctrl + Z 键回到移动前的状态。

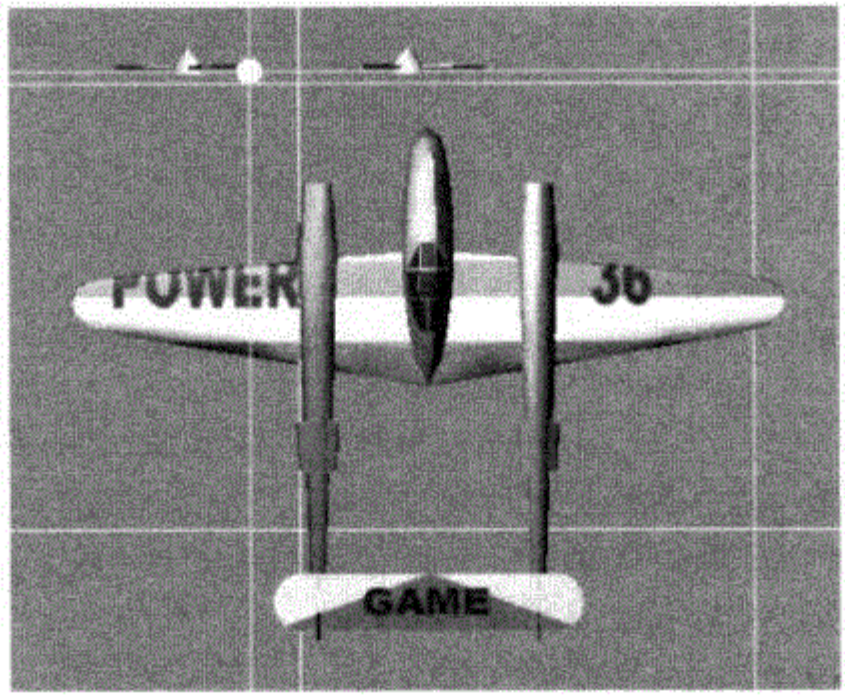


图 3.43　未设置继承关系时移动机身的结果

这样分散的结构会对未来的编程造成不便，如果通过脚本来控制整架飞机的移动，将需要同时发出五条指令，才能让机身和螺旋桨轴、螺旋桨同步移动。使用“继承关系”可以解决这一问题。

图 3.44　打开 Hierarchy Manager 面板

首先执行 Editor | Hierarchy Manager 指令，在菜单中打开“继承关系编辑面板”，如图 3.44 所示。

随后在右侧找到 Hierarchy Manager 标签并打开，这里显示了所有物体（除了数组这类纯逻辑物体外）的继承关系，单击 3D Root 前的小箭头可以看到 3D Object 之间的继承关系，得到如图 3.45 所示的继承关系图。

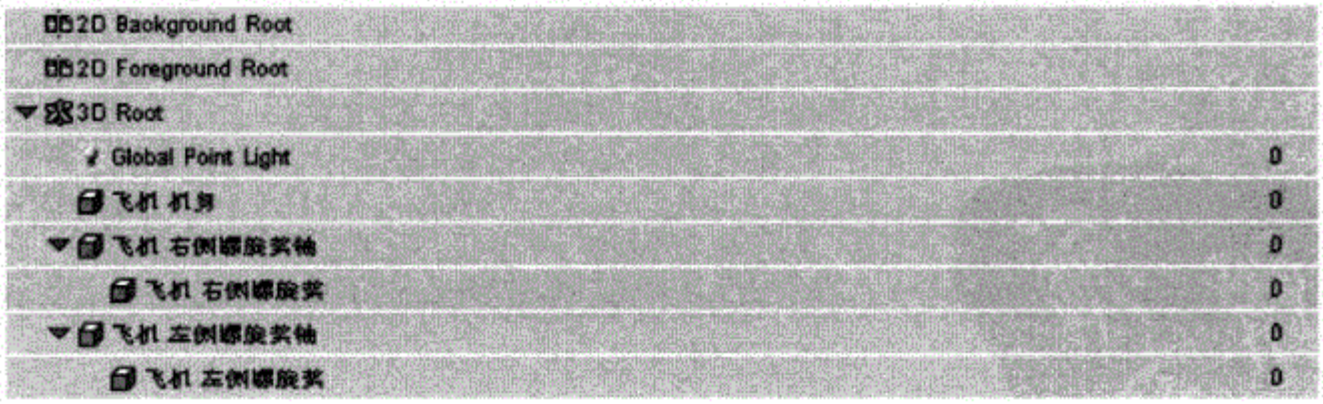


图 3.45　Hierarchy Manager 面板

所有位于下一个层级的物体都是上一个层级物体的“子对象”，当对上一个层级的物体进行移动旋转和缩放操作的时候，“子对象”相对于该物体的大小比例和

位置关系都不会有任何变化。为此，左右两侧的螺旋桨要设置为螺旋桨轴的子对象，螺旋桨轴则要设置为机身的子对象。

设置螺旋桨轴为机身的子对象，只需要拖动“飞机 左/右侧螺旋桨轴”到“飞机 机身”条目上就可以实现，新的继承关系如图3.46所示。此时再尝试移动机身，就会发现螺旋桨也会跟随移动。

图3.46 飞机继承关系设置结果

3.2.2 设置飞机的初始状态

接下来，通过手工输入数字来设置飞机的位置，并将其设置为“初始状态”。

首先，打开机身的设置面板。进入设置面板有如下几种方法：

（1）在Level Manager中找到“飞机 机身”，双击或右键选择Setup；

（2）在3D Layout（三维预览窗口）中右键选择机身物体然后选择Setup；

（3）在下方的便签页中找到3D Object Setup标签页，然后在左上角的Name下拉列表中选择“飞机 机身”；

（4）在刚刚打开的Hierarchy Manager面板中，右键选择“飞机 机身”然后选择Setup。

根据需要可以选择不同的方式进入某一物体的设置面板，3D Layout最为直接，而Level Manager最为正式。

进入机身的设置面板后，可以看到四个选项设置，如图3.47所示。

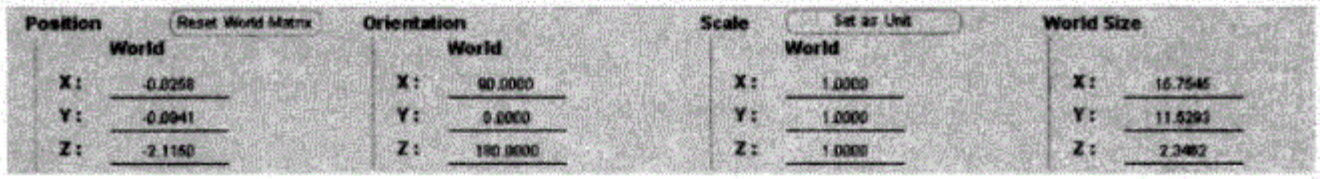


图3.47 机身设置选项

在此将飞机移动到世界坐标的（0，0，0）位置上，将Position的X、Y、Z都设置为0，在3D Layout中可以看到飞机的新位置。

同理进入飞机螺旋桨轴的设置面板，可以看到增加了一系列的Local设置选项，如图3.48所示。

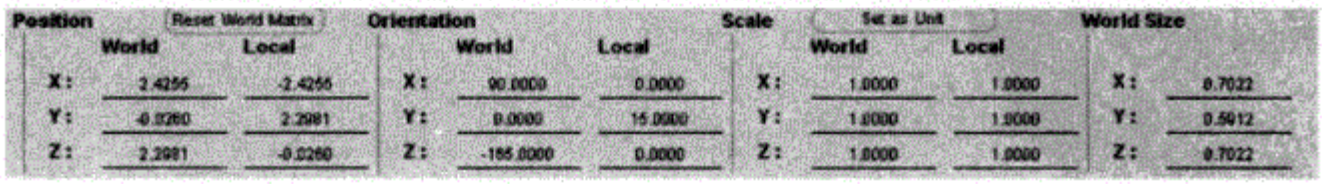


图3.48 飞机螺旋桨轴设置面板

以 Position 为例，World 下的 X、Y、Z 坐标标识的是该物体相对于世界坐标系的位置，而 Local 下的 X、Y、Z 值则表示相对于其“父对象”的位置。调整 World，Local 下的数值也会相应改变，反之亦然。

以“飞机 右侧螺旋桨轴”为例，如图 3.49 所示，现在它的 Orientation（旋转角度）的 Local 数值为（0，15，0），即表示它相对于机身坐标系沿 Y 轴旋转了 15 度，在此将其设置为 0，可以观察到螺旋桨轴的变化，如图 3.50 所示。

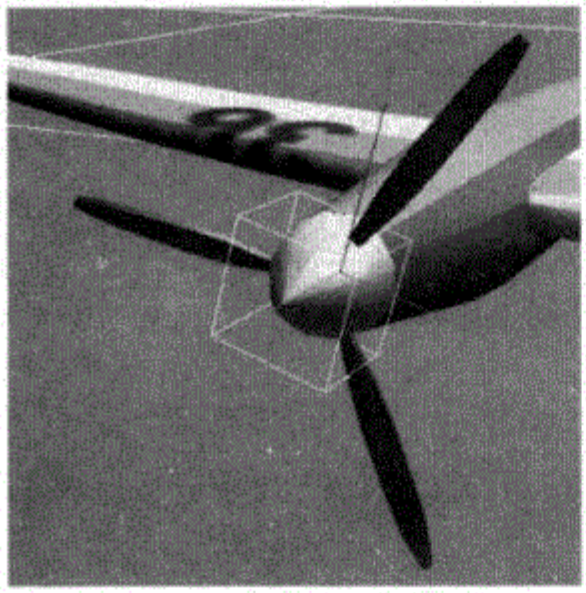


图 3.49　默认螺旋桨轴

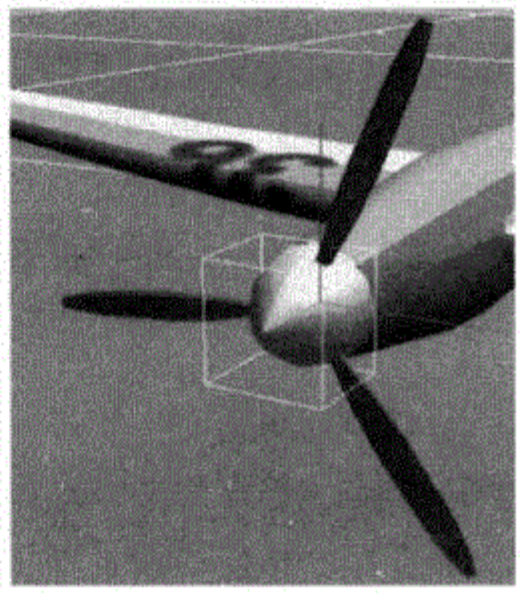


图 3.50　调校后的螺旋桨轴

接下来进入“飞机 右侧螺旋桨”的设置面板，可以看到 Orientation 的 Local 数值，如图 3.51所示。

在此将其设置为（0，0，0），按照类似的方法，完成对左侧螺旋桨轴和螺旋桨的调整，最终效果如图 3.52 所示。

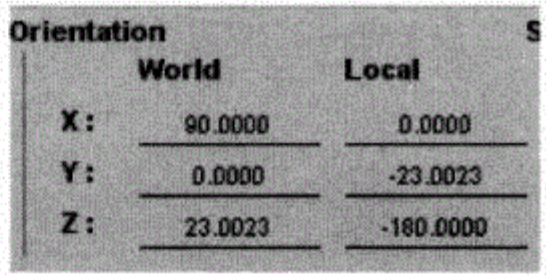


图 3.51　右侧螺旋桨轴旋转角度

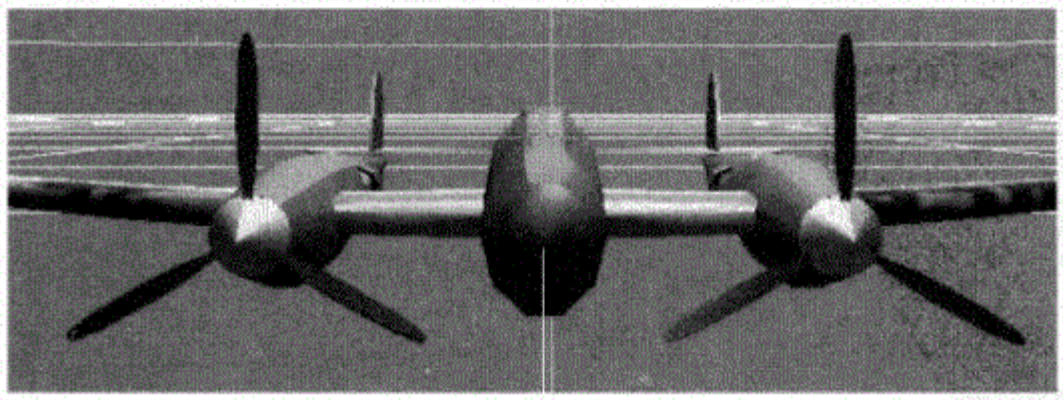


图 3.52　调整螺旋桨轴旋转角度后的效果

此时飞机的螺旋桨位置都是正确的，要想保留设置后的效果，并且以后能够随时回到这一状态，可以将其设置为初始状态，在 Level Manager 中，选中飞机的全部五个部件，单击右键，在弹出菜单中选择 Set Initial Conditions 选项，如图 3.53 所示。

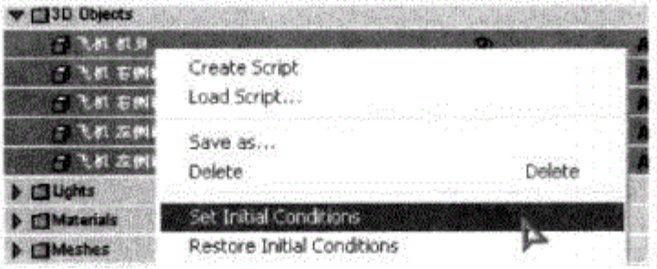


图 3.53 Set Initial Conditions 选项

也可以单击 Level Manager 标签下方的长条按钮 Set IC For Selected（IC 为 Initial Conditions 的缩写），如图 3.54 所示。

所有设置过初始状态的物体，在 Level Manager 中，IC 一栏都会显示一个叉号，表示该物体设置有初始状态，如图 3.55 所示。

图 3.54 Set IC For Selected 按钮

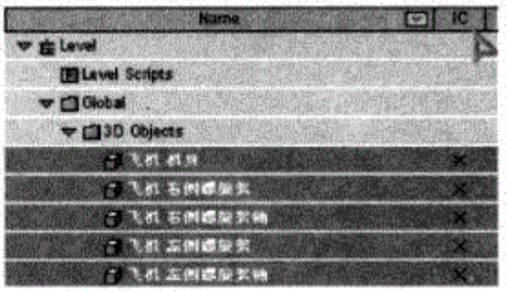


图 3.55 设置 IC 后的状态

所有设置过初始状态的物体，通过单击右下角的“恢复 IC”按钮，可以“一键恢复”到初始状态，如图 3.56 所示。

图 3.56 Restore Initial Conditions

尝试通过移动、选择、缩放工具对飞机的任何部件进行调整，即使到如图 3.57 所示的四散分离状态，只要单击“恢复 IC”便可以重新恢复到“初始状态”。利用 IC 可以定义一个游戏的初始状态，当游戏进行一段时间或结束时，游戏世界被弄得一团混乱时，通过 IC 可以轻易地回到从前并重新开始新一轮的游戏。

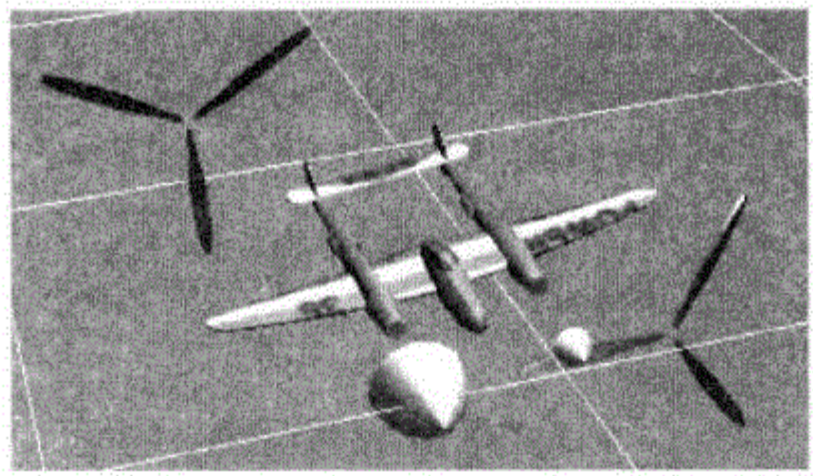


图 3.57　对飞机各部件进行任意缩放移动

设置 IC 的技巧是很常用的，需要养成在更改后重新设置 IC 的习惯，它的作用有些类似“暂存”当前的状态。否则单击“恢复 IC”键后，没有保存的更改都会随之失效。

3.2.3　编写第一个脚本

下面通过脚本模块来编写第一个脚本，使飞机的螺旋桨旋转起来。Virtools 的脚本编写方法不是传统的编写程序代码，而更像是画流程图。

1. 创建脚本

首先，需要为螺旋桨轴创建一个新的脚本，在 Level Manager 中（或在 3D Layout 中），单击右键选择“飞机 右侧螺旋桨”，在弹出的右键菜单中选择 Create Script 选项，创建一个新的脚本。所有物体的脚本，都会在 Level Manager 中显示为该物体的下一层级，如图 3.58 所示。

图 3.58　新建脚本

为了使用方便和更容易理解，需要更改名字。选中该脚本名按 F2 键，为其更名为“旋转”。

双击该脚本进入脚本编辑器 Schematic。顾名思义，Virtools 的脚本编写并不是传统的编写程序代码（Coding），而更像是画流程图（Schematic）。

如图 3.59 所示，左侧的方格为脚本头，标识了该脚本的名称以及它所属的物体，右侧的空白区域则是用来编辑脚本的。Virtools 的脚本编写是使用 Building Blocks（简称 BB）来完成的，它们是一系列被封装了各种功能的程序模块，为了让螺旋桨旋转，需要使用 Rotate（旋转）模块，首先打开右侧 Building Blocks 标签页，选取 3D Transformations | Basic | Rotate 选项，如图 3.60 所示。

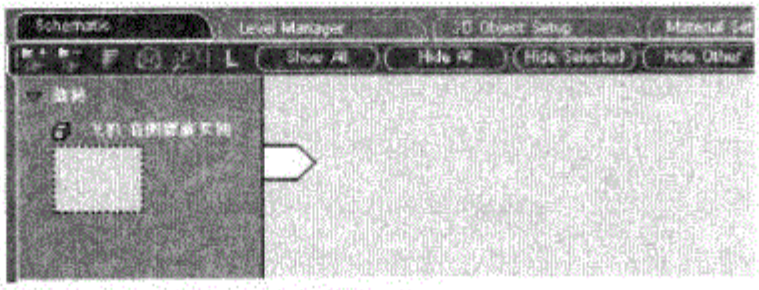


图 3.59 Schematic 空白脚本

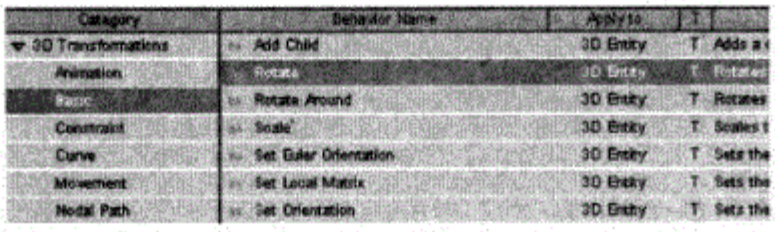


图 3.60 Rotate BB

每一个模块的右侧都标有一些基本信息，比如 Apply to（可以赋予给何种物体）、T（是否可以指定对象）、Description（简要描述），用户可以根据模块的名字以及这些信息大致猜测出它的作用。拖动该模块并放置到脚本书写区域之上，模块所放置的位置无关紧要，方便操作即可，如图 3.61 所示。

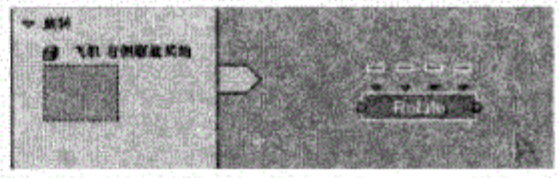
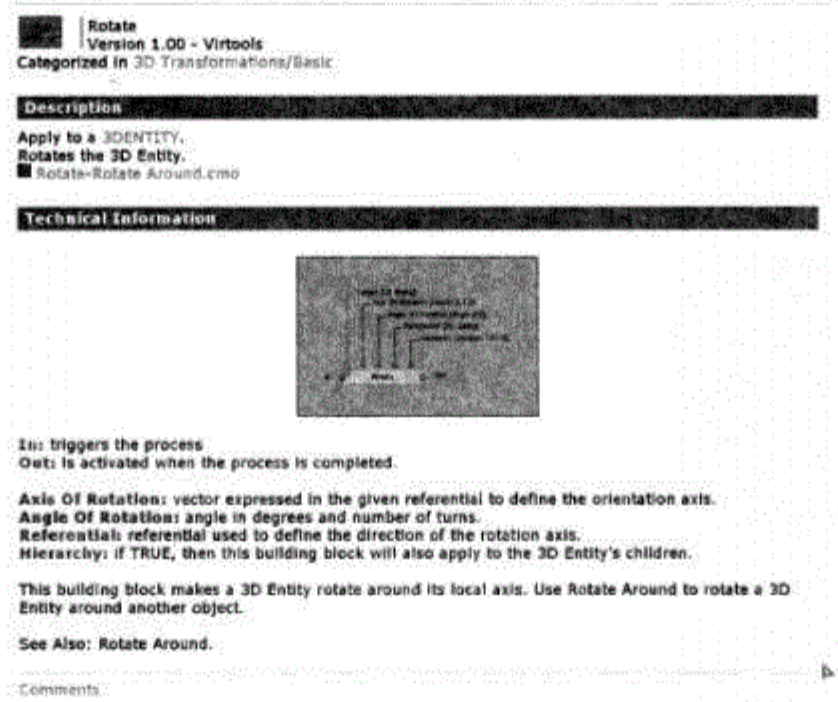


图 3.61 为脚本添加 Rotate BB

如果需要更细致的 BB 说明，可以在脚本书写区域选择该模块并按 F1 键，就会自动进入该模块的帮助页面，如图 3.62 所示。



Rotate
Version 1.00 - Virtools
Categorized in 3D Transformations/Basic

Description

Apply to a 3DENTITY.
Rotates the 3D Entity.
Rotate=Rotate Around.cmo

Technical Information

In: triggers the process
Out: is activated when the process is completed.

Axis Of Rotation: vector expressed in the given referential to define the orientation axis.
Angle Of Rotation: angle in degrees and number of turns.
Referential: referential used to define the direction of the rotation axis.
Hierarchy: if TRUE, then this building block will also apply to the 3D Entity's children.

This building block makes a 3D Entity rotate around its local axis. Use Rotate Around to rotate a 3D Entity around another object.

See Also: Rotate Around.

Comments

图 3.62 Building Block 帮助页面

Virtools 中每一个 Building Block 都有讲解说明，并自带一个简单的范例程序展示该 BB 的使用方法，它位于 Description 中的最后一项。如 Rotate 的范例程序名称为 Rotate-Rotate Around. cmo，用鼠标直接单击后可在 Virtools Web Player 打开并运行，若需要打开工程文件查看，可以到 Virtools\Documentation\CMOs\BBSamples 中的相应目录中找到该 cmo 文件并用 Virtools 打开。掌握查看帮助的方法是非常重要的。

下面以 Rotate 为例来介绍脚本模块。该 BB 的主体为长方形，上标有该 BB 的名字，左侧小箭头为输入端，如图 3.63 所示。右侧小箭头为输出端，如图 3.64 所示，当左侧箭头被激活的时候，程序开始执行，程序执行结束后，则自动激活右侧小箭头。并非所有的 BB 都有输出端，但所有 BB 都具有一个或多个输入端，只是这些输入端的功能各不相同。

图 3.63 BB 输入端

图 3.64 BB 输出端

BB 上方，指向 BB 内部的三角箭头是该 BB 的输入参数，如图 3.65 所示。每一个箭头代表一个输入参数，用鼠标双击其中任意一个箭头，可以看到该参数的详细信息，图 3.66 为左侧第一个参数的详细信息。

图 3.65 BB 输入参数

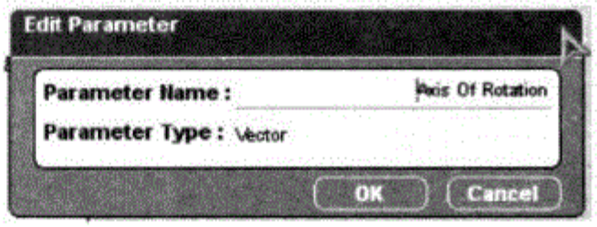


图 3.66 输入参数类型

其中，Parameter Name 为参数名，Parameter Type 为参数类型。参数的名称是可以更改的，并不会对 BB 的功能起到任何影响。这其实和程序中的函数参数是一个道理，对于一个函数（Virtools 中可以视为 BB），参数的类型是固定不可更改的，而参数的名称只是为了编程的方便而使用的，在 Virtools 中一样可以更改参数名以使脚本更加易读。

BB 除了输入参数还有输出参数，但是在本例（Rotate）中没有用到输出参数，将在用到时再做介绍。

BB 上方的小长方格为参数值，如图 3.67 所示，表示赋给 BB 输入参数的实际数值，将任何一个小方格向上拖动一定距离，会发现它和输入参数之间被一条虚线所连接，如图 3.68 所示。

图 3.67 参数值

图 3.68 参数连线

虚线连接用来表示某一个数值被赋给了某一个参数。可以这样理解三角（输入参数）和长方块（参数数值）之间的关系：三角表示该 BB 需要什么类型的参数，而小方块表示实际赋给该 BB 的数值是什么。双击任意一个小方块，可以看到该数值的详细信息，如图 3.69 所示。

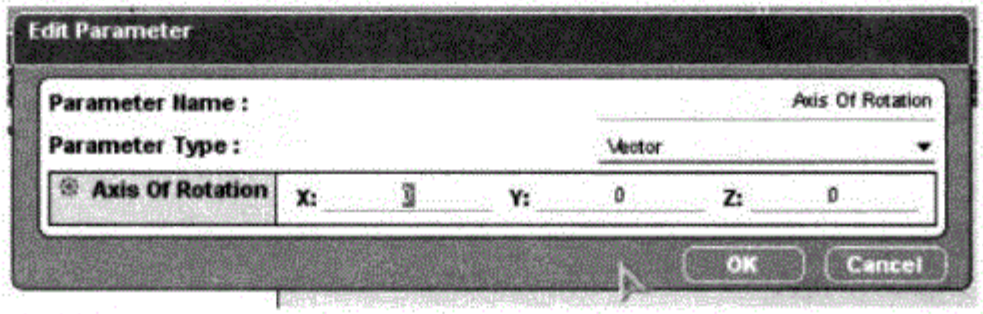


图 3.69 参数值详细信息

根据参数类型的不同，参数数值设置会各不相同。与输入参数不同，参数数值不仅可以改变名称，也可以改变参数类型。一个参数数值，它可能是任何类型的，但若想将其赋给某一个 BB 的某一个输入参数，它必须能够和该输入参数的类型匹配。以上面例子（Rotate 的第一个参数）为例，输入参数的类型为 Vector（向量，即三维向量），因此只有参数数值的类型为 Vector 才能成功赋给该输入参数（连接虚线），若将该参数数值的 Parameter Type 改为 Vector 2D（二维向量），如图 3.70 所示，单击 OK 按钮会发现参数数值和输入参数之间的虚线自动断开了，如图 3.71 所示，这是由于参数类型不匹配所导致的。

图 3.70 更改参数值类型为 Vector 2D

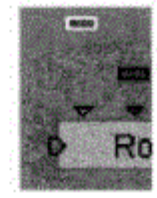


图 3.71 参数类型不匹配导致连线断开

下面用同样的方法将该参数的类型改回 Vector，连线并没有自动恢复，需要手动将该参数值赋给 Rotate 的第一个输入参数。具体方法是：鼠标左键单击小三角并拖动鼠标形成连线，在参数的数值上方松开鼠标左键完成连线，如图 3.72 所示。

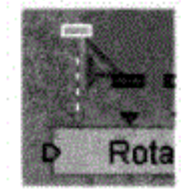


图 3.72 参数连线

双击 BB 可以进入该 BB 的参数设置窗口，如图 3.73 所示，在这里可以同时调整所有输入参数的数值，其中 Axis Of Rotation 为旋转轴，即绕着哪个轴旋转。

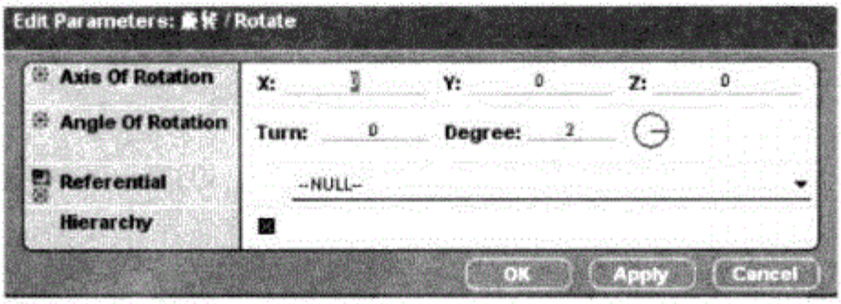


图 3.73　BB 参数设置窗口

在 3D Layout 中选中螺旋桨轴，可以看出螺旋桨轴应该绕着绿色的坐标轴进行旋转，即 Y 轴，如图 3.74 所示。在此将 Y 设为 1，X 和 Z 都设为 0，表示不在 XZ 轴上进行旋转。

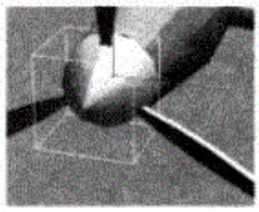


图 3.74　螺旋桨轴坐标轴

Angle Of Rotation 为旋转角度。在这里先采用默认值，后面会根据测试结果再进行更改。

Referential 为参考系。参考系是指这个旋转操作是相对于什么物体进行的旋转，默认的 NULL 表示无参考系，系统将自动使用世界坐标系，即世界中心作为旋转的参考点，由于希望螺旋桨轴绕着自己的坐标轴进行旋转，因此需要在下拉列表框中选中“飞机 右侧螺旋桨轴”。

Hierarchy 表示继承。这是一个 Boolean（逻辑）参数，只有 true 或 false（真或假）两个值，叉号表示 true，否则为 false。继承参数表示该动作是否会影响到这个物体的所有子物体，螺旋桨轴的子物体是螺旋桨，螺旋桨需要随轴运动，因此采用默认值 true。最终的参数设置结果如图 3.75 所示。

图 3.75　Rotate BB 参数设置结果

接下来需要让脚本工作起来，从 Rotate BB 左侧的输入端箭头按住鼠标左键拖出一条实线，并将其连接到脚本书写区域最左端的路标形突起上，松开鼠标左键完成连线，如图 3.76 所示。

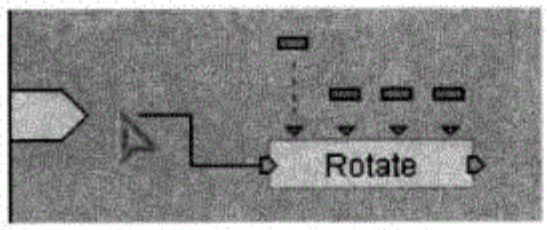


图 3.76 连线 Rotate BB

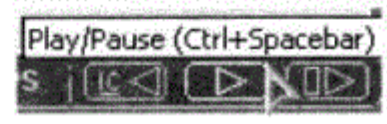


图 3.77 播放按钮

在 Virtools 脚本书写中，实线连线表示 BB 脚本之间的“激活路径”，可以理解为一套脚本运行时的线路。任何一个脚本，都是从最左侧的突起开始执行的，当一个脚本执行的时候，首先会激活该突起，然后沿着该突起连接出来的实线路径顺序执行其他脚本。这里的脚本执行逻辑比较简单，开始执行脚本后，激活 Rotate BB。单击右下角的 Play（播放）按钮进行测试，如图 3.77 所示。

在 3D Layout 中，将视角拉近右侧螺旋桨，运行后可以看到螺旋桨稍微向右偏移了一点（偏 2 度角），如图 3.78 所示。

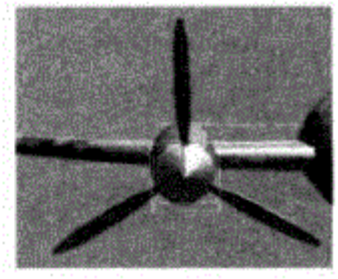


图 3.78 螺旋桨旋转 2 度

由于 Rotate BB 并不带有自动循环的功能，它只是完成绕轴旋转一个角度的操作，然后便会激活右侧的 Out 端并停止运行，因此需要手动设置循环旋转。

首先停止运行并回到初始状态（Restore IC）。单击右下角的后退箭头，螺旋桨轴回到初始的方位，这是由于它被设置了初始状态（Initial Condition/IC）的缘故。

用鼠标左键单击并拖动 Rotate BB 右侧 Out 端（小三角形），将连线连接至 Rotate BB 左侧的 In 端，如图 3.79 所示。

图 3.79 Rotate BB 自循环连线

下面再运行测试，将看到螺旋桨开始慢慢旋转起来。这个连线方式的实际逻辑如下：

脚本开始→激活 Rotate 的输入端（In）→脚本执行完毕→激活 Rotate 的输出端（Out）→顺着返回的连线，在下一帧继续激活 Rotate 的输入端→以此类推……

在激活的过程中，只有顺着箭头的指向激活才是有效的，而 In→Out 的箭头是逆着箭头的方向，因此无法激活。通过连接 Out 和 In 得到的“BB 自循环”在编写脚本中十分常用。

在测试中发现，螺旋桨旋转的速度太慢，若要让其变快，无须暂停运行或回到初始状态，可以直接在运行中双击 Rotate BB，打开设置面板，更改其中的 Angle Of Rotation，单击 Apply 或 OK 按钮后立刻能看到效果。Virtools 具有“在程序运行中进行调试”的功能，这样不但可以立刻看到更改后的效果，也会利于错误的发现。

最终将 Angle Of Rotation 设置为 31 度，也可以根据自己的感觉调整到满意的

数值。

2. 复制脚本

当前已经为右侧螺旋桨添加了一个旋转的脚本，接下来需要为左侧螺旋桨复制一份同样的脚本。

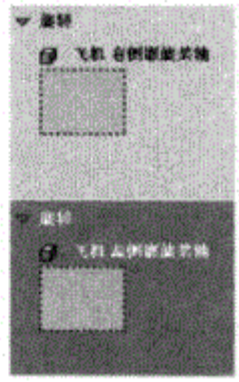


图 3.80 复制得到的脚本

在 Level Manager 中，用鼠标右键单击“旋转”脚本，然后在弹出菜单中选择 Copy 选项，再用鼠标右键单击“飞机 左侧螺旋桨轴”，在弹出菜单中旋转 Paste，完成粘贴，双击新粘贴的脚本进入编辑页面，可以看到它的内容完全一样，只是赋予的物体不同，如图 3.80 所示。

运行测试一下，可以看到两个螺旋桨同时旋转的效果。截止到此的工程文件可以在 CMOs\Plane\Plane_04. cmo 找到。

3.3 脚本书写进阶

上一节中主要介绍了一些脚本书写的常用技巧和如何调试 Virtools 程序（使用 Trace 和 Breakpoint）。在本节中，将进一步完善所做的游戏原型，为它增加更多的互动内容，可以直接打开（CMOs\Plane\Plane_06. cmo）的工程文件继续本章的学习，也可以使用自己的工程文件。

