动漫·游戏专业系列教材

二维游戏设计与制作

主编 李 炯 袁 蔚

高等教育出版社

内容简介

本书是动漫游戏专业系列教材之一。

全书共分8章，第1章主要介绍2D游戏的基本概念和游戏模式，以及游戏研发的主要分工和相互之间的关系；第2章为游戏的美术制作部分，通过多个游戏场景和角色像素图制作的实例，使学生掌握游戏中美术制作的技巧；第3章、第4章、第5章是对2D游戏引擎的讲解，其中第3章是对游戏引擎的使用说明，第4、5章通过两个实例指导学生制作纵向射击游戏和横向冒险游戏；第6、7、8章主要讲解RPG游戏制作工具RPG Maker XP，第6章是对它的使用说明的详解，第7章通过实例指导学生制作RPG游戏，第8章是脚本部分，通过使用编程语言的方式，使学生对RPG Maker XP制作的游戏进行扩展等。

为了便于学生学习，本书配有光盘，光盘中收录了书中所涉及的3个实例以及书中用到的素材资源，以供学生学习和借鉴。

本书同时配有学习卡，按照本书最后一页"郑重声明"下方的学习卡使用说明，可登录http://sve.hep.com.cn，可上网学习，或下载资源。

本书采用出版物短信防伪系统，用封底下方的防伪码，按照本书最后一页"郑重声明"下方的使用说明进行操作可查询图书真伪并有机会赢取大奖。

图书在版编目(CIP)数据

二维游戏设计与制作/李炯，袁蔚主编. —北京：高等教育出版社，2009.9

ISBN 978-7-04-026056-4

Ⅰ.二… Ⅱ.①李…②袁… Ⅲ.二维-动画-设计-教材 Ⅳ.TP391.41

中国版本图书馆CIP数据核字(2009)第139662号

出版发行	高等教育出版社	购书热线	010-58581118
社　　址	北京市西城区德外大街4号	咨询电话	400-810-0598
邮政编码	100120	网　　址	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	北京人卫印刷厂	畅想教育	http://www.widedu.com
开　　本	787×1092　1/16	版　　次	2009年9月第1版
印　　张	18.5	印　　次	2009年9月第1次印刷
字　　数	430 000	定　　价	31.20元（含光盘）

物料号　26056-00

前　言

游戏是每个人儿童时代的最爱。随着科技的日新月异以及新兴媒体的出现，游戏设计与制作的发展也越来越快。现在，游戏专业已经被全国多所高等学校开设，学生们能通过学习来打造自己儿时的最爱，因而受到普遍的欢迎。

由于目前国内的教育情况，游戏专业中美术与程序存在脱节现象。本书的编写正是基于以上的考虑，同时更注重适应中等职业教育的人才培养目标以及课程设置的总体要求，力图使本书在内容丰富、概念明确、重点突出、结构合理的基础上，突出实用性和实践性强的特点。

在本书的编写过程中，首先要感谢2D游戏的先驱者，正是由于他们才有了2D游戏的过去。其次要感谢在这个3D游戏随处可见、却仍然坚持不懈地热爱着2D游戏的人们，由于他们，2D游戏才有了继续发展下去的机会。最后要感谢本书的读者，因为有了读者2D游戏才有未来。

感谢霆泰公司艺术总监任志峰的鼎力协助，还要特别感谢惠普公司李富伟在本书RPG Maker XP程序中的帮助，另外感谢张翔王对本书的帮助，最后衷心地感谢张苏中教授给予的大量意见以及帮助。

本书第1、3、4、5章由李炯、任志峰编写，第2、6、7、8章由袁蔚、张翔王、李富伟编写。

作者联系方式：lijiong2@sohu.com。

作　者

2009年4月于上海

目　录

第 1 章　2D 游戏概述

1.1 2D游戏基本概念

1. 2D 游戏的概念

我们现在所说的游戏是指在计算机上运行的单人或者多人的游戏，而 2D 游戏是相对三维游戏而言的，是用二维的描画方式表现的游戏，如图 1-1 所示。

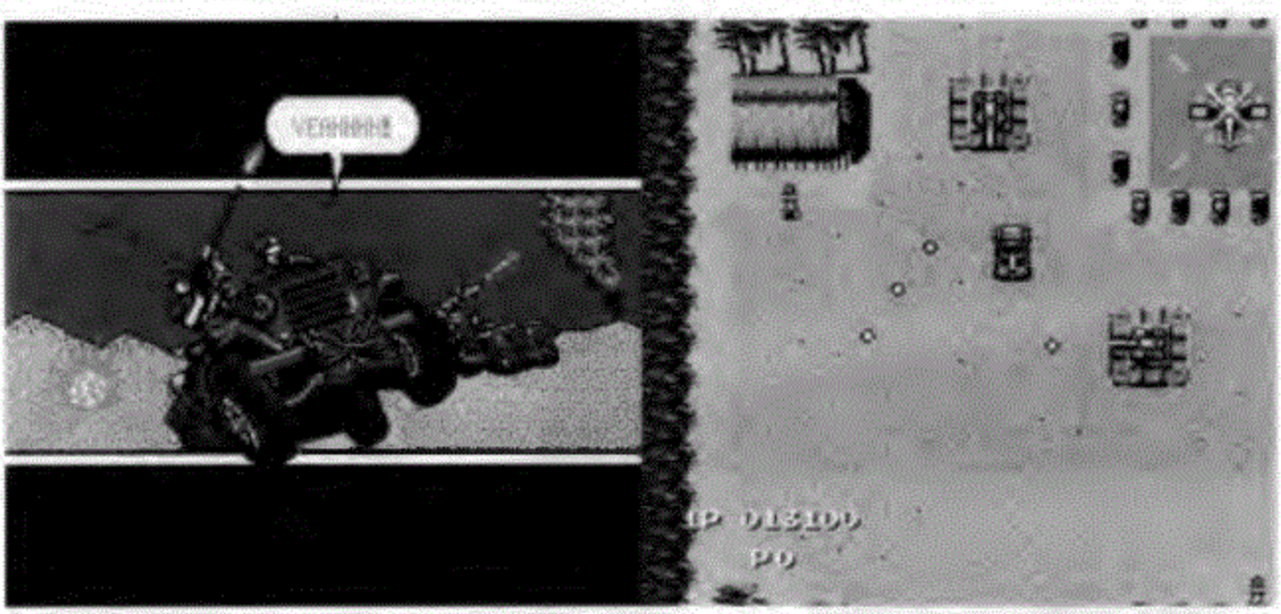


图 1-1

2. 2D 游戏的主要运行平台

2D 游戏是最早的游戏形式，从最早的任天堂 FC（红白机，如图 1-2 所示）、任天堂 GB、任天堂 GBA、任天堂 SFC（超任，如图 1-3 所示）、世嘉土星，到现在的 PS（如图 1-4 所示）、PSP、XBOX 360（如图 1-5 所示）等都是 2D 游戏的主要运行平台。

3. 2D 游戏存在的意义

现在，越来越多的游戏正向 3D 发展，这是由于硬件的快速发展在一定程度上满足了人们在游戏模拟现实方面的需求所造成的。在游戏有了 3D 表现能力以后，出现

了许多新的游戏规则，为玩家带来更多的乐趣，但与此同时一些 2D 游戏固有的规则不能在 3D 游戏中实现，使其丧失了不少乐趣。因此，2D 游戏虽然会逐渐减少，但绝不会消失，2D 游戏和 3D 游戏将会像两条平行的轨道一样各自发展，如图 1-6~ 图 1-11 所示。

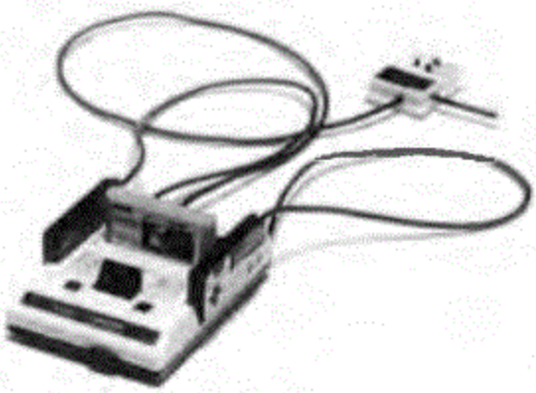


图 1-2

图 1-3

图 1-4

图 1-5

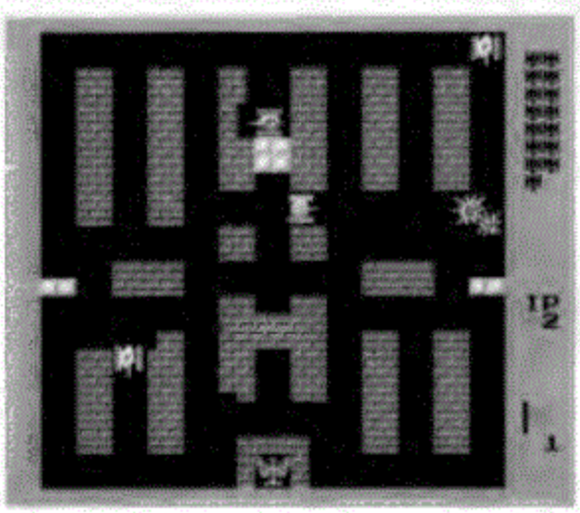


图 1-6

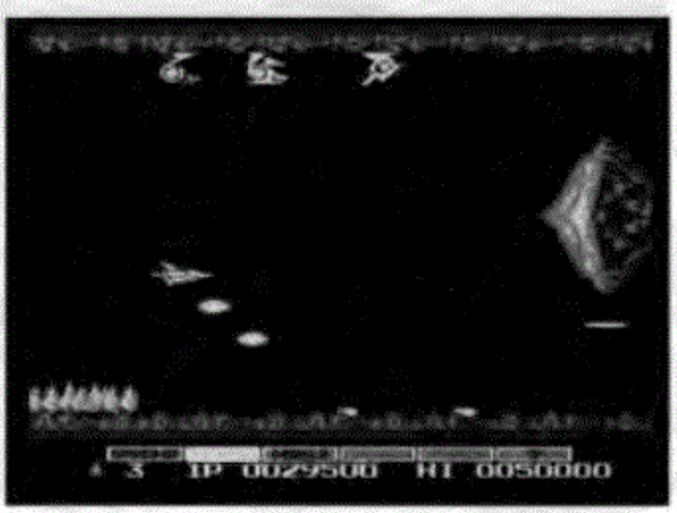


图 1-7

图 1-8

图 1-9

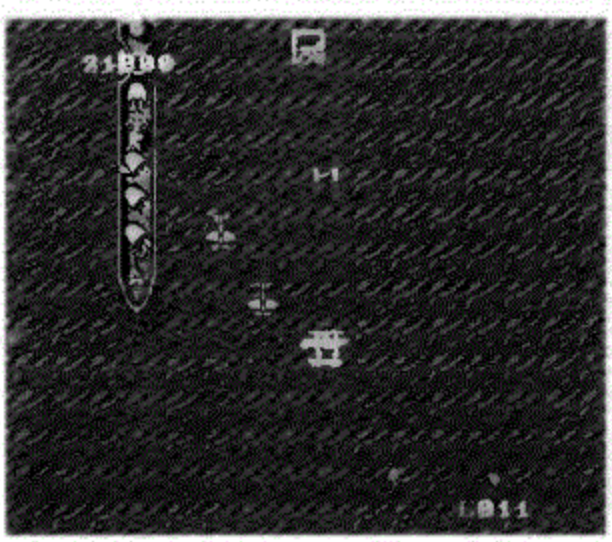


图 1-10

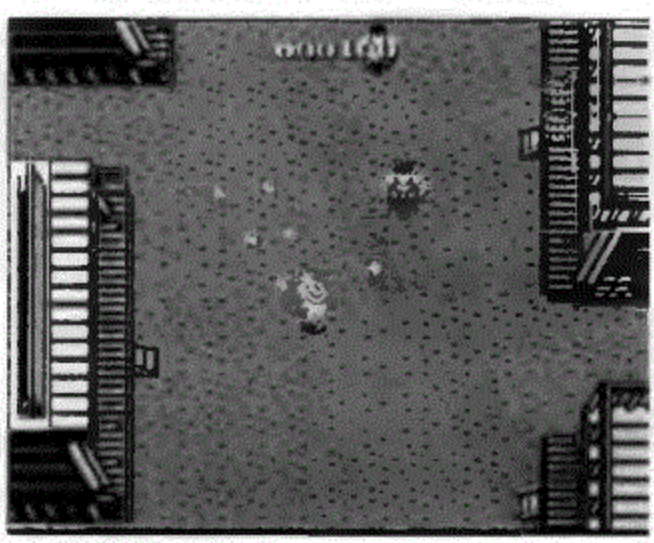


图 1-11

4. 2D 游戏的图形表现

在 2D 游戏盛行的年代，只要有足够的制作时间和预算，就算没有先进的 3D 绘图技术，游戏一样还是可以凭借外表漂亮的 2D 人物及背景图形来传达整个游戏和游戏中角色的情感。从这点上来讲，那些角色扮演和养成类游戏更加能够体现这种特色，它们大量使用了此类方式来表现游戏角色的喜怒哀乐。《美少女梦工厂》，如图 1-12 所示，就是一个养成类游戏，它以精美的画风传递游戏角色的情绪，不仅受到了男性玩家的喜爱，而且也受到了许多女性玩家的青睐。

图 1-12

由于使用 2D 图形是比较经济、实用的表现方法，因此甚至连一些本身已经比较成功的 3D 游戏，也会使用 2D 卡通的表现方法来体现游戏的整体风格。

1.2 2D游戏研发中的主要分工以及它们之间的关系

游戏的制作是一个大型项目，需要不同能力、不同特长的人员通力合作，各个环节必须环环相扣才能最终完成。一款受欢迎的游戏尤其如此。基本上每款游戏的制作架构都如图 1-13 所示。

1. 游戏制作人

游戏制作人类似于电影制作人、唱片专辑制作人，主要负责寻找、组织并且控制游戏研发所需的经费、人员以及一级发行渠道等，可以说他是游戏成功与否的直接负责人。当然，由于当前游戏制作的规模不断扩大，有时甚至需要多个团队协作

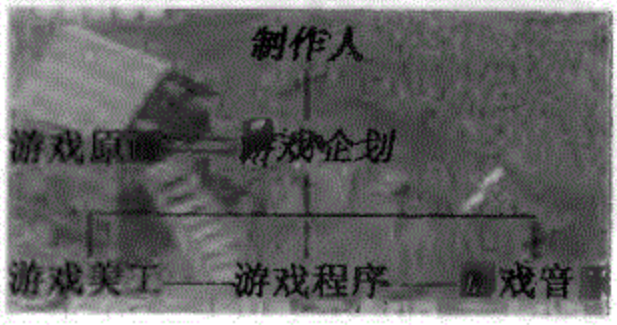


图 1-13

完成，出于每个团队各自的利益以及它们之间的协调，有时会需要多个级别的游戏制作人。

2. 游戏企划设计

游戏企划类似于电影导演。他们根据团队特点以及市场预测对游戏进行具体规划并指导游戏研发。由于游戏会有不同的类型和不同的规模，游戏企划往往是由几个人组成一个企划团队。企划团队的成员分别承担游戏总体企划、剧情企划、关卡企划、协调企划等工种，但无论哪方面的企划都必须符合游戏企划的总体风格、特色以及游戏性。游戏企划不仅要确保企划本身的质量，还要对其他工种起到指导作用。所以一个好的企划者往往是这个研发团队和游戏的灵魂。

3. 游戏程序设计

游戏程序设计是指根据既定的游戏企划和运行平台，选择相对适合的编程语言完成游戏的编辑、生成最终的可执行文件，从而实现游戏的运行。游戏编程是游戏研发中最为重要的部分，也是所需人力资源较多的部分。没有程序设计，就不可能产生游戏软件。游戏规模越大，所需的人力资源也越庞大，而且必须分工明确，协调性强，还要由一个有经验且高水平的主程序工程师制定基本框架、统一思想、协调合作、掌控进度，主程序工程师对游戏软件的顺利完成起到举足轻重的作用。

4. 游戏美术设计

游戏美术设计是指根据既定的企划对游戏画面进行规划和创作，最终制作完成程序所需要的数据。最初的游戏美术设计只是对游戏编程的补充，但随着游戏研发技术的快速发展和人们对视觉享受的不断追求，美术设计在游戏研发中所占的比重越来越突出，在一个中型或大型的游戏研发中，美术设计的人力资源是最为庞大的。

5. 游戏原画

游戏原画是指根据企划的设定对游戏中的人物、场景以及其他物件等进行美术创作。其根本目的是将企划设定图形化，将企划的理念通过视觉感受传达给程序员、美术设计人员以及音乐音效设计者，它对游戏画面的制作起到视觉感官的指导作用。

6. 游戏音乐音效设计

音乐音效设计是指根据游戏种类或者故事背景对游戏中所需要的音乐和音效进行创作，是对玩家除视觉以外的感官享受的补充，从而使游戏更好地体现引入感。

1.3 2D游戏研发中各种美术数据之间的关系和形成理念

简单地说，游戏是对现实的模拟，是现实逻辑的延续和超越，所以游戏绝大部分必

须符合现实规则。2D 游戏是在上下左右的二维空间规则下的游戏，其最大的特点是所有的物件都置于一个平面的矩阵之中，所有的碰撞判定只能在 X 坐标和 Y 坐标上产生，因此 2D 形象和场景都是能够通过直接绘图制作出来的，当然也可以通过 3D 软件渲染成 2D 绘图。在二维世界里无法做到场景的即时视角转换，绘制二维角色形象时，制作者必须为角色的动画效果画每一幅图片。以下是 2D 游戏研发中几种基本美术数据。