3.3.1 旋转的浮空山

图 3.81 浮空山模型

首先在场景中制造一些飞行中的障碍物——浮空并且不断旋转的山峰，如图 3.81 所示。

此时游戏的目标是，当玩家控制飞机飞行的时候，不断有山峰从远方飞来，为了增强视觉效果，山峰会绕着 Y 轴旋转，而旋转的速度是随机的。

将以上问题分解，首先要解决的是山峰绕 Y 轴随机旋转。这里将单独制作山体的旋转，然后将它们整体导出，这是一种常用的开发方式，适合团队合作时的分工协作。

1. 导入模型资源

新建一个 Virtools 工程，在菜单中选择 Resources | Import File 命令，将 CMOs\Plane\mountain. nmo 导入到场景中。

注：Import File 可以导入后缀为 nmo 的资源，使用它可以不借助资源库面板完成资源的导入工作，相对而言，Import File 更适合一次性地导入，而资源库更适合存放那些需要反复导入并使用的资源。

2. 旋转

导入后可以看见一个名为 Mountain 的三维物体，为它新建一个脚本并命名为“随机旋转”。在脚本编辑面板中开始编辑脚本，下面使用一种更为快捷的方式来添加 BB：按住 Ctrl 键，然后用鼠标左键双击脚本书写区域的空白区域，会出现如图 3.82 所示的选择框。

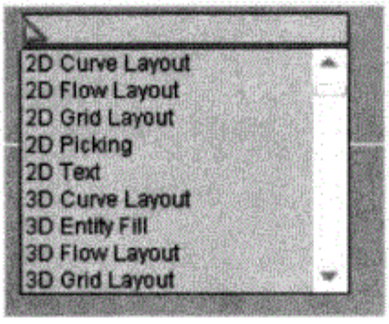


图 3.82 Ctrl + 双击添加 BB

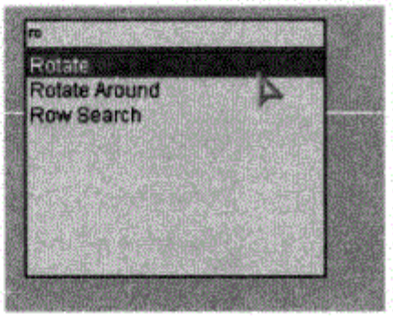


图 3.83 使用首字母索引搜寻 Rotate BB

在输入栏中，输入 rotate（小写即可），通常只需输入“ro”两个字母就能自动检索到以下三个 BB，如图 3.83 所示。

用鼠标或键盘方向键选择 Rotate 即可完成 BB 的添加工作，这种添加 BB 的方式适合添加已经熟悉的 BB。

首先让山体自动旋转起来，这和旋转螺旋桨如出一辙，具体脚本和设置如图 3.84 和图 3.85 所示。

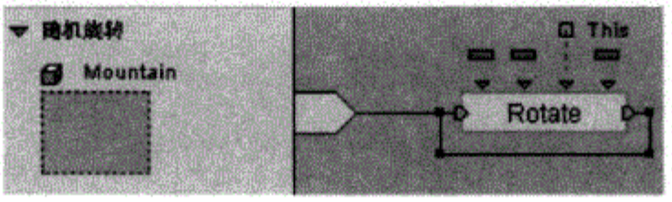


图 3.84 山体旋转脚本

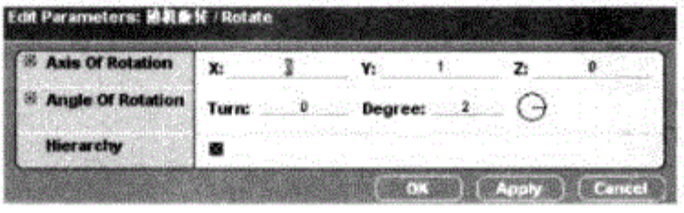


图 3.85 山体旋转 Rotate BB 设置参数

在运行测试之前，不要忘记先为 Mountain 设置 IC。

3. 随机旋转

接下来将山体绕 Y 轴旋转的角度设为随机，为旋转添加一些随机性。为了产生一

个随机的数值，需要 Random BB，它位于 Building Blocks | Logics | Calculator 分类下，也可以使用“Ctrl + 双击”的方法来添加 BB。

Random BB 的作用是产生一个介于 Min 和 Max 之间的数值，在此需要先指定输出参数的类型，输入参数的类型会随着输出参数类型的变化而变化。双击 Random BB 下方的输出参数三角，默认的 Parameter Type 为 Float，即浮点数，在此需要产生一个角度值（Angle）用作 Rotate 中的输入参数 Angle Of Rotation，故将其更改为 Angle 并单击 OK 按钮，如图 3.86 所示。

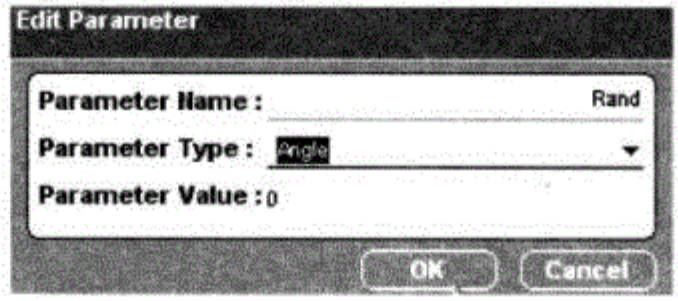


图 3.86　更改 Random BB 输入参数类型为 Angle

接下来双击 BB 进行输入参数设置，在这里将 Min 设为 -5 度，Max 设为 5 度，如图 3.87 所示。

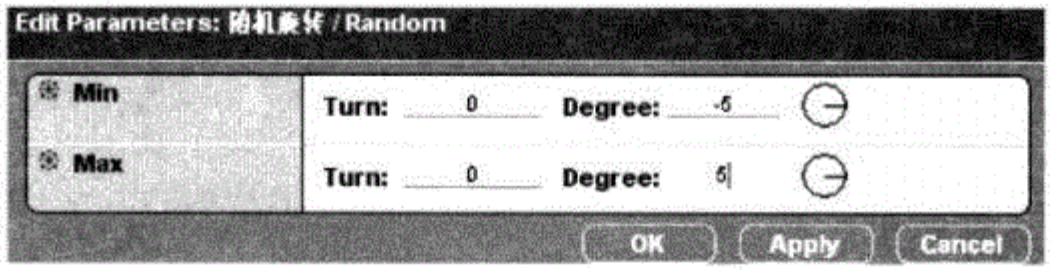


图 3.87　Random BB 输入参数设置

这样在程序运行时可产生一个介于 -5° ~5°之间的随机角度，然后将这个角度赋值给 Rotate BB 的输入参数 Angle Of Rotation，具体脚本连接方法如图 3.88 所示。

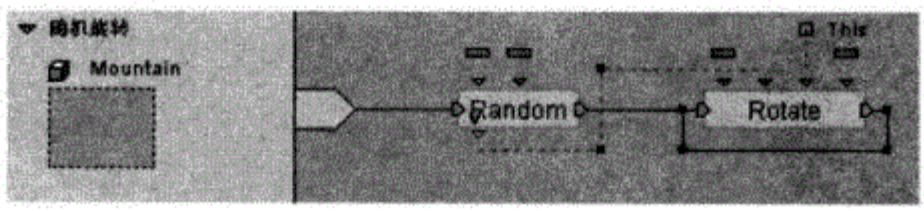


图 3.88　山体随机旋转脚本连线

测试后即可看到随机旋转的效果。

4. 导出资源

上面的工程文件可以在 CMOs\Plane\rotate_mountain. cmo 找到。

为了在游戏场景中使用已经制作的山，需要将其导出。在 Level Manager 中，选中 Mountain 然后在右键菜单内选择 Save As，指定导出路径和导出文件名（本书使用 rotate_mountain）后单击 OK 按钮完成导出，可以在导出路径内找到后缀为 nmo 的导出资源，如图 3.89 所示。

图 3.89 导出得到的 rotate_ mountain. nmo

5. 导入资源至飞机场景

将导出的资源使用 Import File 导入飞机的工程文件（也可以直接使用 CMOs\Plane\Plane_06. com）。

在实际游戏中，所有的浮空山都是动态创建的，简单的理解就是使用程序在游戏进行中不断复制浮空山并且加入到场景之中，因此需要保存一个最原始的模板，即刚刚加入场景的这座山，它在游戏进行的时候既不会被显示也不会被激活（不会旋转也不会移动），只是隐性地存在于场景中作为模板供程序进行复制，因此首先需要隐藏它并且使它的脚本不运行。

设置隐藏：在 Level Manager 中，用鼠标左键单击 Mountain 右侧的眼睛图标使其关闭（如图 3.90 所示），可以看到浮空山从场景中消失。

图 3.90 隐藏 Mountain

使浮空山不被激活：在 Level Manager 中，用鼠标左键单击 Mountain 右侧 A 图标使其上方出现红色叉号表示不激活。一个物体不被激活表示该物体所有脚本不会被运行。

更改完这几项设置后重新设置 IC（初始状态），否则更改便会在 Restore IC（恢复初始状态）后失效。

运行测试一下，浮空山并未被显示出来，打开脚本的 Trace 模式，可以看到浮空山的“随机旋转”脚本并没有运行，如图 3.91 所示。

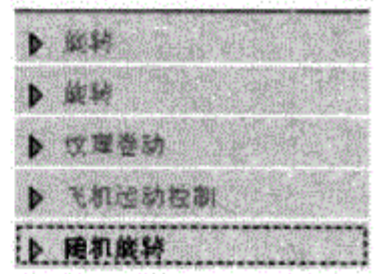


图 3.91 Trace 结果

6. 动态复制浮空山

接下来需要创建一个脚本，它的功能是在游戏运行过程中动态创建浮空山并将其加入到场景之中，这种在游戏全程激活的脚本最适合被赋给 Level，即游戏的最顶层结构，因为 Level 在游戏运行中是始终激活的。

在 Level Manager 中为 Level 创建一个脚本并重命名为“创造浮空山”，如图 3.92 所示。创建脚本的方式和其他物体一样，只是 Level 无法在 3D Layout 中找到。

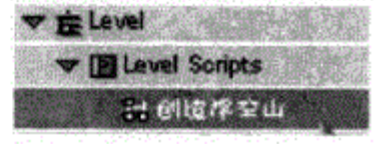


图 3.92 创建“创造浮空山”Level 脚本

所有属于 Level 的脚本都被放置在 Level Scripts 中。在新建的脚本中，添加如图 3.93 所示的模块（建议使用 Ctrl + 双

击的方法）。

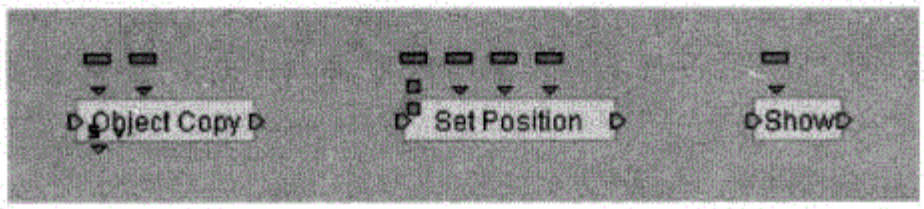


图 3.93 “创造浮空山”所用到的 BB

Object Copy 模块可以复制一个物体。此时需要复制的是 Mountain。双击该 BB，Original即为复制的“源”，拉动下拉列表框可以看到很多物体可供选择，这是由于 Object Copy 可以复制一切加入到工程文件之中的物体。为了检索方便可以先指定该物体的类型再做寻找——在 Class 一栏选择 3D Entity（可以使用键盘输入的方式进行快速选择），然后再回到 Original 选择 Mountain 即可，如图 3.94 所示。单击 OK 按钮完成设置。

图 3.94 Object Copy – Original 参数设置

复制得到的新物体即为 Object Copy 下方的输出参数，可以随时引用它。

Set Position 模块用来设置三维物体的位置。此时希望浮空山出现在飞机的前方（而不是默认的世界中心，否则会将飞机挡住），因此将 Set Position 的输入参数 Position 设置为（0、0、100），Referential 设置为 Null 表示使用世界坐标系，如图 3.95 所示。

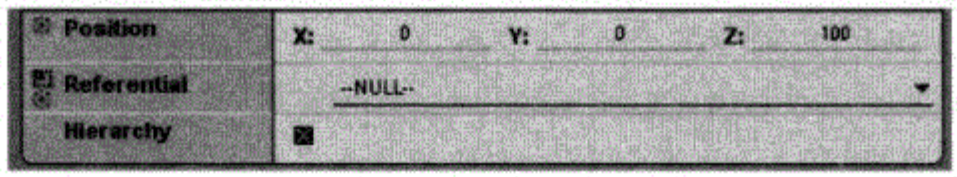


图 3.95 Set Position 输入参数设置

Target（目标）指向的是 Object Copy 脚本模块复制出来的物体，用来设置被复制的浮空山位置，具体连线如图 3.96 所示。

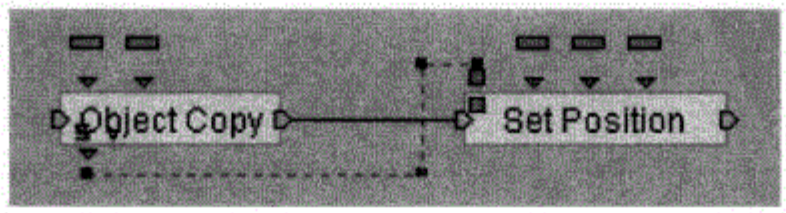


图 3.96 Set Position – Target 连线

Show 代表显示一个被隐藏的物体。由于之前已将模板浮空山隐藏，因此被复制的浮空山仍然是隐藏的，此时需要将其显示出来。Show BB 默认不带 Target 参数，需要为其增加一个 Target 输入参数使其可以控制被复制的浮空山的显示状态。右键选择 Show BB 模块，并在弹出菜单中选择 Add Target Parameter（增加目标参数），然后将 Object Copy 的输出参数与 Target 相连即可，如图 3.97 所示。

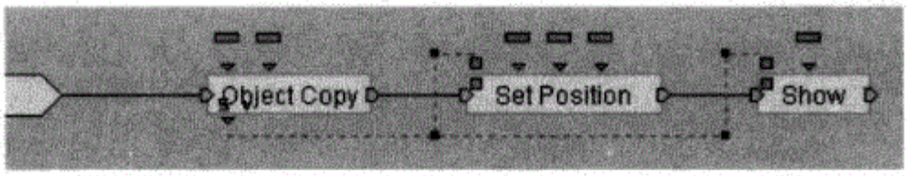


图 3.97 复制浮空山脚本连线

运行测试脚本可以看到一座浮空山出现在飞机的前方，如图 3.98 所示。

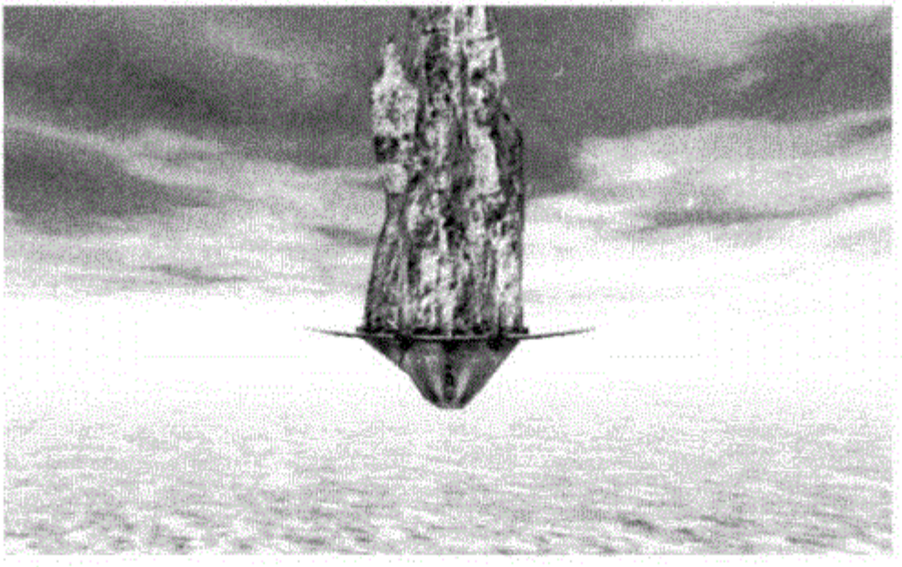


图 3.98 测试复制得到的浮空山

检查 Level Manager 可以看到新创建的 Mountain 副本被用红色字体显示出来，如图 3.99 所示，它表示动态创建的物体，当恢复初始状态后将被删除，而在 Schematic 中，“随机旋转”的脚本也被复制并且正在运行，如图 3.100 所示。

图 3.99 Level Manager 中动态复制得到的 Mountain

图 3.100 动态复制得到的 Mountain 所运行的脚本

7. 向飞机逼近的浮空山

接下来让浮空山与地面一样，从远处向飞机逼近，这需要再为 Mountain 创建一个新的脚本，将其重命名为“移动”。

注：对于一个物体，将不同功能的脚本分别存放在不同的脚本文件中是一个良好的编程习惯，将它们都放到一个脚本中不利于将问题分解，同时在动态激活或关闭某一个功能时也会比较麻烦。

图 3.101 Translate 自循环连线

如图 3.101 和图 3.102 所示，尝试自己直接完成脚本。

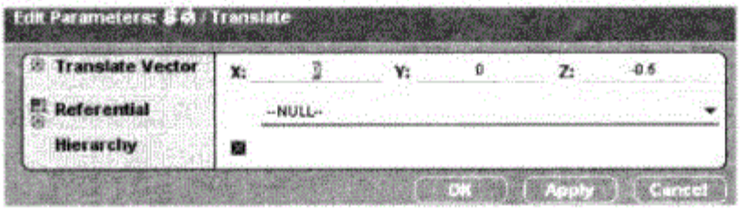


图 3.102 Translate 输入参数设置

调整参数，在运行测试中找到最适合的移动速度。

8. 调整山体大小

经过测试，发现山的尺寸过大，需要将其缩小一些。在 Level Manager 中打开 Mountain 的可视开关（小眼睛图标），使用缩放工具将其缩小，然后关闭眼睛图标隐藏 Mountain，最后重新设置初始状态（IC）。

9. 动态删除浮空山

测试时会发现新的问题。动态创建的浮空山在飞过摄像机之后仍然会继续向前飞行，此时它不会被显示在摄像机上，脚本却还一直在执行，不再起任何作用的浮空山也会一直存在于舞台之上（只是不被显示出来），这是对资源的浪费，如图 3.103 所示。

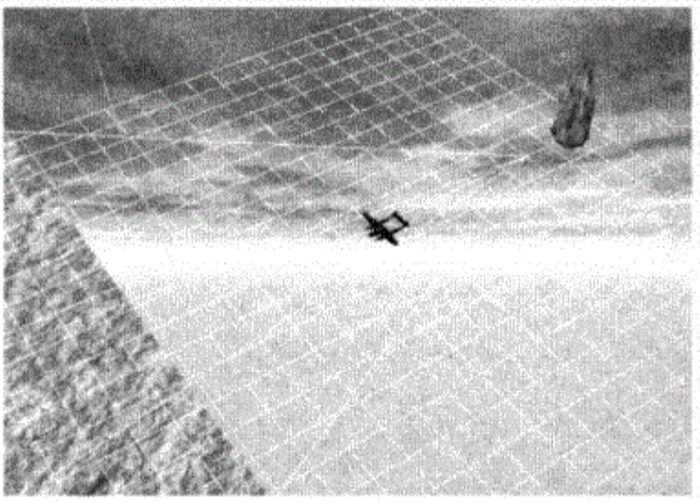


图 3.103 在飞机后方未被删除的浮空山

因此，需要在浮空山飞过摄像机之后将其删除，这个问题可以被解释为：浮空山当前位置的 Z 值小于某一个数值（摄像机的位置的 Z 值）时，删除浮空山。

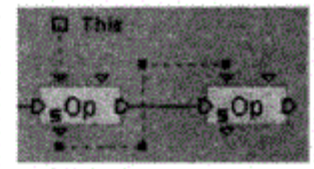


图 3.104 动态删除浮空山脚本

为 Mountain 新建一个脚本，起名为“动态删除”。若要判断是否应该删除浮空山，首先应该知道当前位置的 Z 值（见图 3.104）。

如图 3.105 所示，第一个 Op 的运算为 Get Position，第二个 Op 的运算为 Get Z。

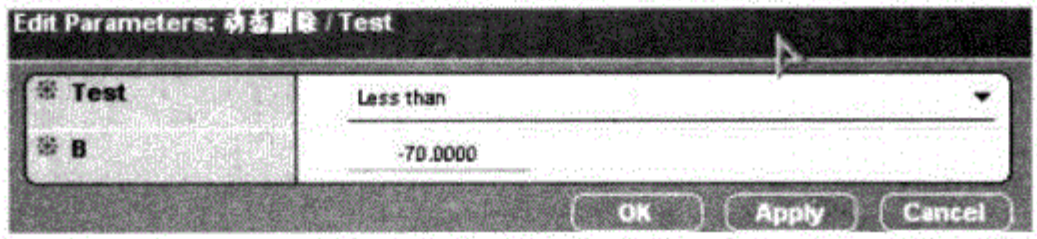


图 3.105 Test BB 输入参数设置

接下来，进行 Z 值判断，与飞机控制脚本一样，使用 Test BB。这里 Test 的 B 参数应该指定为摄像机位置的 Z 值，在例子中该值为 -70，如图 3.106 所示。

沿着脚本的运行路线，在 Test 的两个输出端进行下一步的测试。

若判断成立（第一个输出端为 True），即浮空山已经飞过摄像机，应该被删除，在这里使用 Object Delete BB，它的功能是删除一个物体，将输入参数 Object 指定为 This，令其删除自身，如图 3.107 所示。

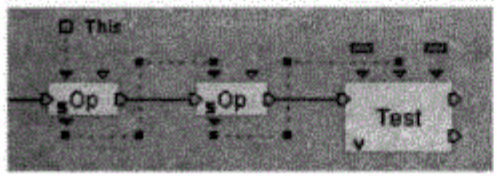


图 3.106 Test BB 连线

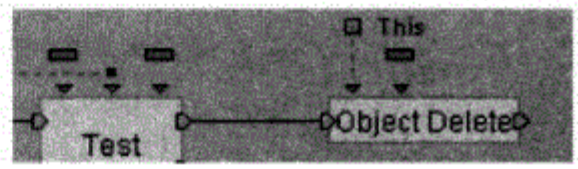


图 3.107 Object Delete BB 连线

若判断不成立（第二个输出端为 False），即浮空山还未飞过摄像机，则继续循环判断，将其与第一个 Op 的 In 端连接起来，如图 3.108 所示。

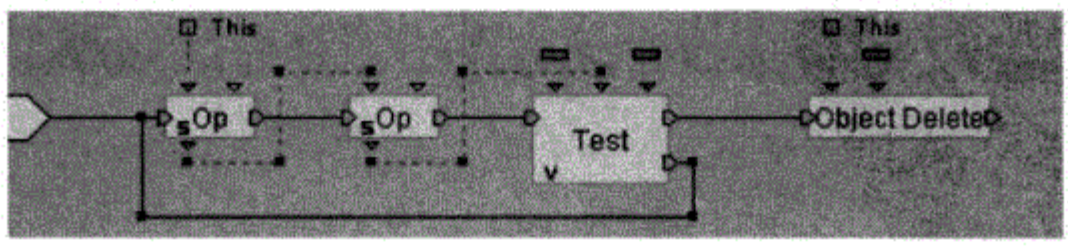


图 3.108 循环判断连线

运行测试即可看到动态删除的效果。

10. 不断产生浮空山

目前已完成了一个浮空山的动态创建，并可以向前移动和动态删除，接下来需要让场景中不断出现浮空山，这需要将之前创建的 Level Script “创造浮空山” 稍作修改。如图 3.109 所示，这里使用了一个新的模块——Delayer，它的功能是延迟一段时间，双击设置该 BB 可以更改延迟的时间（Time to Wait）。

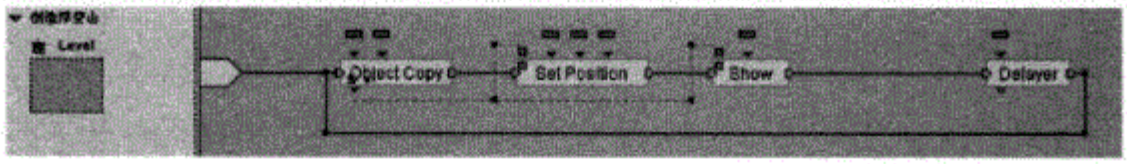


图 3.109　不断产生浮空山的脚本连线

在这里将延迟时间更改为 5s，如图 3.110 所示，进行测试后即可看到不断被创建的浮空山。更改 Set Position 可以让山在更远的地方被创造出来，这样效果会更好。在例子中将 Set Position 的数值调整为（0、0、1000）。

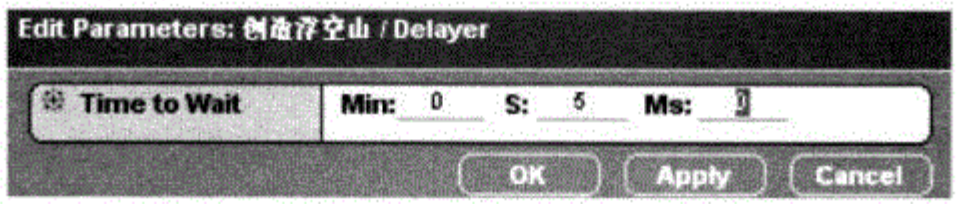


图 3.110　Delayer 输入参数设置

通过 Perspective View 可以看到一连串被创建出来的浮空山，如图 3.111 所示。

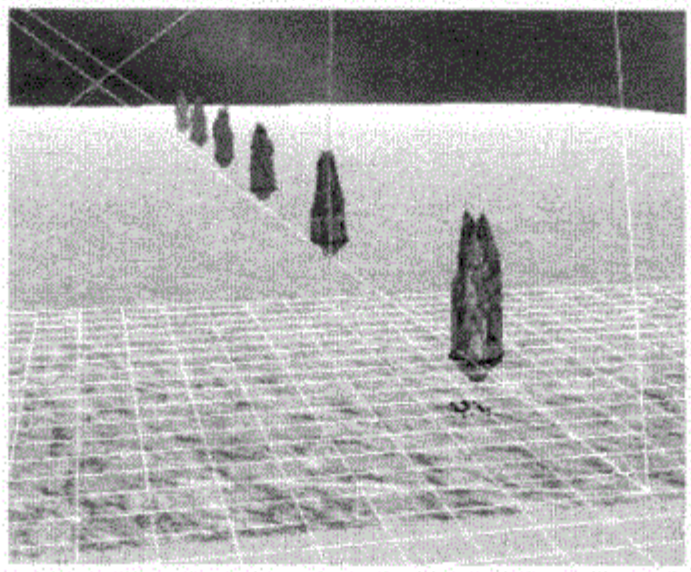


图 3.111　不断被复制出的浮空山

11. 增加更多随机性

此时创建出来的山，位置和间隔都完全固定，需要为它添加更多的随机性。

设置随机浮空山位置：目前创建浮空山的 Z 轴位置固定为 1000，目的是让其在远端

出现，无须改变。将恒定为0的XY方位做出一些随机变化，可以使山峰出现在画面的不同方位，会使游戏画面更丰富。

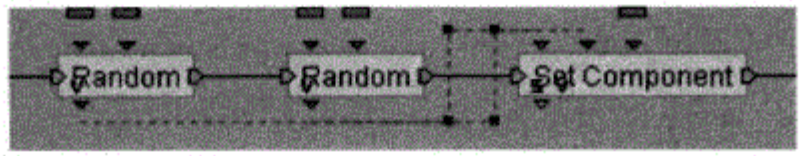


图3.112 随机位置脚本

图3.112即为产生随机位置的脚本，前两个Random已经介绍过，它们的作用是分别随机产生用于方位坐标的X和Y。最后一个Set Component也是一个常用的BB，它的功能是设置一个参数的各个部分。例如在这里就需要分别设定一个Vector类型变量（用作Set Position的坐标）的X、Y、Z三个分量，双击Set Component的输出参数可以看到Parameter Type有很多选项，它们都是可以通过Set Component进行设置的。在这里使用默认的参数类型Vector，如图3.113所示。

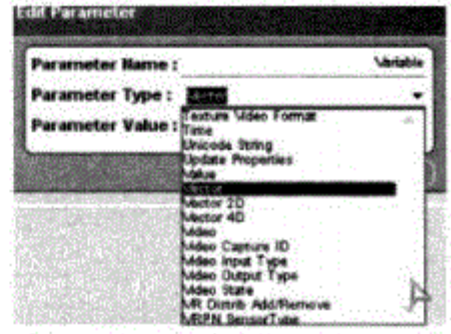


图3.113 Set Component BB输入参数类型Vector

由于希望浮空山总是出现在Z=1000的地方，因此在这里将Set Component的第三个参数设置为1000。

接下来需要确定产生X、Y值的两个Random BB的上限和下限。首先将Mountain显示出来，首先确定X的最大值，即山峰可以到达的最右方——利用移动工具沿X轴移动到最右方然后读取当前Mountain坐标的X值。如图3.114所示，在3D Layout中移动Mountain至画面最右方。读取此时X坐标，如图3.115所示。

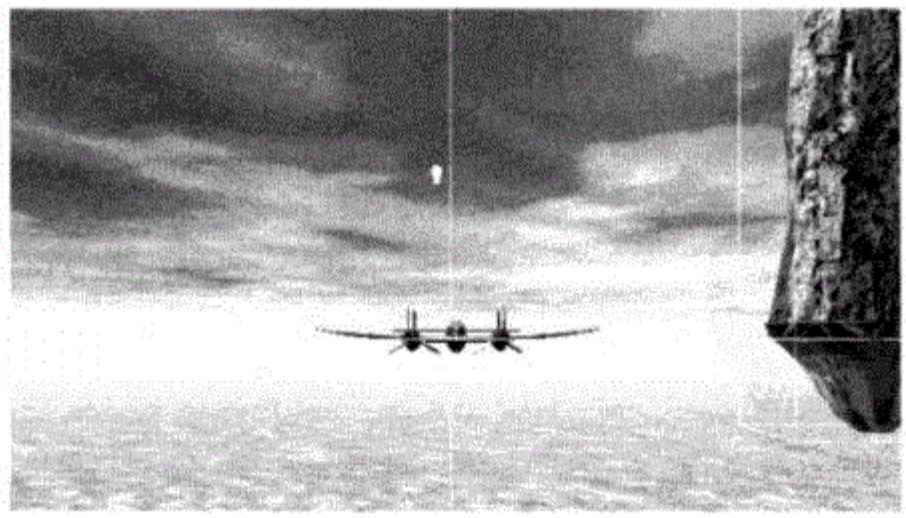


图3.114 将浮空山移动到画面右方

设置第一个Random的Max参数，如图3.116所示。

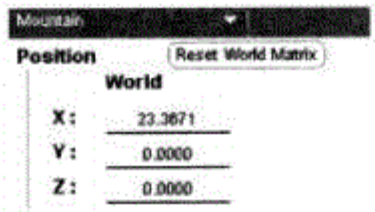


图 3.115 浮空山位于画面右方时的位置坐标

图 3.116 设置第一个 Random BB 的 Max 参数

由于山峰在左右两边（对应于 X 的负方向和正方向）是完全对称的，X 的最小值应为 Max 的相反数，即 -23，将其填入第一个 Random BB 的 Min 一栏，完成第一个 Random BB 的设置。

下面以相同的方法确定 Y 坐标随机的最大值和最小值，只是这一次浮空山在 Y 轴上的运动并不像 X 轴那样对称，需要分别确定边界值。此时上边界为 15，如图 3.117 所示。下边界为 -40，如图 3.118 所示。

图 3.117 将浮空山移动到画面上方

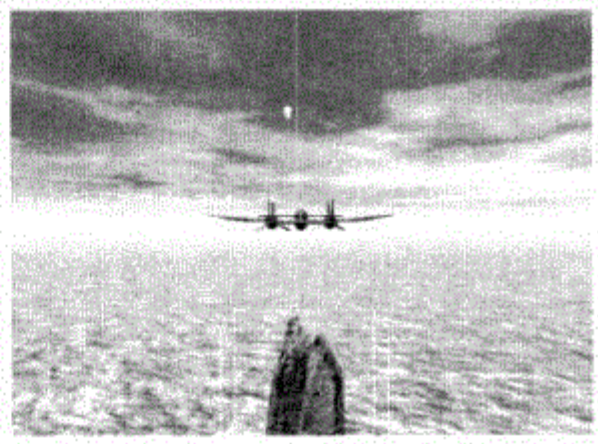


图 3.118 将浮空山移动到画面下方

将确定的边界值填入第二个 Random BB，如图 3.119 所示。

图 3.119 设置第二个 Random BB 的输入参数

最后，将第一个 Random（用来随机 X 坐标）的输出参数连接至 Set Component 的第一个输入参数，将第二个 Random（用来随机 Y 坐标）的输出参数连接至 Set Component 的第二个输入参数，并确定 Set Component 的第三个输入参数被设定为了 1000（Z 坐标），完成的连线图如图 3.120 所示。

图 3.120 随机坐标参数连线

完成随机坐标后，将 Set Component 的输出参数（随机坐标的结果）连接至 Set Position 的 Position 输入参数，如图 3.121 所示。运行测试结果，如图 3.122 所示。

图 3.121 完成随机浮空山脚本连线

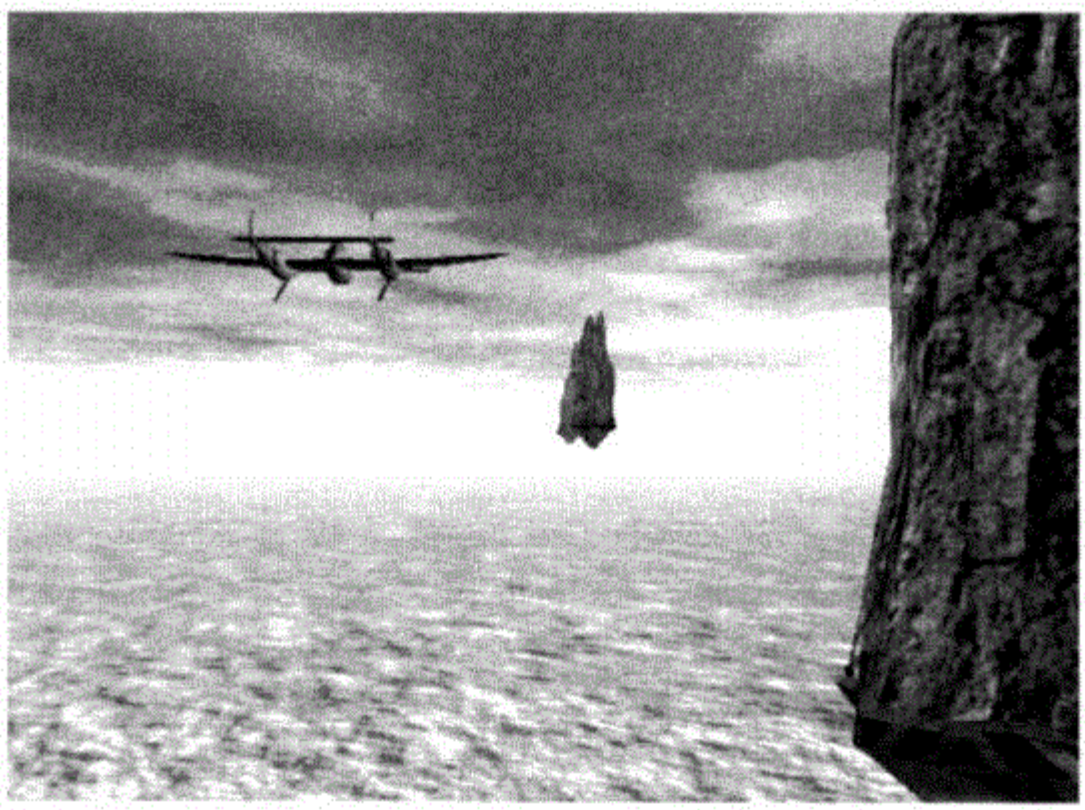


图 3.122 测试随机浮空山运行效果

仅仅是位置上的随机还稍显不够，最后还可以让浮空山出现的时间间隔具备一定的随机性，原理也比较简单，即将 Delayer 的延迟时间由 Random 产生，具体脚本和参数设置参见图 3.123 和图 3.124 所示。

图 3.123 随机浮空山最终脚本

图 3.124 随机创建时间间隔 Random BB 参数

在测试中，可以根据自己的测试结果来调整各项数值以达到满意的效果，经过调整后的最终工程文件可以在这里找到 CMOs\Plane\Plane_07. cmo。

3.3.2 旋转的陨石

下面继续为场景中添加陨石。它与浮空山非常类似，只是在旋转的时候可以绕任意轴旋转（浮空山只绕 Y 轴旋转），因此可以将大部分脚本复制并粘贴给陨石以简化开发流程。

1. 导入陨石模型

陨石模型的资源文件为 CMOs\Plane\Meteor. nmo，使用 Import File 导入模型，将其隐藏，关闭激活开关，并设置为初始状态（与浮空山一样），如图 3.125 所示。

图 3.125 隐藏并关闭 Meteor

2. 为陨石复制脚本

在 Level Manager 中，将 Mountain 的三个脚本复制一份给 Meteor，如图 3.126 所示。将 Level Script “创造浮空山” 复制一份并更名为 “创造陨石”，如图 3.127 所示。

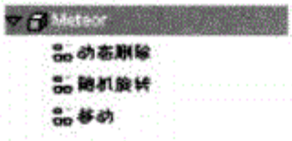


图 3.126 复制 Mountain 脚本给 Meteor

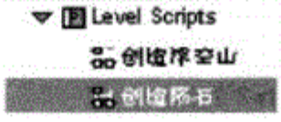


图 3.127 重命名脚本

3. 更改脚本参数

首先进入 “创造陨石” 脚本，在这里需要将 Object Copy 的 Original 更改为 Meteor（见图 3.128）。

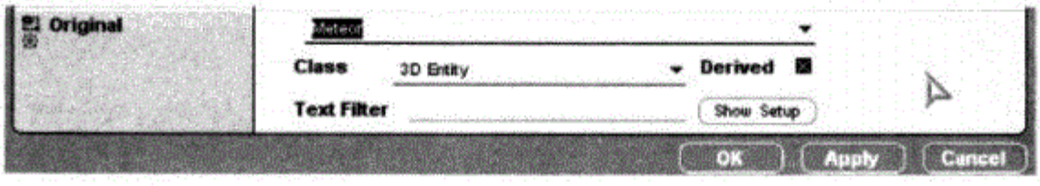


图 3.128 修改 Object Copy BB 的 Original 输入参数

然后依照得到浮空山 X、Y 随机数范围的方法来确定陨石的 X、Y 数值范围，如图 3.129 和图 3.130 所示。

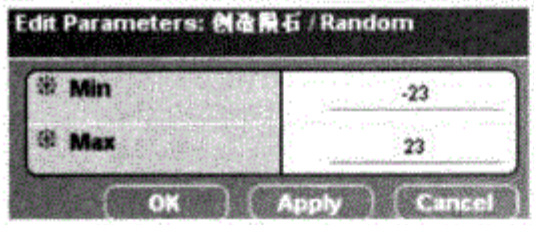


图 3.129 第一个 Random BB 输入参数

图 3.130 第二个 Random BB 输入参数

陨石的产生时间间隔可以更短暂一些，如图 3.131 所示。

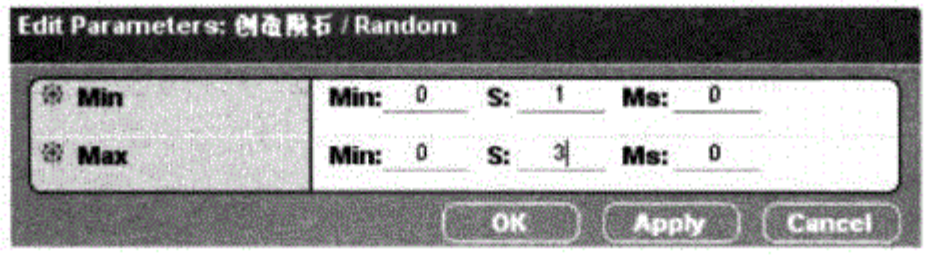


图 3.131 陨石随机时间间隔

接下来进入陨石的“移动”脚本，让陨石的移动速度具有一定的随机性。

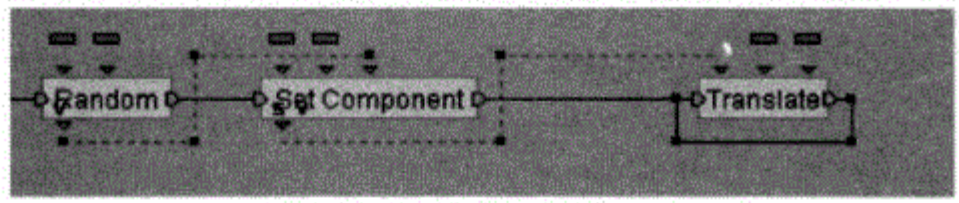


图 3.132 随机陨石移动速度脚本

在图 3.132 和图 3.133 所示的脚本中将陨石每一帧的移动向量由固定的(0，0，-1.5)改为随机的（0，0，随机数），即达到了随机移动速度的效果。

陨石原本的旋转坐标轴为固定的（0，1，0），现在让旋转坐标轴也随机产生，此时无须使用 Set Component 搭配三个 Random 来产生一个随机向量，直接将 Random 的输出参数类型设置为 Vector 即可，如图 3.134 和图 3.135 所示。

图 3.133 随机移动速度 Random BB 参数

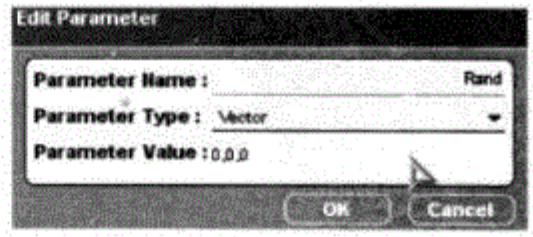


图 3.134 随机旋转 Random BB 输入参数类型

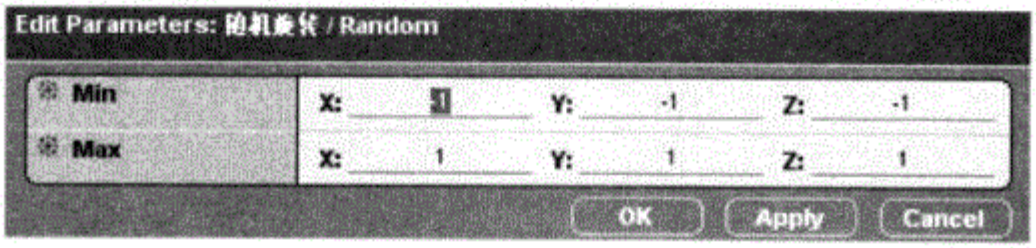


图 3.135　随机旋转 Random BB 输入参数

更改过的随机旋转脚本如图 3.136 所示。

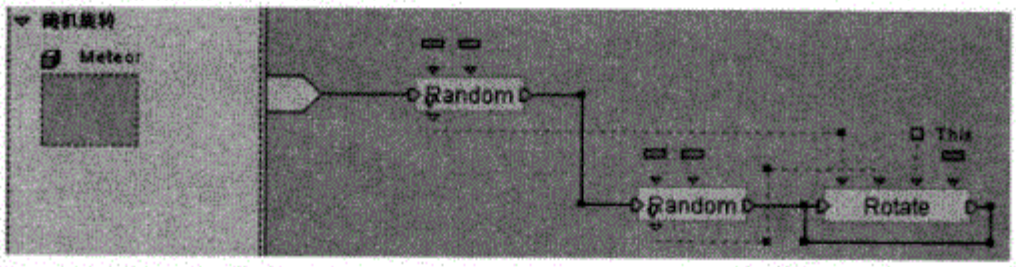


图 3.136　随机旋转脚本

可以适当提高陨石的旋转速度，获得更好的效果，如图 3.137 所示。

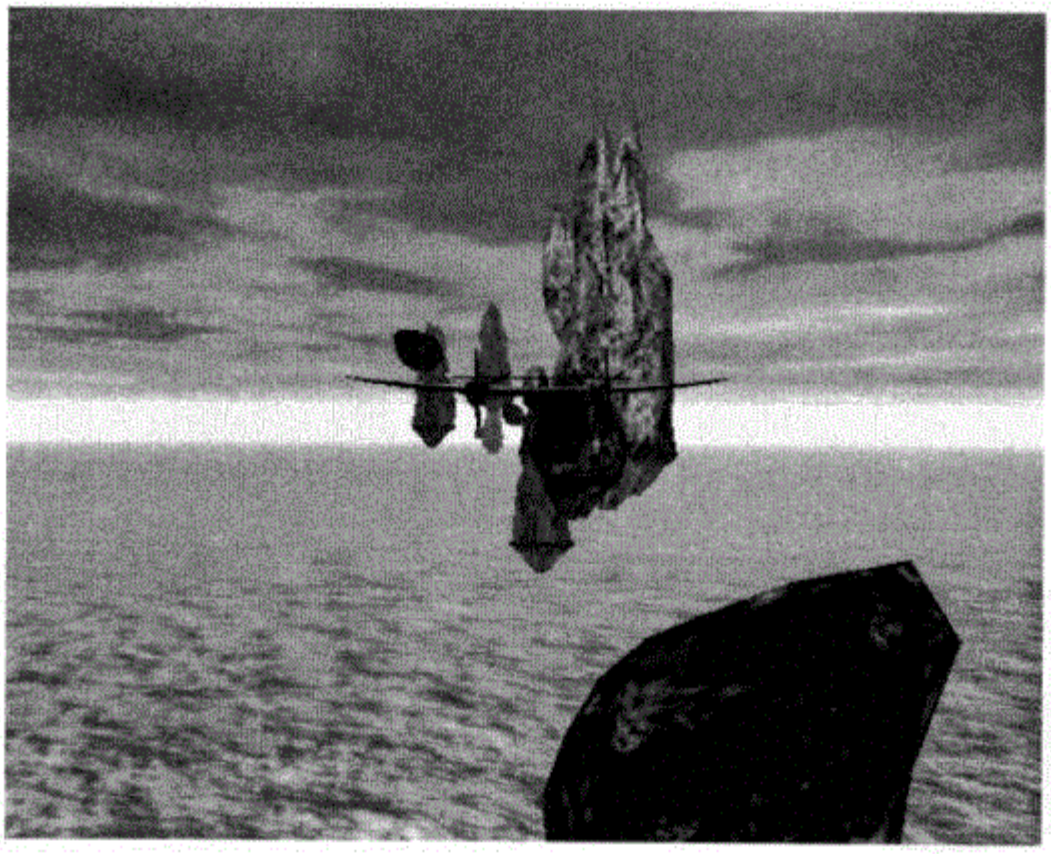


图 3.137　最终测试效果

至此的工程文件可以在 CMOs\Plane\Plane_08. cmo 找到。

3.4 本章回顾

本章首先介绍从3DS Max中导出Virtools的NMO文件的方法，并介绍了导入文件的调整、新建灯光、设置Specular属性以及调整材质的方法，最后尝试编写第一个简单的脚本——让螺旋桨不停旋转。这部分需要重点掌握的概念有：

（1）材质各个颜色通道的意义；

（2）灯光和材质的Specular属性及其应用；

（3）物体之间的继承关系；

（4）Initial Condition（IC）的意义及作用；

（5）Building Block各个部分的含义；

（6）脚本运行的基本逻辑；

（7）BB自循环的原理。

在进一步的学习中，添加了浮空山和陨石两种物体，它们都是在游戏的进程中动态创建的物体，并且具备一些随机性。这里需要重点掌握的概念是：

（1）动态物体的概念以及使用方法；

（2）Random BB；

（3）Set Component BB。

3.5 课后练习

1. 尝试从3DS Max或Maya等建模软件中导出一个模型，并导入Virtools进行调整，使其达到较好的显示效果。

2. 尝试向飞机游戏中加入更多障碍物，利用随机性来丰富游戏的效果。

第 4 章　完善游戏原型

数字游戏中的特效具有增强游戏氛围、完善游戏表现的作用。大部分的特效都模拟了真实世界的效果，学习中要注意了解游戏元素的物理规律，并结合引擎的特性及相关的脚本运算来具体实现。本章介绍 Virtools 粒子系统（Particle System）的相关知识，并结合游戏制作的需要，介绍具体的参数设置方法。

4.1　粒子

粒子是三维游戏中不可缺少的视觉特效，本节将基于上一章的游戏原型，学习 Virtools 中的粒子使用方法。可以直接打开本节的工程文件 CMOs\Plane\Plane_08. cmo 开始练习。

4.1.1　浮空山火箭喷射器

本节的第一个粒子效果是为浮空山的基座增加火箭喷射的效果。

1. 关闭 Level Script

在进行粒子制作和调整之前，先关闭一些不必要的脚本以方便制作，在这里将两个动态创建物体的 Level Scripts 关闭，如图 4.1 所示。

图 4.1　关闭动态创建物体 Level Script

然后显示 Mountain 并重新设置初始状态（IC），如图 4.2 所示。

图 4.2　显示 Mountain

2. 新建 3D Frame

火箭喷射器实际上是一个粒子喷射装置，它位于浮空山的底部，不断向下喷射粒

子。在 Virtools 中，通常情况下是使用 3D Frame 作为粒子系统的承载体。

Virtools 中的 3D Frame 是一种特殊的三维物体，它没有 Mesh 结构，因此无法被渲染，即无法在游戏运行的时候显示 3D Frame，也可以把它理解为一种用作标记三维世界中某一个点的特殊物体。它的用途非常多，粒子是它的重要应用之一。

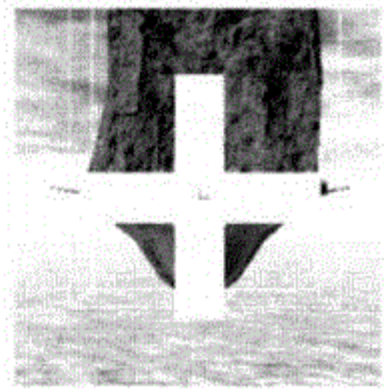


图 4.3 新建得到的 3D Frame

首先创建一个 3D Frame，在工具面板中单击 Create 3D Frame 按钮。创建一个新的 3D Frame，如图 4.3 所示，它可以被旋转、移动和缩放，与操作三维物体一样。

在 Level Manager | Global | 3D Frames 下找到新建的 3D Frame，为它重命名为“浮空山火箭喷射器”，如图 4.4 所示。

在继承关系设置面板（Hierarchy Manager）中，将其指定为浮空山的子对象，如图 4.5 所示，以便浮空山能带着喷射器进行旋转和移动。

图 4.4 新建“浮空山火箭喷射器”脚本

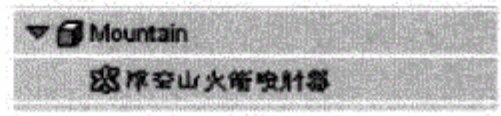


图 4.5 设置继承关系

进入 3D Frame Setup 面板，将 Position-Local 的 Z 值设置为 0，调整 Y 值，使 3D Frame 居于浮空山底座下方，如图 4.6 和图 4.7 所示。

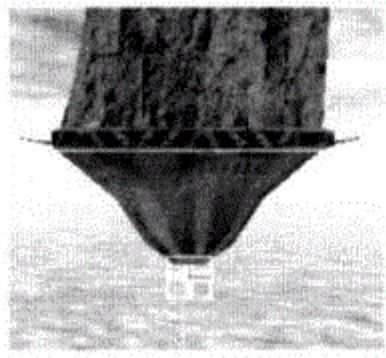


图 4.6 移动 3D Frame 至浮空山下方

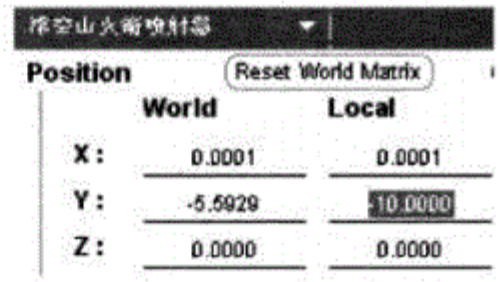


图 4.7 3D Frame 位置坐标

3. 将 3D Frame 转换为粒子发射器

为“浮空山火箭喷射器”新建一个脚本，命名为“火箭喷射器”。将 Building Blocks | Particles 目录下的 DiscParticleSystem BB 拖曳到脚本中并连线，如图 4.8 所示。

图 4.8 DiscParticleSystem BB

此时 3D Layout 中的 3D Frame 外形发生了变化，如图 4.9 所示。新的外形像一个盘子，切换到 Perspective View 近距离的效果如图 4.10 所示。

这里使用的是“碟式粒子系统”。圆形轮廓线标识的即为碟形粒子发射口，粒子将从“盘子”中发出，三角箭头表示发射粒子的方向，可以直接运行测试一下，如图 4.11 所示。白色的小立方体不断从圆形中发出并沿着三角箭头的方向飞行，这样就完成了粒子系统的构建。

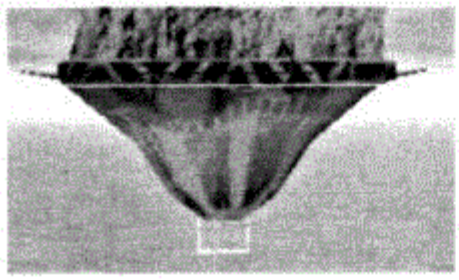


图 4.9 外形改变后的 3D Frame

图 4.10 近距离观察 3D Frame

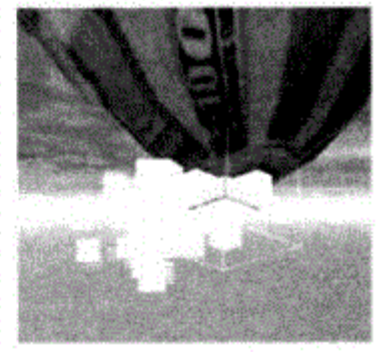


图 4.11 测试粒子效果

4. 设置粒子系统

火箭喷射器是向下喷射粒子，这样可以产生让山体悬浮的动力。如图 4.12 所示，旋转“火箭喷射器”使三角指示标志朝向正下方，粒子发射器旋转方向的参数如图 4.13 所示。

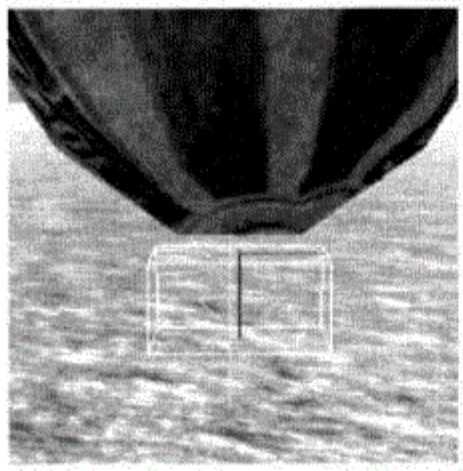


图 4.12 调整粒子发射器方向朝下

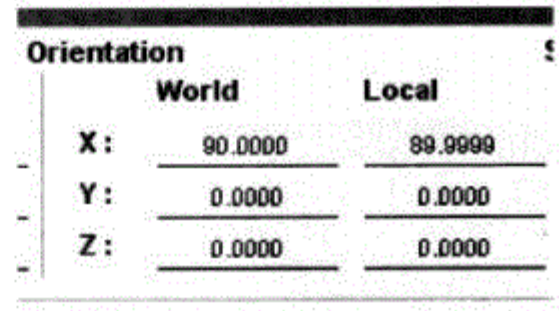


图 4.13 粒子发射器旋转方向

接下来添加粒子的贴图，没有贴图的粒子便会像被渲染为小方块。打开 Virtools Resources，在 Texture | Particles 目录下将 Simple. jpg 拖动至工程中并将其重命名为“火箭喷射粒子”。

回到脚本书写区域，双击 DiscParticleSystem BB，弹出的设置面板比较复杂，在此

只做如图4.14所示的更改：

Texture（粒子纹理贴图）设为火箭喷射粒子；

Initial Color and Alpha（起始颜色及透明度）设为黄色（A=255不透明）；

Ending Color and Alpha（结束颜色及透明度）设为红色（A=0全透明）。

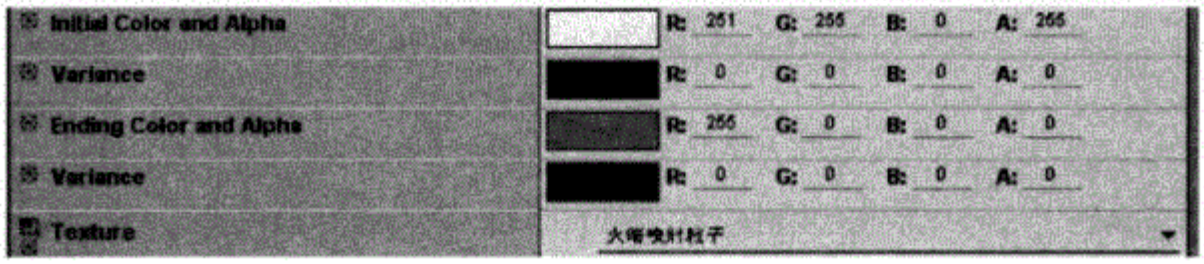


图4.14 粒子设置参数

运行测试可以看到如图4.15所示的效果。

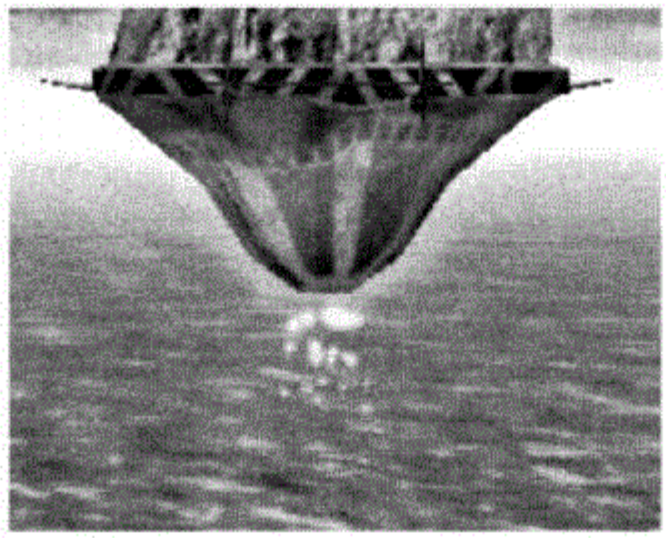


图4.15 粒子测试效果

喷射的粒子成了圆形，每一个粒子的颜色从刚刚被喷射出的黄色渐变为红色并消失。

不必终止运行，再次打开设置面板，如图4.16所示进行调整，注意画面效果的变化，单击Apply可以观察更新后的效果。其中：

Yaw Variance为偏角，粒子发射时偏角的变化范围。

Pitch Variance为仰角，粒子发射时仰角的变化范围。

调整后粒子由原来的“散射”变为向下方的“直射”，如图4.17所示。

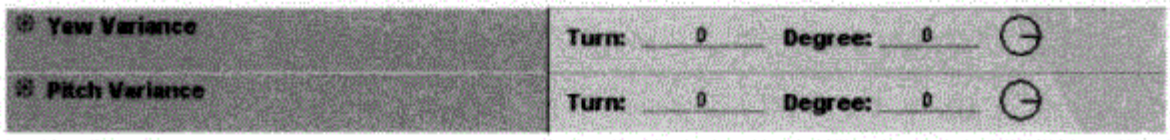


图4.16 粒子发射角度参数设置

Emission 为每次发射粒子的数量。

Emission Variance 为每次粒子发射数量的变化值（增加随机性），如图 4.18 所示。

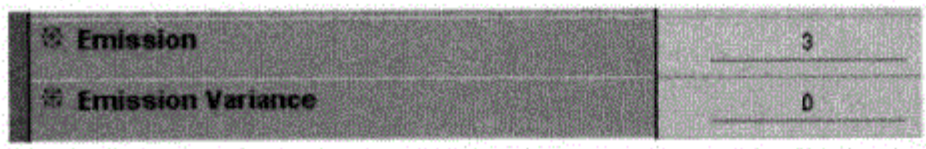


图 4.18　粒子发射数量参数设置

图 4.17　调整发射角后的粒子效果

Emission Delay 为每两次发射粒子之间的时间间隔，如图 4.19 所示。

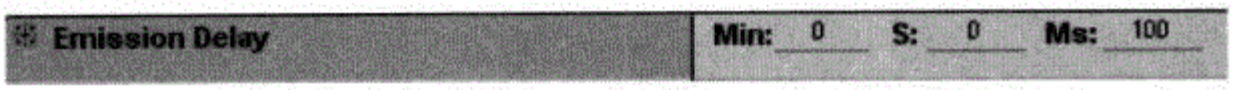


图 4.19　发射时间间隔参数设置

调整后，粒子发射的间隔变短，每一次发射的粒子数量减少，如图 4.20 所示。

图 4.20　调整发射数量和时间间隔后的效果

如图 4.21 所示，调整粒子尺寸参数设置，其中：

Initial Size 为粒子起始大小，表示每一个粒子被发射时的大小；

Initial Size Variance 为粒子起始大小变化值（增加随机性）；

Ending Size 为粒子终止大小，表示每一个粒子在消失之前的大小；

Ending Size Variance 为粒子终止大小变化值（增加随机性）。

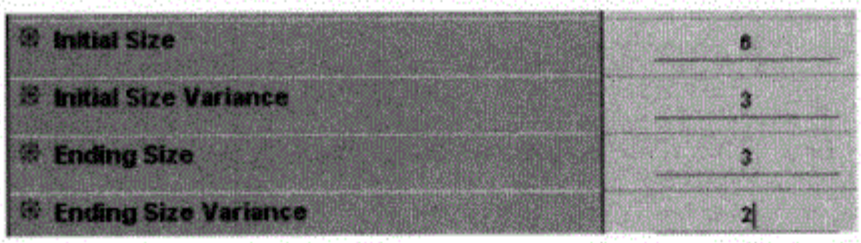


图 4.21　粒子尺寸参数设置

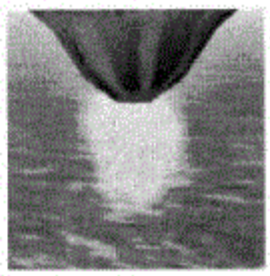


图 4.22　调整粒子尺寸后的效果

调整后粒子变大，如图 4.22 所示。

如图 4.23 所示，调整粒子飞行速度。其中：

Speed 为粒子的飞行速度。

Speed Variance 为粒子飞行速度变化值（增加随机性）。

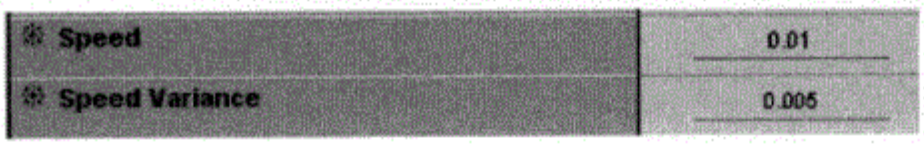


图 4.23　粒子飞行速度参数设置

调整后火焰喷射的距离增长了，如图4.24所示。

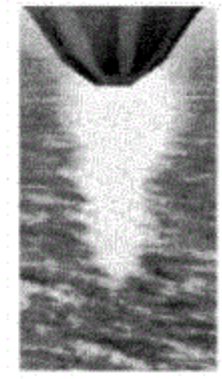


图4.24 调整粒子飞行速度后的效果

5. 在游戏中测试效果

下面在游戏中实际测试一下效果。首先，将“浮空山火箭喷射器”的激活开关关闭，如图4.25所示，和浮空山一样，所有火箭喷射器都将在游戏中动态创建。重新设置初始状态（IC）。

图4.25 关闭“浮空山火箭喷射器”

将浮空山隐藏，在关闭眼睛图标后，再单击一下眼睛图标，使关闭的眼睛图标上出现字母“H”，表示隐藏所有继承的子对象（H是Hierarchy的缩写），如图4.26所示。

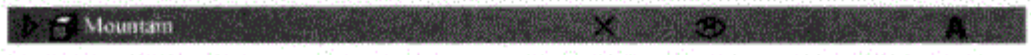


图4.26 隐藏所有继承子对象

此时浮空山火箭喷射器也被隐藏了，眼睛图标变为了透明关闭效果，这是被上层对象隐藏的标志，如图4.27所示。

图4.27 被继承隐藏的火箭喷射器

开启两个Level Script，如图4.28所示。

图4.28 开启Level Script

运行测试，可以看到每一个浮空山下方都增加了一个火箭喷射器并不断喷射着烈火，如图4.29所示。

4.1.2 陨石星尘

接下来再为陨石增添一些粒子效果。

1. 增加物体粒子系统

这里将直接使用Meteor作为粒子发射器，而不像浮空山火箭喷射器那样使用一个3D Frame作为粒子系统的承载体。

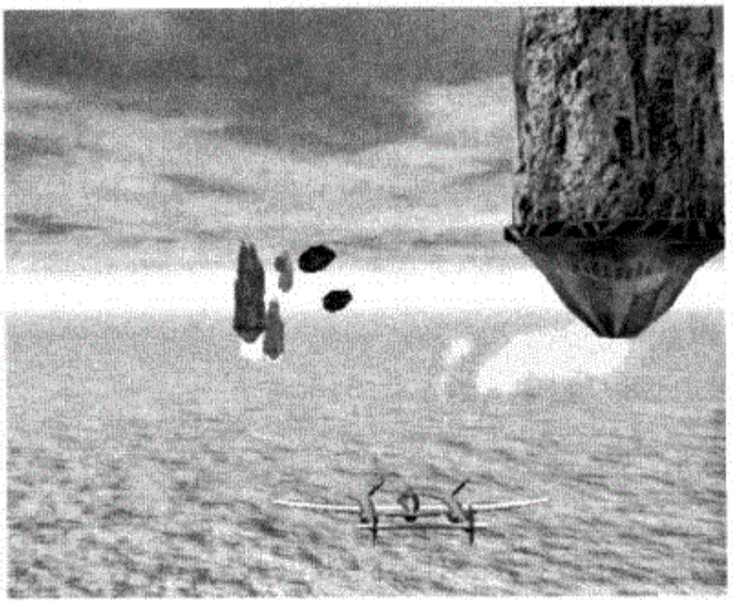


图 4.29　测试添加粒子后的游戏效果

为 Meteor 新增一个脚本并重命名为“星尘粒子”。在脚本中增加 ObjectParticleSystem BB，如图 4.30 所示。

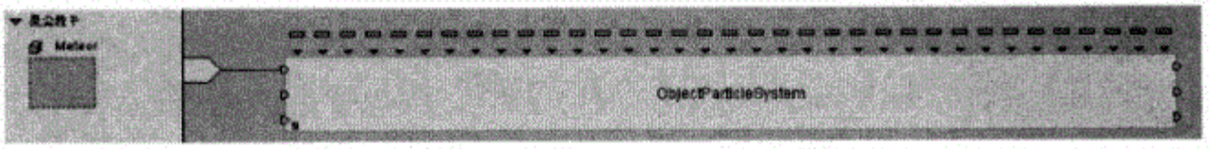


图 4.30　为 Meteor 增加 ObjectParticleSystem BB

运行测试效果如图 4.31 所示。

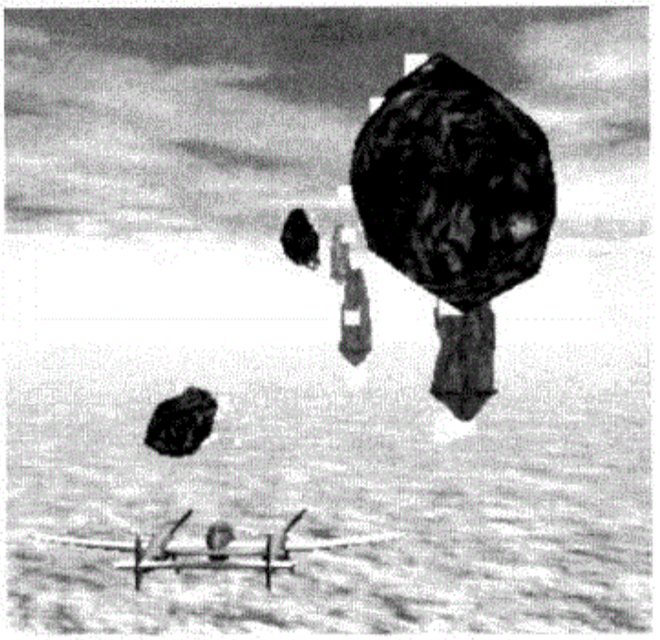


图 4.31　测试陨石粒子效果

2. 调整设置选项

在 Virtools Resources 素材库 Textures | Particles 的目录下将 Particle. bmp 加入到工程文件中，并设置其为 ObjectParticleSystem 的 Texture，如图 4. 32 所示。

图 4. 32 设置陨石粒子贴图

运行测试，效果如图 4. 33 所示。

根据需要可以自行试验并调整粒子系统的各个选项，直到达到满意效果，如图 4. 34 所示。

图 4. 33 测试陨石粒子贴图效果

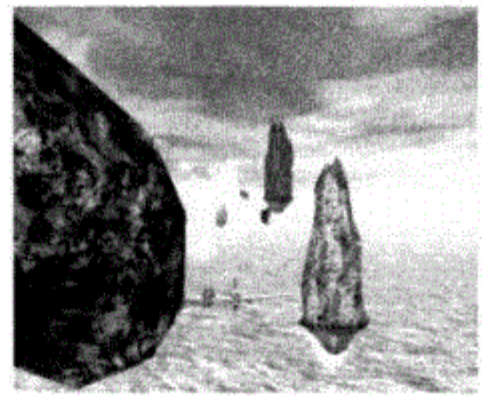


图 4. 34 最终陨石粒子效果

4. 1. 3 飞机爆炸效果

接下来将使用粒子制作飞机碰撞到障碍物后的爆炸效果，并介绍一些关于粒子的新知识。

1. 新建 3D Frame

图 4. 35 新建“爆炸”脚本

与制作“浮空山火箭喷射器”一样，首先需要新建一个 3D Frame 作为爆炸粒子效果的载体，并将其重命名为“爆炸粒子效果”。为其新建脚本并命名为“爆炸”，如图 4. 35 所示。

在脚本书写区域新建 PointParticleSystem BB，如图 4. 36 所示。

图 4. 36 PointParticleSystem BB

将 3D Frame 的方位坐标设置为（0，0，0），以便检测实际效果。

2. 调整粒子系统参数

与之前所做的两个粒子系统一样，首先应该为其指定纹理贴图（Texture），然后再

调整设置。选择 VirtoolsResources\Textures\Particles\目录下的 ParticleSmoke. tga 为纹理贴图，如图 4. 37 所示，用来模拟爆炸的烟雾效果。爆炸的最终效果如图 4. 38 所示。

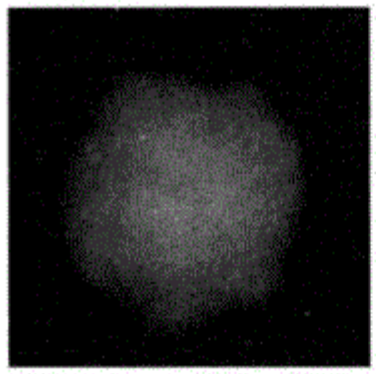


图 4. 37　爆炸烟雾效果粒子贴图

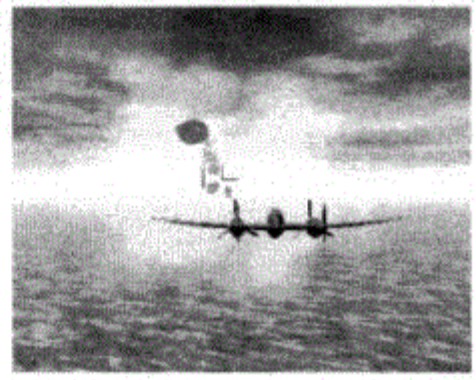


图 4. 38　爆炸效果

设置中需要注意，图 4. 39 展示的设置面板中 Maximum Emission 和 Emission Variance 都被设置为 50，它的意义是最多允许的粒子数量为 50，而一次发射的粒子数量