1. 有动画和没有动画的场景

① 纵向卷轴，如图 1-14 所示。

② 横向卷轴，如图 1-15 所示。

③ 跟随卷轴。

（a）90° 视角，如图 1-16 所示。

（b）120° 或 45° 视角，如图 1-17 所示。

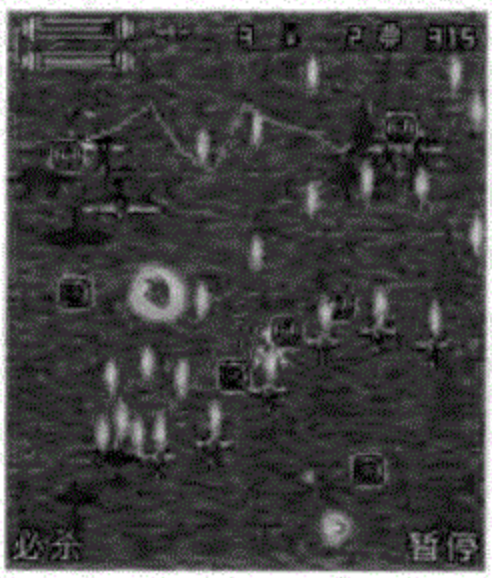


图 1-14

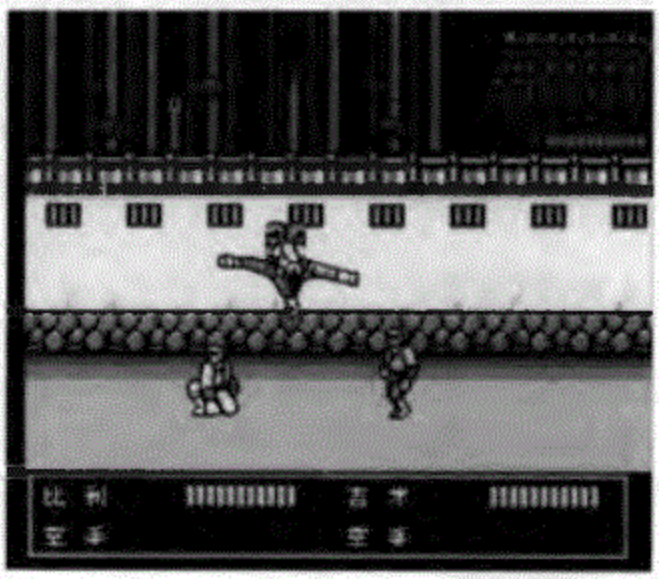


图 1-15

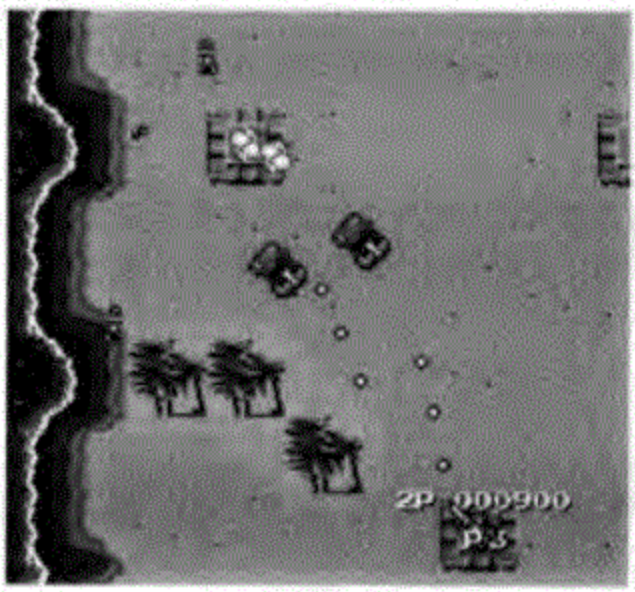


图 1-16

图 1-17

说明：游戏中的卷轴（或称之为滚屏）是相对游戏主人公而言的。

2. 可移动和不可移动的物件

如图 1-18 所示，可移动和不可移动的物件有如下几种。

① 主人公。

② 敌人。

③ 情节所需的人物和物件。

图 1-18

说明：背景和物件的视角要统一，如横向卷轴物件的视角是侧视，纵向卷轴物件的视角是俯视，而跟随卷轴物件的视角是侧视和俯视相结合。

3. 游戏周边画面和游戏界面

① 游戏开始画面，如图 1-19 所示。

② 游戏结束画面，如图 1-20 所示。

图 1-19

图 1-20

③ 游戏过场画面，如图 1-21 所示。

④ 各种选择或设置画面，如图 1-22 所示。

图 1-21

图 1-22

⑤ 游戏界面，如图 1-23 所示。

⑥ 菜单画面，如图 1-24 所示。

图 1-23

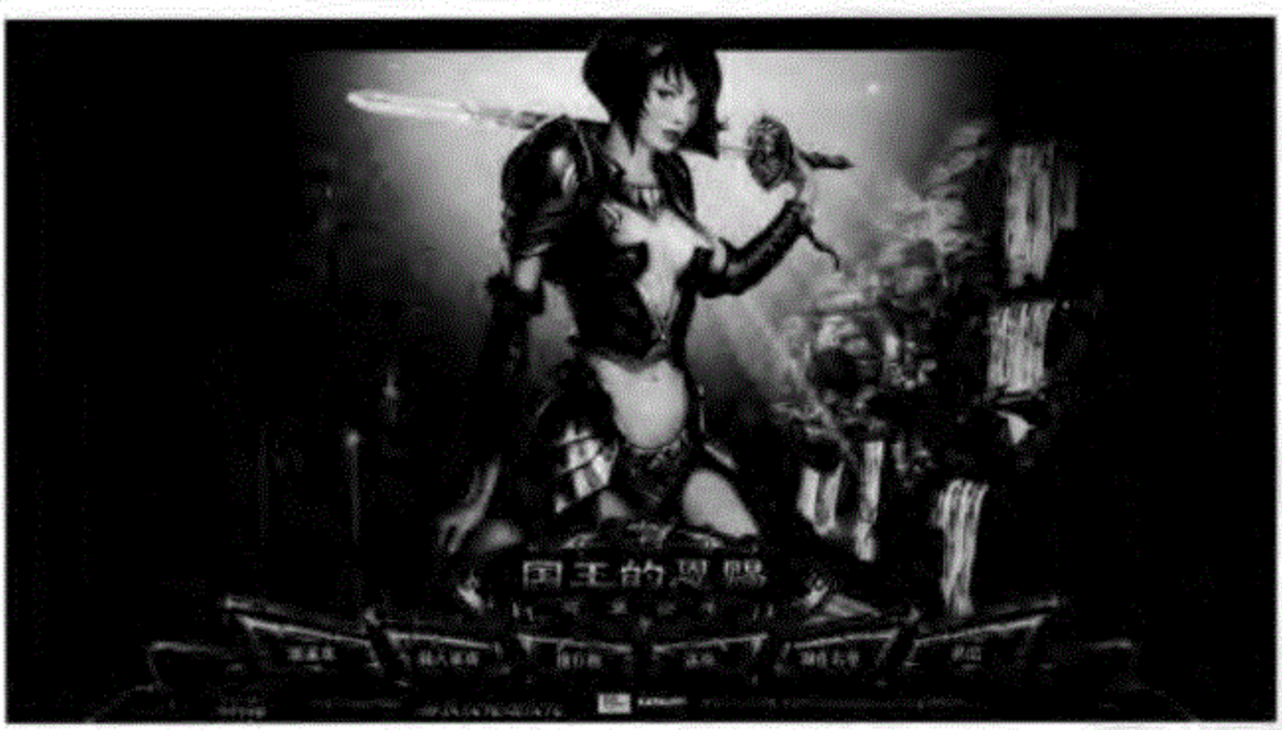


图 1-24

1.4 怎样成为一个合格的游戏美术设计师

1. 了解限制

美术设计师必须了解游戏引擎所带来的限制，如画面的尺寸、调色板等，以及了解这些限制的重要性，这样才有可能在现有的条件下制作出最出色的画面。

2. 不要过分依赖软件

无论如何设计师必须具备很好的美术才能。要成为一个合格的游戏美术设计师，仅仅知道如何使用各种绘图软件是远远不够的。计算机软件是不会替设计师完成美术设计工作的，美术设计师要有良好的绘画基础和美术鉴赏力，再精通 1~2 种绘图软件，就会在工作上游刃有余了。

3. 尽可能地吸收多方面的知识

如果美术设计师只把自己定位成一个单纯的绘画者，是很难从事游戏美术设计这项工作的。他还要尽可能多地吸收来自多方面的知识，如：科幻、神话、宗教、地理、历史、生物等，这样对快速理解研发中来自其他工种的信息和指导是很有帮助的。美术设计师还要在工作中尝试用程序员的思维方式去考虑问题。

4. 培养良好的团队合作精神

游戏通常由许多人共同制作完成，而每个人的工作都相互关联，因此美术设计师也不可能把自己封闭起来。美术设计师的工作态度将在团队中起到很大的作用，如果团队中所有成员的关系都十分融洽，游戏开发就会进展顺利，反之就可能毁于一旦。

5. 准备接受来自任何人的批评

美术设计师应学会去积极地接受批评。批评可能来自于其他的美术设计师、企划、程序员、甚至非专业人员。

6. 要学会给自己充分的思考时间

当美术设计师在接到一项任务的时候，不要急于马上进入制作阶段，要给自己安排充分的思考的时间，将任务了解清楚，以免出错。游戏中的资源正在变得越来越多，因此美术设计师必须能够组织自己的创作并且合理安排每个部分的时间。

第 2 章　像素图绘制

2.1 像素图基本概念和用途

1. 像素图的概念

像素图是一种以“像素”（Pixel）为基本单位来进行制作的计算机绘图表现形式。

其表现方式为点阵图（Bitmap），这是计算机最原始的表现方式并沿用至今。计算机绘图分为点阵图（又称位图或栅格图像）和矢量图。顾名思义，点阵图是指以点阵方式保存的图像。由于计算机系统在保存点阵图时保存的是图像中各点的色彩信息，因此，其优点是画面细腻，它主要用于保存各种照片和图像；但是，点阵图的缺点是文件尺寸太大，且和分辨率有关，因此，将点阵图的尺寸放大到一定程度后，图像将变得模糊，如图 2-1 所示。

2. 为什么要画像素图

首先需要理解点阵图缩放后为什么会变得模糊，这里涉及一个“插值”的概念。

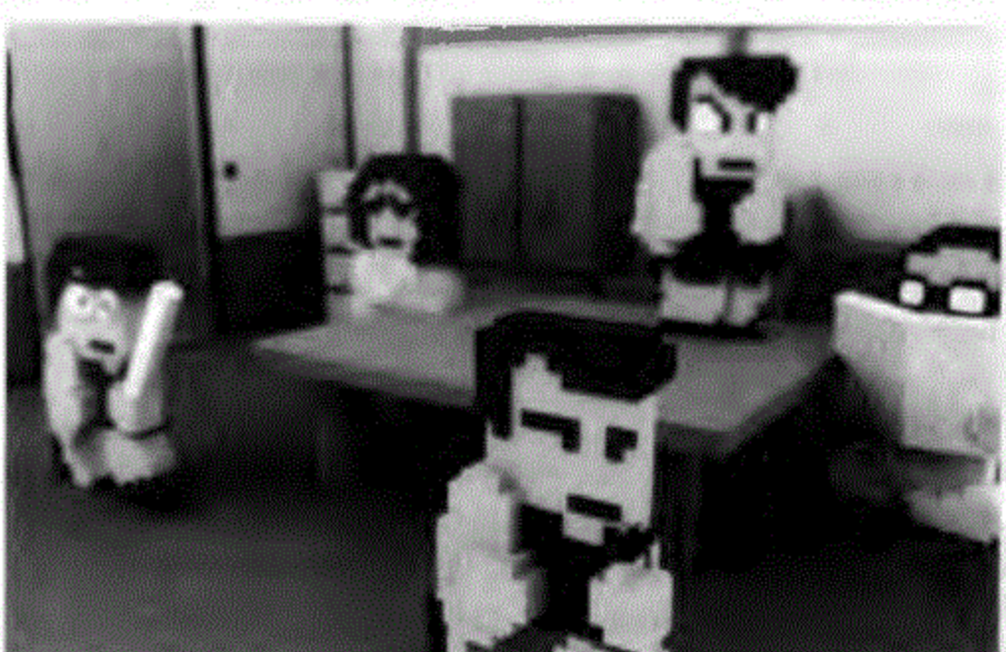


图 2-1

举例来说，9个像素缩小2/3是3个像素，但9个像素缩小一半就是4.5个像素，但屏幕最小单位就是1个像素，不存在半个像素单位的屏幕，所以这就有了插值的说法，放大时同理。因为在没有像素的地方需要补上像素，而此时计算机会自动补上像素。

一般像素图中会有像素的描边。当图像缩小时，会发现点阵图中有些曲线变得不连续了；相反当图像放大时，会发现这条线有些地方变成了2个或3个像素的描边，像素图中的线条也变粗了。为了保持描边像素的宽度，就需要在缩放后重新填补颜色修改线条，靠近颜色变化的地方都需要重新上色或描边，因而它并不是真正地可以无损缩放的方式，也不能说将位图转换成为像素图了。

在绘制低分辨率图像的情况下，如果事先绘制较大图像再根据分辨率限制对其进行缩小，图像将会变得模糊。所以，为了在有限范围内得到最好的图像效果、最多的细节，就需要根据图像大小量身定做，就是一个点一个点地画出来。这就是像素图及其特殊的制作过程诞生的原因，所以制作像素图之前确定图像大小是首要条件。

3. 如何画像素图

像素图的入门十分简单，但绘制优秀的作品就需要很好的绘画功底加上一定的技术能力。难点在于提高绘制的效率，而这一切都建立在对像素图原理的了解和丰富的绘制经验的基础上。

选择什么样的工具、软件应因人而异。无论选择什么软件，其制作像素图的基本流程都是一样的。

当然，绘制者熟悉并利用软件特点也是提高效率的关键，甚至可以在熟悉的软件上创造出适合自己的“独门秘籍”。基本上只要具备“铅笔工具”的绘图软件都能制作出像素图，最典型的代表是Windows自带的“画图”程序。

制作像素图的软件一般可分为两种类型，一种是用来制作Icon图标的专用软件，比如Microangelo、IconCool、Articons、Aha-soft等；另一种则是用来编辑位图的图形软件，比如：Photoshop、Fireworks、PhotoImpact等。本教程使用的软件是Photoshop CS3和Fireworks CS3，两种软件在绘制像素画时各有优势。本教程中大部分技巧可以在这两种软件上通用。

像素图的存储格式有多种，主要有bmp、ico、gif、png等。其中最常用的存储格式为gif与png。png能保存图层信息（包括透明图层），gif能以动态形式出现，而ico为操作系统图标专用格式。Fireworks默认图片保存格式为png、导出动画格式为gif。使用Photoshop绘制的过程中则要先保存为PSD格式。

4. 像素图的历史和应用

像素图最早出现在计算机应用程序的图标（Icon）以及早期8位元电子游戏中。由于点阵图图像是由一个个点构成，加上早期计算机图片处理能力较低，才衍生出“像素图”这种图像风格。应用范围从FC家用红白机画面直到今天的NDS手掌机；从黑白的手机图片直到今天全彩色的掌上电脑中的小游戏；另外有计算机图标、即时通信软件（如QQ表情、QQ秀）、网页图标等。

虽然像素图最初的诞生原因是由于计算机性能的限制，但现在它已经演变成一种独

立的艺术创作风格，通常被人称为“像素艺术”（Pixel Art）。此风格图像强调清晰的轮廓、明快的色彩，同时像素图的造型往往比较卡通，因此深受大众喜爱，如图 2-2 和图 2-3 所示。

图 2-2

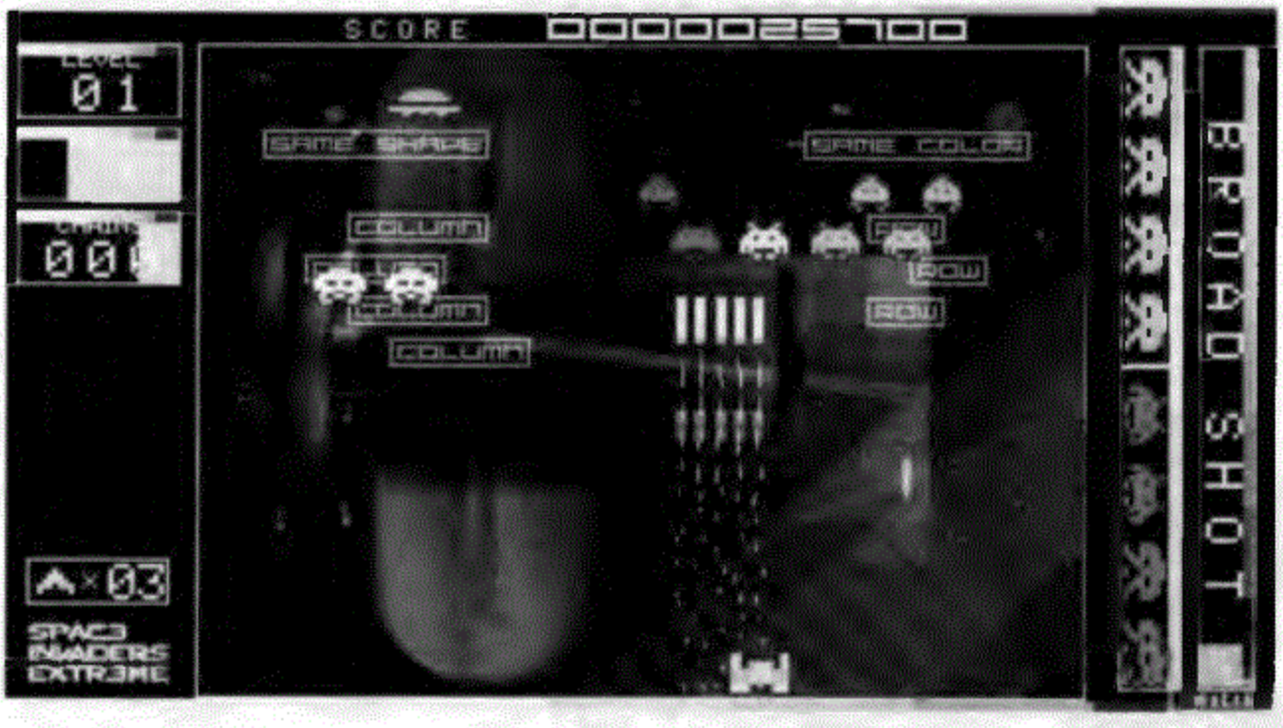


图 2-3

其以点（像素）为单位的构成特征与马赛克（Mosaic）艺术相似，生活中类似的表现形式还出现在编织中，例如十字锈，如图 2-4 所示。

图 2-4

5. 像素图的发展和未来

在游戏领域，因为早期 8 位元电子游戏都使用此风格表现，所以现在也被当做一种有趣的怀旧风格，成为游戏文化的一部分。一些新游戏将此风格夸张化以作为卖点，如 PSP 上的“勇者别嚣张”系列，如图 2-5 和图 2-6 所示；“勇者 30”，如图 2-7 所示。甚至 CAPCOM 公司和 SEGA 公司分别于 2008 年、2009 年在家用机 Wii 上推出的新游戏《洛克人 9》和《珍道中 !! 保罗的大冒险》都使用与 FC 游戏相同的 8 位机的画面风格，连音乐也是 8 位元，所以像素图并不会随着计算机和游戏机性能的提升而退出历史舞台，相反它将作为一种数码艺术风格继续存在下去。如今一些“像素图”作品已经脱离了“像素”的束缚甚至使用 3D 的立方体来表现，但它们的基本原

图 2-5

图 2-6

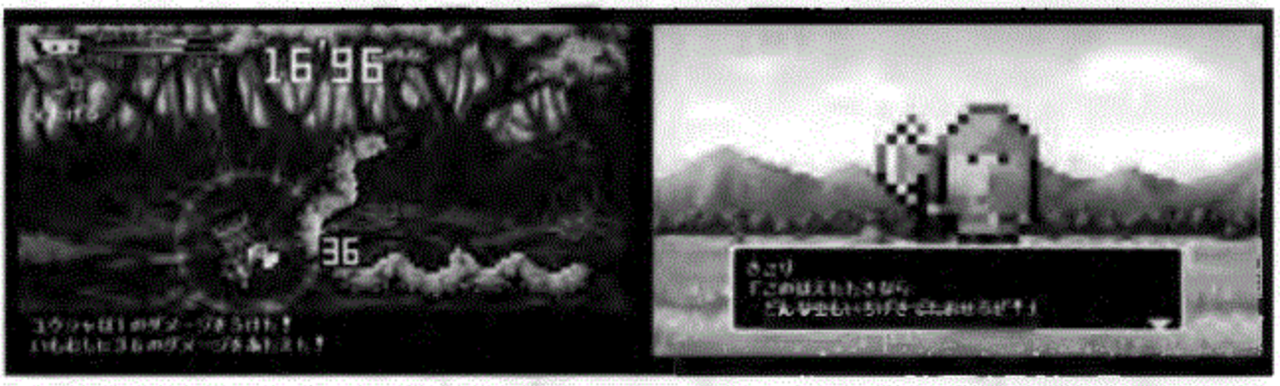


图 2-7

理与实用领域的像素艺术是完全相同的。

2.2 像素图绘制技巧

1. 基本绘制方法

下面就以对简单实例的分析向读者介绍使用 Photoshop 绘制像素图的流程和技巧。这些实例将会涉及一些基本工具的使用和一些绘制像素图的基础知识。

（1）线的画法

像素图的绘制不同与普通的绘画，由于画布大小的限制，要求每一个点的位置都要精确，所以在绘制时也要以精确为前提。我们首先练习精确曲线的画法。在绘制前利用PS的一些功能做好准备工作可以在后续的绘制中大幅提高绘图效率。

首先将画布放大至800%。在实际绘制过程中，要常常使用缩放工具（Z）来调整图像的显示大小。在绘制较小的图像时可以先将图像放大至800%甚至更大后再进行绘制，但在绘制过程中要经常将图像缩小以观看整体效果。

打开并调整网格工具。在“编辑”菜单的“首选项”子菜单中将网格线间隔设为1个像素，这样图像放大后就可以在绘制时通过看网格知道1个像素的大小，使用铅笔工具时也可以准确知道当前光标的有效范围（右下角）处在哪个像素中。

明确我们的绘制工具。Photoshop中使用“画笔工具”在绘制斜线或弧线时会自动消除锯齿，产生渐变，在绘画时能使线条更柔和，但它们会给绘制精确的像素图带来极大的不便，所以更精确的铅笔工具是我们绘制像素图的主要工具。

提醒：使用“铅笔工具”时鼠标指针会变成方形，放大后可以发现光标有效范围是方形的右下角而非方形光标中心的十字形。

线条由连续的点构成，在像素图中也是一样。我们要通过逐点绘制的方法绘制精确的斜线和曲线。

像素的限制要求我们在绘制曲线时精确控制每一个点的位置。将像素图放大可发现，图中的斜线是直线交错排列组成的，而每一根直线的长度决定了它们最终组成的斜线的角度和曲线的弧度。直线越长，它们构成的弧线的弧度就越缓和，如图2-8和图2-9所示。

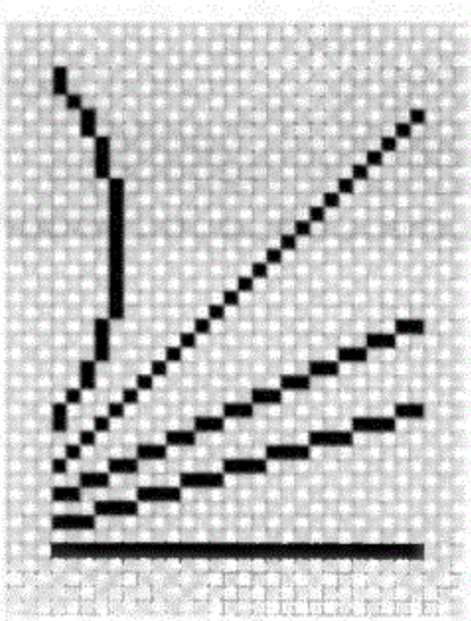


图2-8

图2-9

如果将这些直线的长度缩减到1像素，它们将会组成直角。所以需要注意的是，若每根直线的长度相等，最终将组成两根以某个角度相交的直线而非单根弧线。绘制弧线的原则一般为：中间的直线较长，两头的直线较短，最后再以连续的弧线构成曲线。

提醒：在“编辑”菜单中“首选项”子菜单的“常规”选项卡中，将“图像插值”下拉列表框选择为“邻近（保留硬边缘）”。这样设置后，再使用变换工具对图像进行拉伸和旋转就不会出现自动消除锯齿和渐变的情况。不过为保证精确性，这种方法仅仅作为提高效率的辅助手段，不能代替逐点绘制。

（2）常用线条绘制

熟练掌握一些常用的标准线条的绘制方法是很有必要的。使用不标准的、不符合规范的线条容易使绘制物体的边缘显得粗糙。下面进行逐个介绍，如图 2-10 所示。

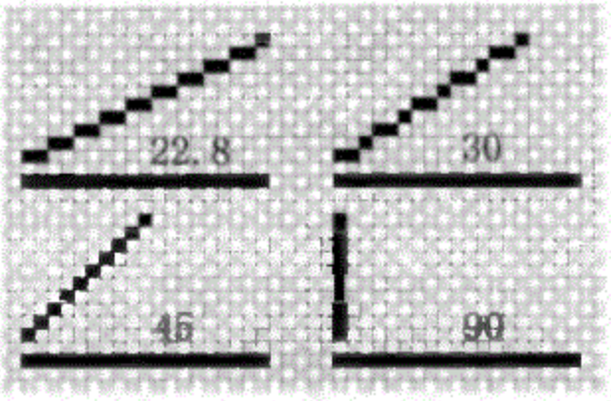


图 2-10

1）绘制 22.8° 的斜线

以两个像素并排的方式斜向排列。绘制这种斜线常用的线条是像素建筑图和像素几何体的标准线条。

2）绘制 30° 斜线

以两个像素间隔一个像素的方式斜向排列，当竖向排列时则形成了 60° 的斜线。绘制这种斜线经常与其他线条配合使用，以便完成一些特殊的造型。

3）绘制 45° 斜线

以一个像素的方式斜向排列。

4）绘制 90° 角

按住 Shift 键可以拖动鼠标绘制出完美直线。

5）绘制各种弧线

参照图 2-9 中的 1/4 圆。

（3）平面图形绘制

在绘制有立体感的图形之前，我们先从简单的平面图形着手练习，如图 2-11 和图 2-12 所示。

注意：不同颜色线条之间的角度关系，图 2-11 中是使用标准线拼凑的三角形。利用这些基本线条能绘制出我们需要的绝大多数元，然而仅仅描绘轮廓再填色是不够的，图 2-13 中绘制的是像素游戏中的道具和各种图标。我们可以发现，几乎每一个元都或

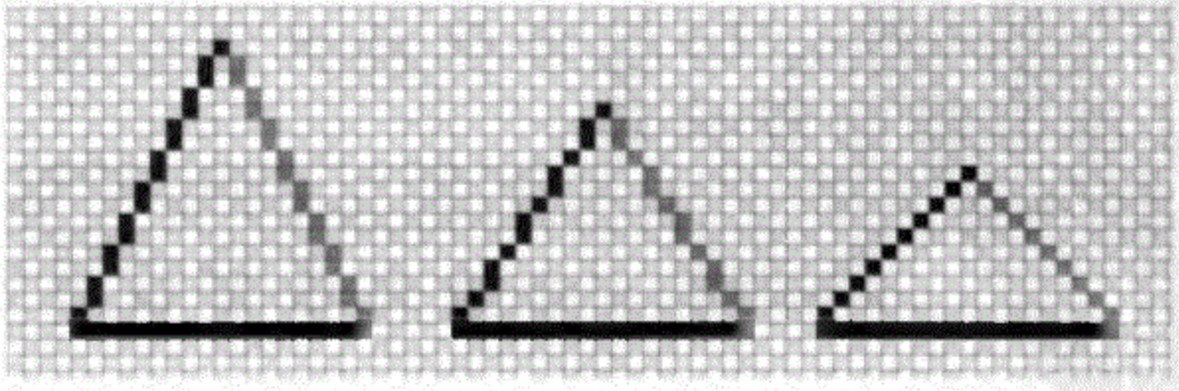


图 2-11

图 2-12

多或少地带有立体感。制作像素图的过程与其说是“绘画”不如说是“塑造”。下面我们就来学习如何使用像素点塑造出立体图形。

（4）立方体的绘制

如何在有限的尺寸内表现出物体的立体感是像素图绘制过程中的难点，也是接下来几个例子重点讲解的内容。我们先从学习绘制基本的立体几何图形——立方体

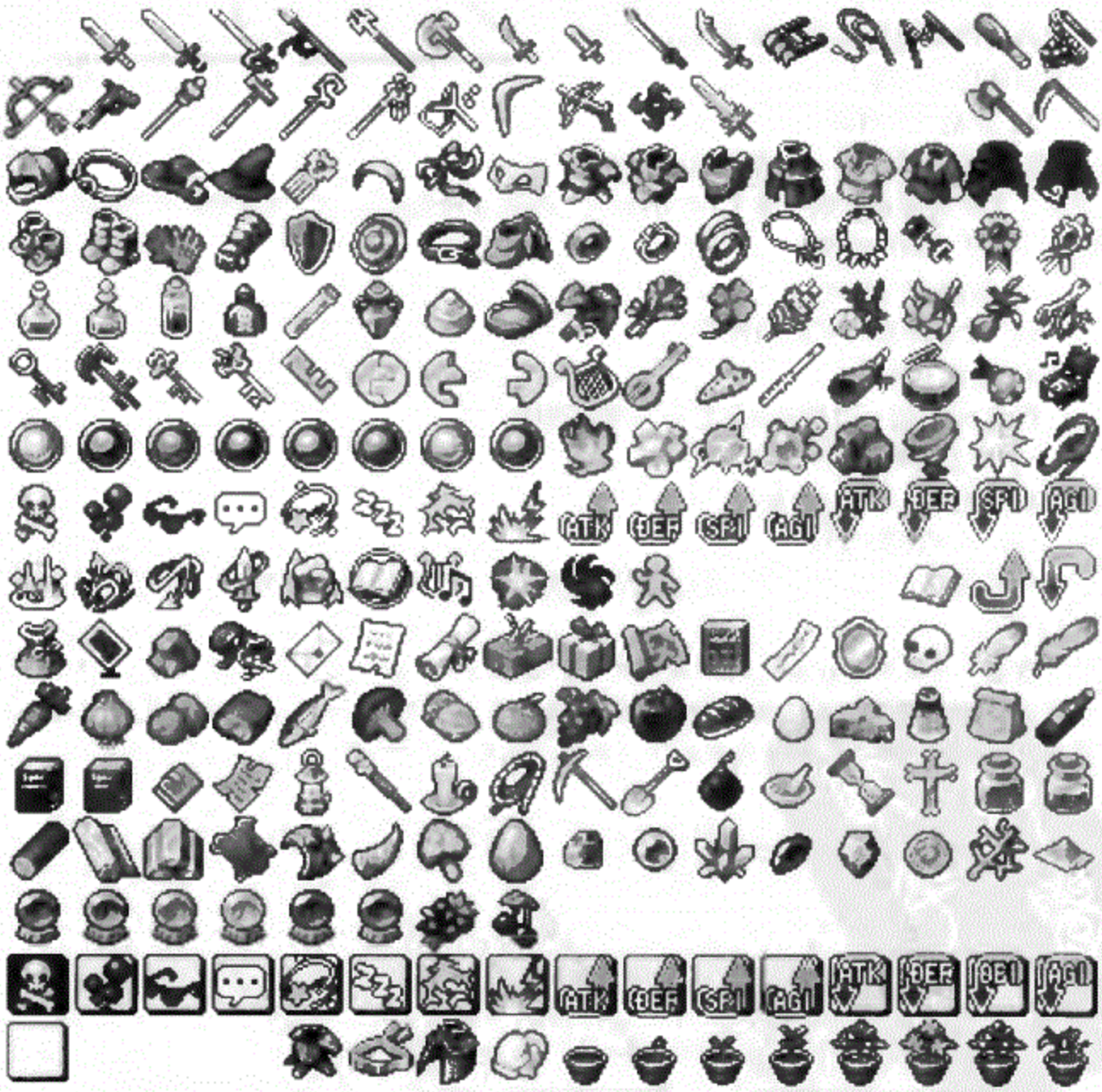


图 2-13

开始，如图 2-14 所示。

首先使用铅笔工具绘制立方体的轮廓。为了提高效率，我们并非逐点绘制立方体的轮廓而是首先绘制立方体的顶面。绘制时使用 22.8° 的斜线。

绘制完四边形的顶面后，按住 Alt 键拖动鼠标进行复制，沿 Y 轴方向复制出一个相同的四边形轮廓。

使用直线工具分别连接两个四边形对应的顶点，立方体的大致轮廓就绘制完成了。

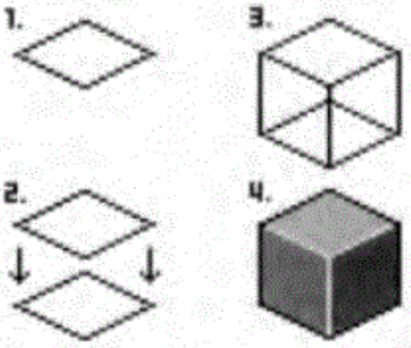


图 2-14

提醒：在复杂图形的绘制过程中，对相似部分复制后进行修改是提高效率的基本技巧，在后文中将大量使用这种方法。为方便进行复制，建议在绘制可能重复利用的部分时，新建图层以方便以后选取和修改。

以块面为单位分出简单明暗关系，使用油漆桶工具和铅笔工具进行大块面的填色，用 3 种颜色分别画出暗部、亮部和高光。更复杂和细致的填色方法将在后面的例子中详细说明。

简单立方体的绘制完成了，尝试使用相同方法绘制以下几何物体，尤其注意球体上明暗的划分，如图 2-15、图 2-16、图 2-17 所示。

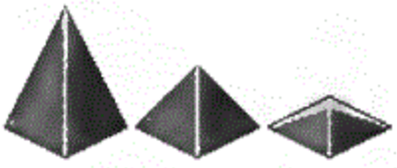


图 2-15

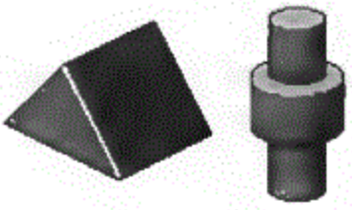


图 2-16

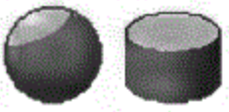


图 2-17

（5）立体字符的绘制

在掌握简单几何图形的绘制方法后，我们使用这种方法来绘制稍复杂的立体字符。选用的例子是反向的英文字母 E，如图 2-18 所示。

图 2-18

使用多边形工具和直线工具或直接用文字工具绘制出字符。需要注意的是，若直接用文字工具输入，字符周围为半透明，这时需要使用颜料桶工具对整个字符进行一次填色。

单击鼠标右键，选择变换工具对字符的角度和透视关系进行调整，这里使用变换工具中的斜切工具——使用前需要将“图像插值”下拉列表框设置为“邻近”（保留硬边缘），

以确保字符角度与场景中其他元一致，之后将字符镂空，使用上例中的方法拖动复制边框并对相应顶点进行连接，这样立体字符的边框就绘制完成了。

在像素图中，尤其是像素游戏制作过程中，不同物件往往是分开制作的，为确保它们的透视关系一致，需要在绘制时将已绘制完成的元导入作为背景进行参考。若由多人同时开始分工制作，可先绘制一个如本例中的立体图形分发给所有成员作为绘制时作品视角的标准。

确定主光源并上色，我们依旧使用以块面为单位的简单上色方法。立体字符的光影分布比简单几何图形复杂，若不确定一个主光源，就无法通过光影准确地塑造出立体感。当一个场景中元数量增多时，要尽量确保它们光源的一致性。透视和光源都要在所有元件绘制前进行计划，一旦开始绘制将很难再进行纠正，如图 2-19 所示。