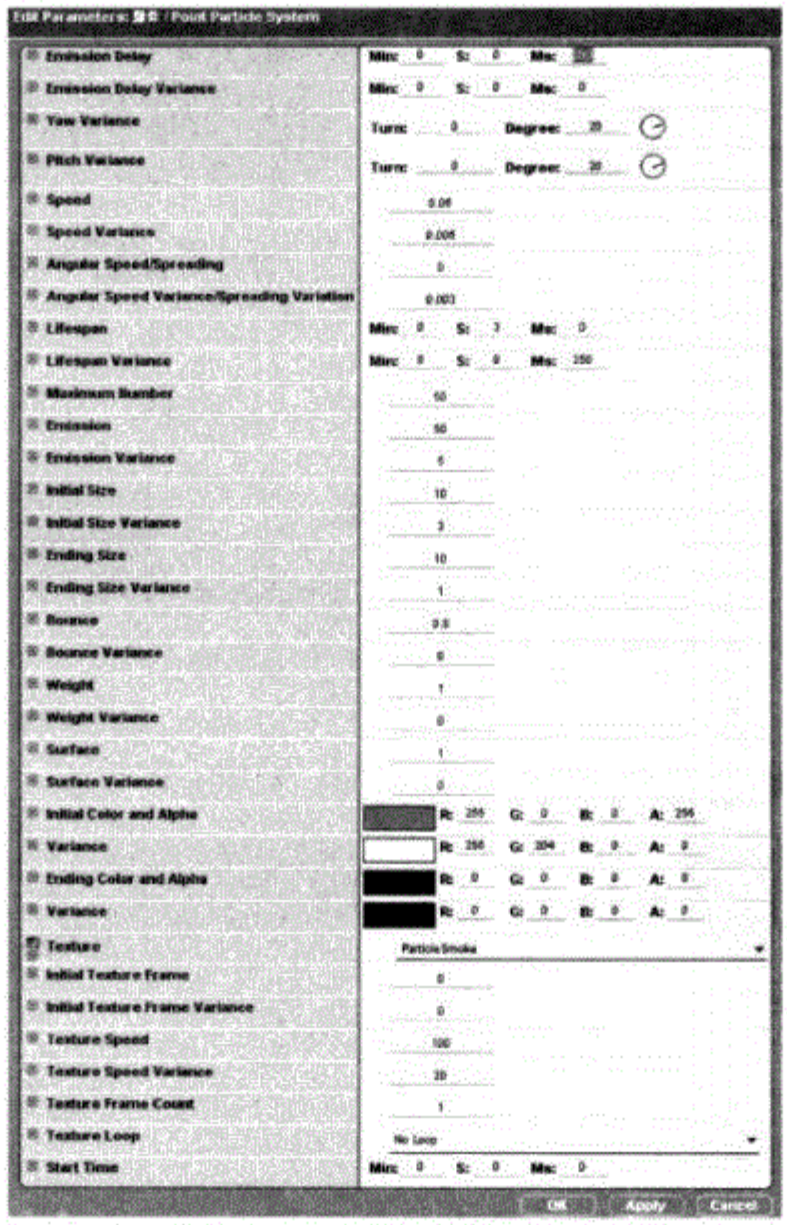


图 4. 39　爆炸粒子效果参数设置面板

也为50，在测试中可以看到此时粒子发射器发射粒子并不像之前那样是连续的，而是“一下一下”的发射，这是由于人为限制粒子数量的缘故，由于爆炸的效果只是一刹那的，这种设置方式比较合理。

为了让爆炸只发生一次，可以设定脚本在PointParticleSystem被激活后的下一帧关闭该BB，这样爆炸便会只发生一次。

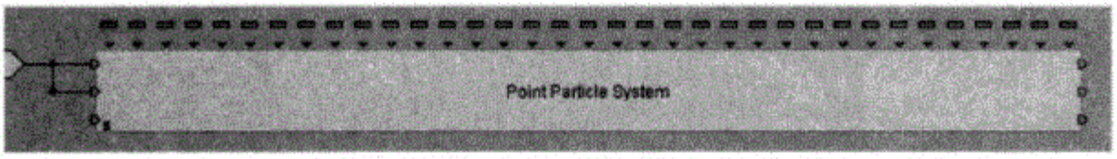


图4.40 动态关闭粒子脚本连线

如图4.40所示的脚本，第二根连线所连接的In端为Off，即关闭粒子系统，双击第二根连线可以设置连线延迟，默认的连线延迟是0，即表示无延迟，在此将其设置为1，表示一帧的延迟，如图4.41所示。

图4.41 连线延迟设置面板

运行测试可以看到，“爆炸”只发生了一次便停止了。

3. 增加重力影响因素

下面让粒子受到重力的影响向下做加速运动，并将介绍Attribute（属性）的设定。

作为施加重力的物体，地面最为合适。在Level Manager中打开Ground的设置面板，单击左侧的Attribute按钮，如图4.42所示，进入属性设置面板。单击下方的Add Attribute按钮，如图4.43所示，为Ground新增一个属性。

图4.42 Attribute按钮

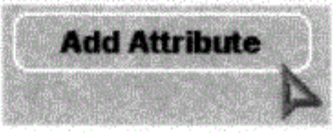


图4.43 Add Attribute按钮

在Particle System Integrators（粒子系统互动）分类下选择Particle Gravity并添加该项，如图4.44所示。

添加后保持默认值不变，如图4.45所示。

运行测试效果，如图4.46所示。飞机爆炸的火焰迅速下落，这是所受重力影响过大的缘故，但是陨石和浮空山喷射的粒子也都受到了重力的影响，如图4.47所示，因此需要调整一下三个粒子系统的设置。

首先让浮空山和陨石的粒子系统不受重力的影响，打开相应脚本，然后在ParticleSystem BB上单击右键并选择Edit Settings进入BB设置面板，如图4.48所示。

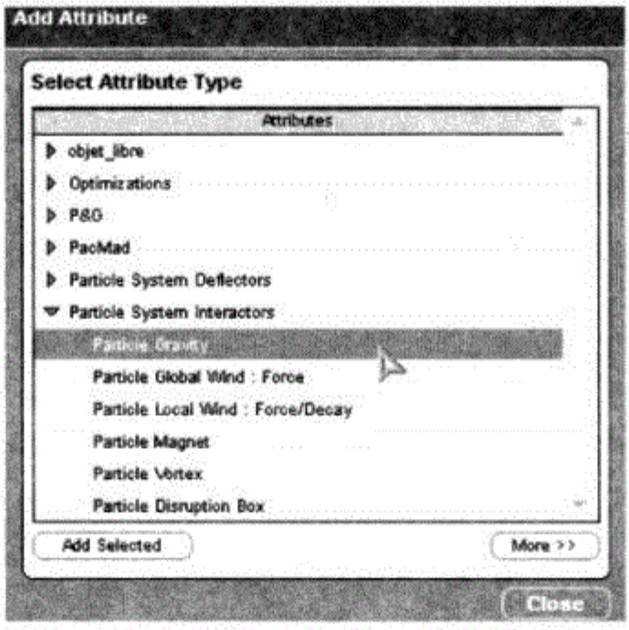


图 4.44 Particle Gravity 属性

图 4.45 添加 Particle Gravity 属性

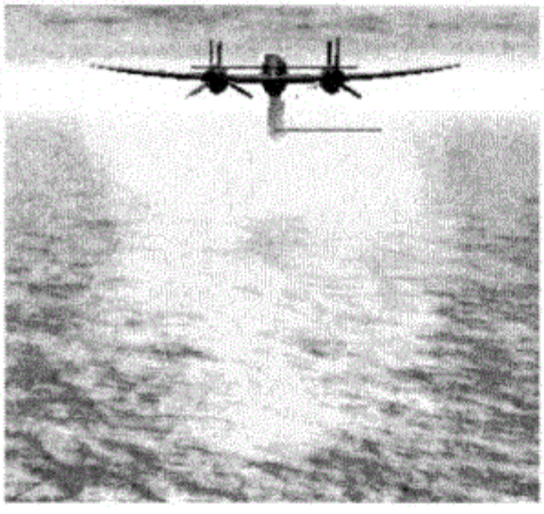


图 4.46 添加重力后测试效果

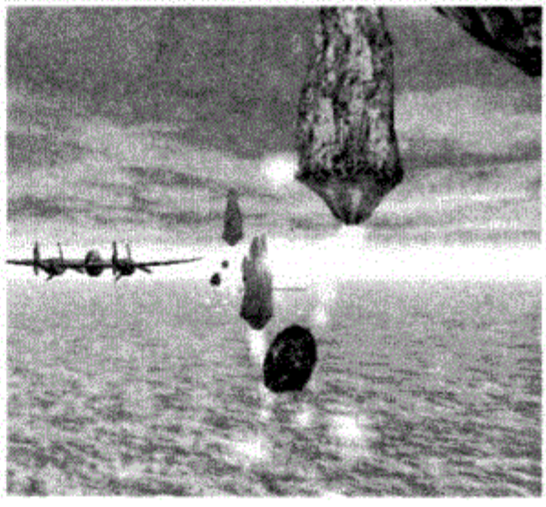


图 4.47 添加重力后测试效果

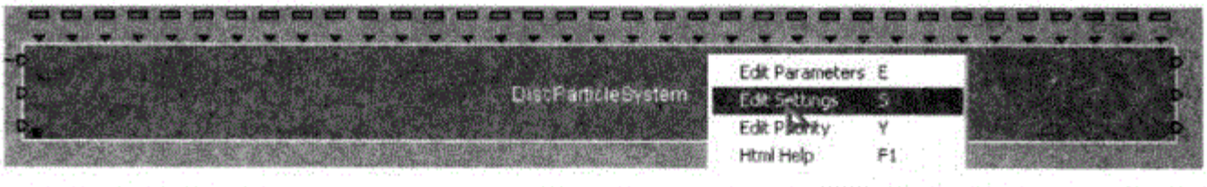


图 4.48 Edit Settings

在 Manager Inflectors 中将 Gravity 的选项删掉，如图 4.49 所示，这表示该粒子系统所发射出的粒子将不受到重力的影响。测试后可以看到浮空山和陨石的粒子不再受到重力的影响。

图 4.49 删除 Gravity 选项

然后需要调整爆炸粒子效果使重力影响的效果不那么明显，这可以通过两种方式得到，一是更改 Ground 的 Particle Gravity 属性的数值（Value），减小该数值以使得重力的效果变小，另外还可以通过更改粒子系统的设置参数来达到同样的效果，在此将使用第二个方法。

在爆炸粒子效果的 PointParticleSystem 的设置面板中更改 Weight（默认为 1）和 Weight Variance（默认为 0）的数值即可改变粒子的重力，如图 4.50 所示。

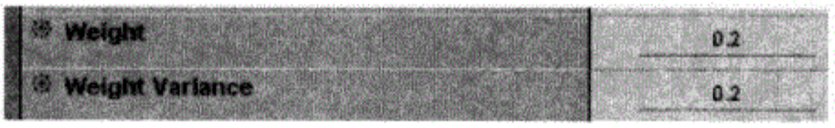


图 4.50 更改爆炸粒子效果重力参数设置

测试即可看到爆炸的粒子不再像之前那样快速下落了，调整该数值直到达到满意效果。本节结束的工程文件可以在 CMOs\Plane\Plane_08. cmo 找到。

4.2 添加核心互动要素

通过前几节的学习和制作，游戏原型已经初具规模，如图 4.51 所示。

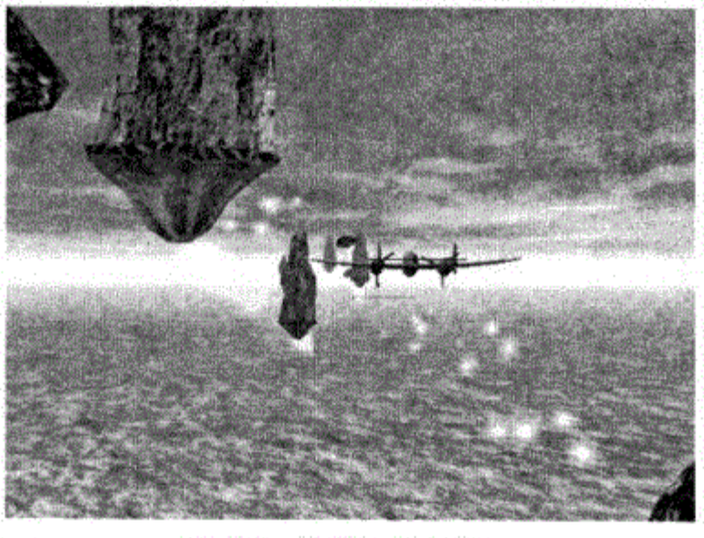


图 4.51 当前游戏原型画面

但是它还欠缺一个游戏完整的“循环”，即游戏的开端和结束，在本节中将为游戏世界添加碰撞检测，并为飞机设置“生命数量”。每当飞机撞上陨石或者浮空山后，都会损耗生命，生命为0后，游戏便会自动结束。

4.2.1 碰撞检测

Virtools中有多种方式能实现碰撞检测，这里仅以Collision Detection BB为例进行介绍，其他碰撞检测方式将在以后的制作中学习。

1. 新建脚本

首先为“飞机 机身”新建一个脚本，重命名为“碰撞检测”，在Building Blocks面板中拖曳Collision | Collision Detection模块至脚本中。该模块的作用是判定某一时刻，模块所在物体（脚本承载体）是否和“障碍物”发生了碰撞，关于何谓“障碍物”将在下面进行介绍，在此先要为该模块设置自循环，如图4.52所示。

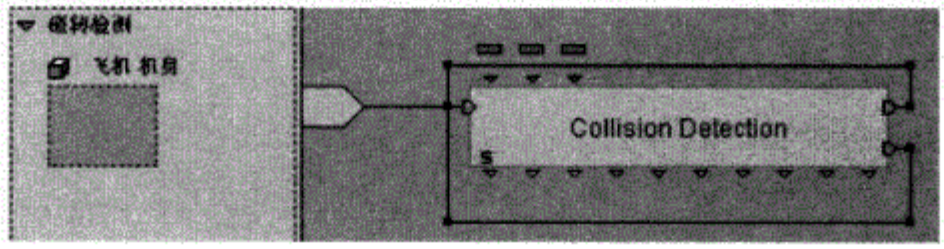


图4.52 Collistion Detection自循环连线

2. 设置Obstacle（障碍物）属性

Collision Detection模块的作用是检测某一时刻是否和“障碍物”发生碰撞，因此首要的问题便是确定障碍物，让程序知道哪些物体是障碍物，这需要通过Attribute（属性）来实现。

首先将浮空山设置为障碍物，打开Mountain的设置面板，单击左侧Attribute按钮，如图4.53所示，进入属性设置面板。

单击Add Attribute为Mountain增加一个新的属性，在列表中选择Collision Manager分类中的Moving Obstacle（移动障碍物），如图4.54所示。

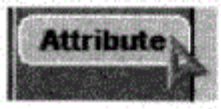


图4.53 Attribute按钮

图4.54 添加Moving Obstacle属性

为了测试碰撞检测的有效性，需更改“碰撞检测”脚本，使得发生碰撞（Collision Detection的True输出端被激活）后，运行Hide模块，隐藏飞机，如图4.55所示。

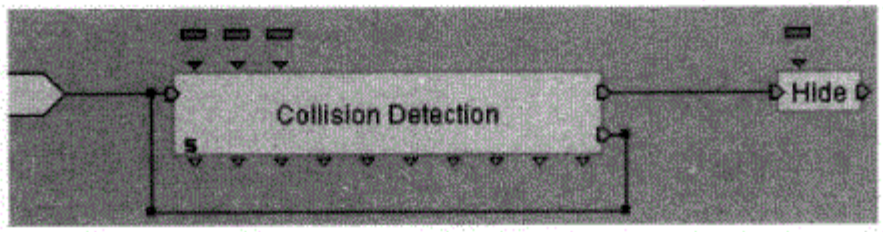


图 4.55 添加 Hide BB

此处 Hide 与以前常用的 Show 模块作用相反，即隐藏一个物体。如图 4.56 所示，将 Hide 的设置选项 Hierarchy 打开，使飞机连同它的子对象（螺旋桨等）一同被隐藏。

图 4.56 设置 Hide BB 的 Hierachy 选项

运行测试，会发现飞机未碰上任何东西就消失不见了，如图 4.57 所示。

实际上这是由于所隐藏的 Mountain 正位于世界坐标原点，即（0，0，0）点，和飞机的起始位置相同。所以飞机实际上是碰上了隐藏的浮空山，因此需要将浮空山移开，在此将 Mountain 的方位 Z 坐标设置为 -100 即可，然后重新设置初始状态（IC）。

再运行测试，当飞机碰撞到迎面飞来的浮空山后才会消失不见，如图 4.58 所示。

图 4.57 测试运行 Bug

图 4.58 测试碰撞效果

下面依照类似的方法，添加陨石的 Obstacle 属性，并注意将其位置也调整到飞机碰不到的地方，以免发生撞上隐形的陨石模板的问题。

3. 设置爆炸效果

以上实现了碰撞后飞机消失的简单效果，下面为其添加“爆炸粒子效果”。其原理很简单，即当飞机碰撞上障碍物后，隐藏飞机，并在飞机爆炸的地方动态复制一份“爆炸粒子效果”物体，使其播放粒子效果。

首先将 3D Frame“爆炸粒子效果”隐藏并关闭激活开关，设置初始状态（IC），如图 4.59 所示。

图 4.59 隐藏并关闭“爆炸粒子效果”

修改“碰撞检测”脚本，如图 4.60 所示。

图 4.60　修改后的“碰撞检测”脚本

Object Copy 用来复制 3D Frame“爆炸粒子效果”，Set Position 设置新复制的 3D Frame 的位置为在 This（飞机 机身）坐标参考系下（0，0，0）方位，即飞机当前的位置，Show 用来显示复制得到的“爆炸粒子效果”。

运行测试即可看到飞机撞到障碍物后的爆炸效果，如图 4.61所示。

至此为止的工程文件为 CMOs\Plane\Plane_09. cmo。

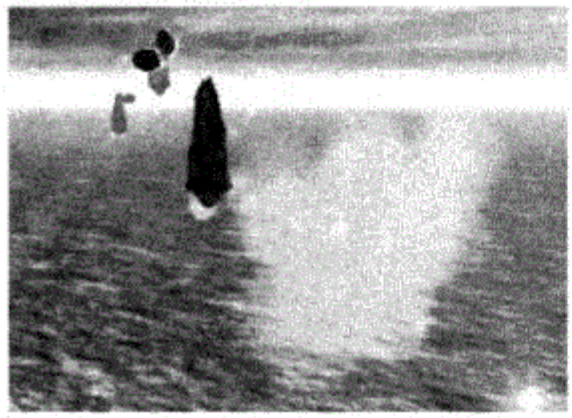


图 4.61　碰撞后的飞机爆炸效果

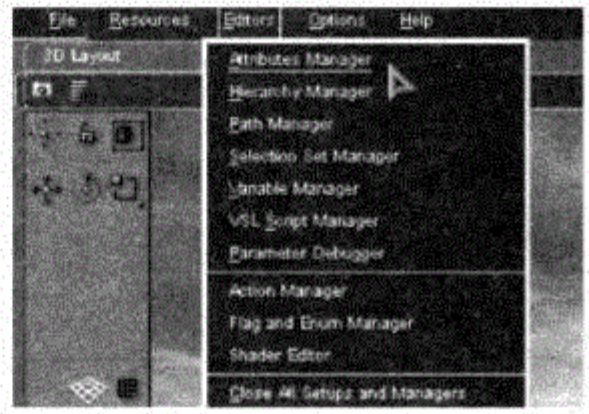


图 4.62　菜单打开 Attributes Manager

4.2.2　设置飞机生命数量

接下来为飞机设置生命数量。在生命没有耗光之前，飞机不会消失在游戏的世界中。

1. 增加生命属性

首先，需要自定义一个 Attribute（属性），用它来记录飞机还剩下多少生命值。在 Virtools 菜单中选择 Editors | Attributes Manager，打开属性控制面板，如图 4.62 所示。

该面板管理所有工程中的属性设定，列表中包括了所有自定义属性和 Virtools 自动的属性，下面新建一个自定义属性，选择左上角的按钮 Create Attribute，如图 4.63 所示。

首先新建一个分类用来存放该属性，在这里将其命名为 Plane，表示该属性是和飞机有关的，如图 4.64 所示。

图 4.63　Create Attribute 按钮

图 4.64　创建 Plane 属性分类

单击 Create 按钮可以看到在下方 Create New Attribute Type 中 Category 被自动指定为 Plane，即新建的分类，如图 4.65 所示。

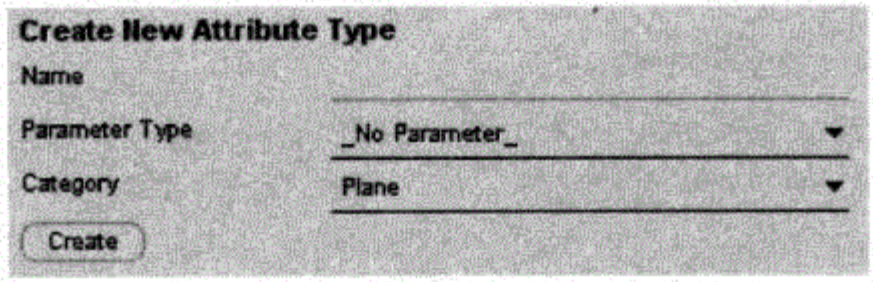


图 4.65 将新建属性的分类指定为 Plane

然后，将新属性的 Name 设置为 Lives Left，表示剩下的生命数量，在 Parameter Type 中将参数类型指定为 Integer（整数），如图 4.66 所示。

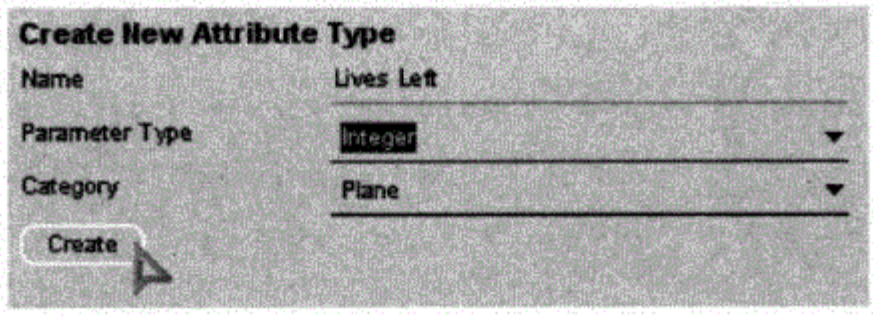


图 4.66 新建参数设置选项

单击 Create 按钮完成新属性的创建。

在 Attribute Manager 中即可找到新创建的 Plane 分类和其中的属性 Lives Left，如图 4.67所示。

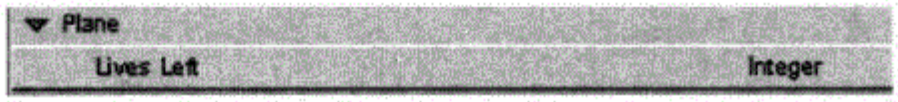


图 4.67 Attribute Manager 中新建的属性

打开“飞机 机身”的设置面板，为其添加 Lives Left 属性，如图 4.68 所示。双击 Value 将参数值设置为 3，表示飞机剩余生命为 3。

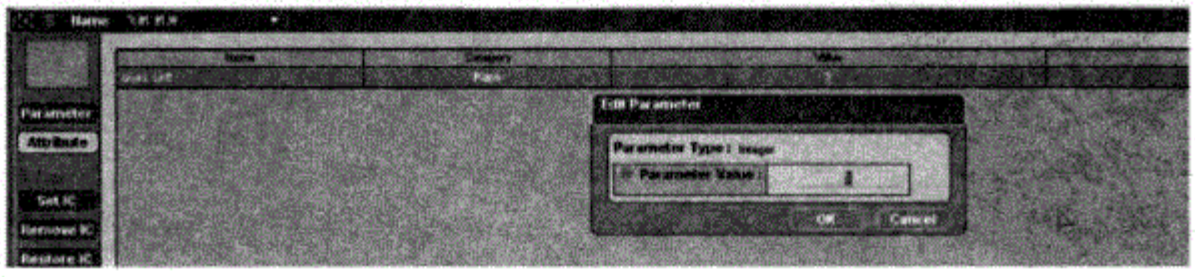


图 4.68 为飞机添加 Lives Left 属性

接下来设置它为初始状态（IC）。需要特别注意的是，一个物体的参数（Parameter，包括位置、旋转、缩放等）和属性（Attribute，包括所有属性信息）的初始状态是分开管理的，需要分别单独进行设置。

2. 编写飞机消耗生命的脚本

设置好属性后，便需要利用该属性编写飞机消耗生命的脚本。打开飞机的“碰撞检测”脚本，其中需要增加如下逻辑：

- 飞机碰到障碍物→让飞机消失，播放爆炸效果；
- 检测飞机剩余生命是否为0；
- 若为0，则让飞机消失，游戏结束；
- 若不为0，延迟3秒→减少 Lives Left 属性→让飞机重新出现在（0，0，0）的位置→飞机闪烁（此时无敌）→恢复碰撞检测。

具体脚本如图4.69所示。

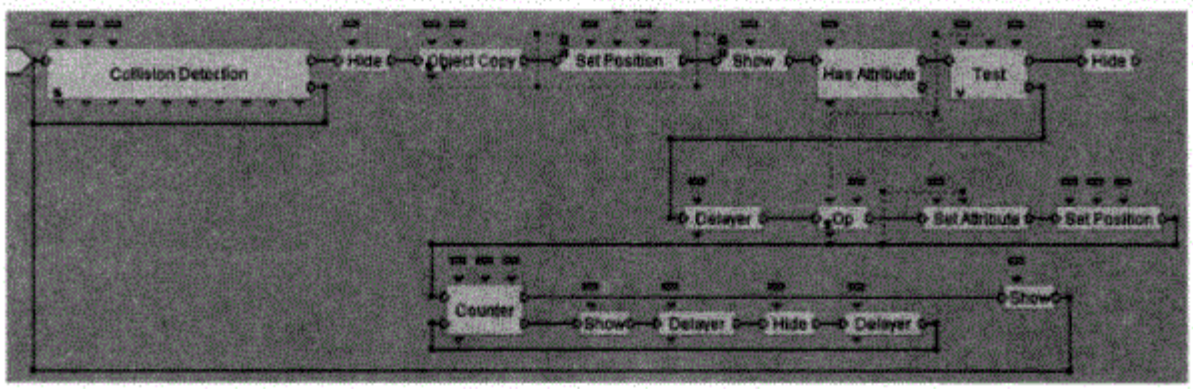


图4.69 飞机生命脚本

分部分解释脚本如下：

如图4.70所示，当 Collision Detection 输出端 True 激活后，首先执行飞机爆炸并隐藏飞机的脚本，这和以前写的脚本是一样的。

如图4.71所示，随后需要检测当前飞机的 Lives Left 属性值，即检测当前飞机剩下多少生命。

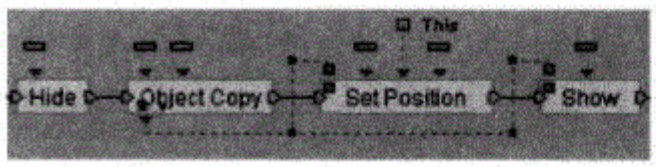


图4.70 飞机生命脚本之一

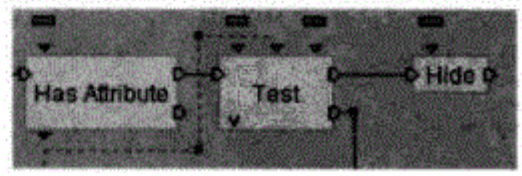


图4.71 飞机生命脚本之二

其中，Has Attribute（Logics | Attributes 分类下）的功能是读取当前物体的一个属性值，双击设置面板需要指定所读取的属性为 Plane 分类下的 Lives Left，如图4.72所示。

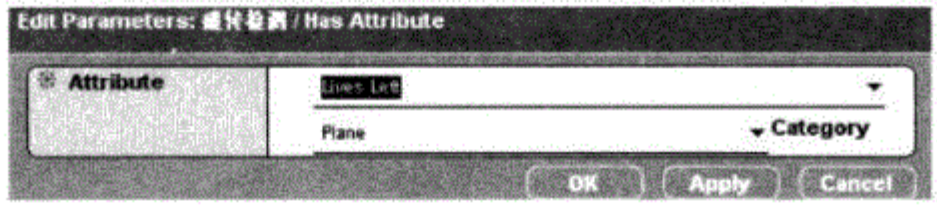


图 4.72 Has Attribute BB 输入参数设置

在 Test 中，需要比较的数值即为 Has Attribute 读出的属性值，将参数 A 和 Has Attribute 的输出参数相连，参数 B 设置为 0，检测方式定为 Equal，表示判断 A（剩余生命）是否等于 B（0）。在这里需要指定 Test 的 A 和 B 的输入参数类型为 Integer，如图 4.73 和图 4.74 所示，但是不做更改也没有关系，因为整数和默认的 Float（浮点数）类型之间是可以互相转换的。

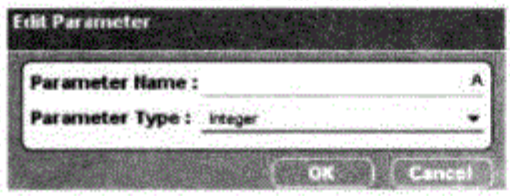


图 4.73 Test BB 输入参数 A 参数类型设置

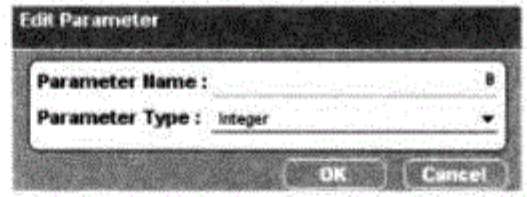


图 4.74 Test BB 输入参数 B 参数类型设置

当剩余生命为 0 的时候（Test 的 True 输出端激活），隐藏飞机，当剩余生命不为 0 的时候，执行图 4.75 所示的脚本。

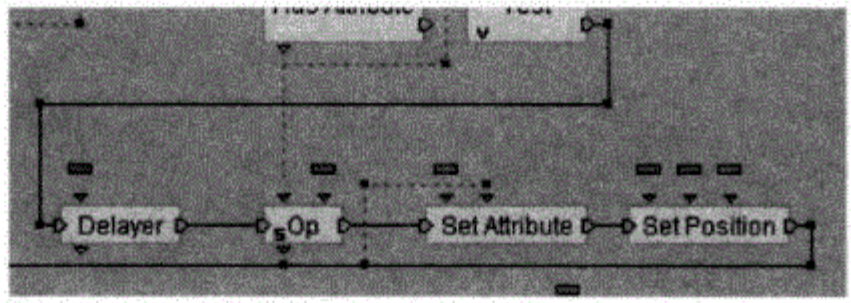


图 4.75 飞机生命脚本之三

首先使用一个 Delayer 模块进行延迟，延迟的时间为 3s，这是为了给游戏者一定的反应时间。然后，执行 Op，这个 Op 的作用是让整数类型参数 A 减去整数类型参数 B，如图 4.76 所示。

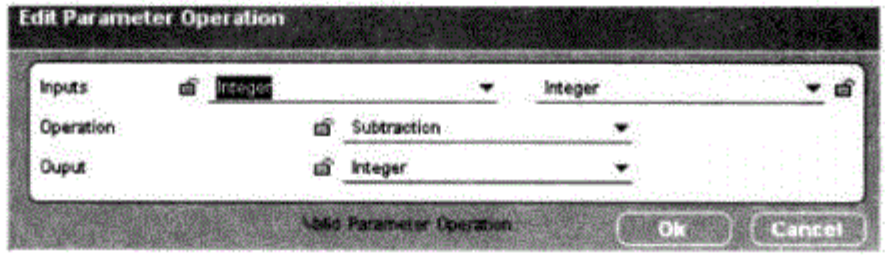


图 4.76 减法 Op BB 设置选项

参数 A 为 Has Attribute 的输出参数，即当前 Lives Left 的数值，而参数 B 设定为 1，因此 Op 的功能是让当前生命数减一。完成减一的操作后，需要将新的生命数重新设置给飞机。在这里使用的模块是 Set Attribute，这个模块和 Has Attribute 类似，需要先对属性进行设置，如图 4.77 所示。

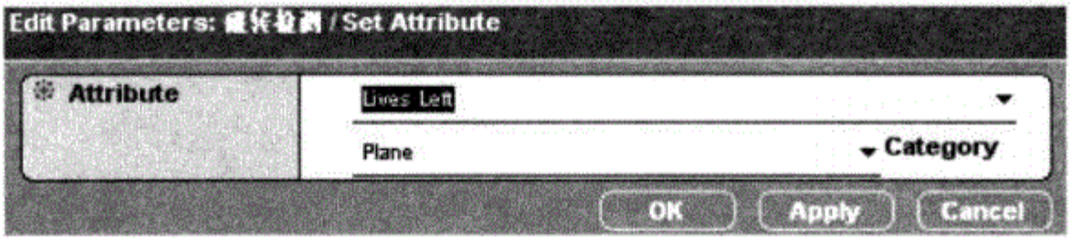


图 4.77 Set Attribute BB 的 Attribute 输入参数设置

指定属性后，Set Attribute 会根据所选择的属性为自己新增输入参数。比如这里的 Lives Left 属性为 Integer 类型，故 Set Attribute 会新增一个 Integer 类型的输入参数供新的属性值传入，在这里将其与 Op 的输出参数相连即可做到更新 Lives Left 的效果。

Set Position 的作用是让飞机移动到世界坐标中心，参数设置如图 4.78 所示。

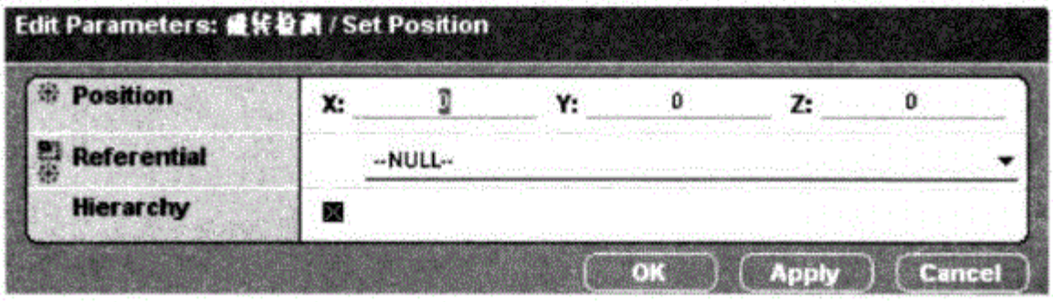


图 4.78 Set Position BB 输入参数设置

如图 4.79 所示，这一部分脚本是让飞机“闪烁”的脚本。其中，图 4.80 所示部分的作用是通过 Show Hide 的交替来实现飞机“闪烁一次”的效果，第一个 Delayer 的延迟时间为 300ms，第二个 Delayer 的延迟时间为 100ms。