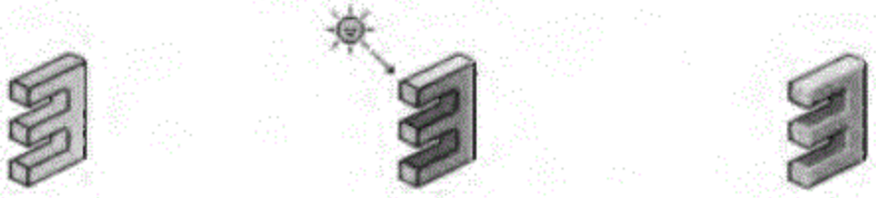


图 2-19

（6）几何体绘制在像素图中的实际应用

我们在前面的学习中介绍了像素中简单立体图形的绘制。几何体的绘制不仅仅是在基础练习中，还可以应用于成品的制作。接下来将综合运用之前学到的几种基本技巧，绘制一间游戏商店的店铺，如图 2-20 所示。

图 2-20

图 2-20 中这栋建筑的基本形状是一个立方体，所以首先可以使用立方体的绘制方法绘制出轮廓，并在这一步确定透视关系和观察角度，因为之后部分的绘制将围绕这个立方体展开，如图 2-21 所示。

绘制 3 个与顶面形状相同但面积递增的边框，直接使用缩放工具会使线条变粗。按图 2-21 中样式叠在建筑主体的立方体上方；在转角处绘制垂直线条，塑造出体形，如图 2-21 所示。

清除多余线条，使用主色调填满作为底色。这样房屋的主体就绘制完成了，与我们之前绘制立方体的过程基本一样，如图 2-21 所示。

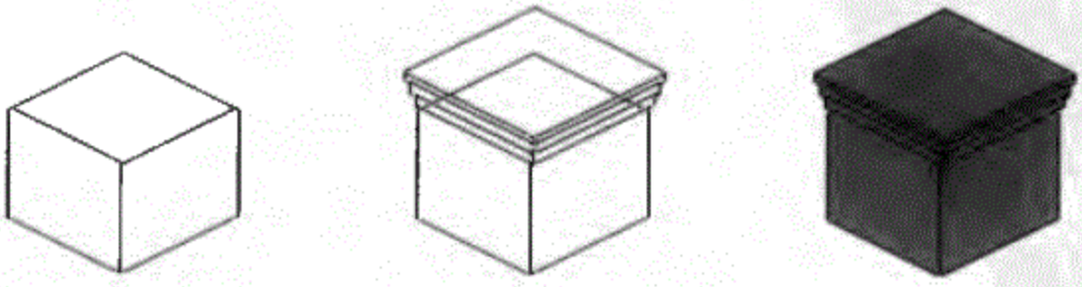


图 2-21

高效率的像素图绘制方法是将复杂的图形分解为多个相对简单部件后进行绘制再组合。这一点与平时一般意义上的绘画有所不同，需要特别注意。

屋顶的凹陷部分的绘制方法比绘制立方体更简单。只需在绘制菱形后将靠上方的两条边向下拖动复制并删除多余部分即可，如图 2-22 所示。

确定主光源并上色，以块面为单位根据光源方向进行填色。以之前的底色作为标准色，在“拾色器”（前景色）中选取两种与之相近的颜色分别绘制亮部与暗部。

使用白色或接近白色的颜色绘制高光，覆盖高光处线条。屋顶处周围黑色线条过于密集，使用灰色或深蓝色线条代替，如图 2-22 所示。

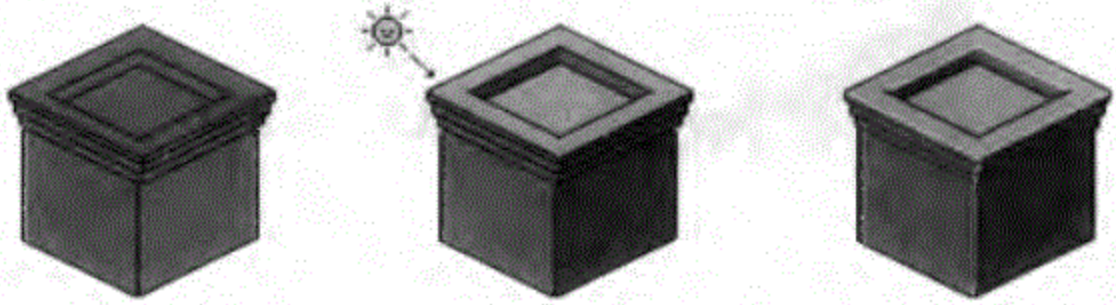


图 2-22

提醒：黑色是像素图绘制过程中要特别注意的颜色。在有限空间内，黑色的线条和点可以直观地描绘出物体的轮廓和结构，如图 2-23 所示。然而在有限的尺寸内，每一个黑点将占用一个像素，意味着在描绘的物体上不得不减少一个彩色的点，而在像素图中，不同颜色对塑造物体的立体感是至关重要的。但是在绘制过程中过多依赖黑色，不但会使画面显得平面、无体积，而且容易使原本就紧凑的画面显得一团黑。正确的绘制方法应主要依靠颜色和光影的变化塑造物体。当描绘结构不得不大量使用黑色线条时，应选择与底色同一色系的深色代替黑色。通过光影塑造物体的技巧将在后续章节中详细介绍。

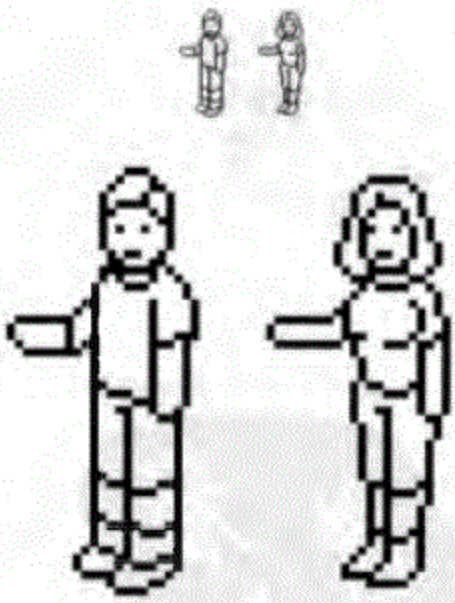


图 2-23

绘制橱窗和门的方法与绘制屋顶凹陷的方法相同。仔细观察图 2-20 可以发现门上分出了明暗并绘制了高光，往往是这样的小细节决定了画面的最终效果。这也是优秀的像素图作品虽简单但显得特别精致的原因。

绘制招牌。这里用到了立体文字字符绘制的技巧。参照以前学习的方法进行，只要注意在使用斜切工具对字符的角度和透视关系进行调整时，使文字底面与商店正面墙体平行；拖动复制边框并对相应顶点进行连接时，字符侧面的底边与商店侧墙底边平行。其实不光字符，绘制任何一个组件时都要检查并与其他组件比较，以确保最终组装时透视关系没有错误。

添加细节。像素图中没有留很大的空间绘制细节，所以要做到小而精，对一些图

形进行简化后绘制。观察图 2-20 中靠左边的橱窗玻璃，可以发现该图作者仅仅用 12 个点就描绘出了 KONAMI（科乐美）的 LOGO，如图 2-24 所示，最左边任天堂标志也隐约可见。

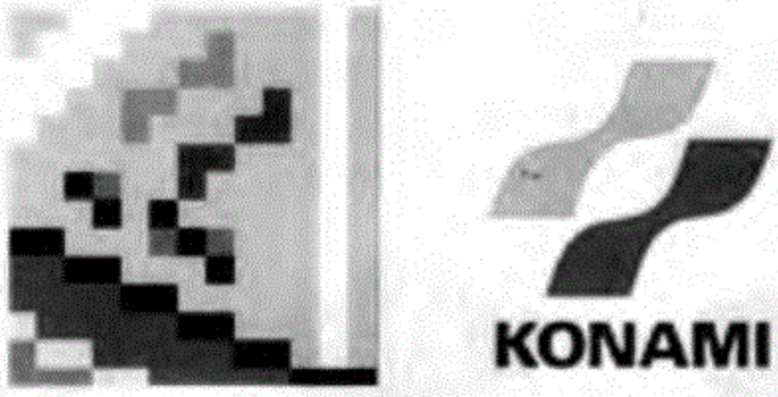


图 2-24

玻璃上的高光可以新建图层后用白色半透明绘制，这样就能透出下面的橱窗装饰，并且使橱窗装饰物在高光处自然变亮，如图 2-25 所示。

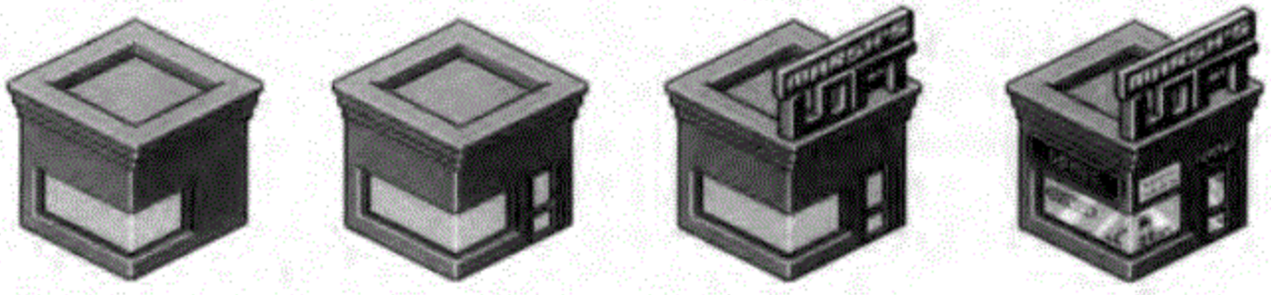


图 2-25

游戏商店绘制完成了。在绘制过程中，这个游戏商店由许多单独绘制部件拼合而成，而这个店铺又将成为整个场景中的一个组件。当许多这样的组件被拼合到一起后将十分壮观，如图 2-26 所示。

（7）光与影

前文我们提到过像素图绘制完全靠明暗表现立体感，可以说对明暗的处理决定了一幅像素图的好坏。这一步也是最考验基本功的部分，绘制时需要大量借用素描和水粉画的技巧和理论。接下去我们将通过学习绘制一个桌球来了解像素图光影效果的表现方法，如图 2-27 所示。

1）底色

明暗的绘制应放在像素图绘制的后期，之前应该先给像素图上好底色。底色将是绘制明暗时的重要参照。

2）初步明暗

确定光源方向后，以底色作为标准色，吸取底色并调整颜色亮度，用两种颜色分别

图 2-26

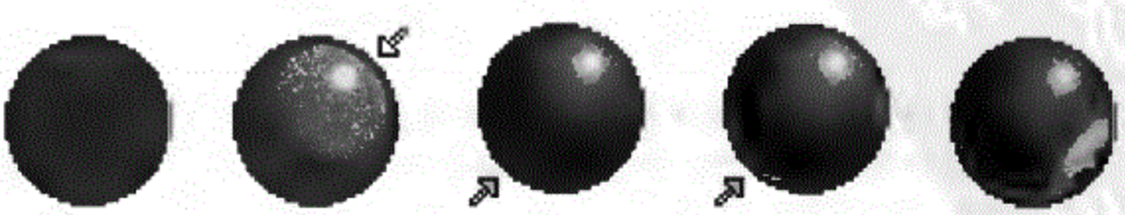


图 2-27

画出亮部与暗部。这步也可以在绘制底色的同时完成。与之前对几何体的简单上色方法相同，只是不再以块面为单位而是对每块面都分出单独的明暗。当然，我们要绘制的球形只有一个面，以高光为圆心向外扩展，然后逐渐加深。

3）过渡色

在明暗的交界处使用中间色画出过渡，在新产生的交界处可多次重复这个步骤以使过渡更自然。但是若过渡次数过于频繁会减小亮部和暗部的对比，使画面显得“平”，起到反效果。

注意：光滑物体除了高光外还有反光。这里没有什么特别的技巧，主要考验绘画基本功。一般像素图以卡通风格为主，反光只会在强调写实风格的像素作品中出现。所以在学习像素图的绘制过程中并不做特别要求。

4）颜色的穿插

绘制明暗时，在不增加新颜色作为过渡色前提下，如何使两种颜色之间的过渡更自然呢？我们通过颜色之间的穿插来实现这一点，如图 2-28 所示。单单使用中间色过渡，各层颜色之间容易有脱离的感觉。若使两种颜色的在交汇处以网状交叉，可以达到与添加中间色相似的效果并且过渡更加自然，如图 2-28 所示。

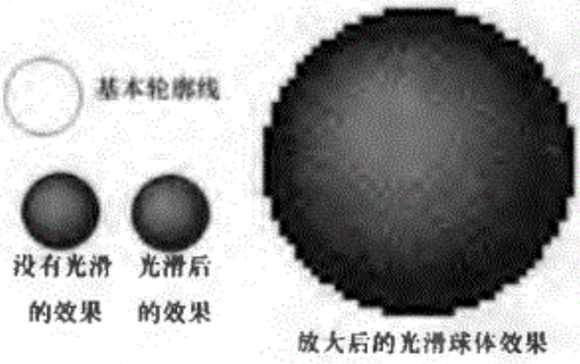


图 2-28

5）深入

为了在有限的尺寸内表现出效果，明暗的对比应当适当地进行夸张处理。实际效果以像素图 100% 大小显示时为准。

在进行明暗绘制时要考虑被绘制物体的材质。绘制金属等光滑物体时，要尽可能加深暗部（不要使用黑色），亮部的最亮处要接近白色或直接使用白色；绘制布料等物体时则可用较浅的阴影表现出布料的纹理。

6）添加细节

绘制桌球上的图案与数字。建议图案与球体分开图层进行绘制。在图案上绘制出光影，也可以尝试调整图层透明度以透出球体上原本已经绘制好的阴影。

拥有复杂光影效果的桌球绘制完成了。在实际绘制过程中并非光影越复杂越好，当物件很多时，过分复杂的光影可能会使画面显得混乱。添加光影时还要考虑到物体的质地等因素，切忌机械地死扣明暗——这一点和画素描是一样的。

2. 绘制技巧

以绘制像素人物为例。在绘制像素图前一般会先绘制尺寸较大的草图进行造型的设计。但是，因为受最终成品的尺寸限制，像素图往往无法完全还原出原设定。这就要求我们在绘制像素图前对人物的造型进行简化，卡通化是一个很好的选择，如图 2-29 所示。

（1）卡通化例子一

参看图 2-30 中作为 RPG 游戏《最终幻想 6》的主角之一的 Tina，图片右上角为游戏场景中人物的一个特殊动作（惊讶）。我们总结出的经验如下。

图 2-29

图 2-30

1）颜色

卡通更容易用色块进行表现。我们在绘制时要尽量使用饱和度高的颜色绘制人物各部分的底色。

2）造型

Q 版可以在缩小尺寸的情况下表现更多的细节，所以像素图人物大多绘制成 Q 版造型。即使不想使用 Q 版造型的“大头风格”也不宜超过 4~5 个头身的比例。使用 Q 版造型的最大好处是由于头比例的增大，可以绘制面部细节。“大头”的缺点是很难表现动作，一般用于 RPG 行走图。4~5 个头身造型则在制作硬派风格游戏和动作游戏时使用。更大头身比的人物只在强调纪实风格的游戏时使用，对绘制的要求较高并且很难表现细节，本书不作推荐。

3）细节

对比原画，细节的流失是不可避免的。我们要做的就是有针对性地保留和放大某些细节，抓住原画的特征进行像素图的绘制。保留太多琐碎的细节，会显得凌乱和没有主次。

（2）卡通化例子二

根据以上几点，我们再来分析《最终幻想 7》中的主角克劳德（PS 游戏中的造型）的像素化造型，如图 2-31 所示。很明显，尽管 Q 版后流失了大量细节，但我们仍然很容易就能认出该人物。观察后可以发现简化后的造型很好地把握并继承了人物的重要特征（一般集中在上半身），如头发、肩甲、手套等。

图 2-31

同样，缺点也同样明显，过分使用黑色使得人物缺少细节，如光影、布料纹理等。更严重的是，在静态的图片上看上去可能还不错，一旦动起来就会发现无法分辨人物的手脚所处的位置。结果是我们无法看到人物做的动作，只能看到一团蠕动的并且可疑的黑色物体。

前文的头身限制是针对主角与一般角色而言，对于大体积的 BOSS 型角色，因为尺寸足够大，我们完全可以参照原画的头身比来绘制造型。关于这一点可以参照早期《最终幻想》（《FINAL FANTASY》），如图 2-32 所示。其中 6 代《FINAL FANTASY Ⅵ》（简

图 2-32

称 FF 6）无论剧情与画面都是像素 RPG 时代的巅峰。在没有 3D 处理能力的游戏主机 SFC 上，FF 6 在游戏中大量使用像素呈现出假 3D 效果。

（3）分层绘制

在之前的章节中我们已经学会了像素图重要的基本绘制技巧。但是本书中的内容仅仅作为一种参考，在实际绘制过程中可能会发现更有效率或更方便的绘制方法，譬如可以尝试在像素图制作过程中使用滤镜。本段的内容是讲述如何使用图层功能快速制作像素人物。

之前的章节我们一直使用 Adobe Photoshop CS3 来进行教学，而在下文中会用到 Adobe Fireworks CS3。Fireworks CS3 在进行分层绘制时选取和管理图层效率更高。当然，本段内容依旧适用于 Photoshop，不过在后文提到连续动作的绘制时会重点使用 Fireworks。

注意：Fireworks 的默认保存格式为 PNG 文件（即图片导入游戏引擎时使用的格式）。Fireworks 能识别 PSD 文件的图层，但是 Photoshop 无法识别出 PNG 文件的图层。

（4）分层绘制例子一：飞行器

我们先从一个简单例子开始，如图 2-33 所示，这是飞行射击游戏中的一个 BOSS，左上角为由 3 个图层绘制完成的完成图，其他 3 个为拆开的图层。因为除了位移外不需

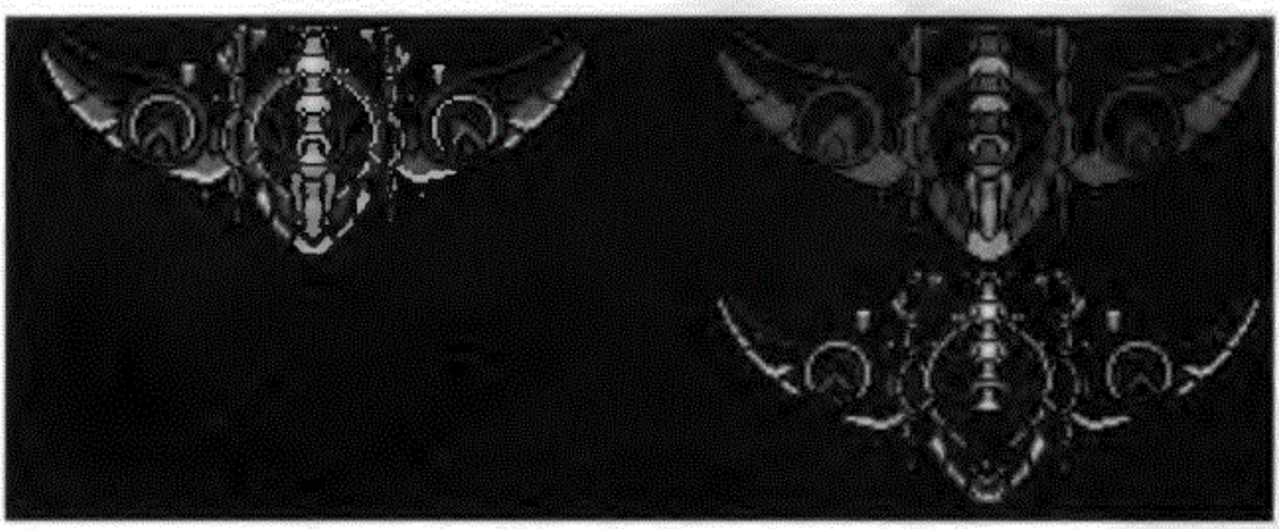


图 2-33

要做活动，所以采用了比较标准的“线条轮廓 + 填充颜色”的绘制方法。不过这个例子的明暗与底色是分图层制作的，将花纹、标识、明暗等繁琐并且需要经常改动的部分放在独立的图层更容易修改，尤其是使用橡皮工具进行擦除的时候。至于具体对哪些部分进行分层与个人绘制习惯以及绘制前的计划有关，需要绘图者自己摸索和把握。

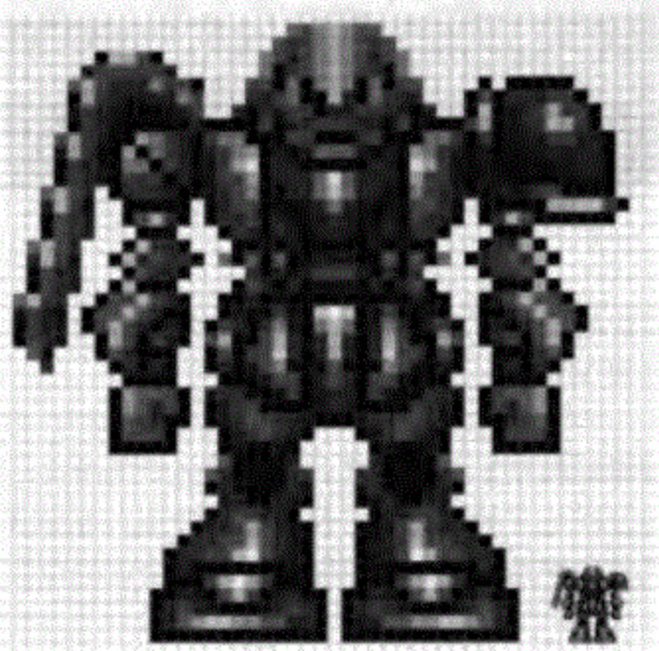


图 2-34

观察这个例子左下方的纯线条部分，会发现内容非常复杂。这就是“一体化”绘制方法的弊端，虽然只绘制了必要的部分，但实际却花费了很多时间。

更有效的方法是在绘制线条阶段就分层为机翼、机翼上的炮台、机身、驾驶舱等部分，进行分别制作。下一个稍复杂的例子将详细介绍这种方法。

（5）分层绘制例子二：机器人正面行走图（如图 2-34 所示）的详细制作流程

4 个方向行走图分为正方向行走图和斜方向行走图两种，实例中演示的是正方向行走图的绘制。经典 2D 日式 RPG 游戏采用的就是这种移动方式。

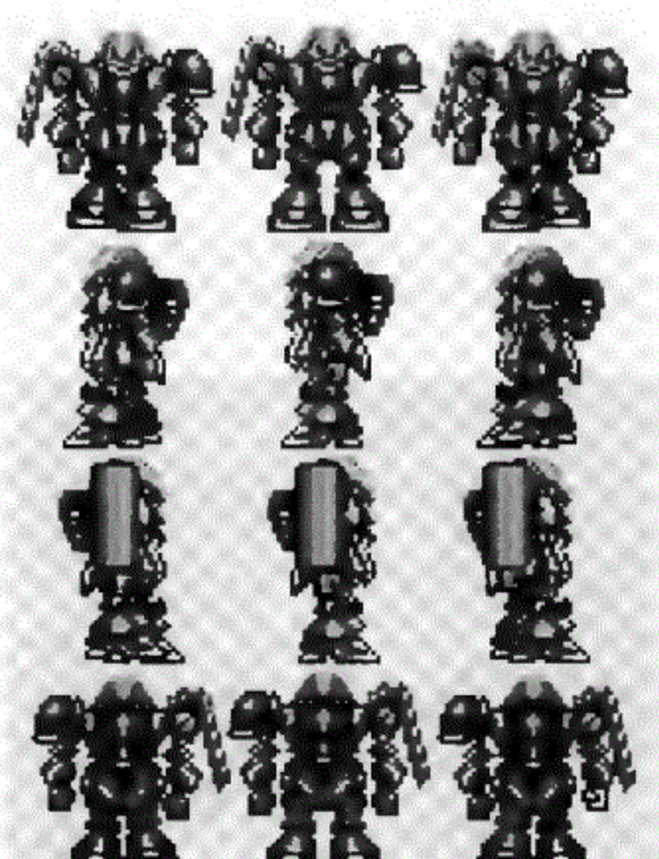


图 2-35

我们要绘制的机器人行走图包括正面、侧面和背面。因为该机器人左、右不对称，所以要绘制两种侧面，如图 2-35 所示。

其中每一面都包括一张站立图和跨左脚、跨右脚。当以“站立——跨左脚——站立——跨右脚——站立……”的规律进行播放时就合成了行走的动画。

下面的例子以学习绘制技巧为主，只绘制机器人的正面行走图。同时这个例子也作为 Fireworks 入门的基础教学。

① 首先绘制草图，大小不限。只需要画出大致轮廓，不必画出细节，也不必上色。如果是简单图形也可以省略这步。

② 打开 Fireworks CS3，新建文件，大小设为 48 × 48。

③ 在“视图”菜单中的“网格”里选择“显示网格”命令，再在“编辑网格”中将网格调整到合适大小，这里设为 1 × 1。

④ 单击“文件”→“导入”命令，导入预先绘制草图或参考图片。在“属性”面板左边输入宽和高，调整草图大小，这里都输入“48”；再用工具栏中的“缩放工具”微调至合适大小。因为考虑到机器人的活动空间被限定在 48 × 48 像素内，正面站立图中的机器人要离边缘有 1~2 像素的剩余空间。动作幅度越大，留出的剩余空间也要适当加大。

⑤ 在层面板中选中草图或参考图所在的层，命名为“参照”，并调整不透明度为 30%。锁定该层，如图 2-36 所示。

⑥ 大致按可动关节将机器人分为数块，这里分为“躯干和头部”、“左臂”、“右臂”、“左腿”和“右腿”这 5 大块。我们将逐一绘制这些部分，也可以先绘制整个机器人再进行切割。这里我们主要使用前一种方法，图 2-37 中为已经绘制完成的机器人拆开后的状态。

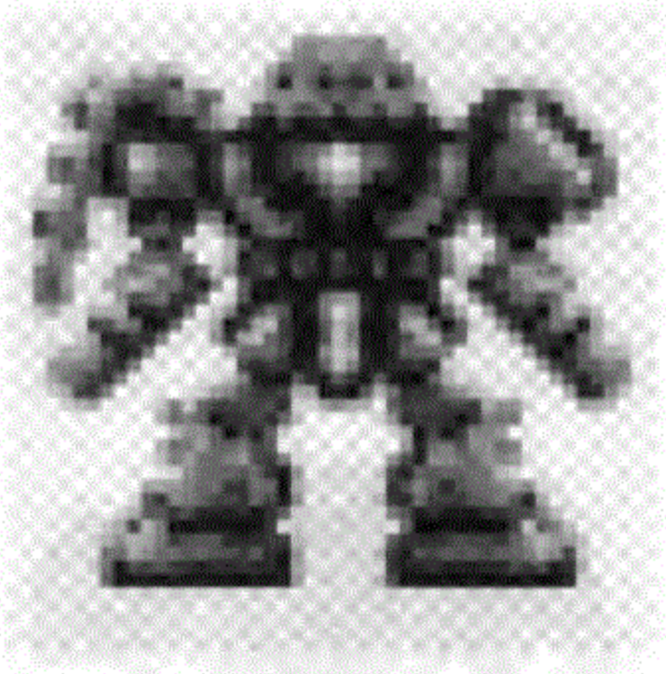


图 2-36

⑦ 在“层”面板中单击“新建 / 重制层”按钮。为区分方便将新建层命名为“躯干”，我们将在这层上绘制机器人的躯干和头部。

⑧ 在工作区域单击鼠标右键，在弹出菜单中的“缩放比率”里将视图调整到合适大小以方便我们绘制，这里设为 800%。

⑨ 使用工具栏中的刷子工具，在“属性”面板中调整笔刷的设置，将笔触调整为没有柔化效果的硬直线。

⑩ 参照底下的草图，用 1 像素的黑色绘制轮廓，用油漆桶工具和刷子工具填上底色。眼睛部分会在以后单独绘制。

对称的部分，可以先绘制其中一边，用“选取框”工具选取后再用右键菜单中的“编辑”菜单中的“复制”和“粘贴”命令或按住 Alt 键拖动鼠标复制，用右

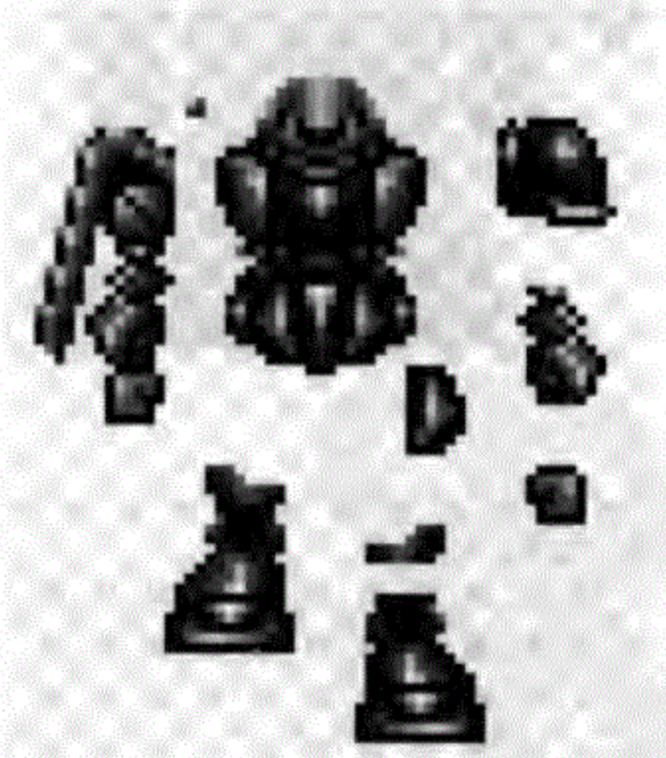


图 2-37

键菜单中的“变形”命令进行水平翻转或垂直翻转。最后选中图层中的位图并在图层面板右上角的“选项”下拉菜单中选择“平面化选项”或“向下合并”命令来合并图层中的位图。

使用数位板能加快工作速度。轮廓和形状以实际效果为准，不必完全照搬草图。必要时要进行一些简化，但过分简化会降低细节感使像素图色块化。绘制过程中要经常用右键菜单中的“缩放比率”命令将视图调小观察效果。

对要擦除的部分，在属性面板中将橡皮擦工具的“边缘”设为0，再进行擦除，或使用工具栏中的选取框工具，配合键盘上的Shift键和Alt键选取要擦除部分，再按Delete键删除选区。

绘制完躯干和头部的轮廓并填上底色后，新建层，并把“躯干”图层锁定。用以上方法逐层绘制“左臂”、“左腿”，并对图层进行相应命名。分别将“左臂”、“左腿”所在的层拖动到“新建/重制层”按钮上复制层，将复制出的“左臂”和“左腿”层里的位图进行水平翻转并移动到相应位置，对肩膀处不对称的部分进行重新绘制和加工。将复制出的层分别重命名为“右臂”和“右腿”。对绘制完成的层进行锁定。

肢体连接处和被躯干遮住的部分也要进行绘制，在调整动作时这些部分会外露。如果是先绘制整个机器人再进行切割，并将切割开的部分放置于各个图层中，则要在切割后把这些部分追加上去。这样可以避免调整动作时反复修改关节连接处。这样一个没有明暗效果的机器人正面站立图就绘制完成了，如图2-38所示。

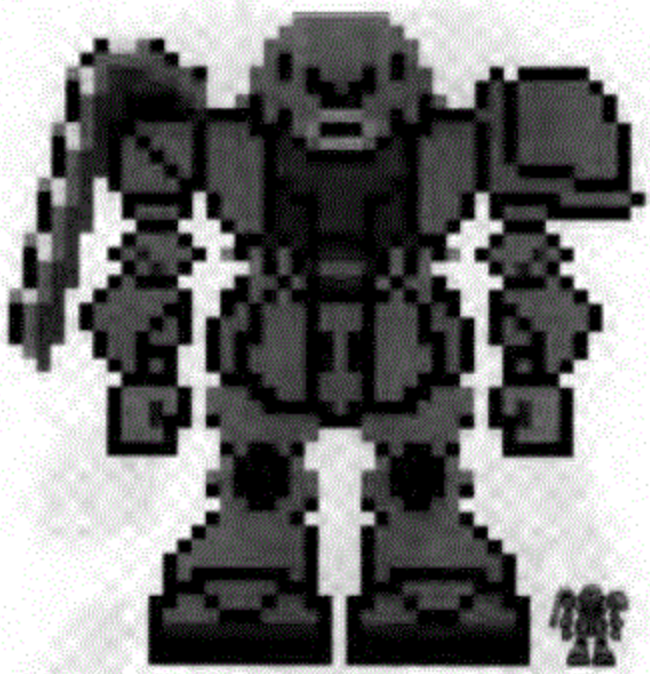


图2-38

绘制明暗效果。将“躯干”层解锁，单击层面板中的“新建位图图像”按钮在“躯干”层里新建一个位图。该位图位于原位图的上方。

使用工具栏中的滴管工具吸取躯干上的底色，单击工具栏上选取的颜色，在弹出的色板上单击“系统颜色吸取器”。在“颜色”窗口中调整颜色的亮度，用这些颜色绘制阴影和高光。用滴管工具可以从机器人身上吸取已使用过的颜色，如图2-39所示。

绘制明暗的手法与用铅笔画素描是相同的，用明暗塑造出图像的立体感。把一部分之前黑线绘制的轮廓用不同亮度的底色覆盖，融入明暗。因为明暗是绘制在新的位图图层内，所以不会影响到之前画的底色和轮廓，可以方便地进行修改。

对明暗效果修改至满意后，使用层面板选项中的“向下合并”功能将明暗位图合并到原底色所在的位图。

在“左臂”、“右臂”、“左腿”、“右腿”各层里新建位图，直接从绘制完成的“躯干”

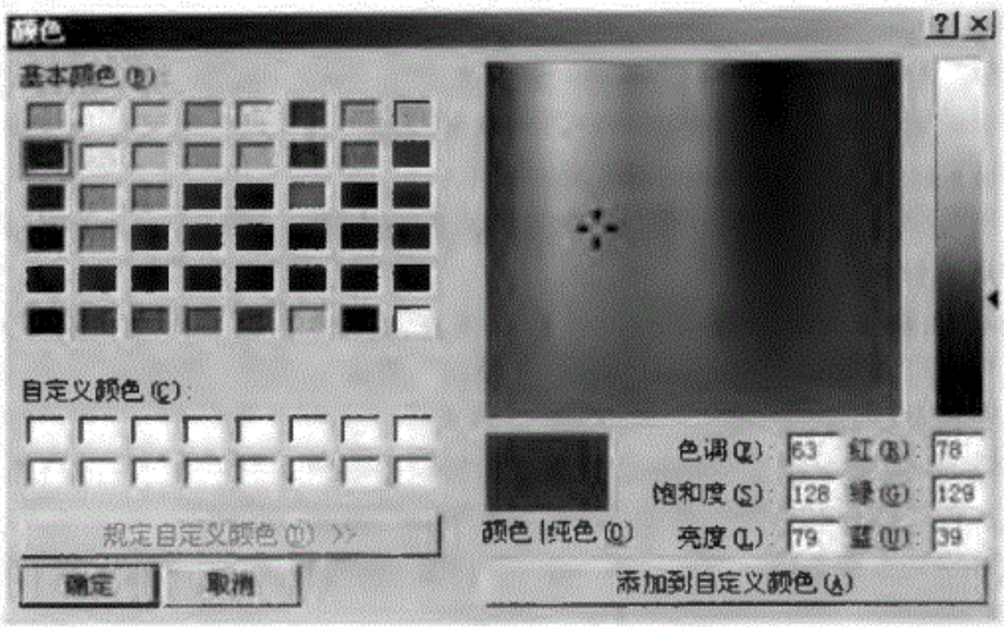


图 2-39

上吸取颜色来绘制这些部分的明暗。完成后将明暗向下合并到底色上。对绘制完成的层进行锁定。

将“左臂”层解锁，用选取框工具选中机器人的手，用右键菜单中的“通过剪切新建位图”命令将机器人的手剪切到新的位图中。用此方法将“左臂”与“右臂”各分为“手”、“臂”、“肩”三个位图。将“左腿”与“右腿”各分为“大腿”和“小腿”两个位图。