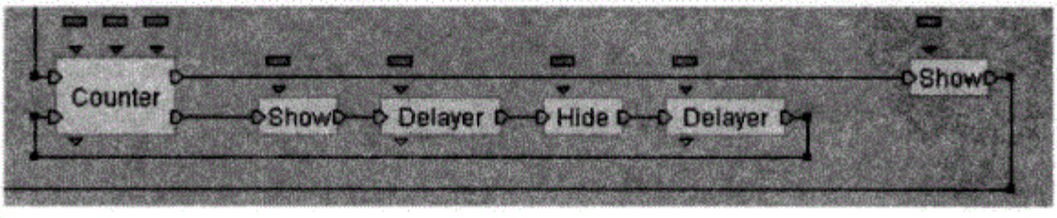


图 4.79 飞机生命脚本之四

这里使用了一个新的 Counter（计数器）脚本模块，它的作用是计数。双击可以打开设置面板，如图 4.81 所示。

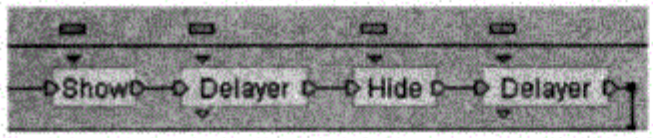


图 4.80 飞机闪烁一次脚本

图 4.81 Counter BB 输入参数设置面板

其中，Count 为计数数量、Start Index 为起始序号、Step 为计数步长。在这里希望飞机闪烁 5 次，因此将 Count 设置为 5，其他选项不做改变。Count 的运行逻辑如下：

- 激活 In 端后开启计数器→检测是否计数完成→
- 若完成→激活 Out；
- 若没有完成→计数→激活 Loop Out→执行用户添加的循环脚本→激活 Loop In→检测是否计数完成→循环；

这里脚本的作用是：利用 Count 循环执行“闪烁一次”这个脚本 5 次，使得飞机闪烁 5 次后，激活 Count 的 Out 端，然后运行 Show，使得飞机显示出来。

飞机显示出来后，即可恢复碰撞检测，开始新一轮的游戏。运行测试一下效果，若有任何问题，可以打开本书的工程文件 CMOs\Plane\Plane_10. cmo 查看。

4.3 本章回顾

本章首先介绍了 Virtools 粒子系统（Particle System）的使用方法，同时结合游戏需要实际添加了三种粒子效果。由于粒子系统包括的各种设置参数非常多，在此无法一一介绍，可以根据需要参考帮助文档中的相关章节进行学习和尝试，应该重点掌握的内容有：

（1）使用 3D Frame 加载粒子系统；

（2）使用 3D Object 作为粒子系统的载体；

（3）粒子系统设置面板中常用参数设置；

（4）设置重力属性影响粒子系统；

（5）开启或关闭粒子系统的重力影响设置。

在 4.2 节中，进一步完善了游戏原型，使其具有了最基本的游戏开始、进行、结束的游戏循环逻辑，本节中需要重点掌握的是：

（1）创建自定义属性；

（2）使用脚本读取并设置自定义属性；

（3）Counter 的运行逻辑。

4.4 课后练习

1. 利用粒子来模拟飞机射击的子弹。一颗子弹即一个不断向前飞行的小型粒子发射器。

2. 在现有的原型基础上，加入控制飞机射击的要素。

3. 在现有的原型基础上，加入浮空山和陨石可以被击毁的要素。

第5章 游戏界面制作

一个完整的游戏要有良好的开始和结尾，游戏界面具有非常重要的作用。游戏的界面与操作密切关联，不同的游戏有不同的玩法，游戏中的各种反馈和设定都要通过界面来体现。例如，在启动界面要展示游戏最具代表性的画面。主菜单界面以功能性为主，通常包括新游戏、读取进度、选项设定、版权信息、退出等项目。进入游戏，显示的则是游戏主界面，游戏界面适当的位置将显示操作中必须的属性、物品和技能设定等项目。

本章主要将介绍一个简单的游戏界面制作方法，并讲解 Virtools 中二维原件的使用和互动程序的编写。

5.1 游戏二维主选单制作

本节将为第3章的飞机游戏原型设计制作二维的游戏主选单，同时学习 Virtools 二维原件的添加使用和互动脚本的编写。本节将继续在原有游戏原型的基础上进行制作，可以打开之前自己的工程文件，也可以直接打开 CMOs\Plane\Plane_10. cmo 文件继续本节的学习。

5.1.1 制作前的准备工作

这里准备制作的二维游戏主选单只有两个选项：START GAME 和 ABOUT，前者用来开始游戏，后者用来显示游戏的版本信息，制作者联系方式和版权信息等。在开始制作之前，应该让游戏在玩家单击 START GAME 之前停下来。

首先关闭两个用来创造浮空山和陨石的 Level Scripts，并将其设置为初始状态，表示该脚本在初始状态下（玩家刚打开游戏）不是激活的，如图 5.1 所示。

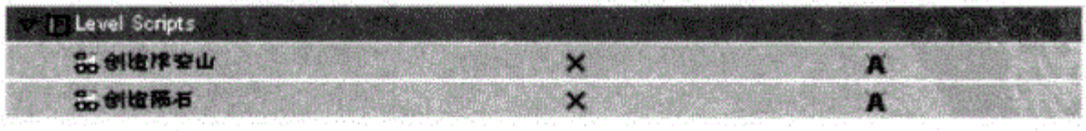


图 5.1 关闭创造浮空山脚本

为脚本设置初始状态后，若对脚本进行了更改但没有重新设置为初始状态，则在Restore IC（恢复初始状态）后，所有脚本更改将全部失效，需要特别注意。

运行测试一下，可以看到虽然陨石和山都不见了，但是飞机仍然可以被操控，地面也在卷动，因此需要关闭场景中其他物体，注意设置初始状态，如图5.2所示。

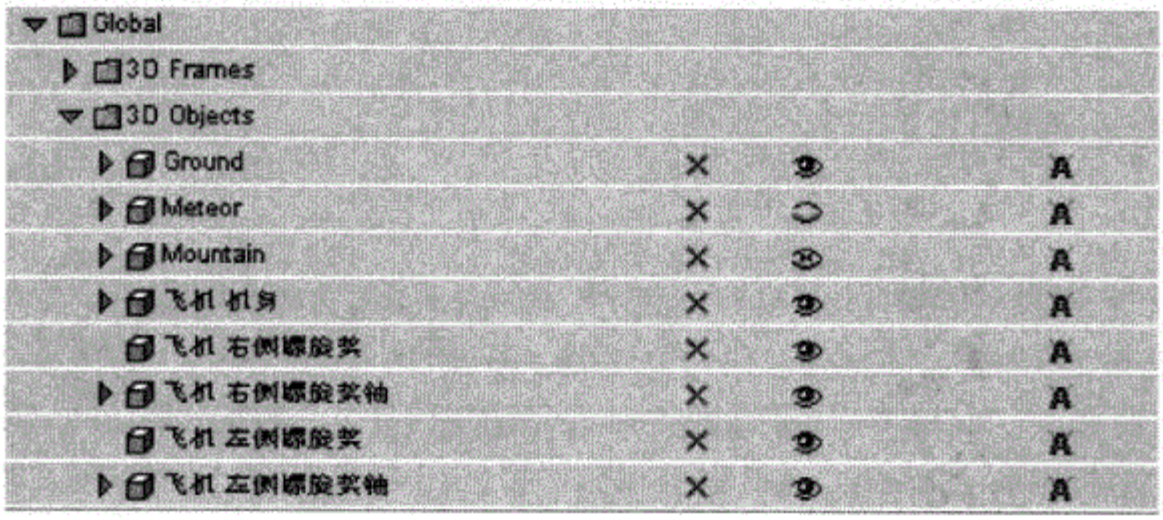


图5.2 关闭其他场景物体

再次进行测试，可以看到游戏已经完全停止。

5.1.2 添加游戏开始界面背景

为了让游戏者在游戏开始之前看不到游戏内的画面，需要添加一张二维图像来覆盖住整个场景，作为游戏开始界面的背景图。

1. 新建2D Frame

在Virtools中，二维平面物体称为2D Frame，首先来创建第一个2D Frame物体，单击3D Layout左侧工具面板的Create 2D Frame按钮，如图5.3所示。

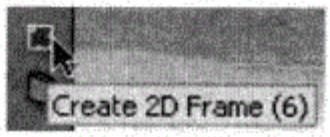


图5.3 Create 2D Frame

此时3D Layout中出现了一个透明灰色的长方块，即新创建的2D Frame，在2D Frame Setup面板中，将其重命名为“开始界面 背景”。

双击3D Layout标签，使3D Layout最大化，此时新创建的2D Frame位于画面上方，如图5.4所示。

这是由于2D Frame的坐标系并没有和Virtools游戏视窗的坐标系绑定的结果，所有2D Frame都有两类坐标系统，即屏幕坐标和画面坐标系统，当前的状态是以屏幕作为坐标系统，2D Frame的位置是用像素来表示的，如图5.5所示。

这个坐标系统以画面左上角作为坐标原点，在该坐标系下的2D Frame并不会随着画面的缩放而改变大小，显然不适合用作背景，因此需要采用第二种坐标系统，将其和游戏画面的坐标系相绑定，使2D Frame可以随画面缩放进行相应缩放。在General

一栏中，激活 Homogeneous Coordinates（齐次坐标系）选项，如图 5.6 所示。

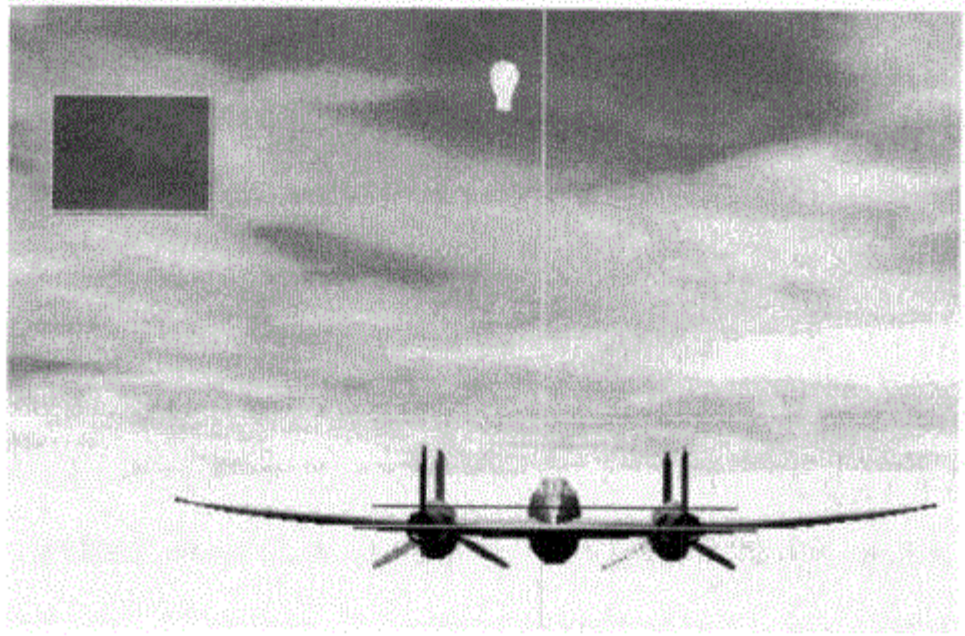


图 5.4 3D Layout 最大化后 2D Frame 跑到画面左上角

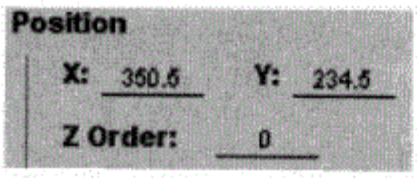


图 5.5 2D Frame 像素方位坐标

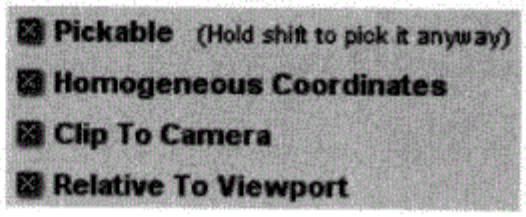


图 5.6 激活 Homogeneous Coordinates 选项

再测试一下，即可看到 2D Frame 随画面缩放的效果，同时可以注意到 Position 一栏的数值已经从刚才的绝对像素位置变为了百分比，如图 5.7 所示。

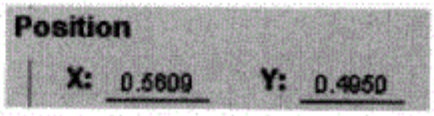


图 5.7 2D Frame 百分比方位坐标

最后更改 Position 和 Size 的数值，如图 5.8 所示，将这个 2D Frame 填满整个屏幕，效果如图 5.9 所示。

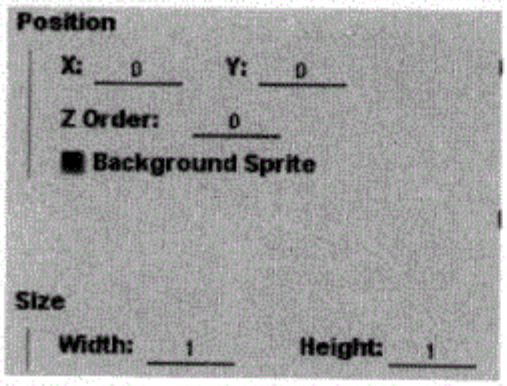


图 5.8 更改参数使 2D Frame 填满窗口

图 5.9 填满窗口的 2D Frame

Position 一栏中的“Z Order”表示该2D Frame的层级，数值越小，层级越靠下，上一层的2D Frame会遮挡下方层次的2D Frame。由于是用作背景，“Z Order”直接设为0。

2. 添加材质和贴图

为背景添加图片之前，首先需要新建一个Material（材质）用作背景贴图（Texture）的载体，单击3D Layout左侧工具面板中的Create Material按钮，如图5.10所示。

将其重命名为“开始界面 背景”。

回到2D Frame Setup面板中，为背景2D Frame指定Material，如图5.11所示。

图5.10 Create Material

图5.11 为2D Frame指定材质

此时整个画面变为了灰色，这是由于材质的Diffuse通道为灰色的缘故，需添加贴图后再进行调整。

找到贴图文件CMOs\Plane\background.jpg，将其放入资料库的Textures文件夹内，然后在Virtools中将其拖曳至3D Layout中的背景图上，为材质添加贴图，如图5.12和图5.13所示。

图5.12 background.jpg

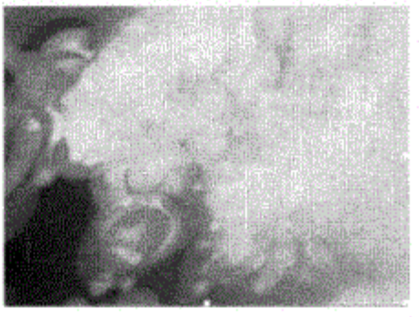


图5.13 2D背景效果

灰蒙蒙的背景色是由于材质的Diffuse通道导致的，如图5.14所示。将其调为纯白色即可恢复画面的光亮，调整颜色后的背景图像如图5.15所示。

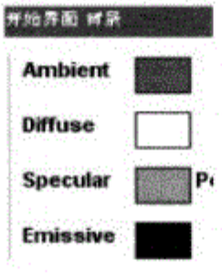


图5.14 将Diffuse颜色通道调为纯白色

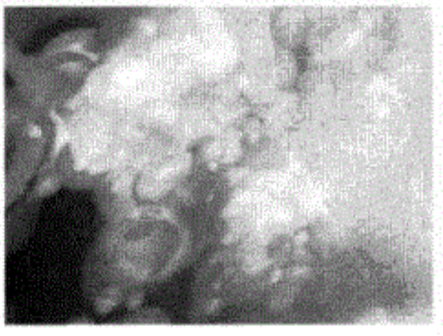


图5.15 调整颜色后的背景图像

最后，不要忘记为2D Frame背景设置初始状态（IC）。

5.1.3 添加 start game 按钮

下面添加 start game 按钮并实现开始游戏的逻辑。

1. 创建2D Frame

首先新建一个2D Frame，重命名为“按钮 开始”，但新建的2D Frame并没有显示在画面中，这是由于该2D Frame所处层级 Z Order 与背景一样，均为0，背景图案被盖住的缘故。更改 Z Order 为1使其位于背景图层的上方，如图5.16和图5.17所示。

图5.16 设置 Z Order 为1

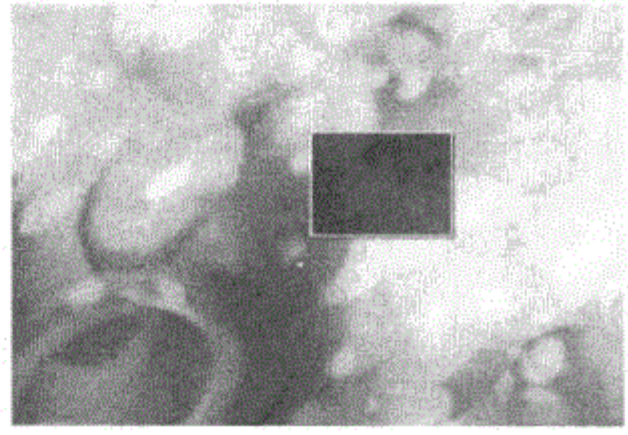


图5.17 调整 Z Order 后2D Frame移动到画面前方

2D Frame正常显示后，还需要重复此前制作背景图片的步骤，使其坐标系统和游戏画面相绑定，如图5.18所示。

图5.18 选择 Homogeneous Coordinates 选项

2. 创建按钮所需材质

接下来制作一个拥有“正常”和“鼠标滑过”两个状态的简单按钮，首先需要为按钮创建两个材质，分别用于“正常”状态和“鼠标滑过”状态。

首先使用 Create Material 创建两个新的材质，分别命名为“按钮 开始 正常”和“按钮 开始 Over”，将 CMOs\Plane\目录下的 start game. png 指定为这两个 Material 的贴图，然后可以将材质赋给按钮2D Frame来检测一下效果，如图5.19所示。

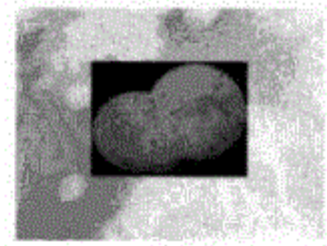


图5.19 添加贴图后的2D Frame

如图5.20所示，“按钮 开始 正常”的周围有黑色，这是由于PNG贴图的透明通道（Alpha通道）没有正常显示的结果，只需将 Material 的 Mode（贴图模式）由默认

的 Opaque 改为 Transparent 即可。调整 Diffuse 通道颜色为纯白色，效果如图 5.21 所示。

下面用同样的方法测试并调整“按钮 开始 Over”材质，为了让鼠标滑过之后的材质显得有所不同，可以调整 Diffuse 通道的颜色，如图 5.22 所示。

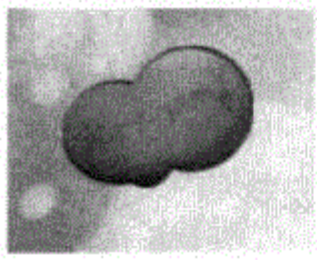


图 5.20 材质模式调整为 Transparent 的效果

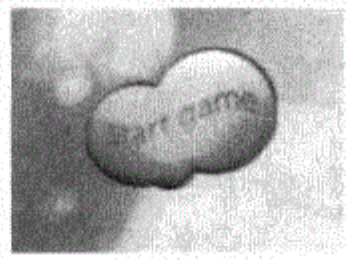


图 5.21 修正 Diffuse 通道后的效果

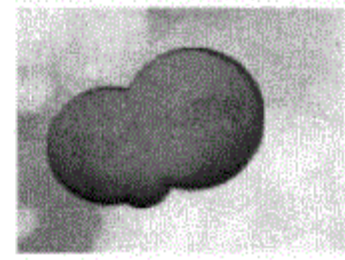


图 5.22 “按钮 开始 Over”材质调校后效果

3. 添加脚本实现按钮效果

在添加脚本之前，首先需要调整好按钮的尺寸和位置，如图 5.23 所示。

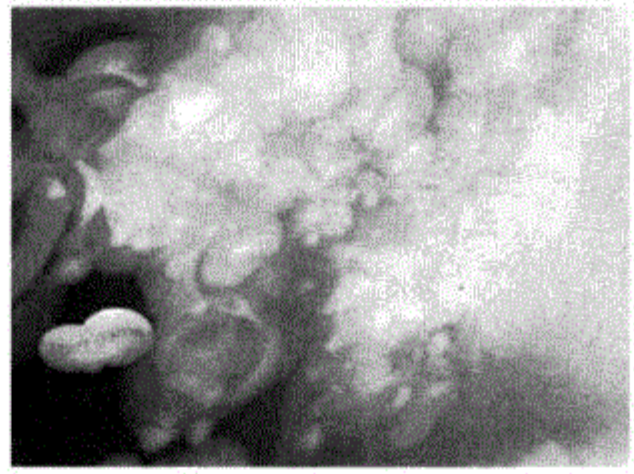


图 5.23 调整按钮的位置和尺寸

然后将“按钮 开始”2D Frame 的材质指定为 None，按钮的材质将由脚本动态指定。

为“按钮 开始”新建脚本，并选取 Interface | Controls | PushButton 选项，添加 BB，如图 5.24 所示。

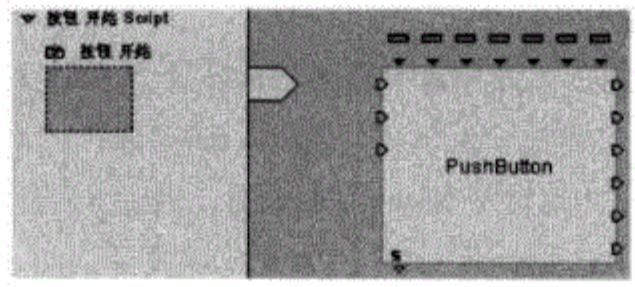


图 5.24 添加 PushButton BB

首先右键单击 PushButton BB 并选择 Edit Settings，进入 BB 设置面板，在这里可以设定对哪些鼠标事件进行监控，如图 5.25 所示。

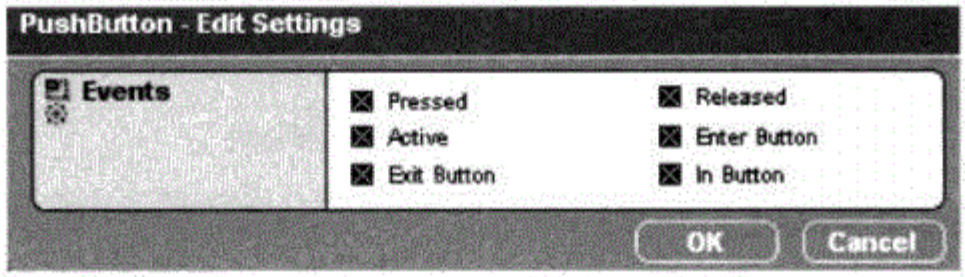


图 5.25 PushButton 设置面板

由于只需要监控用户单击鼠标（Pressed）的动作，因此关闭其他所有的事件监控选项，如图 5.26 所示。

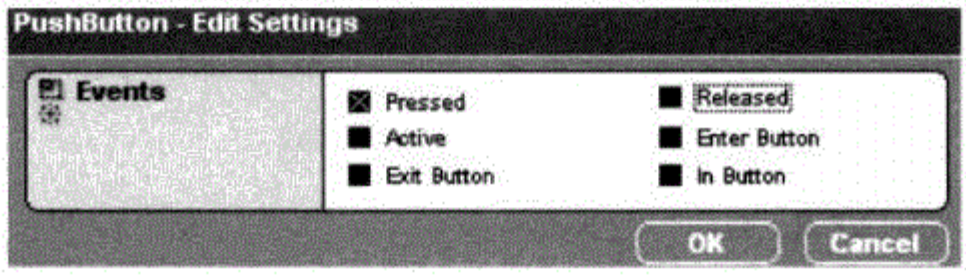


图 5.26 只选择 Pressed 事件

单击 OK 按钮后可以看到 BB 的输出端只剩下了 Pressed 一个，如图 5.27 所示，当用户单击该按钮的时候，便会激活该输出端。

双击 PushButton BB 进入参数设置面板，在这里只需关注前三个选项，如图 5.28 所示。

Released Material 为鼠标松开时的按钮材质，即正常状态下按钮的材质。

Pressed Material 为鼠标单击时的按钮材质。

RollOver Material 为鼠标滑过按钮后的按钮材质。

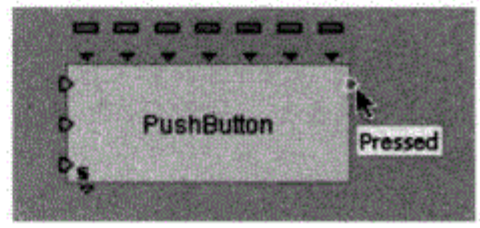


图 5.27 PushButton 更改设置后

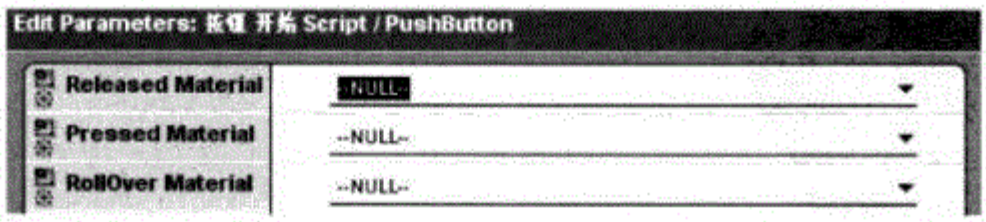


图 5.28 PushButton 参数设置面板

根据之前的设计，分别指定按钮材质如图 5.29 所示。

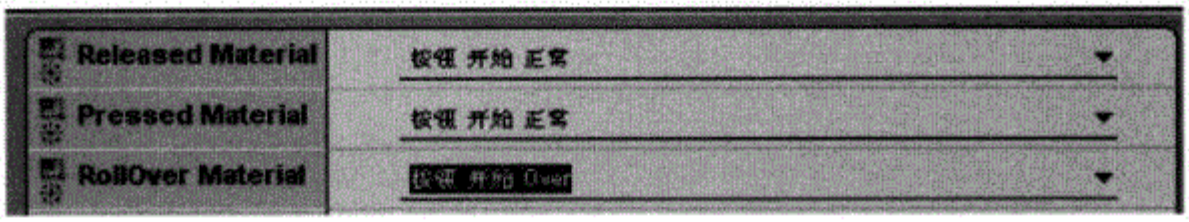


图 5.29　为 PushButton 指定按钮材质

单击 OK 按钮完成参数设置并将 PushButton BB 的 On 输入端和脚本起始端相连接，如图 5.30 所示。

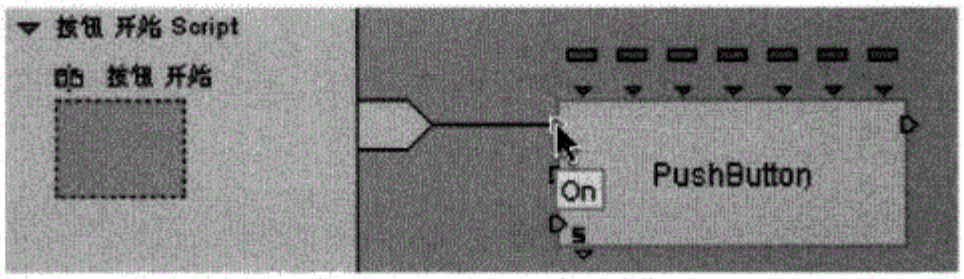


图 5.30　完成 PushButton 连线

测试实际运行效果，如图 5.31 和图 5.32 所示。

图 5.31　鼠标未移动到按钮上方

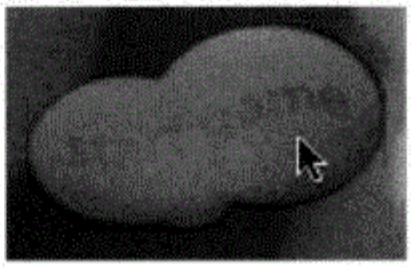


图 5.32　鼠标移动到按钮上方

最后不要忘记设置初始状态（IC）。

4. 实现单击按钮开始游戏功能

接下来添加脚本实现“单击按钮开始游戏”的交互程序。在单击按钮后，首先应该开启两个 Level Scripts，创建动态的山峰和陨石，如图 5.33 所示。

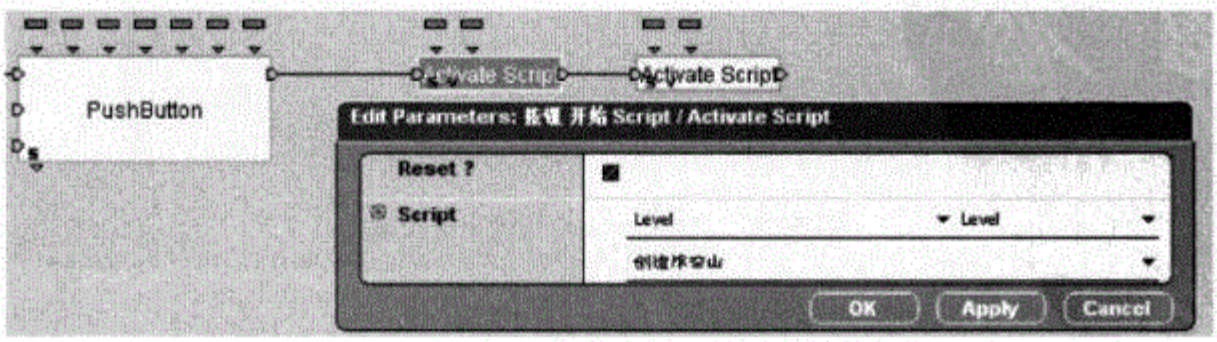


图 5.33　开启 Level Scripts

随后激活飞机的机身、螺旋桨以及 Ground 物体，如图 5.34 和图 5.35 所示。

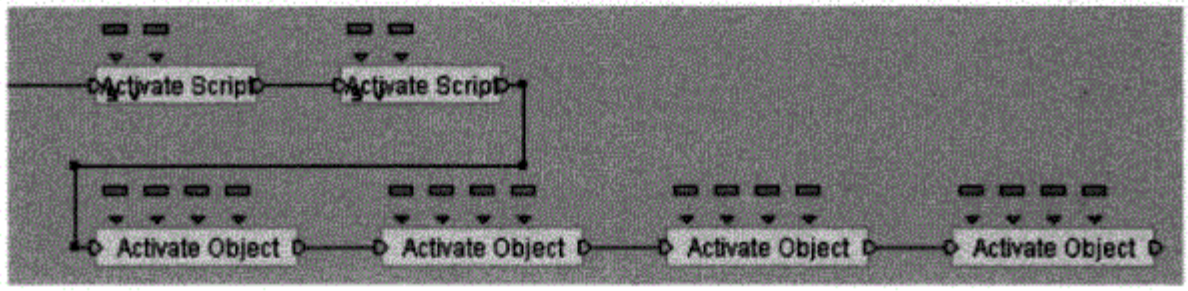


图 5.34　激活飞机等物体

图 5.35　Activate Object BB 输入参数设置

最后，隐藏背景 2D Frame 以及开始按钮，并关闭 Deactivate 开始按钮（使之不起作用），如图 5.36 所示。

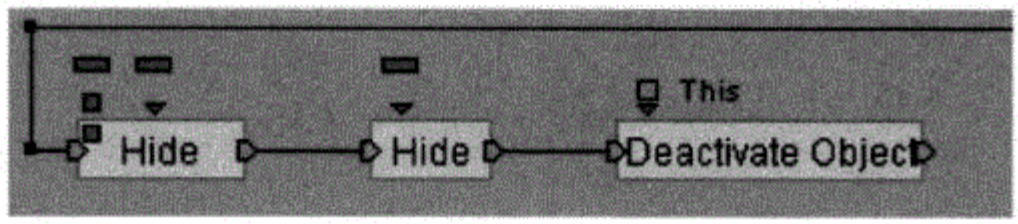


图 5.36　隐藏背景并关闭按钮

最终脚本如图 5.37 所示。

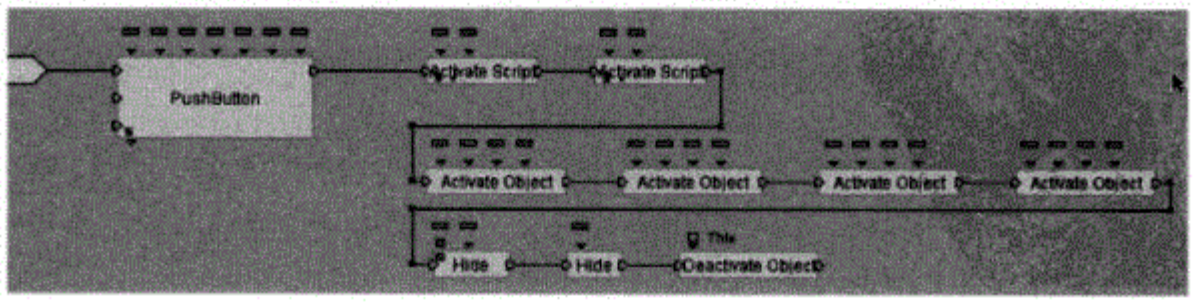


图 5.37　最终开始界面互动脚本

运行测试一下，可以看到按钮已经具备开始游戏的功能，实现了使用二维游戏界面控制游戏开始的简单逻辑。到此为止的工程文件可以在 CMOs\Plane\Plane_11. cmo 找到。