最后在“躯干”层新建位图，绘制机器人的单眼摄像头。这样机器人正面站立图就绘制完成了。

（6）分层绘制例子三：机器人正面行走图

这个例子本来应当放在连续动作的绘制的章节中。不过考虑到例子本身十分简单，而且为了能够加深同学对 Fireworks 的了解，特将这个例子提前。我们继续刚才绘制完成的机器人正面站立图，将其制作成正面行走图。

因为之前绘制的站立图我们已将机器人分解为多个部分，所以我们可以通过调整这些部分的位置很方便地制作出“跨左脚”和“跨右脚”行走图，如图 2-40 所示。

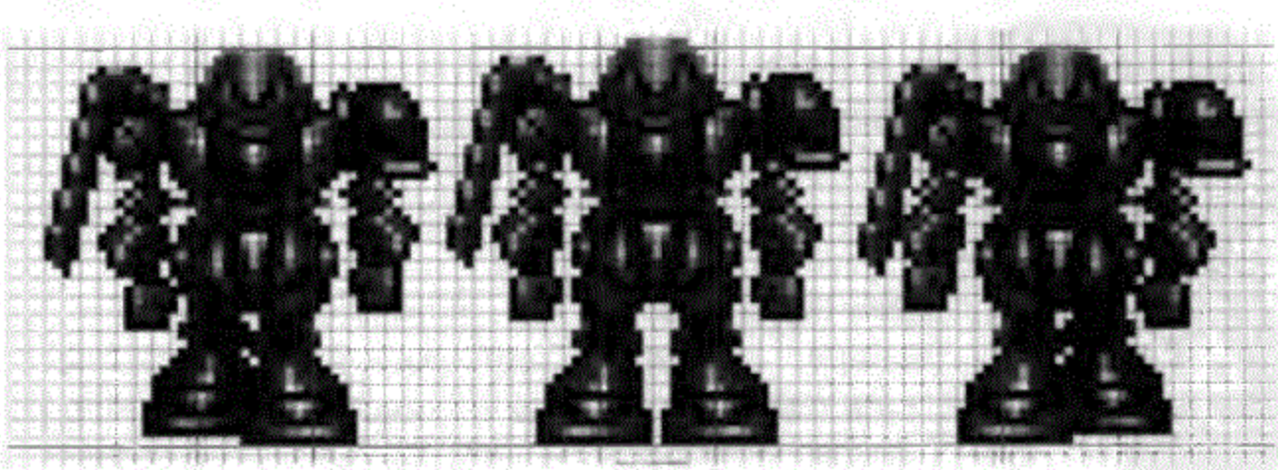


图 2-40

打开之前绘制的机器人正面站立图，选中“帧”面板（如果没有，通过单击“窗口”→“帧”命令，将其打开），将当前帧重命名为“中”。如果“帧”面板与“层”面板在同一面板组中，按住面板组中“帧”面板的标签，将“帧”面板拖动出该面板组并组合到其他面板组或单独成为一个新的面板组。

将“中”帧拖动到“帧”面板下方的“新建/重制帧”按钮上，以复制该帧。将复制出的帧重命名为“左”。

调整帧面板左边的“洋葱皮”工具，或直接点选面板左下角调整“洋葱皮”。Fireworks 中的“洋葱皮”与 Flash 中的“洋葱皮”类似，使用“洋葱皮”可以查看当前所选帧之前和之后的帧的内容。不需要时，用同样方法将“洋葱皮”工具关闭。

按住键盘 Alt 键并点选层的锁定按钮，将所有层解锁。用“指针”工具在工作区中选择机器人各个部分的位图，用键盘上的方向键对位置进行微调。

一般来说站立图最高，“跨左脚”或“跨右脚”时身体下沉，肩膀和手臂相应调低。对应透视把腿缩短，向后摆的腿和手进一步缩短。

根据前后关系调整“层”和“位图”的顺序。调整机器人的重心，将两条腿靠拢作出前后效果，调整手臂各部分位置作出摆动效果。

选中机器人较靠后的那条腿，单击“属性”面板上的“添加动态滤镜或选择预设”按钮，单击“调整颜色”→“亮度/对比度”命令，在相应的对话框中调低亮度。拉开两条腿明暗上的差距。亮度调整完后，如果对效果不满意可以单击下方的“编辑并排列效果”选择相应滤镜进行再一次调整。

在“中”和“左”两帧中切换参看动作是否自然，作出相应调整和修改。

对关节处和细节进行修改，使机器人不同部分之间连接更自然。如果是制作生物的行走图，要对所有关节附近进行修改使关节更平滑、动作更舒展。绘制机器人则不要对关节进行太多修改，在突出机械的僵硬感的同时使机器人动作尽可能舒展。

绘制完成“左”帧后。按住 Ctrl 键单击“中”帧和“左”帧，将这两帧拖动到“帧”面板下方的“新建/重制帧”按钮上，复制这两个帧。将帧拖动调整到“中”—“左”—“中”—“左”的顺序。

将最后一帧“左”帧重命名为“右”。选中该帧，在工作区用“指针”工具框选所有位图，使用右键菜单中的“变形”→“水平翻转”命令对图像进行翻转。

打开“洋葱皮”工具，或单击“视图”→“标尺”命令并从标尺中拖出辅助线。使用键盘方向键对水平翻转的图像进行位置调整。因为机器人行走时身体在 X 轴上没有位移，所以将翻转的图像移动至躯干部分的 X 轴方向与前几帧重合即可。

因为机器人左右不对称，按住 Shift 键用指针工具在工作区选中所有不对称部分，使用右键菜单中的“变形”→“水平翻转”命令对这些位图再次进行翻转，将它们调整至合适位置。

在“中”和“右”两帧中切换参看动作是否自然，作出相应调整和修改。

单击“文件”→“导出向导”→“继续”→“继续”命令，进入“图像预览”窗口，在“选项”选项卡中可以查看绘制机器人行走图所使用的颜色并可以限定导出图像的颜色；在“文件”选项卡中可以限定导出图像的大小；在“动画”选项卡中可以调整每一帧的持续时间，这在 2D 游戏引擎中会用到。

在“图像预览”窗口中单击播放按钮查看机器人行走图的实际效果，在“动画”选项卡中调整每一帧的持续时间。这里我们选中第一帧按住 Shift 键并单击最后一帧以选中所有帧。在“帧延时”文本框中输入“20”，将每帧的持续时间设为 0.2 秒。若发现动作过快并且没有节奏感，则选中第二帧按住 Ctrl 键并单击最后一帧以同时选中这两帧，在“帧延时”文本框中输入 30，将这两帧的持续时间设为 0.3 秒。检查效果并继续调试。

动作调整完毕后单击“确定”按钮，关闭“图像预览”窗口。根据动作调整时遇到的问题，根据情况修改部分帧中的内容。

重复上面步骤，直至对效果完全满意为止。这样机器人的正面行走图就绘制完成了。

（7）分层绘制例子四：横版动作游戏机器人杂兵

RPG 行走图的动作幅度较小，所以下面我们再以这个横版动作游戏中的机器人杂兵为例进行正面行走图的绘制，如图 2-41 所示。绘制的具体方法已经在上面的详细绘制方法中提及了，下面只介绍要点。这个杂兵将会在以后制作连续动作的例子时继续使用。大小限定在 60×60 像素内。

首先在绘制前对绘制对象进行分析，因为该机器人为人形，所以可像对人类那样按躯干、头部和四肢进行简单拆分；再根据具体情况对上下半身、肩膀和手臂、大小腿等进行细分；然后将这些部件各放入一个层进行绘制，如图 2-42 所示。

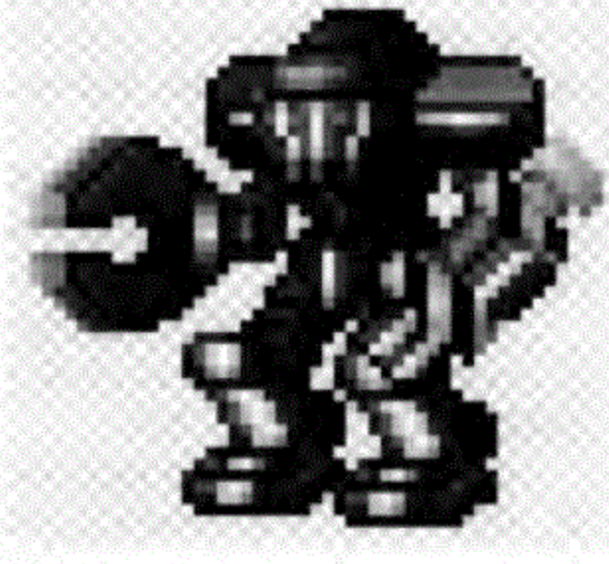


图 2-41

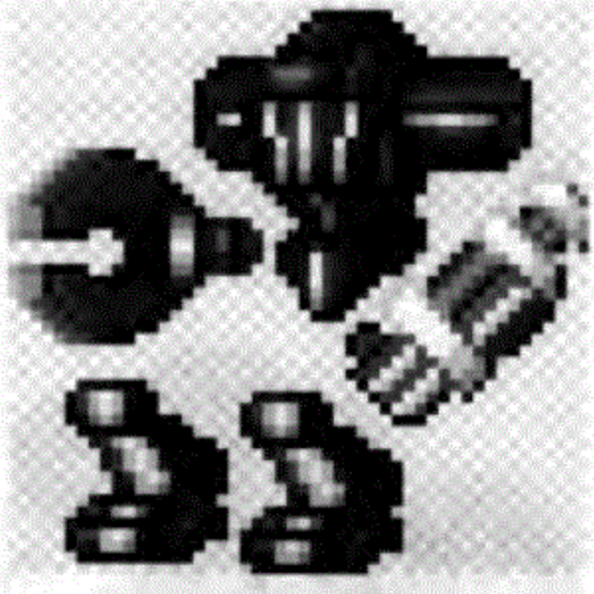


图 2-42

因为是从局部开始绘制，比较难把握整体，所以事先准备好框定人物的大小以及将简单草图置于底层作为参考都是有必要的。

如果使用 Fireworks，请时刻保持除了当前绘制的层外所有层都要处于锁定状态（按住 Alt 键再单击“锁”按钮为锁定所有层）。

使用 4 个层分别绘制躯干、腰和两个肩膀，如图 2-43 所示。将它们组合后观察效果，如图 2-44 所示。注意绘制要完整，互相透叠的部分也要进行绘制，这对以后制作动

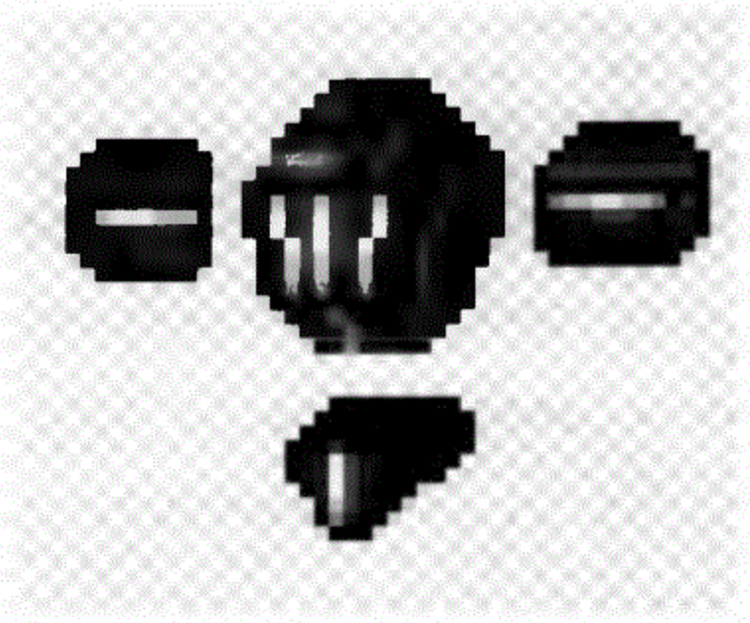


图 2-43

图 2-44

作很重要。颜色方面进行简单上色就行，详细上色和细节绘制放在所有组件都拼合完成以后。

机器人身上的条纹状花纹可以在层中新建“位图图像”后绘制，Fireworks 软件中的层相当于 Photoshop 中的组，位图图像相当于 Photoshop 图层组中的图层。Fireworks 新建层后里面自带一个位图图像，下文提到的“层”或“图层”一律指 Fireworks 中的层。

注意：虽然大小限定在 60×60 像素中，但画布应略大以方便作业。仅依靠辅助线框定一个 60×60 像素的范围即可。

用同样方法绘制手臂和脚，因为部分手臂和大腿暂时空缺，若需要确定位置可以粗略画一个图样来代替，如图 2-45 所示。

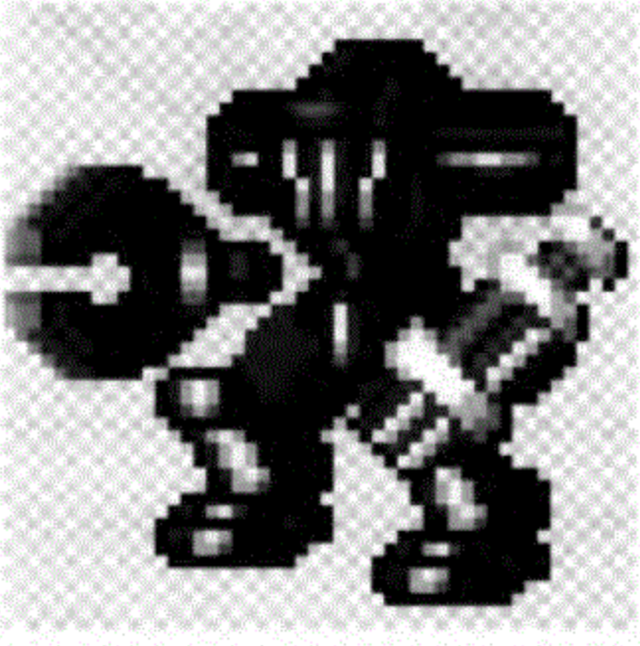


图 2-45

需要指出的是，两只脚是相同的所以可以复制获得，但是必须放在不同图层或同一层中不同的位图图像里。

在各个层中为新建位图图像添加细节，比如在机器人左臂的机械拳上添加黄黑相间的条纹，如图 2-46 和图 2-47 所示。

大腿的绘制方法比较特殊，因为大腿构造是按节排列的机械软管，所以为以制作尺寸与大腿粗细相当的椭圆形，上色并加上简单明暗后进行复制（会自动生成新位图）。每条大腿由 3 个椭圆形拼合而成，如图 2-48 和图 2-49 所示。这样就不必分别绘制两条不同的

图 2-46

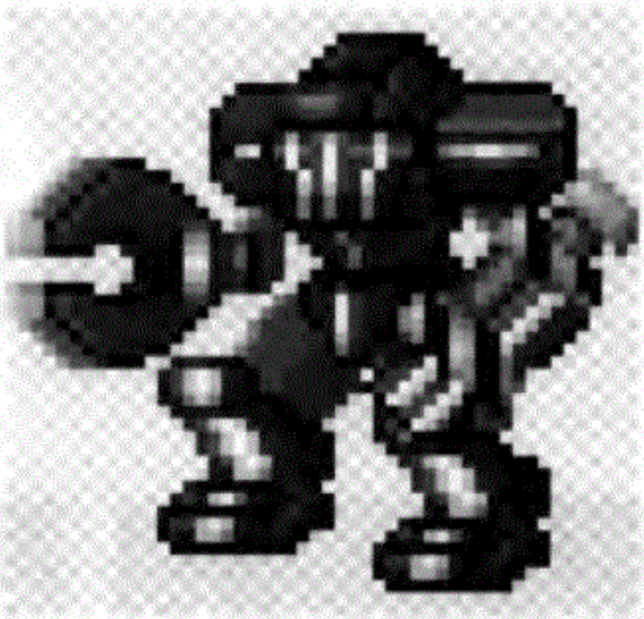


图 2-47

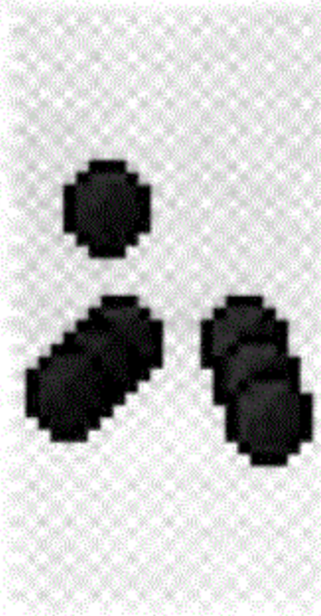


图 2-48

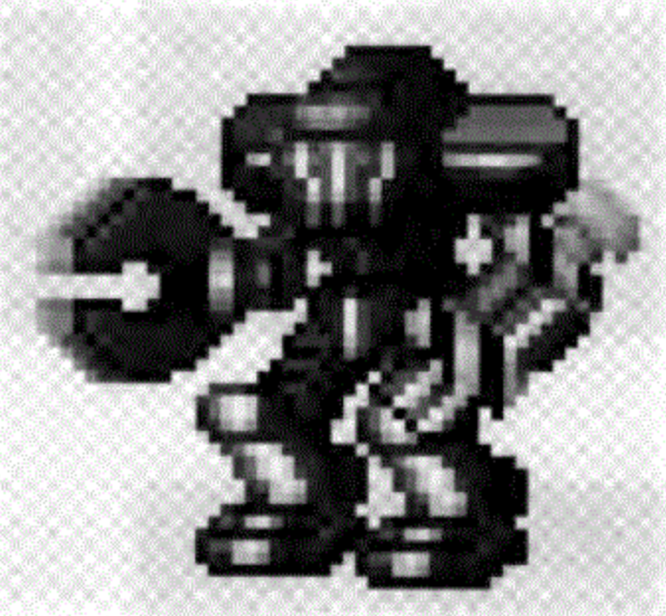


图 2-49

大腿了。

在之后绘制动态效果时，这种每节都可以单独活动的大腿将带来更加直观的便利。

最后再根据之前学习的基础知识对颜色和阴影进行一些修改和深入，这样这个机器人就基本完成了。

注意：机器人站立图绘制完成后一定要保留绘制时的层，这些层是我们以后制作连续动作的关键。

（8）像素游戏场景绘制技巧

像素场景相对于前面学的人物绘制要容易一些，很明显的一点就是尺寸方面要宽裕

很多，制图者可以放心地添加细节而不必做太多的简化。同样因为尺寸比较大的关系，在绘制时可以用滤镜、渐变等工具进行加工，不过使用这些工具后依然要进行一定程度手工修改以强化细节。下面通过绘制一个 RPG 俯视游戏场景来学习一些场景绘制关键点和技巧。

1）什么是场景

首先要明确什么是像素游戏中的场景。简单来说，场景包括地图（即地面）和物件两部分。地图和物件具体的区分根据不同游戏引擎的限制而有所不同。以这个工厂场景来做例子，如图 2-50 所示，一般来说各种筒和罐子是物件，去掉物件后剩下的就是地图，如第二张图，如图 2-51 所示。

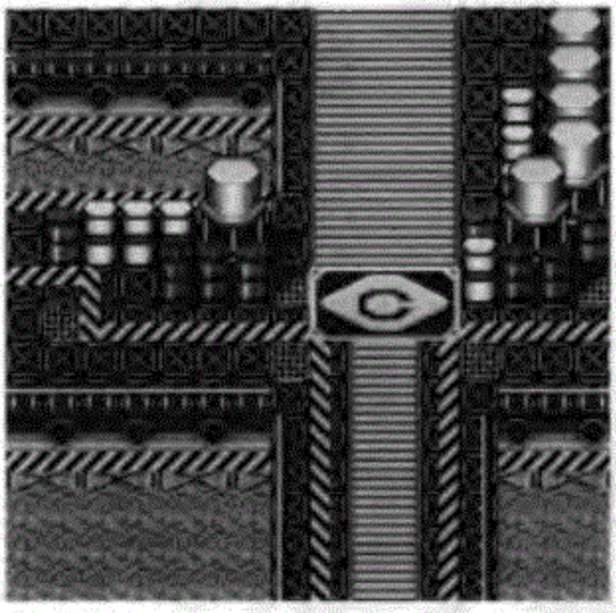


图 2-50

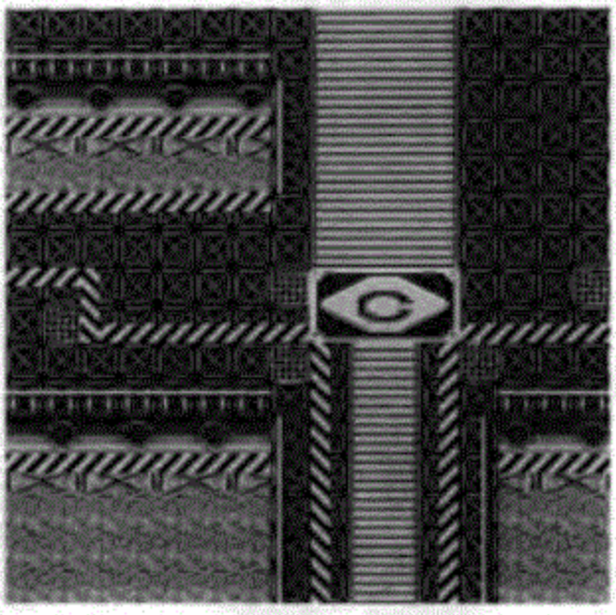


图 2-51

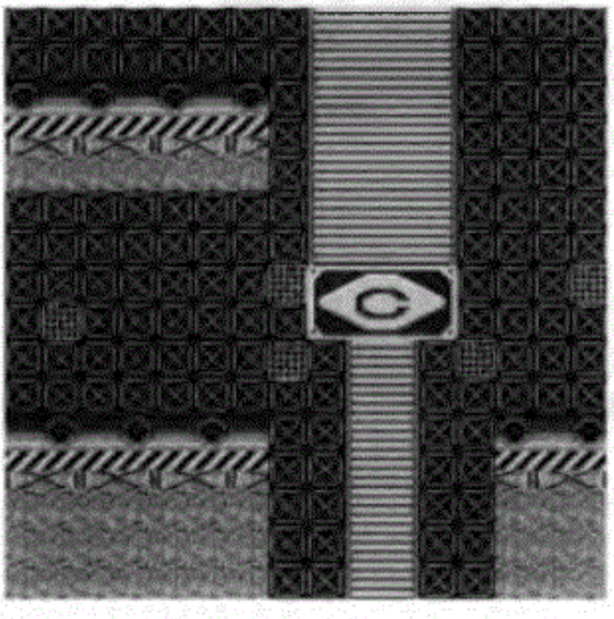


图 2-52

而 RPG Maker 等引擎支持多层的地图，这样就能将栏杆和地面标示物等也划入物件的范畴。去掉物件后的地图如第三张图，如图 2-52 所示。RPG Maker 因为地图分多层、组合的随意性大，并且可以直接在事件层里添加碰撞效果，所以程序里其实并没有地图和无的区别。

另一个例子是游戏引擎，出于节省游戏容量、提高游戏平台运行效率等因素的考虑，可能将整个第一张（如图 2-50 所示）图上的内容都算做地图。这种情况下虽然能通过设置碰撞来设置人物的行走区域，但制作过程尤其是元件的划分会困难很多。

开始制作地图前之所以要了解这些，

是因为地图和元与人物不同，导入引擎前先要对它们进行切割和划分。

2）场景分割原理

玩过 FC 或者 GB 游戏的玩家一定会发现这些游戏的场景明显是由许多连续并且重复的方形区域紧密排列而形成的，如图 2-53 和图 2-54 所示。

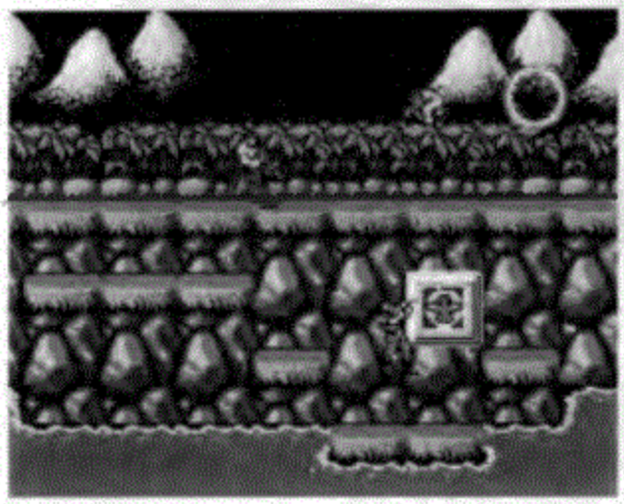


图 2-53

图 2-54

当我们去除那些重复元素后会发现一个场景其实是由有限的几种方形图案组合而成的。这些等大的方形图案被系统调用后拼凑出游戏画面。这种情况一般在越是古老的游戏主机上越是明显，因为游戏容量和色彩限制了这些元素的多样性，如图 2-55 和图 2-56 所示。

我们将前一个例子中如图 2-51 所示的第二幅地图切割并挑出不重复的部分（这里假设引擎支持元件的水平翻转和垂直翻转），原本由 16×16 个方形组成的地图马上就浓缩成了 3×6 个方形元件，如图 2-57 所示。换言之，我们只需要绘制这 18 个元件就能在游戏引擎里拼凑出绘制区域十几倍、几十倍甚至更大的场景地图。

RPG Maker 等软件并不区分地图与物件，若要在游戏中加入筒和罐子，并使之可以与地板等切割排列在一张图上，某些引擎则需要将地图与物件分别导入引擎。地图元件在切割排列时一般以地板元件为最小切割单位。物件最小单位一般同地图最小单位。所有地图元素是等大的，大小超出范围的物件就要被切割为多个部分，比如《超级玛丽》中的下水管就是根据下面地板的大小正好分为 4 块，通过复制底下两块就能无限增加下水管的长度。

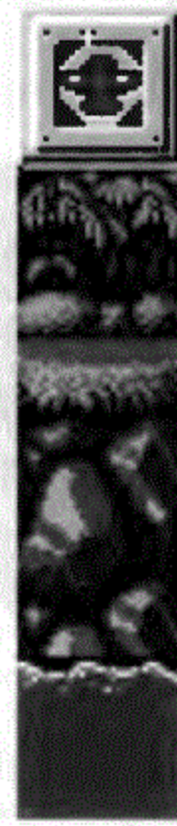


图 2-55

图 2-56

3）元件功能的应用

知道了基本原理后，我们就能有针对性地绘制场景

图 2-57

了。虽然最后的成品要进行切割，但我们绘制时依旧要绘制一块包含所有元素并带一些重复的完整的区域，如前一个例子中的工厂场景。制图时，首先用网格功能将画布分割成预定的最小单位——单块地板的大小，然后绘制一块基本地板，将其复制并铺满画布上所有区域。

当我们要修改时如何处理重复区域的问题呢？同动画软件 Flash 一样，Fireworks 也有元件功能。将绘制的第一块方形地板位图转换成元件，然后从库面板中拖到画布上进行复制和排列，这些元件在选中状态左下角会出现“快捷方式”的标记，如图 2-58 所示。当要修改时，只需修改库中的元件，画布上所有相同部分都会一起发生改变，所以即使使用 Photoshop 进行最初的绘制（例如要使用到 Photoshop 丰富的滤镜），在最后的拼接和修改时还是建议使用 Fireworks（CS3 支持 PSD 文档）。

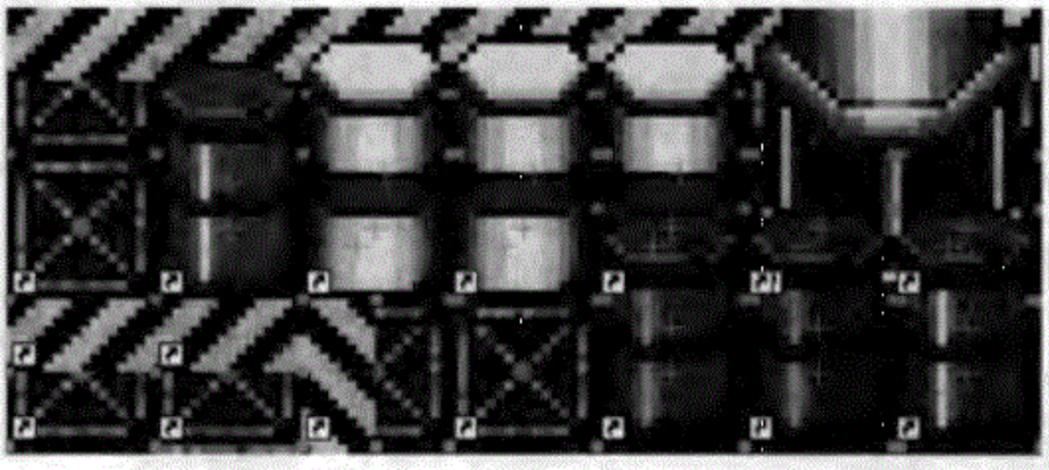


图 2-58

4）绘制物件

配合地板大小，绘制各种物件和装饰，例如油桶和栏杆。绘制的每种东西都单独放在一个层中并在该层进行复制，以便于以后的修改。最初只要绘制到基本明暗阶段即可，首先要确定的是所有的东西组合到一起时大小和比例是否协调，如图 2-59 所示。

5）添加细节

当所有物品的造型都确定后，逐个进行深入加工。要注意整体统一，如光源等。因为东西比较多，所以除了要重点突出的物体外都应该调低对比度与明度，背景也不宜过分花哨。如果有已经制作完的人物，尝试将人物置于场景中，如果无法突出人物，就要对人物或场景进行适当修改。

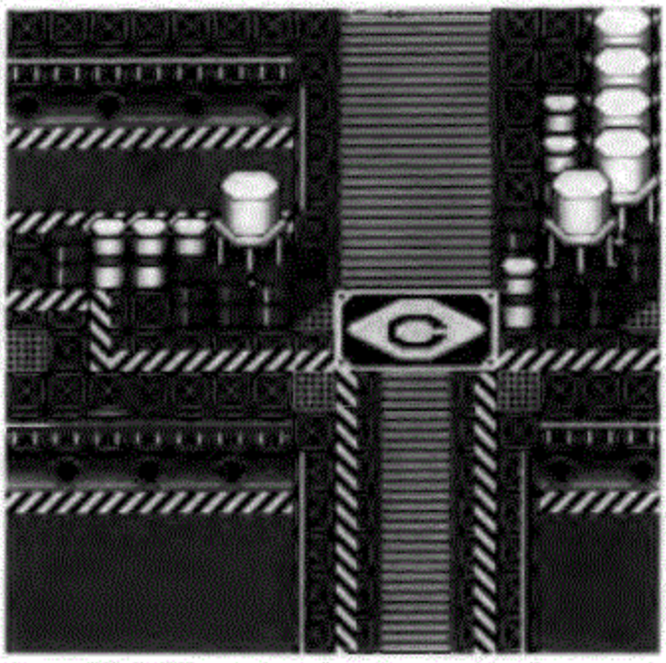


图 2-59

绘制完成的部分可以尝试添加“杂点”滤镜以增加质感、细节感和厚重感。不过一旦添加此滤镜后颜色分布会变得很复杂，将很难再做手动的修改，所以一定要在绘制完成后再进行。

6）场景分割的实际操作

场景绘制完成后进行切割。先将简等物件进行隐藏，只显示地图部分。按照地板大小或按照先前设置的网格大小将不重复的部分挑选出来并在新建画布上进行整齐排列。进行这个步骤前先要了解引擎是否支持水平和垂直翻转，若支持，所有相互对称的元件只需留下一个即可。

注意：如果完全按照本文方法采用 Fireworks 绘制，则只需将画布上所有内容删除，再从库中重新将每个元件添加到画布上进行排列即可。同样要先搞清楚引擎是否支持翻转。

（9）像素图绘制的其他技巧

1）四方连续绘制技巧

绘制地面时总是要求地板可以四方连续。之前例子中我们使用的是 Fireworks 中的元件功能，其实使用 Photoshop 的位移滤镜也能很容易地绘制四方连续图案。

将画布设为方形，大小正好等于单块地面元件的大小，这里设为 10×10 像素，绘制地面元件，如图 2-60 所示。绘制完成后单击“滤镜”→“其他”→“位移”命令，弹出相应的对话框，在“水平”文本框中输入画布边长的一半，在这个例子中是 5。在“未定义区域”中选择“折回”。

位移后画布上的内容如图 2-61 所示。可以在画布中央修改图案的边缘使其能在水平方向连续。修改完成后再次打开位移滤镜，将水平改回 0。

用同样方法修改垂直方向。

2）自制笔触

在 Photoshop 中把常用的线条笔画以及基础图形做成笔刷，就算使用逐点绘制也不用总是一个个点出来，大大地提高了工作效率。

新建一个 PS 文档，选取铅笔工具并选择 1 像素的笔刷，绘制线条或者基础图形。选择“编辑”→“定义画笔”命令，此时刚才绘制的线条或者基础图形就会出现在画笔控制面板中，一个笔刷就做完了。当自制了一定数量的笔刷后，把原有的笔刷删除，然

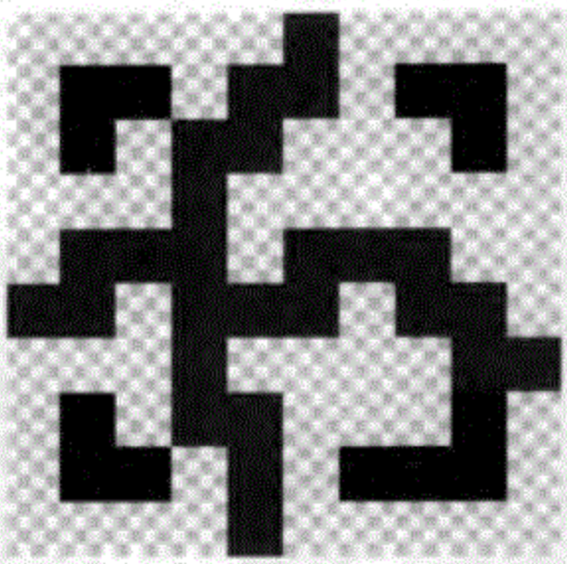


图 2-60

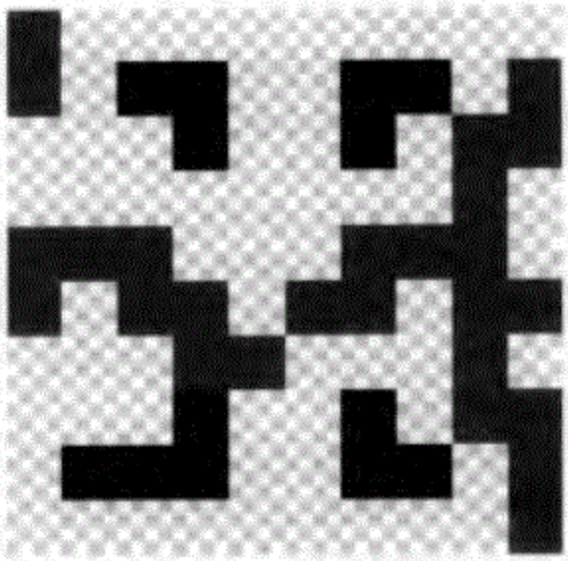


图 2-61

后单击画笔控制面板右上角上的下三角按钮，选择“存储画笔”生成自制笔刷文档（.abr），以后就可以根据需要随时载入画笔或者增添新的笔刷。

3）无损缩小

虽然之前说过，对图像进行缩小会使图像质量细节直线下降。但有一种使像素图在 Photoshop 中无损缩小的方法，只要在图像大小的对话框中，重定义像素选项中选用“邻近”插值的计算方式就可以。在这种情况下，所有插值计算出来的颜色和现有图中的像素图的颜色保持一致，并不会添加过渡色（像素图的颜色其实是被限定在 256 色内的，也就是索引图）。

用此方法能避免画面缩小后变得模糊，但细节依然会不可避免地损失，所以实事上这并不是真正地对像素图无损缩小方式，依然需要手工一个点一个点地进行修改。此方法在一些特定场合如导入草图并缩小时具有实用价值，若通过此方法绘制整幅像素图，那么需要修改的内容将是非常多的，效率并不一定比逐点绘制高。