5.1.4 添加 about 按钮

1. 添加按钮

about 按钮和 start game 按钮在制作步骤上完全相同，故在此不再赘述，仅列出步骤清单如下：

（1）新建2D Frame（命名为“按钮 关于”）；

（2）调整2D Frame 属性并调整大小及位置；

（3）创建按钮所需的两个材质（这里使用 CMOs\Plane\about. png）；

（4）测试并调整材质；

（5）添加 PushButton 脚本实现按钮效果，如图 5.38 和图 5.39 所示。

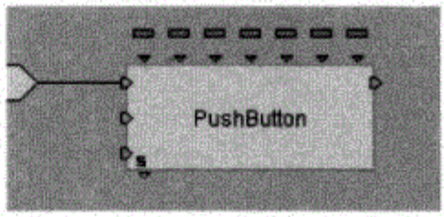


图 5.38 添加 PushButton BB

最终应该得到如图 5.40 和图 5.41 所示的效果。

图 5.39 PushButton BB 输入参数设置

图 5.40 about 按钮 2D Frame 正常态

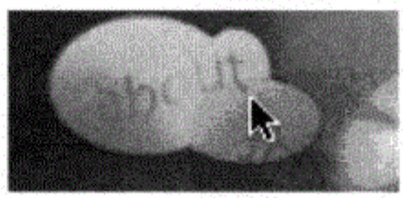


图 5.41 about 按钮 2D Frame Over 态

2. 添加 about 内容面修正板

下面新建一个2D Frame 以及相应材质，为其指定贴图为 CMOs | Plane | about content. png，如图 5.42 所示。

注意这里 Z Order 设置为 2，使该面板可以显示于两个按钮之上，如图 5.43 所示。

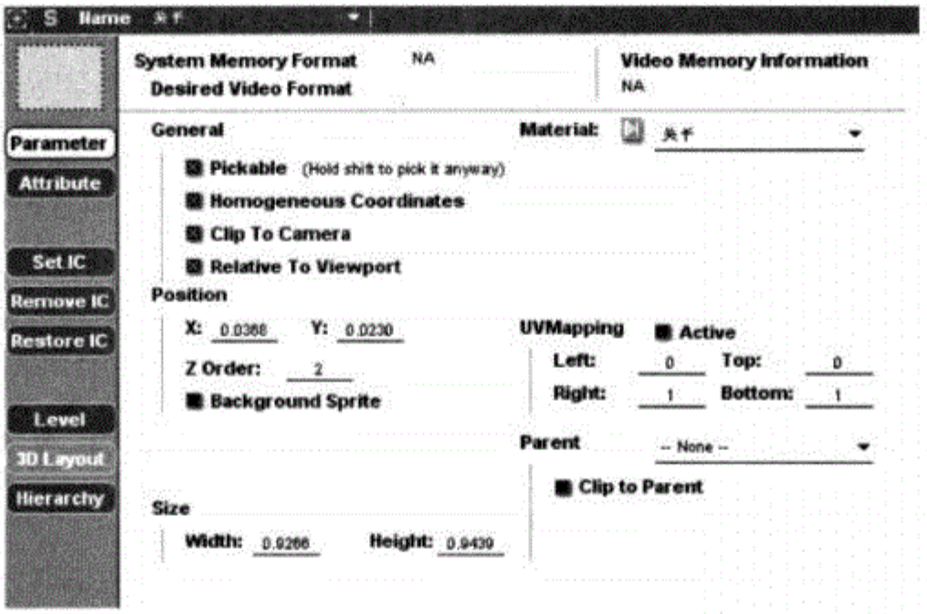


图 5.42　新建“关于”2D Frame

图 5.43　“关于”2D Frame 位于所有其他 2D Frame 之上

然后将其隐藏，并设置为初始状态，如图 5.44 所示。

图 5.44　隐藏“关于”2D Frame

3. *添加互动脚本*

现在添加互动脚本实现“单击 about 后出现内容面板，单击内容面板关闭”的效果。首先，修改 about 按钮的脚本如图 5.45 和图 5.46 所示。

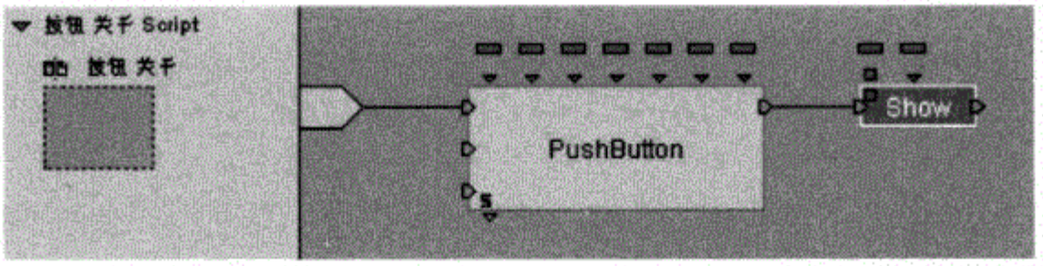


图 5.45 修改后的 about 按钮脚本

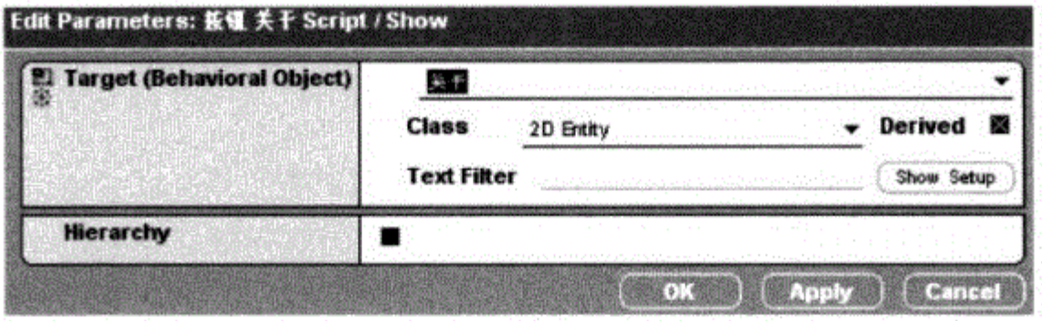


图 5.46 Show BB 输入参数设置

测试可以看到单击后显示“关于”2D Frame 的效果。

然后，为“关于”按钮添加脚本。选取 Controllers |Mouse |Mouse Waiter 选项，添加 Mouse Waiter 脚本模块，如图 5.47 所示。

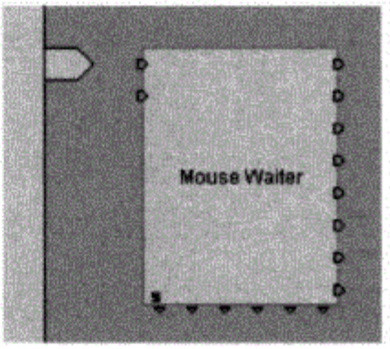


图 5.47 添加 Mouse Waiter BB

右键单击该 BB 并选择 Edit Settings，如图 5.48 所示进行修改设置，使得该 BB 只监听左键按下（Left Button Down）的鼠标事件。

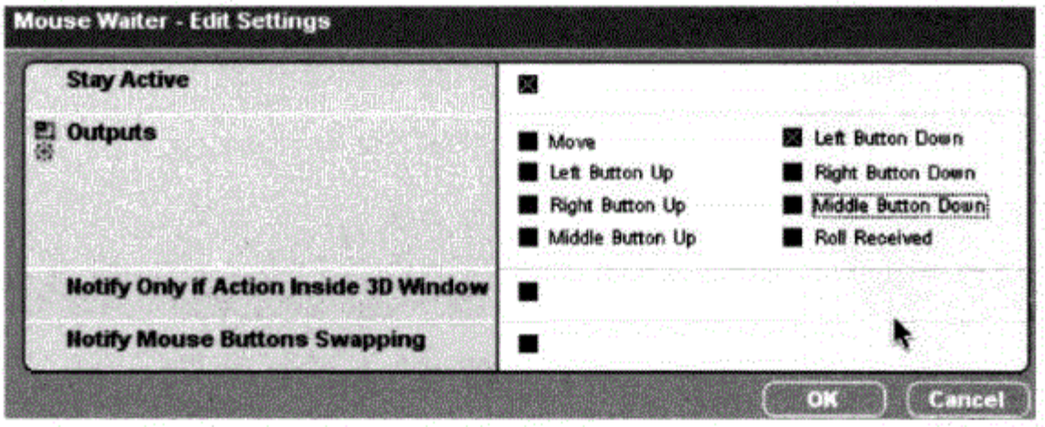


图 5.48 Mouse Waiter 监听 Left Button Down 事件

最终完成脚本如图 5.49 所示，当鼠标左键按下后，执行 Hide 隐藏自己。

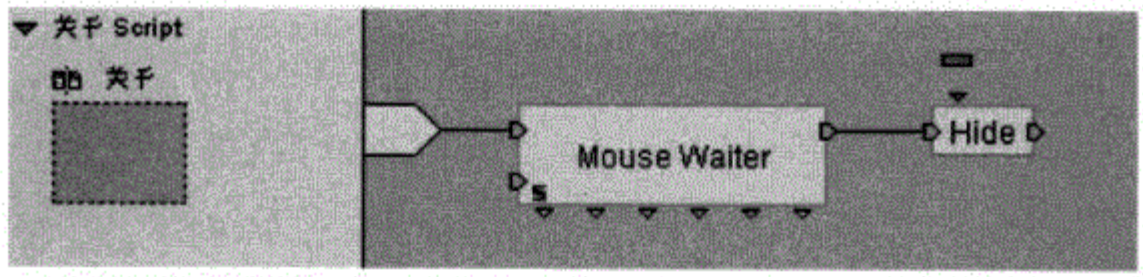


图 5.49 “关于”脚本

4. 修正 Bug

进行测试会发现，当鼠标按下 about 按钮的时候，并没有出现面板，在此需要借助 Trace 和 Breakpoint 进行查错。在“按钮 关于”的脚本上添加 Breakpoint，如图 5.50所示。

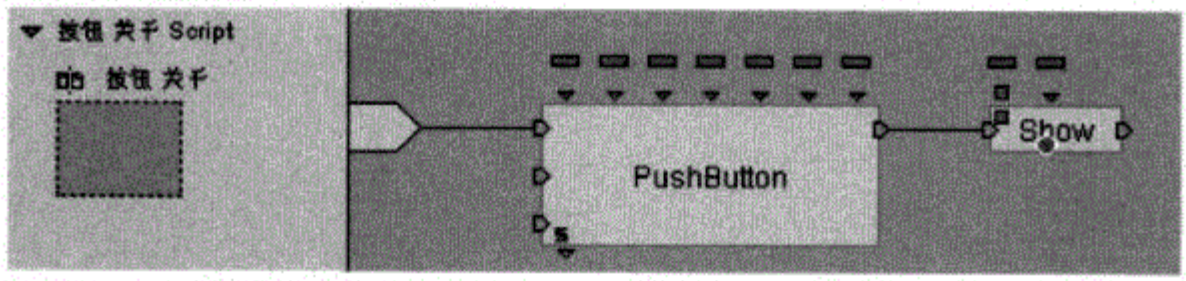


图 5.50 为 Show BB 添加 Breakpoint

然后在脚本栏上方的控制面板中打开 Trace 按钮，如图 5.51 所示。

图 5.51 开启 Trace Mode

这样可以监控单击 about 按钮时候所发生的事情，运行测试并单击 about 按钮。此时弹出 Breakpoint 对话框，所有脚本暂停，可以查看一下此时脚本的运行情况，如图 5.52 所示。

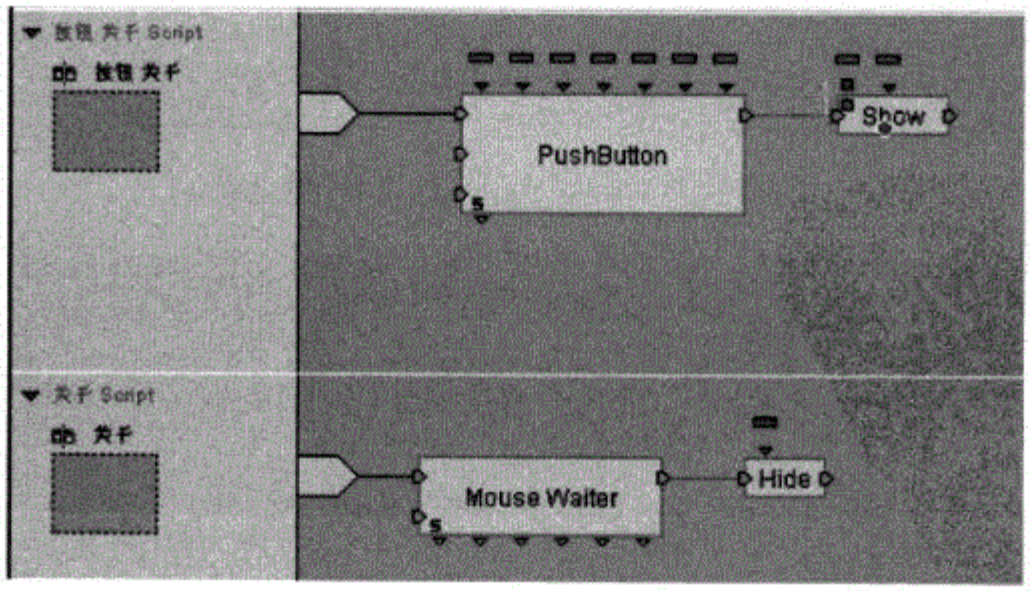


图 5.52 测试 Trace 结果

此时“关于 Script”中的 Hide BB 也被激活了，这是由于用户单击 about 按钮后，“关于”面板弹出，但此时它的脚本中 Mouse Waiter 也由于用户的鼠标左键为按下状态 Left Button Down，而激活输出端从而触发 Hide BB，并立刻隐藏了面板。为了解决这个 Bug，可以延迟脚本的执行，在此之前需要将“关于 Script”关闭，以方便以后用脚本动态控制它的激活状态，如图 5.53 所示。

图 5.53 关闭“关于 Script”

修改脚本如图 5.54 所示。

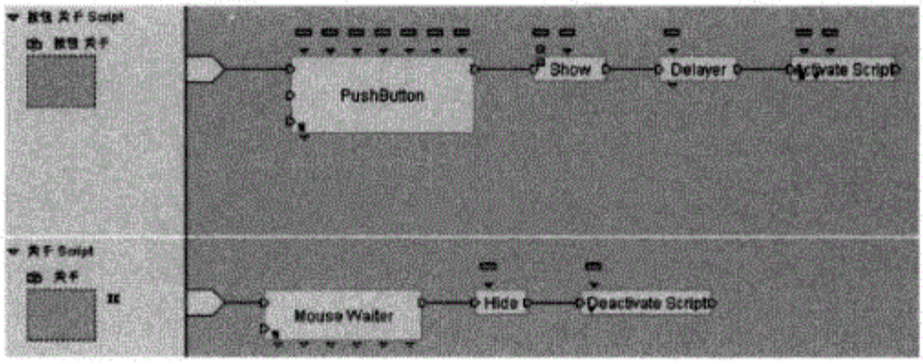


图 5.54 修改过的脚本

新增加的 Activate Script 和 Deactivate Script 都是针对“关于 Script”的，如图 5.55 所示，而 Delayer 是用来控制延迟激活脚本，在这里设置为 300ms（见图 5.56）。

图 5.55 Deactivate Script BB 输入参数设置

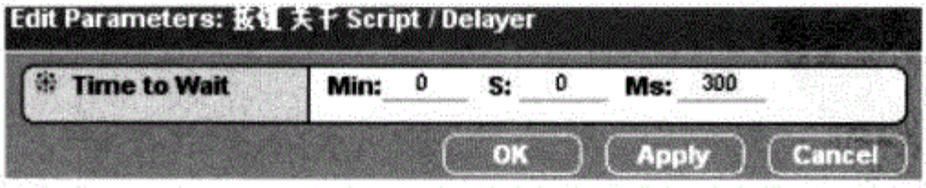


图 5.56 Delayer BB 输入参数设置

测试后即可看到 Bug 已经被解决，about 按钮可以正常运行。

但是，当单击 Start 进入游戏后，about 按钮仍然显示在画面中，此时需要修改

Start 按钮的脚本来解决这个 Bug，如图 5.57 和图 5.58 所示。

图 5.57 修改过的 Start 按钮脚本

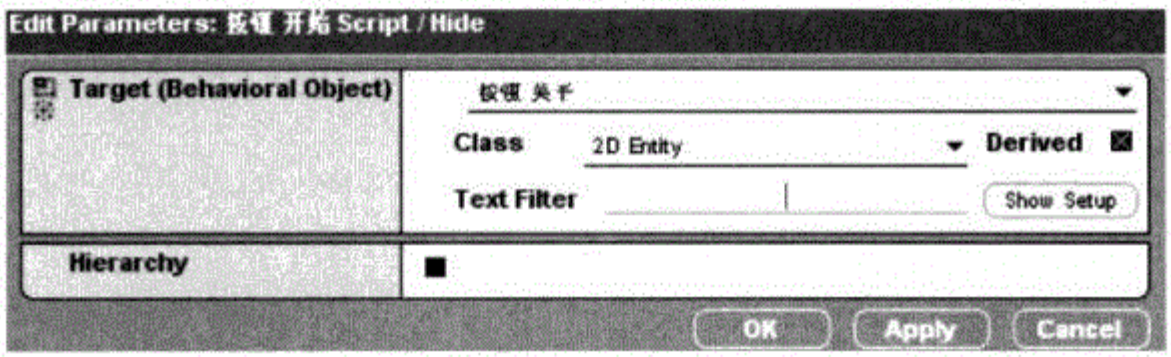


图 5.58 Hide BB 输入参数设置

至此为止游戏内的 Bug 已经全部排除，可以在 CMOs\Plane\Plane_12. cmo 找到工程文件。

5.2 游戏内部界面制作

在上一节中，已为游戏原型制作了简单的开始界面，但是在游戏进行中还缺乏必要的图形界面元素来提供游戏信息，如“剩余生命数量”，在本节中将完成这些内部界面的制作。

5.2.1 制作前的准备工作

为了方便在游戏画面（而非游戏开始界面）内码放原件，可以暂时将开始界面的 2D Frame 全部都隐藏，如图 5.59 所示。

图 5.59 隐藏所有开始界面 2D Frame

此时不必设置 IC，这样在进行更改后还可以方便地回到初始状态。

5.2.2 添加显示“剩余生命数量”的2D Frame

首先，新建一个2D Frame，将其重命名为“剩余生命数量”，并码放于游戏画面的左上角，如图5.60所示。

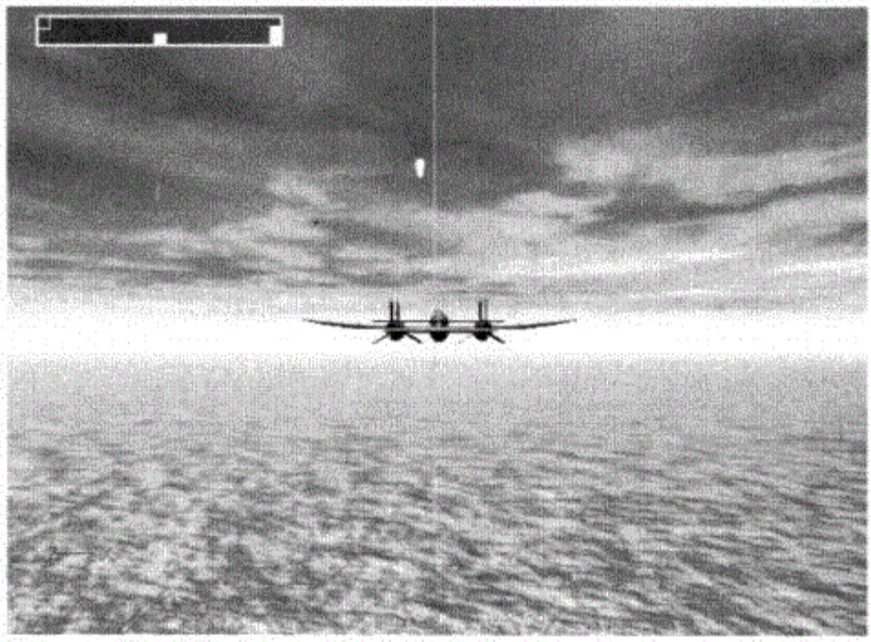


图5.60 将2D Frame码放于画面左上角

注意选择Homogeneous Coordinates选项并设置IC。接下来将使用脚本动态生成二维文字内容并显示于画面之上，因此无须为其创建材质。

5.2.3 添加动态显示文字的互动脚本

在完成动态显示飞机生命数量的脚本之前，先做测试，让刚刚添加的2D Frame可以显示动态文字。

1. 动态创建系统字体

在游戏中，文字并不能直接使用系统提供的字体，而需要使用游戏自带的字体贴图（Texture）。Virtools提供了方便的使用系统字体的方法，即将系统字体动态生成为游戏内的贴图（Texture）来显示文字。

为“剩余生命数量”2D Frame创建一个脚本，选取并添加Interface | Fonts | Create System Font和Set Font Properties两个脚本模块。完成的连线如图5.61所示。

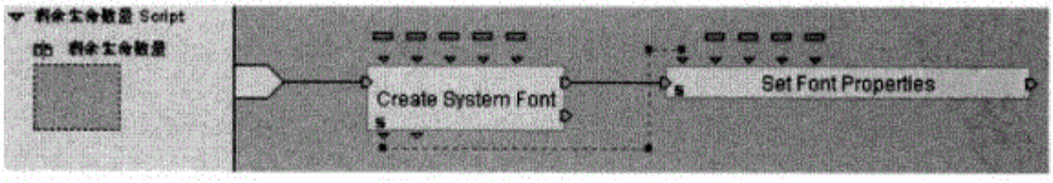


图5.61 创建并设置字体属性脚本

双击 Create System Font BB，进入参数设置面板，如图 5.62 所示。

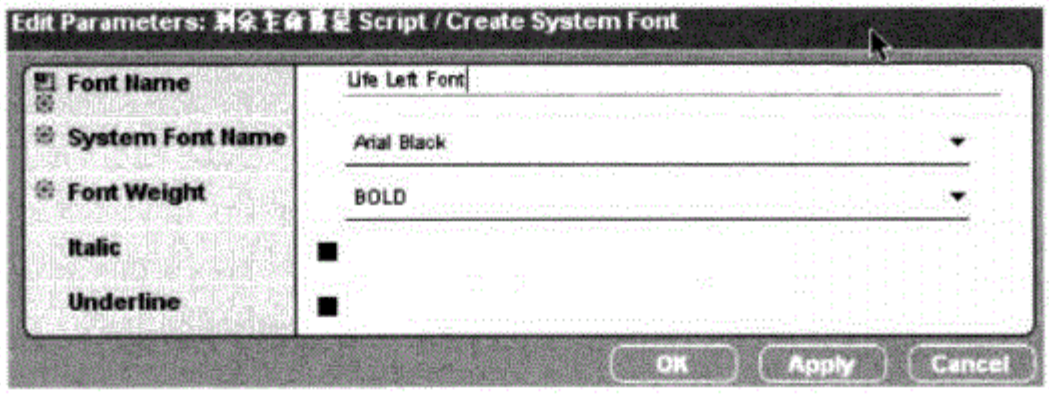


图 5.62 Create System Font BB 输入参数设置

Font Name 为字体名称，即创建的字体名，在这里为了方便记忆将其命名为 Life Left Font。

System Font Name 为系统字体名称，用来创建字体贴图的系统字体名称，在这里选择 Arial Black。

Font Weight 为字体粗细程度，这里选择 BOLD，即粗体。Italic 代表是否为斜体，Underline 表示是否有下划线。

完成修改后单击 OK 按钮。Create System Font 脚本模块创建的字体会被赋给第一个输出参数 Font Created，如图 5.63 所示。

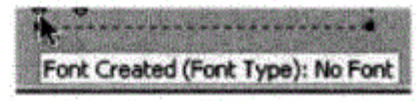


图 5.63 Create System Font BB 输出参数

将其连接至 Set Font Properties 的第一个输入参数 Font，即可对该字体进行设置，双击进入参数设置面板，其中 Space 为字符间隔、Scale 为字体缩放、Italic Offset 为斜体倾斜程度，都不做改动；Color 为字体颜色，将其改成自己喜好的颜色即可。至此完成字体的动态创建以及设置工作，如图 5.64 所示。

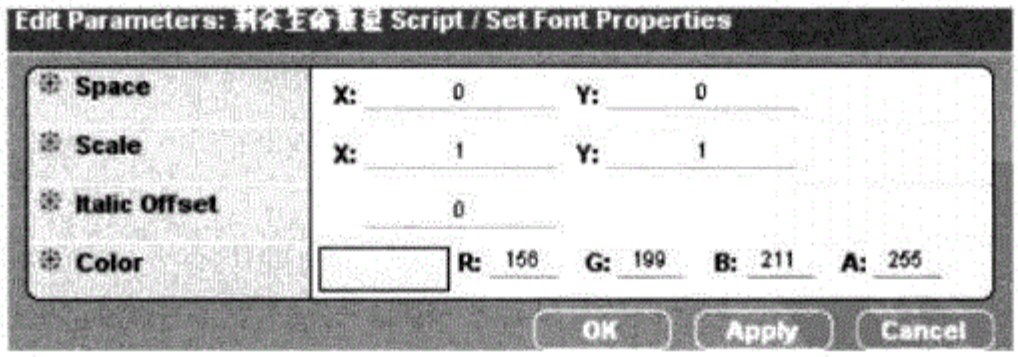


图 5.64 Set Font Properties 输入参数设置

2. 显示文字

设定字体后，便可以使用该字体进行文字的显示。选取 Interface | Text | 2D Text，为脚本添加 BB，如图 5.65 所示。

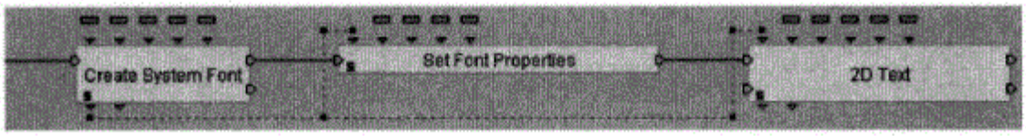


图 5.65　添加 2D Text BB

将 Create System Font 的输出参数 Font 连接至 2D Text 的第一个输入参数 Font，用作显示文字的字体。双击 2D Text BB 进入参数设置面板，如图 5.66 所示。

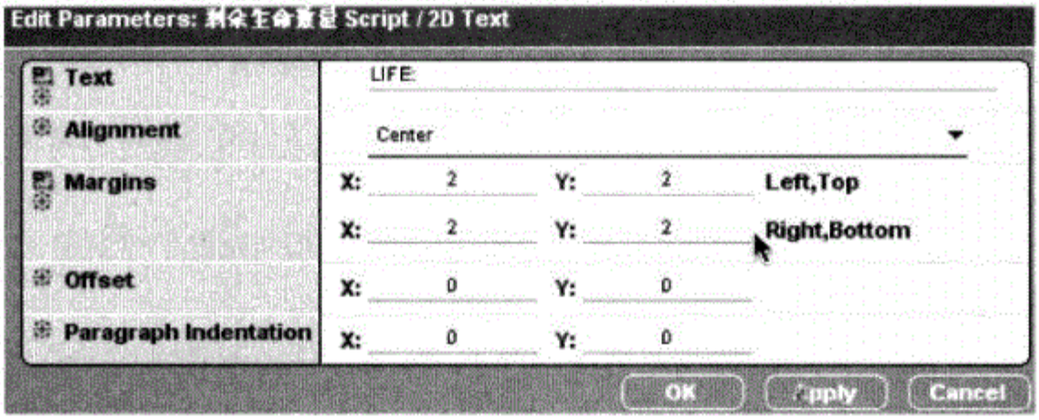


图 5.66　2D Text 输入参数设置

Text 即为显示的文字内容，可以输入任何测试文字（限英文，因为使用英文字体）。将 Alignment 设置为 Center 可以使文字居中显示。完成设置后，进行测试，效果如图 5.67 所示。

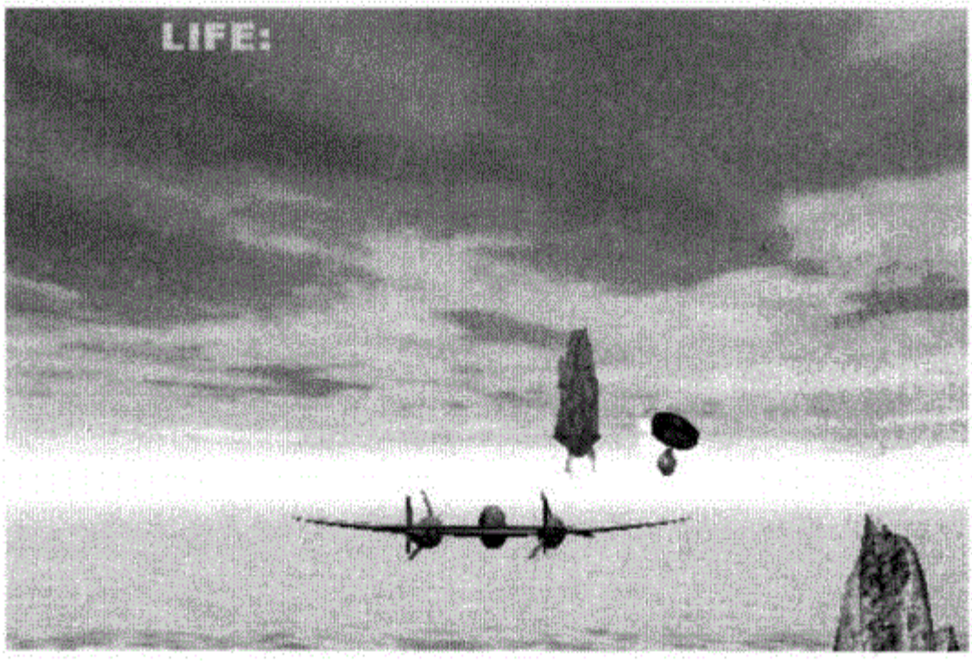


图 5.67　动态显示文字测试结果

5.2.4 动态显示生命剩余数量

若要动态显示生命值，首先需要明确何时更新文字，以及更新的文字内容是什么。本例将使用消息机制来控制文字的更新以及更新的文字内容，即在飞机生命数量改变的时候，通知2D Frame更新文字。

1. 修改“碰撞检测”脚本使其可以发送信息

首先打开“飞机 机身”的“碰撞检测”脚本，它是控制飞机剩余生命数量变化的模块，最适合做消息发送端。

在游戏开始时，需要通知2D Frame显示飞机的起始生命数量。如图5.68和图5.69所示，添加获取飞机生命数量的脚本。

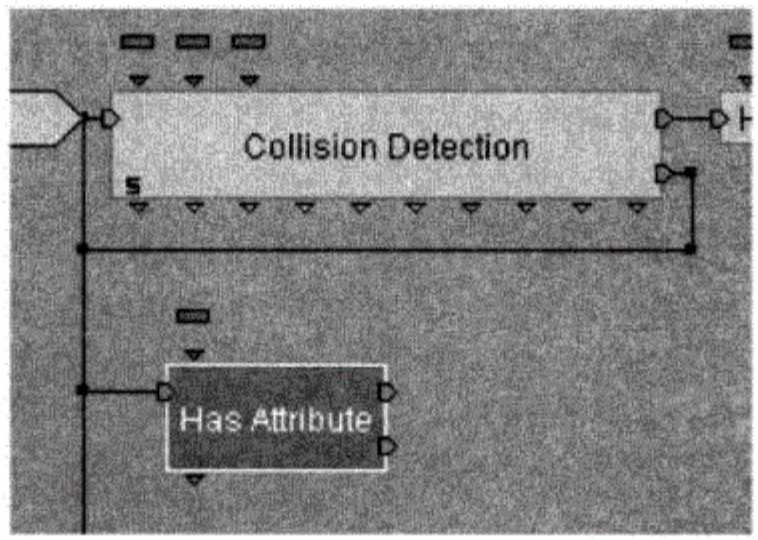


图5.68 添加 Has Attribute BB

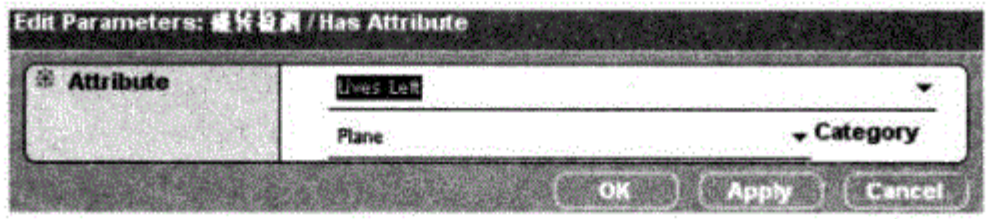


图5.69 Has Attribute BB 输入参数设置

接下来添加Send Message BB，如图5.70所示。

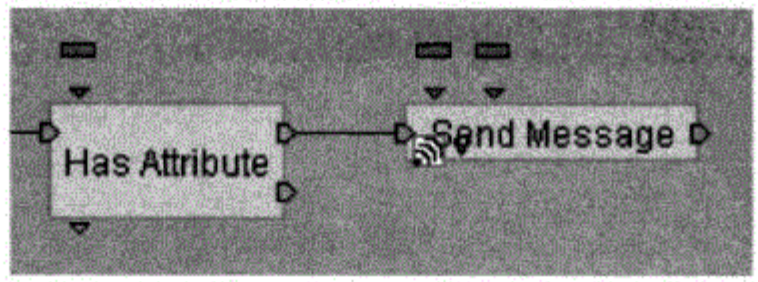


图5.70 添加 Send Message BB

将 Message 手工定义为 UPDATE LIFE 字符串，将 Dest 指定为 2D Frame “剩余生命数量”，如图 5.71 所示。

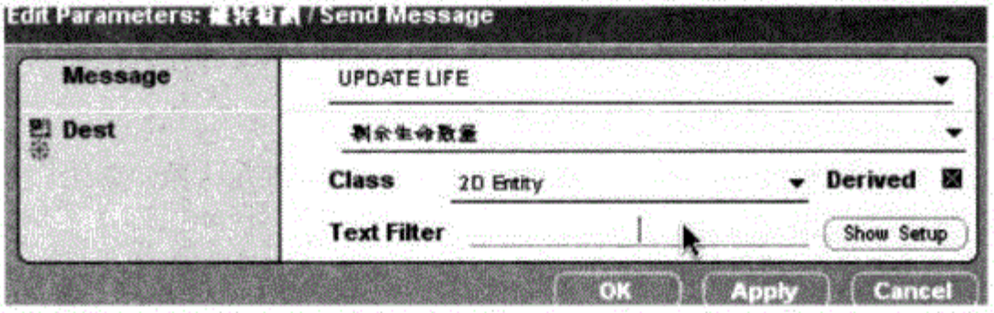


图 5.71 Send Message BB 输入参数设置

下面是至关重要的一步：让发送的消息携带参数。因为仅仅通知 2D Frame 让其更新“生命剩余数量”是不够的，还需要通告剩余的生命数量是多少，因此就需要让这个消息携带着“生命数”，一同发送给 2D Frame。具体方法是为 Send Message BB 增加一个输入参数，单击右键选择 BB 并选取 Construct | Add Parameter Input 选项，如图 5.72 所示。

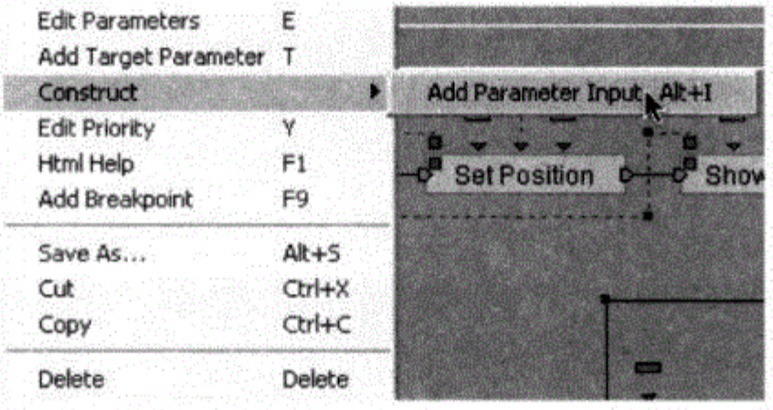


图 5.72 为 Send Message BB 新建输入参数

设置输入参数类型为 Integer（整数），参数名称为 Life Left，如图 5.73 所示。单击 OK 按钮，在 Send Message BB 上方多出了一个输入参数的箭头，如图 5.74 所示。

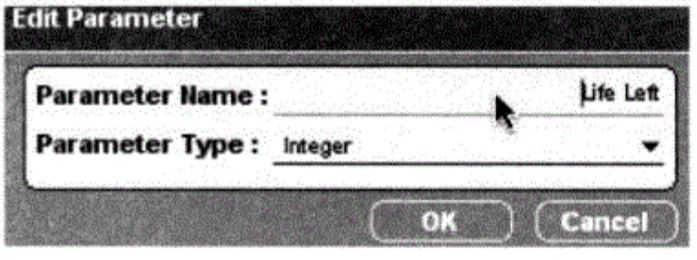


图 5.73 新建输入参数设置面板

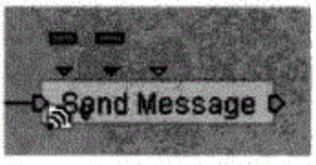


图 5.74 Send Message BB 新建得到的输入参数

将 Has Attribute 的输出参数（剩余生命数量）与之相连，完成的效果如图 5.75 所示。

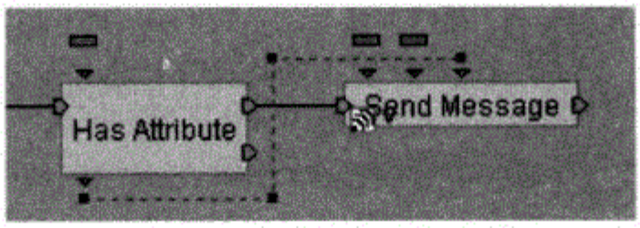


图 5.75 完成连线设置

现在已完成游戏开始时（开始碰撞检测时）发送消息、更新生命剩余数量的脚本，除此之外，在飞机每一次撞上障碍物，减少生命数的时候，也需要更新显示。直接复制刚刚修改过的 Send Message BB（注意连参数一同进行复制），粘贴至 Test 右侧，如图 5.76 所示。

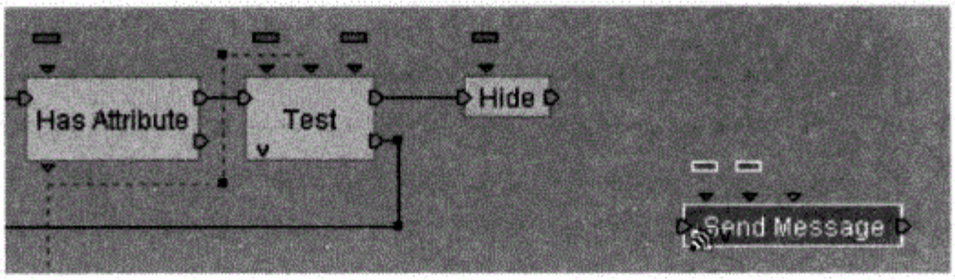


图 5.76 复制 Send Message BB

如图 5.77 所示，进行连线。

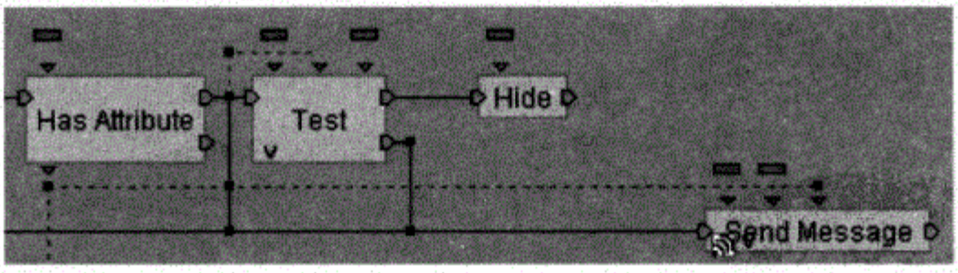


图 5.77 完成连线设置

将 Has Attribute 的输出端与 Send Message 的输入端相连，将输出参数（生命剩余数量）与 Send Message 的第三个输入参数（Life Left）相连。

至此就完成了发送消息的脚本。在游戏开始时以及每一次碰撞发生后，都会发送消息 UPDATE LIFE 至“剩余生命数量”2D Frame 通知其更新显示，并会携带 Life Left 参数，用于更新。

2. 接受消息并显示剩余生命数量

最后还需要为“剩余生命数量”的脚本添加接收信息并更新显示的脚本，具体脚本如图 5.78 所示。

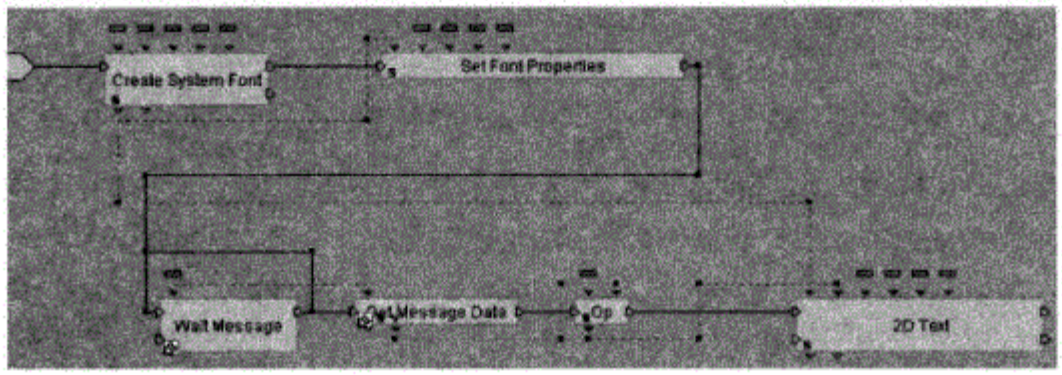


图 5.78　显示剩余生命数量脚本

下面进行逐个解释：

Wait Message 的输入参数设置如图 5.79 所示。

图 5.79　Wait Message BB 输入参数设置

将 Out 输入端与 In 输入端相连，使其自循环，不断监听 UPDATE LIFE 消息。当监听到该消息后，触发 Get Message Data BB（Logics | Message | Get Message Data），该 BB 的作用是获取消息中携带的参数信息。右键选择 Get Message Data BB，并选择 Construct | Add Parameter Output 选项，如图 5.80 所示。

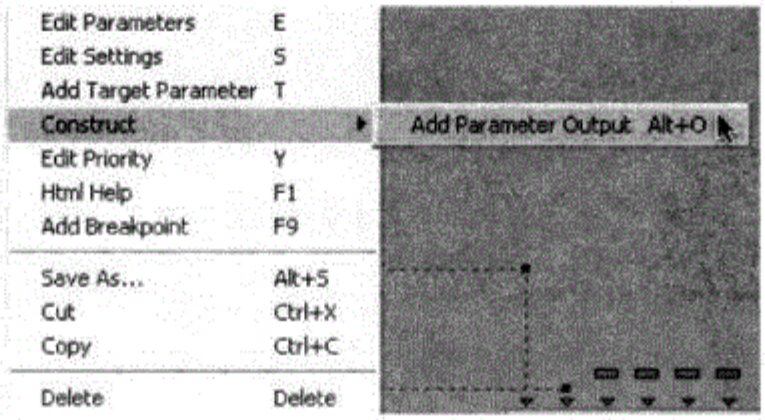


图 5.80　为 Get Message Data BB 添加输出参数

在输出参数设置面板中，指定参数类型为 Integer，参数名为 Life Left（与“碰撞检测”脚本中的 Send Message 脚本相呼应），如图 5.81 和图 5.82 所示。

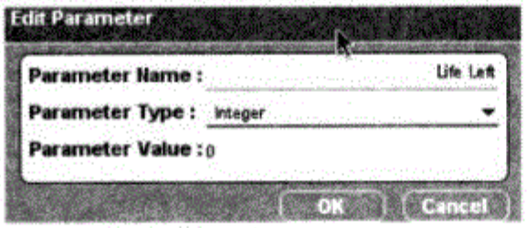


图 5.81 新建输出参数设置面板

图 5.82 Get Message Data 新建得到的输出参数

得到该数值后，需要利用该数值创建需要显示的字符串“LIFE：得到的数值”，使用 Op BB，将运算指定为字符串相加，如图 5.83 所示。

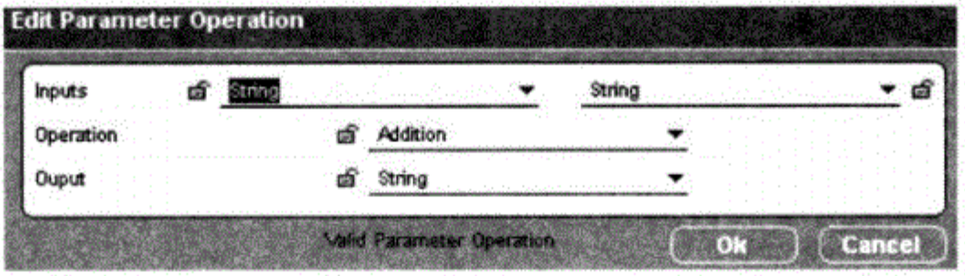


图 5.83 字符串相加 Op

然后直接将 Life Left 输出参数与 Op 第二个输入字符串参数相连，并指定第一个字符串为“LIFE：”，如图 5.84 所示。

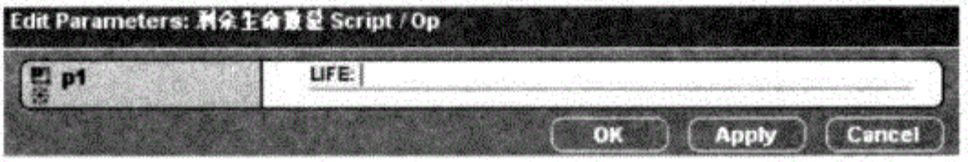


图 5.84 字符串相加 Op 输入参数 p1 设置

此时连线变为了绿色，如图 5.85 所示，这是由于 Integer 类型的数值直接赋给了 String 类型的输入参数，程序自动进行了“变量类型转换”，双击绿色连线可以看到这个转换的运算操作，如图 5.86 所示。

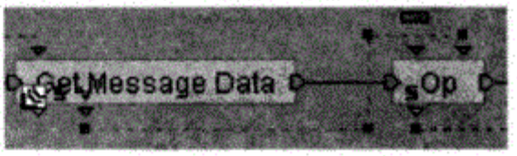


图 5.85 字符串相加 Op 输入参数 p2 连线

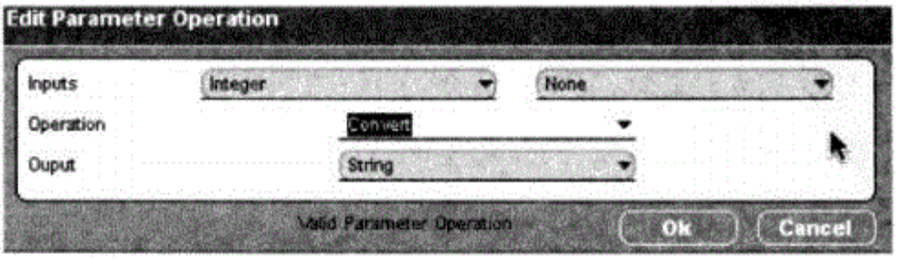


图 5.86 双击绿色连线

最后将 Op 的输出参数，即相加得到的字符串赋给 2D Text 的第二个输入参数 Text，作为文字显示的内容即可。

3. 修正 Bug

现在进行测试。飞机一开始的生命数为 3，此后当发生第一次碰撞后，生命数并没有任何变化，直到发生第二次碰撞，最终飞机三条生命用完的时候，Life 显示为 0，显示的数值始终比飞机实际剩余的生命数大 1 个数，这是由于“碰撞检测”脚本中存在问题，如图 5.87 所示。

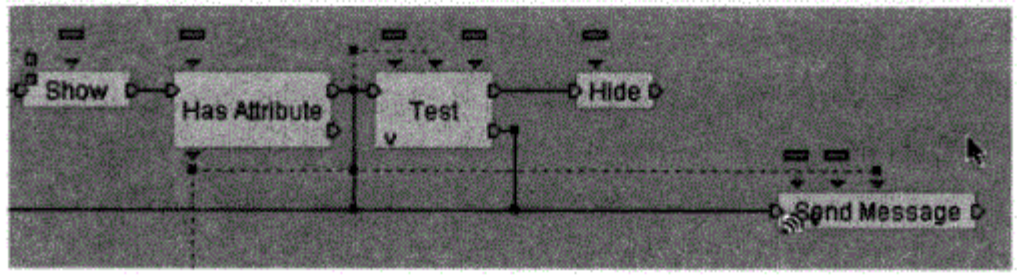


图 5.87　碰撞检测脚本

在飞机发生碰撞的时候，Send Message 在“生命数减 1”这一操作还没有进行的时候就被触发了，因此它显示的并不是当前飞机的剩余生命数，而是始终大 1 个数，对脚本进行修改，如图 5.88 所示。

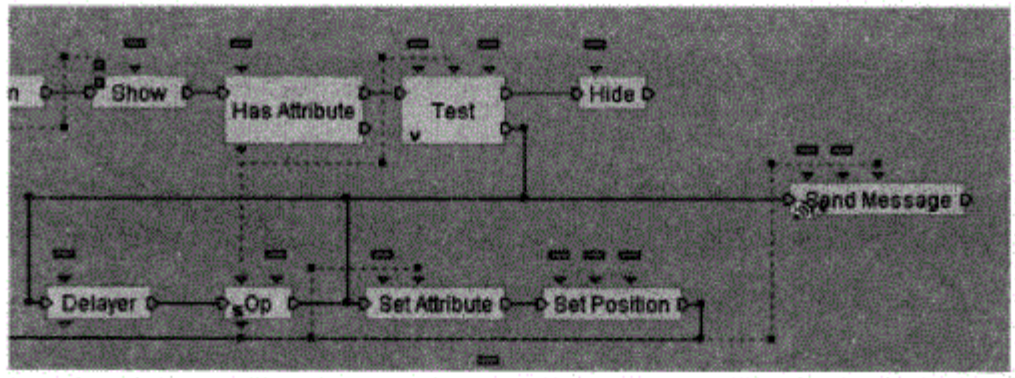


图 5.88　修正后的碰撞检测脚本

将 Op（生命数减 1 运算操作）的输出端与 Send Message 的输入端相连，并将 Op 的运算结果与 Send Message 的 Life Left（第三个输入参数）相连，这样，Send Message 所发送的消息即为更新后的结果。

运行测试后显示正常。到此为止的工程文件可以在如下目录找到：CMOs\Plane\Plane_13. cmo。

5.3　本章回顾

在 5.1 节中，介绍了为游戏原型添加开始界面进行游戏控制的方法，并讲解了 Virtools 中二维原件的使用和互动程序编写，需要掌握的重点是：

（1）创建和设置2D Frame的方法；

（2）2D Frame属性Z Order的含义及使用方法；

（3）使用PushButton创建按钮效果的方法。

在5.2节中，利用消息机制完成了动态显示飞机剩余生命数量的游戏界面制作，本节需要重点掌握的概念有：

（1）Virtools中字体的使用原理；

（2）动态创建系统字体；

（3）显示二维文字；

（4）发送和接受带有参数的消息（Message）。

5.4 课后练习

1. 尝试在游戏原型中，加入游戏结束后返回开始界面的要素。

2. 为游戏加入一个计时器界面原件，使游戏者可以了解当前飞行的时间。

第6章 其他常用制作技巧

由于每一款游戏或交互作品都有不同的制作难点，不可能在一个例子中涵盖所有的制作技巧，因此本章将简要介绍一些游戏制作时的常用技巧。本章介绍的制作技能不仅适用于快速开发游戏原型，也可以满足制作在线的三维交互内容、三维界面设计以及三维虚拟展示设计的需要。

6.1 碰撞检测

在制作飞机游戏原型的时候，接触到了碰撞检测方面的应用。在该游戏中，通过检测飞机是否与障碍物（浮空山、陨石）碰撞来判断飞机是否爆炸，以此类推，可以利用该功能制作攻击命中、子弹命中等其他交互应用。但是，除此之外，在游戏中还有另一大类碰撞检测，它们的作用通常是规定哪些物体可以被穿过，哪些物体应该成为障碍物从而阻拦角色的运动。在一个传统的游戏世界中，游戏者操控的角色无法穿过地图的边界，而箱子、房屋、栅栏以及其他非玩家角色（NPC）也都应该是无法穿过的，下面将介绍在 Virtools 中实现碰撞检测的方法。

6.1.1 Layer Slider 模块

1. 当前存在的问题

首先打开工程文件 CMOs\Chapter7\collision. cmo，如图 6.1 所示。

在该工程文件中，已经添加了一些脚本用来实现玩家控制和简单的光影效果。关于玩家控制部分的脚本，可以参考第 3 章“快速原型制作”中的例子，阴影效果请参考第 8 章阴影部分的相关介绍。进行测试时，可以通过 WSAD 按键控制角色 Jane 的移动，此时角色 Jane 可以穿越任何场景中的物体，如图 6.2 所示，而且即使走到地板边缘也不会停止，如图 6.3 所示。

2. 绘制 Grid

使用 Grid 技术可以将角色的活动范围限制在地板之中。首先切换到 Top 顶视图，如图 6.4 所示，在顶视图中可以更方便地对 Grid 进行操作，然后在工具面板中单击 Create Grid 按钮，如图 6.5 所示，新建一个 Grid。

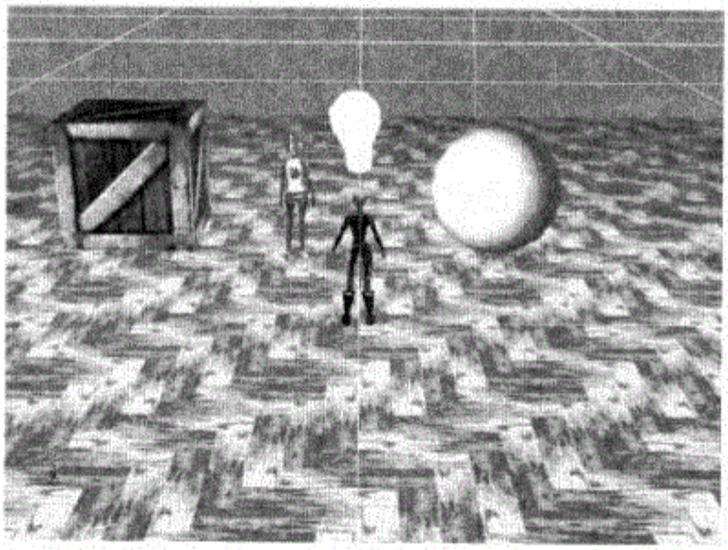


图 6.1　CMOs\Chapter7\collision. cmo 的文件场景

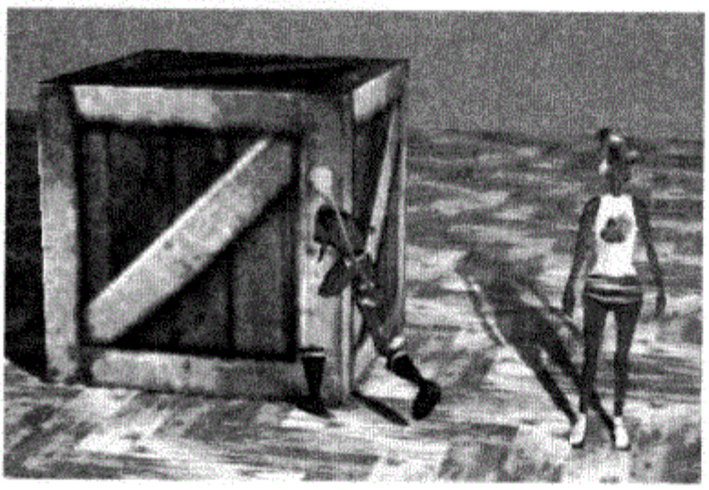


图 6.2　角色可以穿过障碍物

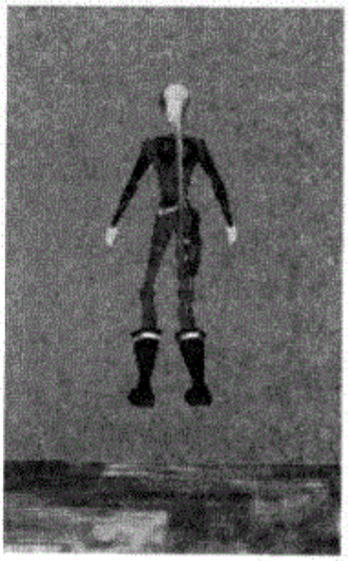


图 6.3　角色会走到地板外面

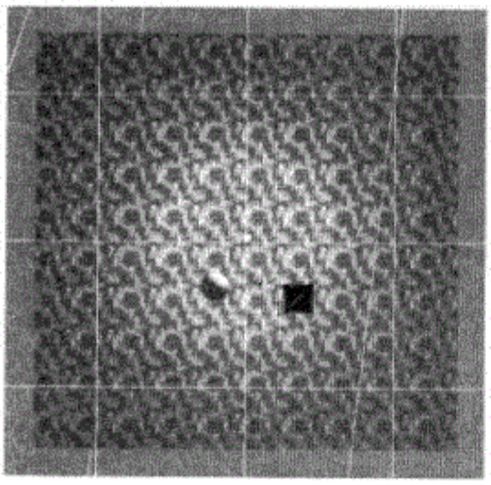


图 6.4　顶视图

图 6.5　新建 Greate Grid 按钮

如图 6.6 和图 6.7 所示，新建得到的 Grid 物体，像是一个矩形的“盖子”，顶面为灰色，而下方的面则统统使用线框来表示，在 Grid 设置面板中，可以看到它是由一个一个的网格构成的，如图 6.8 所示。

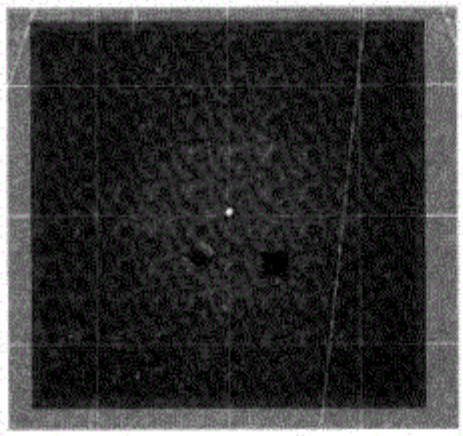


图 6.6　新建得到的 Grid（顶视图）

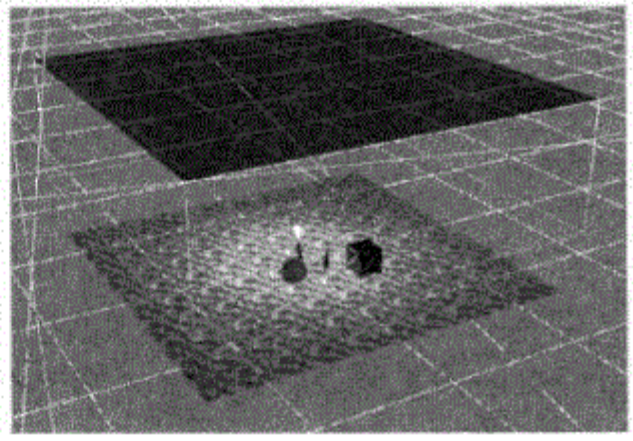


图 6.7　新建得到的 Grid（透视图）

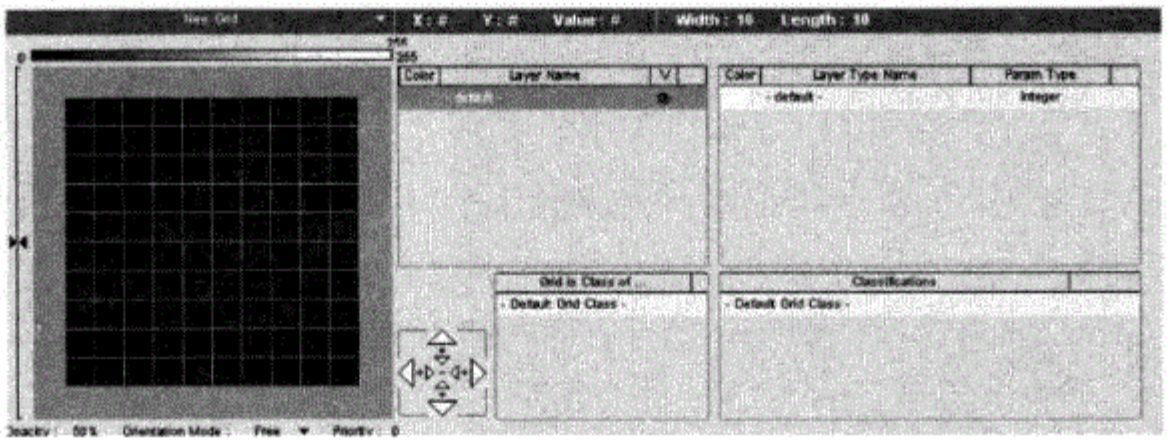


图 6.8　Grid 设置面板

如图 6.9 所示，当使用鼠标在 Grid 设置窗口中移动时，可以在 3D Layout 中相应地显示当前鼠标所处的方格，如图 6.10 所示，方便网格的绘制。

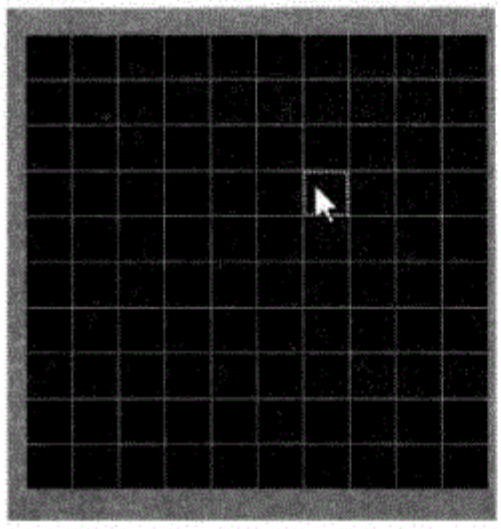


图 6.9　鼠标位于 Grid 设置窗口中

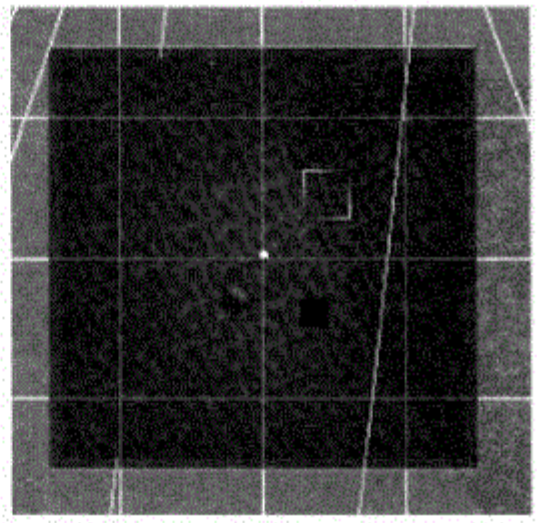


图 6.10　在 3D Layout 中显示出鼠标当前所处的方格

接下来按住鼠标左键，在 Grid 设置窗口中，描画出该网格最外围边缘的格子，可以看到它们变为了黄色（若不慎画到其他地方，可以通过鼠标右键来删除所绘制的格子），如图 6.11 和图 6.12 所示。

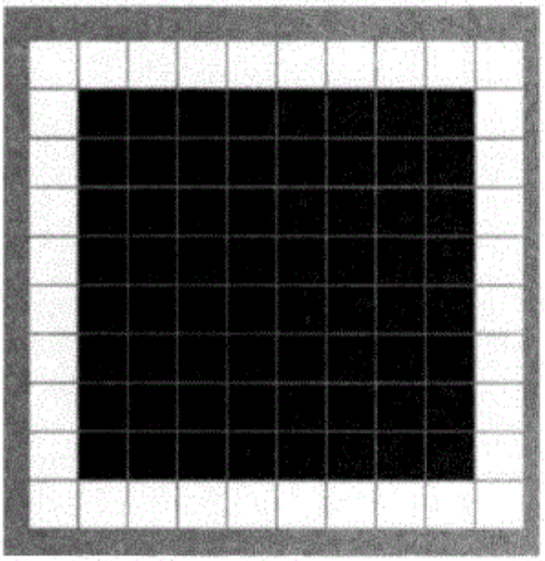


图 6.11　勾画出网格边缘的一排方格

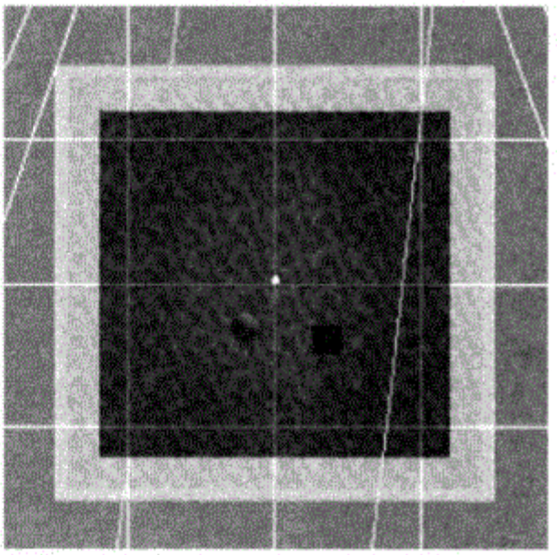


图 6.12　对应的 3D Layout 视图

接下来通过脚本可以使角色无法穿越黄色的方格，从而将角色的活动范围限制在地板之中，但是此前首先需要调整 Grid 的尺寸和位置。在 3D Layout 的顶视图中，利用放大和移动工具，放大 Grid 使黄色方格的内边缘刚好与地板的外边缘对齐，如图 6.13所示。

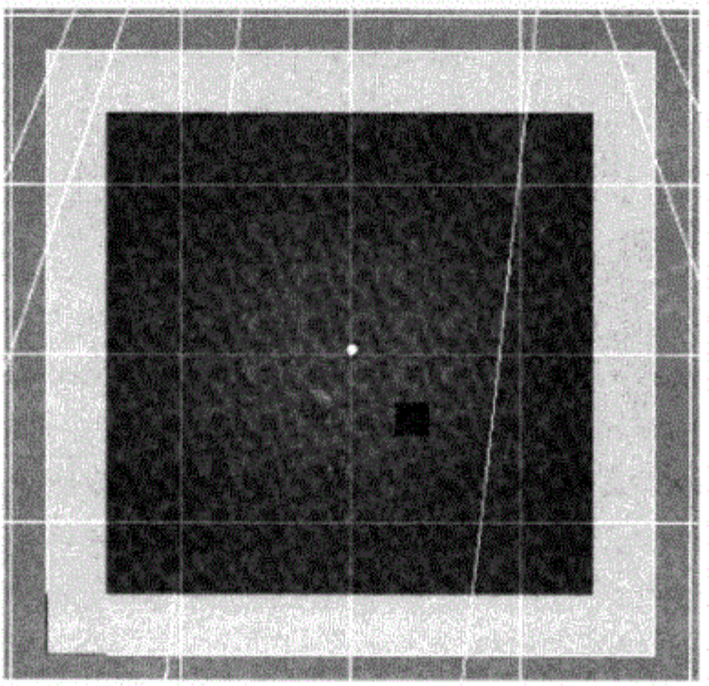


图 6.13　对齐黄色方格内边缘与地板外边缘

然后进入 Front 视角，沿 Y 轴移动 Grid 使它将场景中所有的物体都“包住”（默认情况下它是位于场景物体的上方的），如图 6.14 所示。

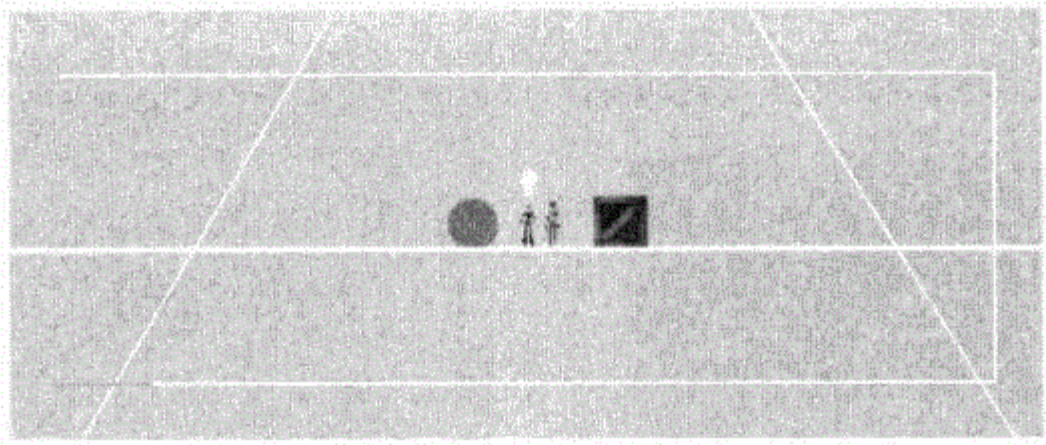


图 6.14　使 Grid 包住场景中的所有物体

最后设置 IC，完成 Grid 的绘制工作。

3. 撰写脚本

接下来将通过脚本使 Grid 真正发挥作用，在 Level Manager 中，为角色 Jane 创建一个新的脚本，并重命名为“场景边缘检测”，如图 6.15 所示。

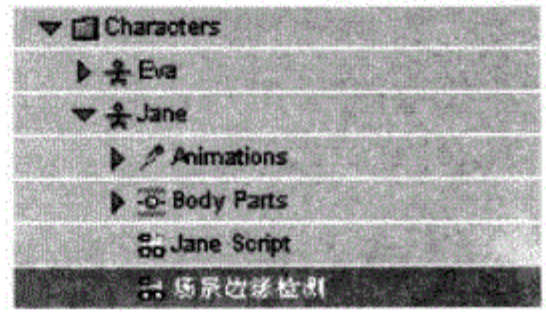


图 6.15　为 Jane 新建“场景边缘检测”脚本

在脚本中，添加 Grids | Basic | Layer Slider 模块，如图 6.16 所示。

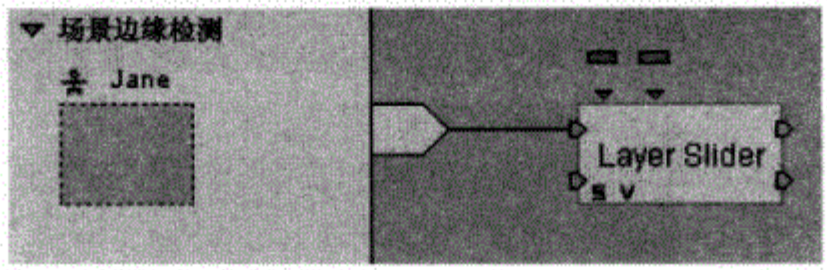


图 6.16　添加 Layer Slider BB

双击进入 Layer Slider 输入参数设置面板。在这里，Influence Radius 是用来控制碰撞检测的半径，稍后再讨论该数值。在此先将 Layer To Slide On 设置为“-default-”即所绘制的黄色方格，该参数用来指定物体无法穿越哪类 Grid 方格，如图 6.17 所示。设置完毕后，运行测试，如图 6.18 和图 6.19 所示，角色 Jane 已经被牢牢地限制在了黄色方格圈定的范围之内，当移动到方格附近时，角色 Jane 无法穿越，只能沿着方格的方向“滑行”，这也正是 Slider 的含义。

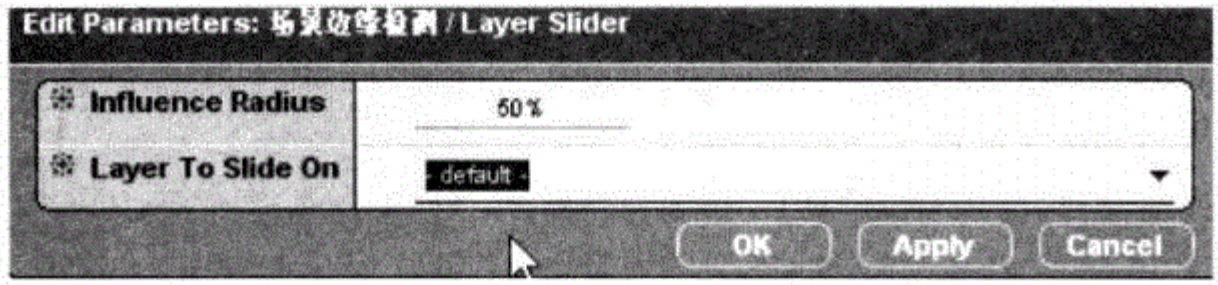


图 6.17　Layer Slider 输入参数设置

图 6.18　Jane 无法穿越黄色方格

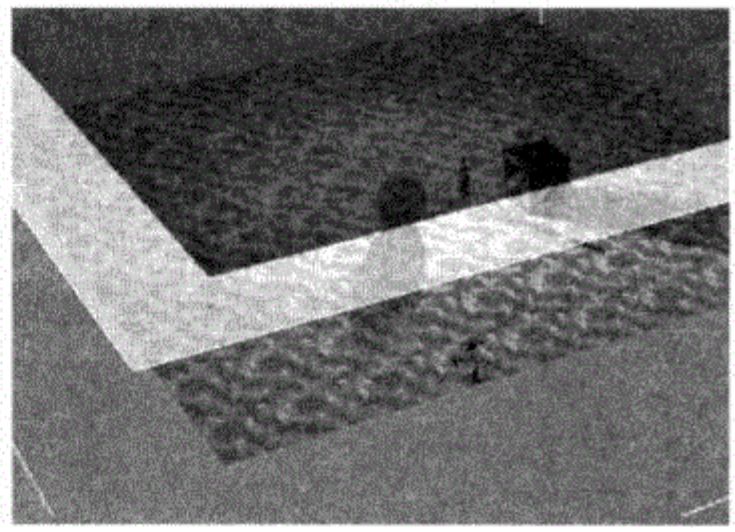


图 6.19　透视视图

最后，可以隐藏 Grid，并设置 IC。即使不显示，Layer Slider 仍然可以正常执行。

Grid 是一种非常方便的标定运动边界的辅助工具，特别适合标定大面积的、比较粗糙的场景边界，例如地图边界，通过细分出更多的方格，房屋树木等不需要复杂碰撞检测的固定物体也可以使用，是一种比较高效的碰撞检测方式。

6.1.2　Prevent Collision 模块

下面使用另一个 Prevent Collision 模块来实现箱子的碰撞检测，使角色 Jane 无法穿过箱子。

1. 编写脚本

为 Jane 新建脚本并重命名为“箱子碰撞检测”，为其添加 Collision | 3D Entity | Prevent Collision 模块，如图 6.20 所示。

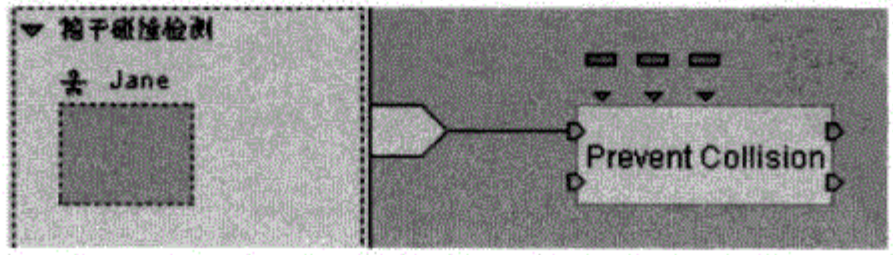


图 6.20　添加 Prevent Collision 模块

Prevent Collision 模块并不会自动循环，当该模块运行的时候，若与障碍物发生碰撞，则激活第一个输出端，并将运行该脚本的物体（这里是 Jane）移动到一个安全的位置上，若没有发生碰撞，则激活第二个输出端。如图 6.21 所示，完成自循环的连线。

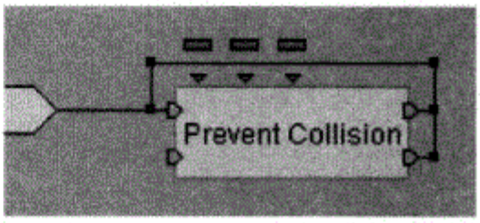


图 6.21 使 Prevent Collision 自循环

2. 设置障碍物属性

设定程序之后，还需要指定场景中哪些物体是障碍物，Prevent Collision 模块才能有针对性的工作，这可以通过为物体添加障碍物属性来实现。

进入箱子的设置面板，单击左侧 Attribute 按钮进入属性设置，为箱子添加 Collision Manager | Fixed Obstacle 属性，如图 6.22 所示。

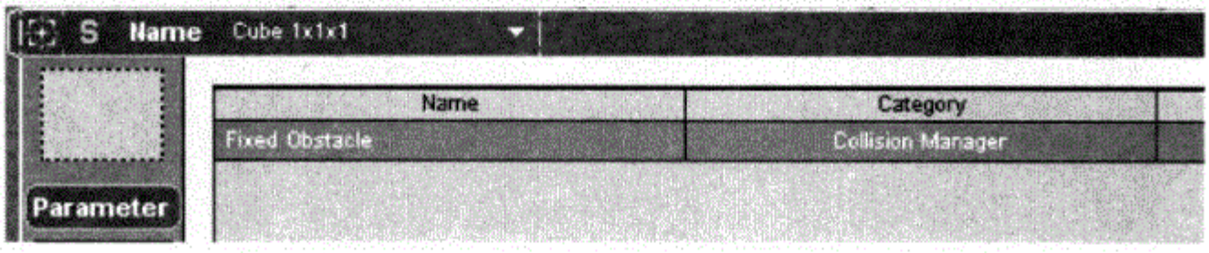


图 6.22 为箱子添加 Fixed Obstacle 属性

该属性表示箱子是一个固定的障碍物（同理，若是移动的障碍物，则需要添加 Moving Obstacle 属性）。运行测试，可以看到角色 Jane 已经无法穿过箱子，如图 6.23 所示。

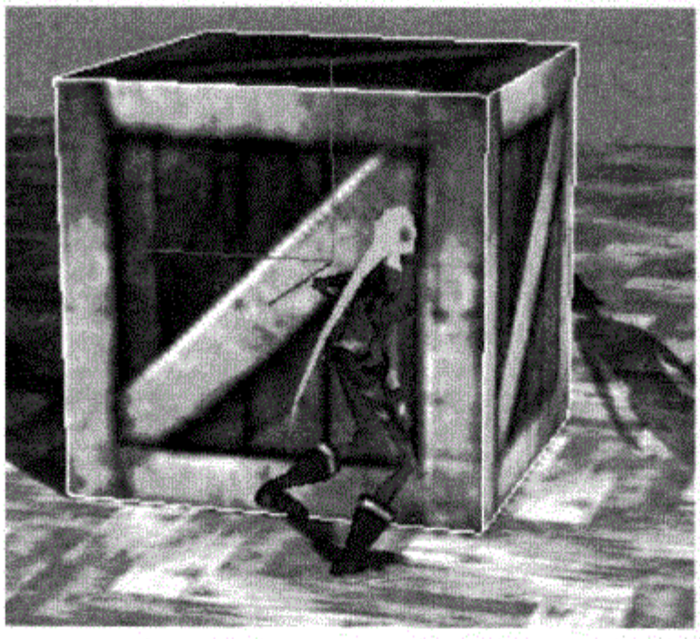


图 6.23 Jane 无法穿过箱子

当角色 Jane 撞到箱子的时候，并不会沿着移动方向滑动，这是由于 Prevent Collision 的作用仅仅是将物体移动到一个安全无碰撞的位置，并不具备 Slider 的功能，因此它的适用范围便受到了一定的局限，而它的优点是运算相对简单。

6.1.3 Object Slider 模块

场景中的圆球表面有弧线，角色在移动中沿着球体表面滑动会比较符合情理，以下将实现这种具备滑动功能的碰撞检测。

1. 撰写脚本

为 Jane 新建脚本并重命名为“圆球碰撞检测”，为其添加 Collision | 3D Entity | Object Slider 模块，如图 6.24 所示。

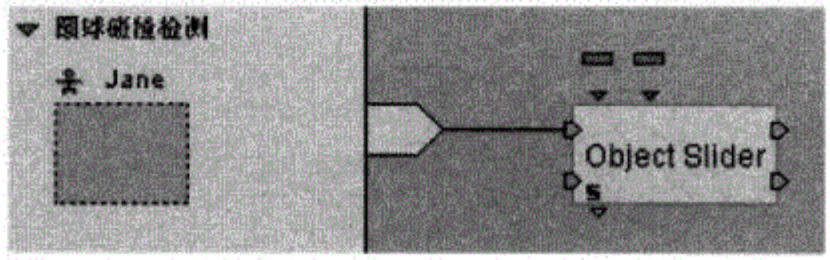


图 6.24 添加 Object Slider 模块

Object Slider 的参数设置面板具有两个参数，第一个参数是用来控制碰撞和滑动的虚拟圆球的半径，其原理是在脚本运行物体（本例中就是 Jane）的中心有一个虚拟的圆球体，该圆球体无法穿过障碍物并且会沿着障碍物表面滑行。半径参数不做改动，留待将来测试再进行调整。第二个参数 Group 是一个 Group（群组）类型的参数，如图 6.25 所示。

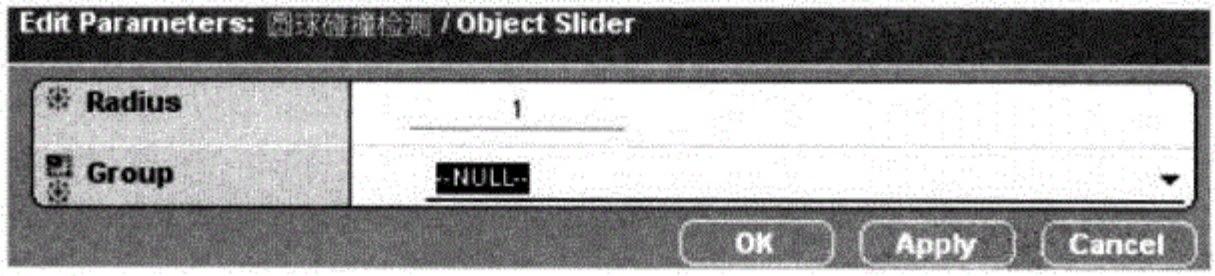


图 6.25 Object Slider 输入参数设置

下面需要新建一个群组用来盛放被 Object Slider 视为障碍物的物体。

2. 创建群组

Group（群组）是一种抽象的物体类型（不同于在 3D Layout 中显示出的三维物体），可以简单地将其视为“文件夹”，在其中可以放置任何类型的物体，例如，在 Object Slider 模块中，指定到 Group 中的所有三维物体便会被视为障碍物，除了脚本方面的应用，在较大规模的工程中，利用 Group 来分门别类管理各种资源也非常方便。

进入 Level Manager 并单击左侧的 Create Group 按钮，如图 6.26 所示，可以新建 Group。

图 6.26 Create Group 按钮

在 Level Manager 中随即多出了一个 Groups 分类，为新建的 Group 重命名为“Object Slider 障碍物”，如图 6.27 所示。

图 6.27 为新建 Group 重命名

下面，需要为 Group 中添加物体，即场景中的圆球，在 Level Manager 中找到 Sphere Low Res 物体并使用鼠标左键拖曳到新建 Group 的图标上，释放鼠标即可完成添加物体的工作，如图 6.28 和图 6.29 所示。

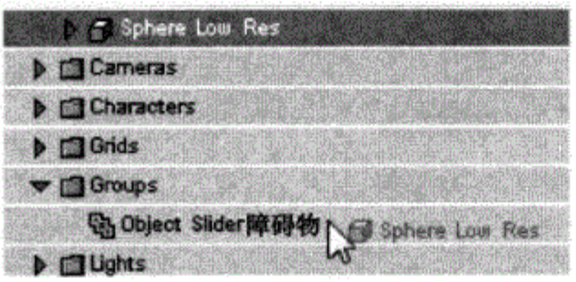


图 6.28 鼠标拖曳为 Group 添加物体

图 6.29 添加物体后的 Group

回到脚本中，为 Object Slider 指定 Group 参数，如图 6.30 所示。

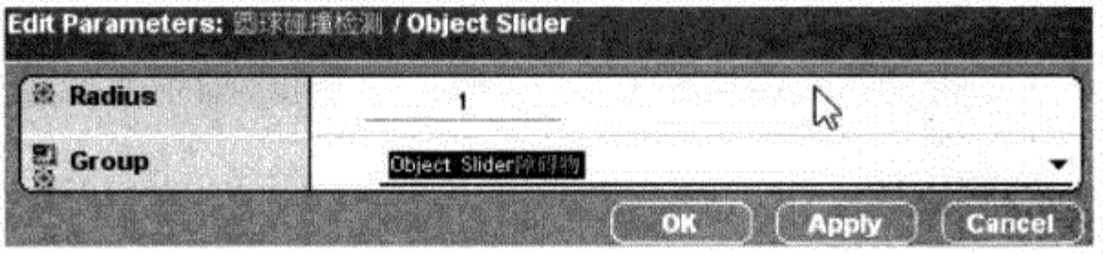


图 6.30 为 Object Slider 指定 Group 参数

运行测试，如图 6.31 所示，可以看到角色 Jane 已经无法穿过圆球并且会沿着圆球的表面滑动，但是角色 Jane 在距离圆球较远的距离就被阻挡了，这是由于 Object 的 Radius 参数过大的缘故，将其缩小为 0.5 即可，如图 6.32 所示。

Object Slider 模块相对于 Prevent Collision 模块具备了滑动的功能，适用范围较广，特别是针对不可移动的物体效果比较理想。

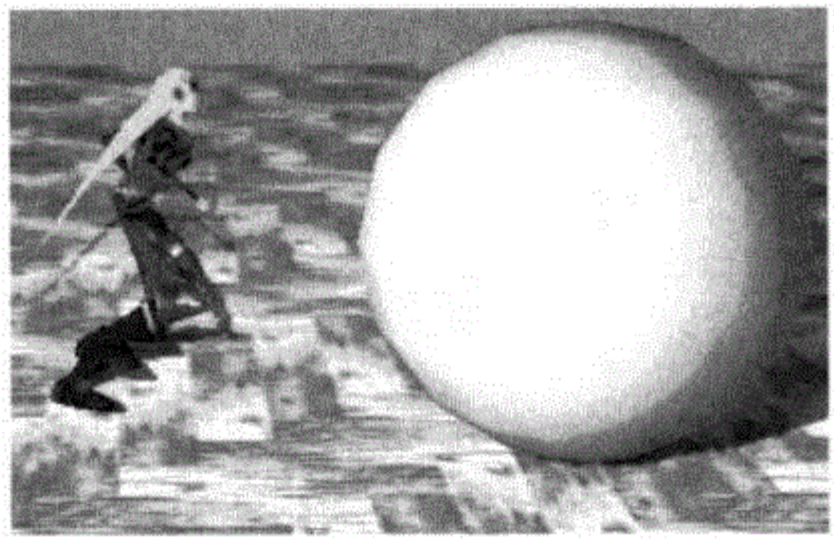


图 6.31　Jane 无法穿过圆球

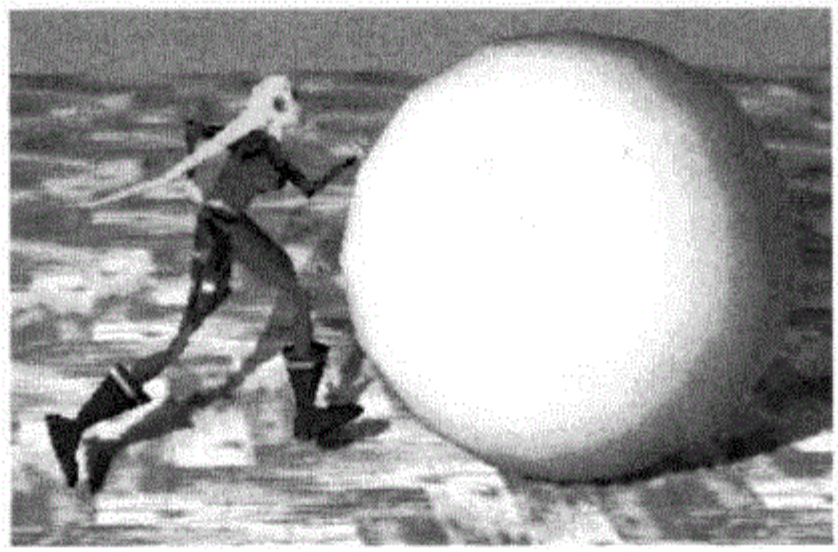


图 6.32　调整 Radius 参数后的效果

6.1.4　Sphere Slider 模块

非玩家角色（NPC）在场景中若是固定的，使用 Object Slider 即可以实现合理的碰撞检测，但 NPC 若是像场景中的 Eva 一样在不断移动，就需要让两个角色在碰到的时候分别“滑动”开从而不影响各自的移动，此时使用 Sphere Slider 模块就可以方便地实现这个功能。

Sphere Slider 模块的使用方法和 Object Slider 模块十分类似，也需要一个群组来盛放被视为“障碍物”的物体，因此应该新建一个 Group 并将 Eva 添加到其中，如图 6.33 所示。

图 6.33　新建名为 NPC 的 Group

然后，为 Jane 新建名为“NPC 碰撞检测”的脚本并添加 Collision | 3D Entity | Sphere Slider 模块，如图 6.34 所示。

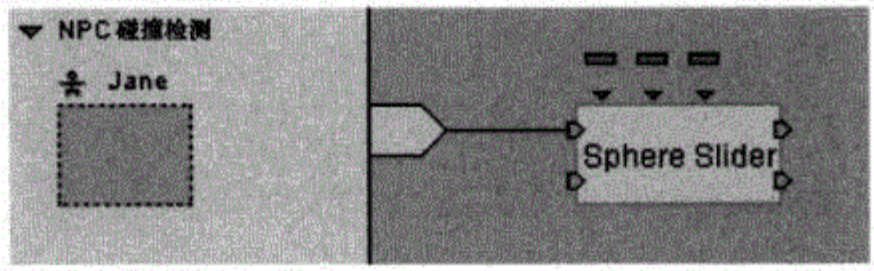


图 6.34　新建“NPC 碰撞检测”脚本

双击 Sphere Slider 进入输入参数设置面板，将 Entities 参数指定为 NPC Group，如图 6.35 所示。

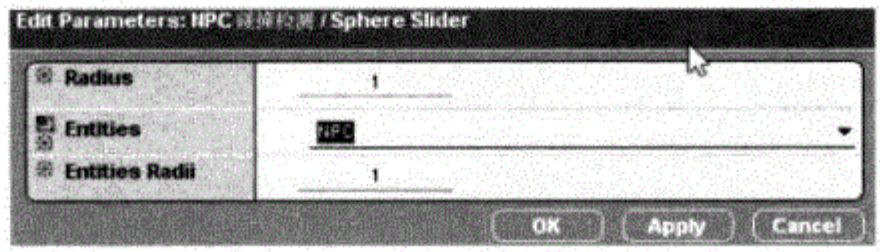


图 6.35　设置 Sphere Slider 参数

运行测试，可以看到角色 Jane 和 Eva 不会撞上并且会相互“滑动”以错开移动的位置，如图 6.36 所示。

图 6.36　角色会相互“错开”行走

现在的问题是两个角色之间的“安全距离”过大，这是由于半径参数造成的，进入 Sphere Slider 的输入参数设置面板将 Radius 和 Entities Radii（障碍物的碰撞检测半径）都改为 0.5 即可，如图 6.37 和图 6.38 所示。

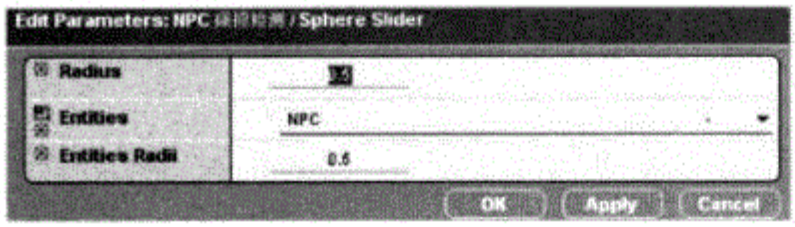


图 6.37 更改碰撞检测半径

图 6.38 角色间的安全距离比较合适

Sphere Slider 模块的工作原理和 Object Slider 模块很像，只不过 Object Slider 模块中的障碍物会根据自身模型的特点进行碰撞检测，而 Sphere Slider 模块则将障碍物的碰撞检测简化为一个虚拟的圆球，从而和模型的具体网格结构无关，因此 Sphere Slider 模块适合应用在模型复杂但是无须精确碰撞检测的情况，例如角色间的碰撞。

6.2 三维环境下鼠标单击物体

在游戏或其他交互应用中，经常需要使用到鼠标单击来激活某个物体或者某个事件，下面将介绍如何在三维环境下获取鼠标选中物体。

打开工程文件 CMOs\Chapter7\mousepick.cmo，如图 6.39 所示。

在该工程文件中，已经加入了一个互动脚本，使 2D Frame “text” 可以监听 UPDATE 事件，并根据 UPDATE 事件所携带的数据 “Picked（String 类型）” 在画面左上角动态显示文字 “Object Picked: ####”，如图 6.40 所示。

下面撰写的脚本逻辑很简单：监听鼠标左键按下的事件，当按下左键时，通过当前鼠标的位置获取场景中的三维物体，若鼠标下方有物体存在，则发送 UPDATE 信息给 2D Frame 并将该物体的名字作为 Picked 参数连同 UPDATE 信息一起发出。

为 Level 新建 Level Script，加入 Mouse Waiter 模块，并设置该模块的参数使其只监听 Left Button Down 的事件（Mouse Waiter 的使用方法在界面制作章节已经做过介绍），如图 6.41 所示。

然后，添加 Interface | Screen | 2D Picking 模块，如图 6.42 所示。

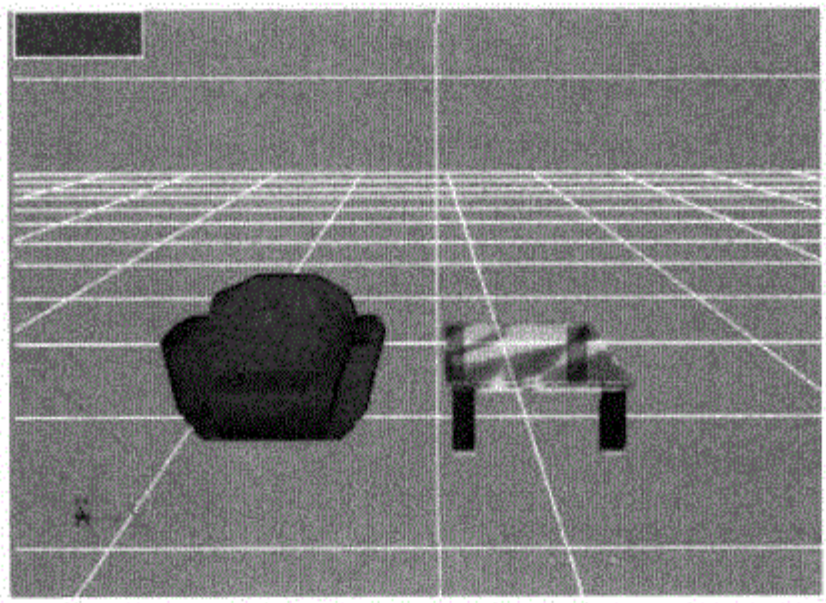


图 6.39　CMOs\Chapter7\mousepick. cmo

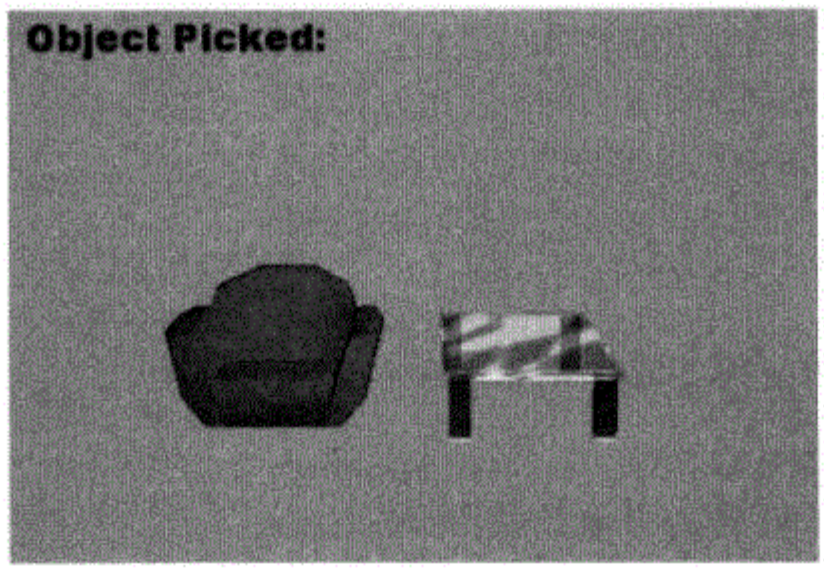


图 6.40　动态显示文字

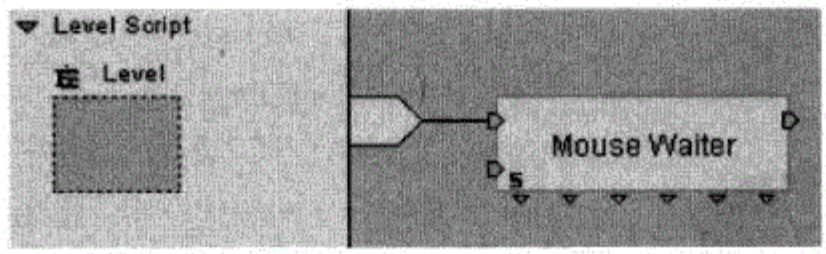


图 6.41　添加 Mouse Waiter 监听 Left Button Down 事件

2D Picking 模块的作用是，通过给定的屏幕二维坐标在三维环境下找到对应的物体。默认情况下，屏幕的二维坐标是通过当前的鼠标位置来指定的，但是也可以通过输入参数的方式来指定二维坐标，右键选择该模块 Edit Settings 进入模块设置面板，并将 Use Mouse Coordinates 选项的叉号去掉即可，如图 6.43 所示。

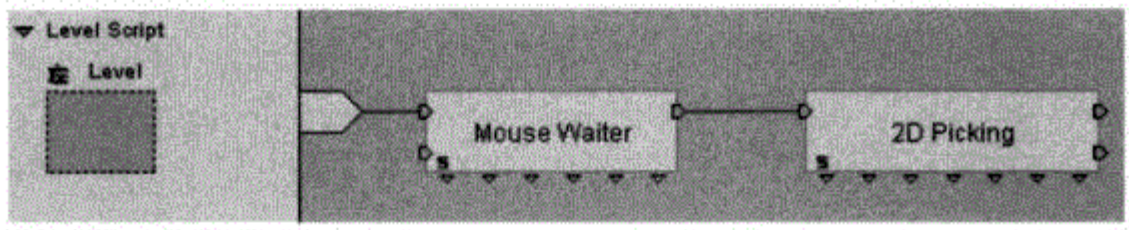


图 6.42 添加 2D Picking 模块

此时 2D Picking 多出了两个输入参数，如图 6.44 所示。

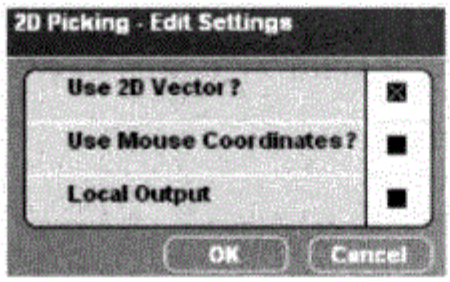


图 6.43 2D Picking 设置面板

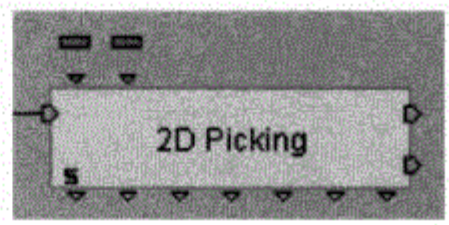


图 6.44 2D Picking 多出两个输入参数

如图 6.45 所示，参数 Pos 为二维坐标，而 Window Relative 则表示 Pos 二维坐标是否为相对于 Virtools 三维视窗而给定的坐标值，若不选择该选项，Pos 则表示整个操作系统屏幕坐标系统中的二维方位。

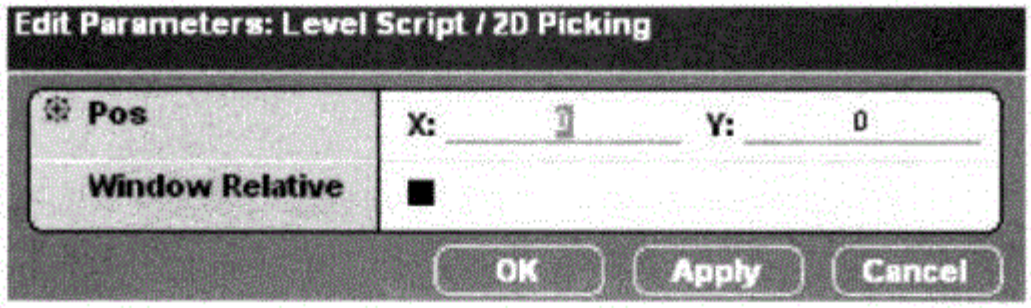


图 6.45 2D Picking 输入参数设置面板

由于在此需要使用鼠标提供的方位，因此仍需选择 Use Mouse Coordinates 选项。

当 2D Picking 得到三维物体后，会激活第一个输出端 True，在此加入 Send Message BB 向 2D Frame “text” 发送 UPDATE 信息，如图 6.46 和图 6.47 所示。

然后，为 Send Message 添加一个输入参数，并设定参数类型为 String，参数名为 Picked，该参数会在 “显示文字” 脚本中被识别并显示出来，如图 6.48 所示。

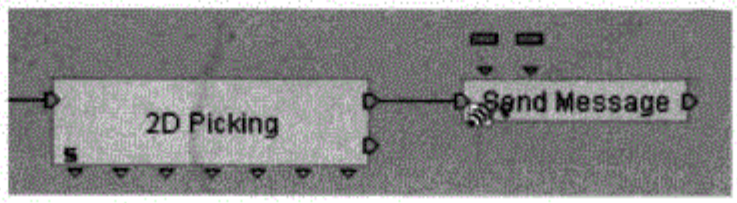


图 6.46 添加 Send Message BB

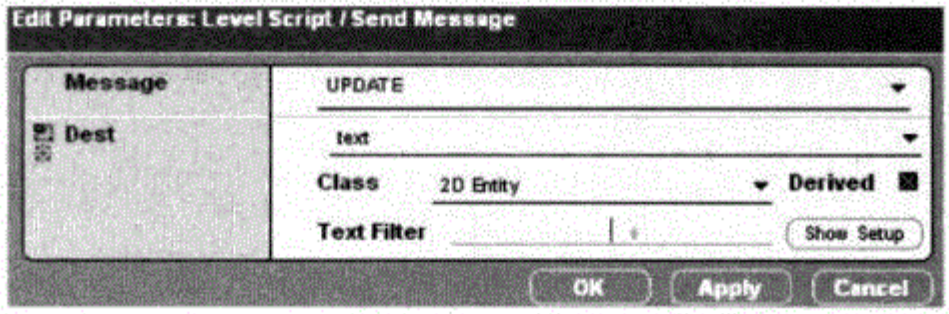


图 6.47 Send Message 输入参数设置

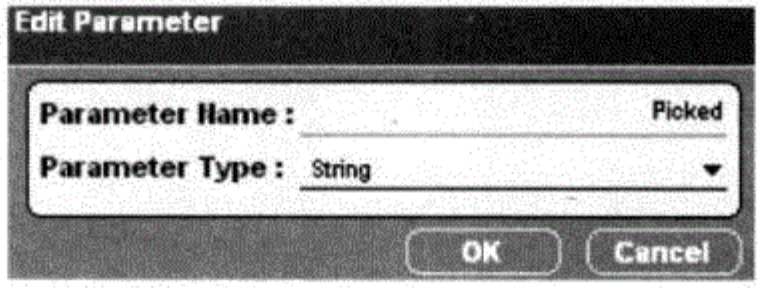


图 6.48 为 Send Message 添加 Picked 输入参数

2D Picking 得到的三维物体被赋予给了第一个输出参数 Object Picked（3D Entity），为了将该物体的名字赋予 Send Message 的 Picked 输入参数，只需要直接将两参数相连，连线将会被自动赋予 Get Name 的运算操作，完成获取物体名称的操作，最后为 Mouse Waiter 设置自循环，如图 6.49 所示。

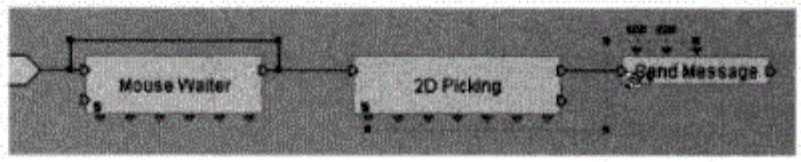


图 6.49 最终脚本

运行测试，用鼠标单击场中的物体，将会显示该物体的名称，如图 6.50 所示。

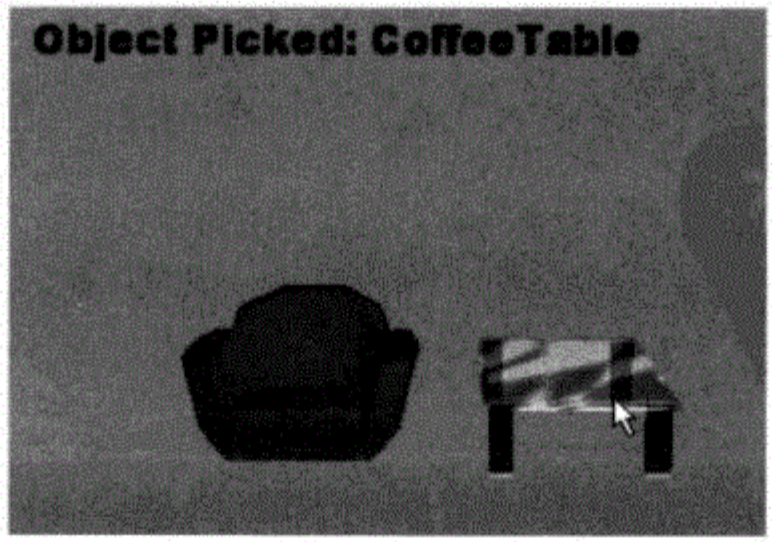


图 6.50 运行测试

掌握了2D Picking的运用方法后，就可以举一反三，利用鼠标单击，实现开启宝箱、开门之类的操作。

6.3 摄像机轨道漫游

三维漫游的方式有很多种，其中常用的方式是控制摄像机沿着预先铺设好的轨道移动漫游。这种方式的好处是摄像机的移动更加平滑，制作者观察景物的可控性强。

6.3.1 绘制漫游轨道

打开工程文件CMOs\Chapter7\curve.cmo，如图6.51所示。

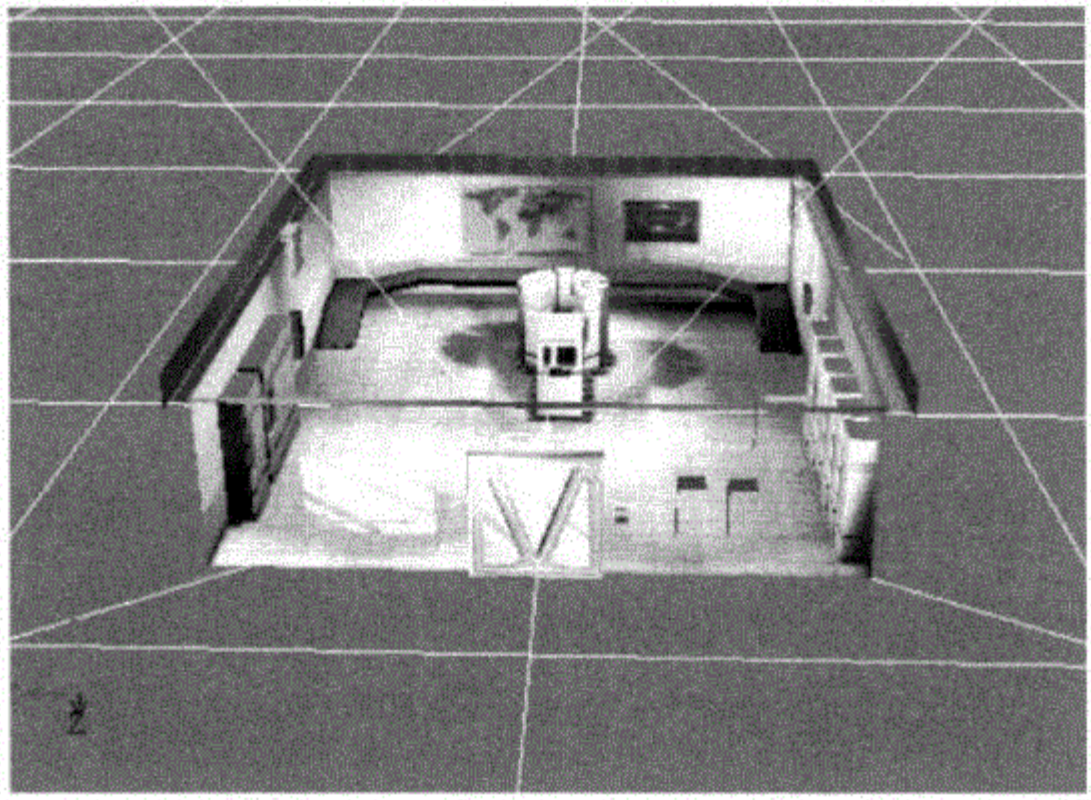


图6.51 CMOs\Chapter7\curve.cmo

图6.52 Create Curve按钮

该工程中包含了一个场景（Virtools自带资源）以及一个摄像机Camera，首先需要绘制一条供摄像机漫游的轨道，这将使用到一个新的物体类型——Curve（曲线）。切换至Top顶视图（这更有利于绘制漫游轨道），然后单击3D Layout左侧创建物体工具栏的Create Curve按钮，如图6.52所示。

在3D Layout中通过单击鼠标左键即可按顺序为新建的Curve添加锚点，并画出一条曲线，如图6.53所示。