4）假组

这个技巧其实我们在之前的场景制作中已经运用到了。将所有已经绘制完成或正在绘制中的人物、物件甚至草稿都导入同一 PNG 文件中，将 PNG 大小设定为游戏画面分辨率，然后将导入的物件都放入场景，以模拟实际游戏效果并对其进行分析，再根据结果进行修改。这样，就不用等到导入引擎后发现了问题却无法修改而后悔了，如图 2-62、图 2-63、图 2-64 所示。

3. 连续动作的绘制

在前面的章节中我们已经掌握了静态像素人物的绘制技巧。那么如何让这些人物在游戏中动起来呢（如图 2-65 所示）？这就需要我们要逐帧绘制这些人物的动作，如图 2-66 所示。听上去似乎是一个很枯燥的过程，但只要方法得当，不但能大大加快进度而且会让过程变得乐趣无穷。下面我们就来详细介绍连续动作的绘制技巧，主要使用的软件为 Fireworks。

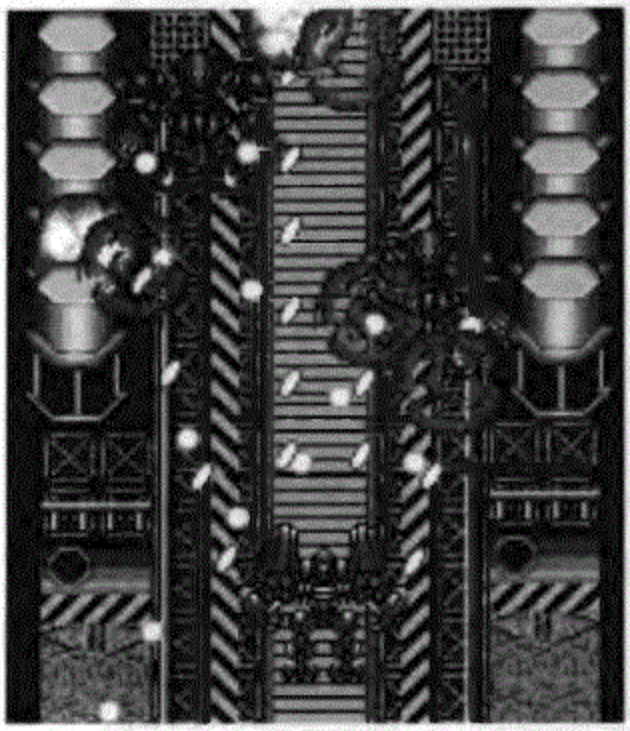


图 2-62

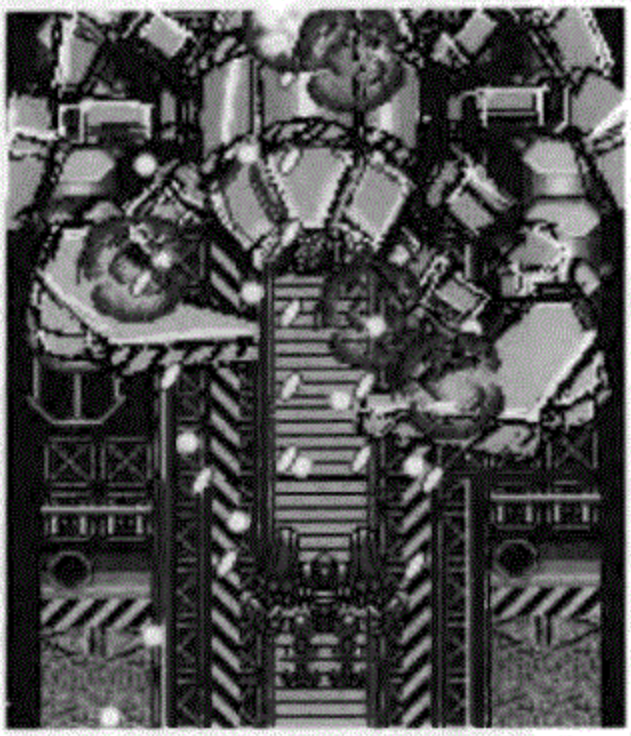


图 2-63

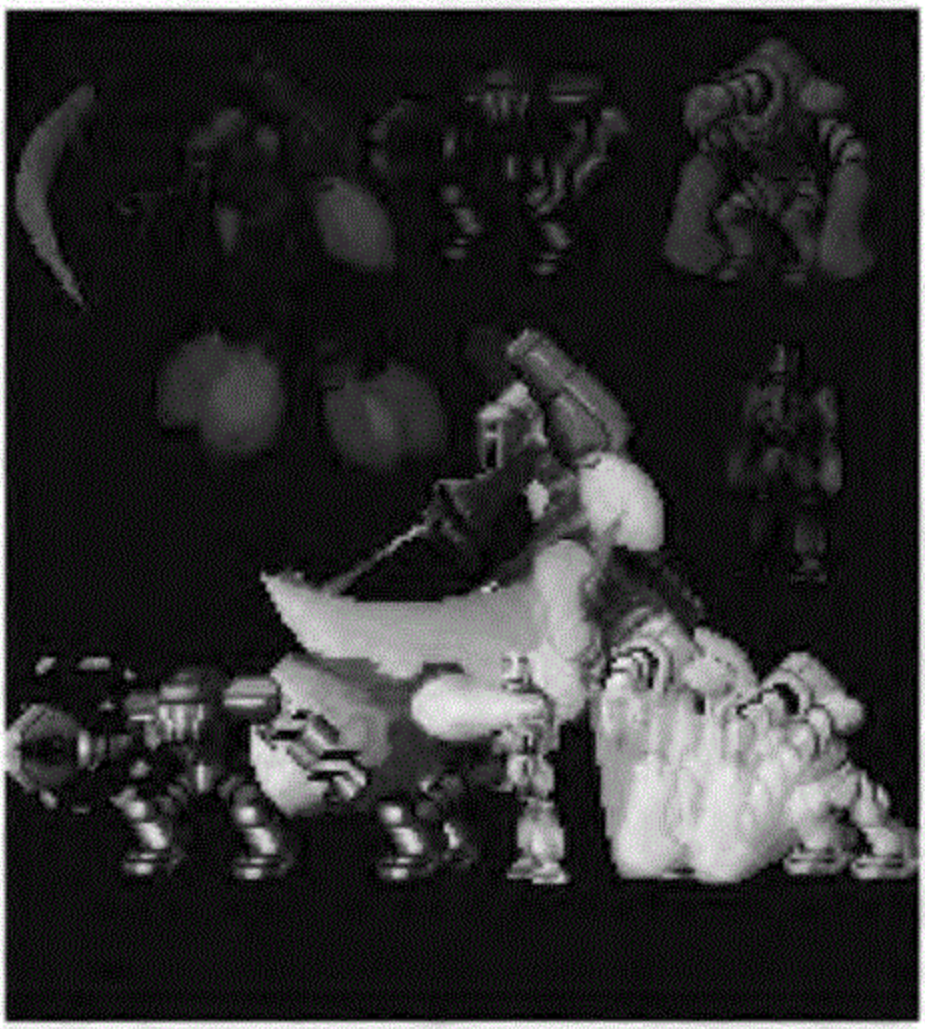


图 2-64

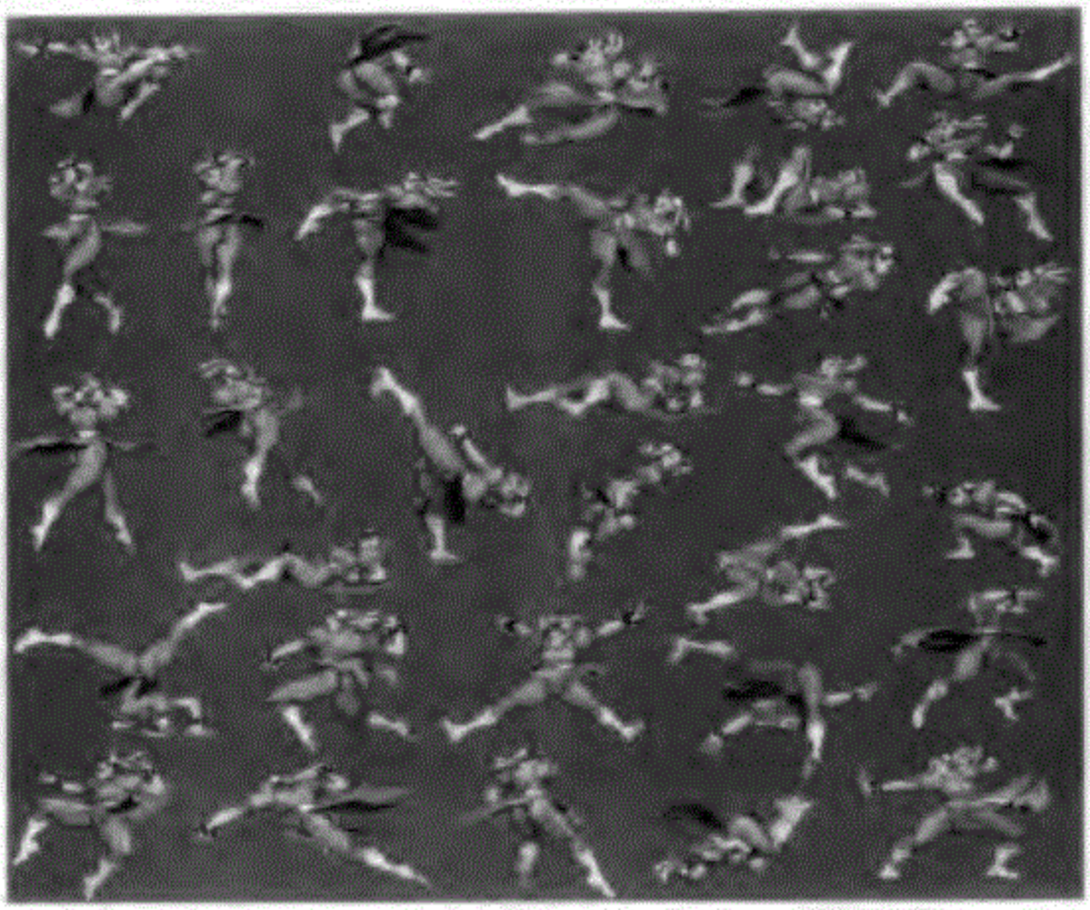


图 2-65

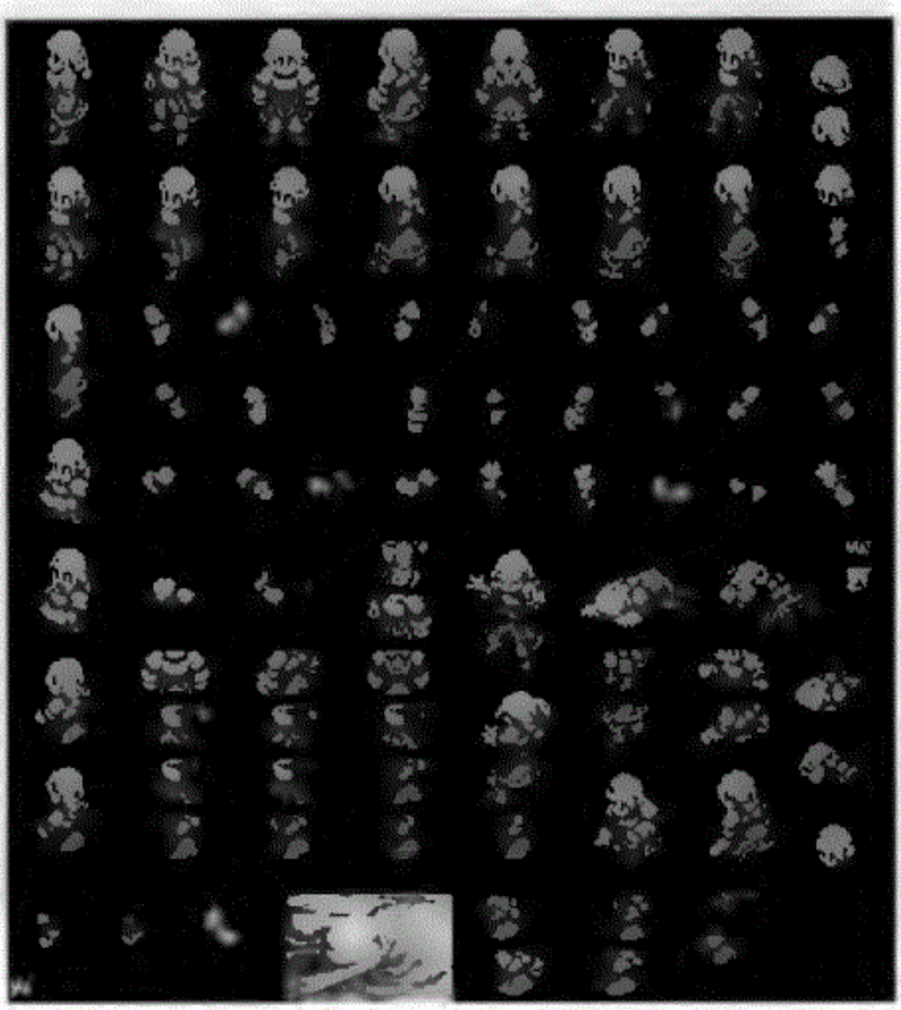


图 2-66

（1）动作绘制

在绘制人物动作之前，先要绘制好一副人物站立图，即人物的基本站立动作。现在我们将之前“1.2 绘制技巧 -2. 分层绘制 - 例子二”中绘制的机器人杂兵作为基本站立动作，如图 2-49 所示，以它为基础绘制一系列连续动作，如图 2-67 所示。

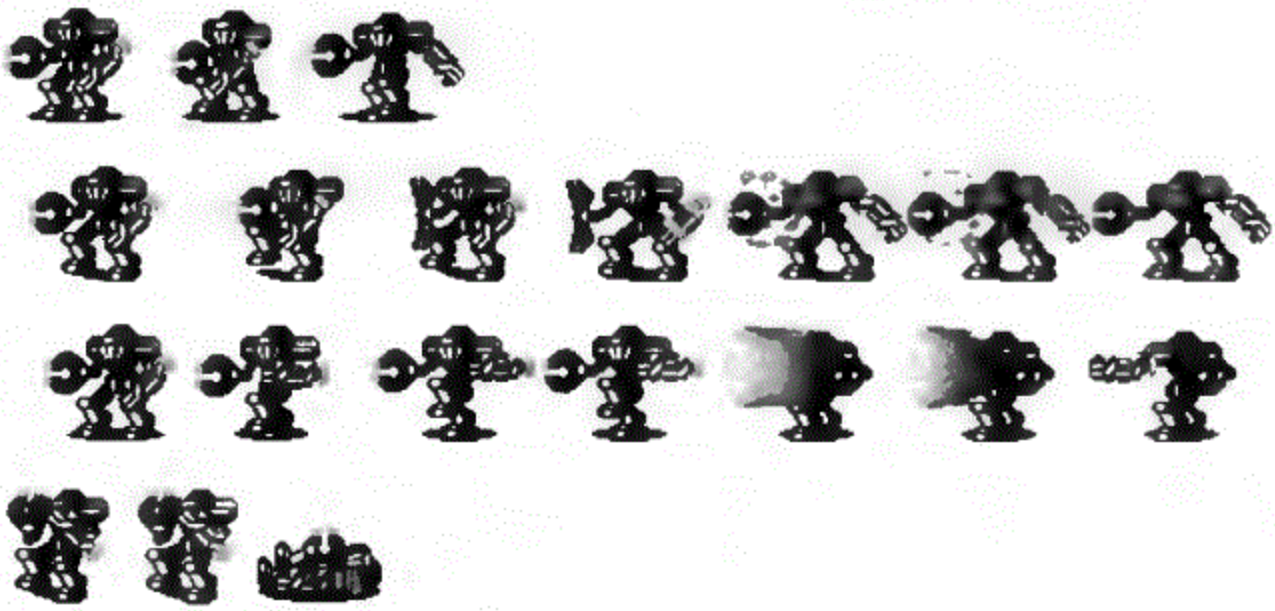


图 2-67

注意：之前绘制的静态机器人站立图一定要保留绘制时的层，这些层是我们以后制作动作的关键。

在绘制之前先要确定要绘制的动作有哪些。一般横版过关游戏角色有移动、攻击、跳跃、死亡等基本动作。我们现在观察图 2-67 中的杂兵机器人，如图 2-67 所示，可以看到它有 4 套动作：行走、攻击动作 1、攻击动作 2、被攻击 / 死亡，所以我们分 4 次绘制。首先从最基本的“行走”开始。

复制并打开之前绘制的机器人杂兵文件（PNG），在帧面板中拖动当前帧到新建按钮上进行复制。选中靠上方的层（图 2-68 所示的帧 1），如图 2-68 所示，框选工作区中事先使用分层绘制法绘制好的机器人，可以看到机器人由许多位图图像组成，如图 2-69 所示，每个位图图像外以蓝框显示。

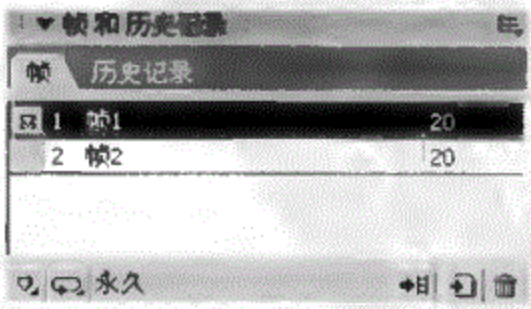


图 2-68

解锁帧 1 中所有层（按住 Alt 键再单击锁图标）。在工作区中拖动各位图，摆出机器人抬右腿的姿势，如图 2-70 所示。注意它着地的左脚不移动垂直方向的位置。图中绿色辅助线即“地面”，注意该机器人左右脚对应不同高度的“地面”以模拟透视关系。移动位图调整机器人动作时可以打开帧面板中的“洋葱皮”工具（与 Flash 中的“洋葱皮”工具类似），或者直接通过在两帧中切换来保证两帧动作的连续性。

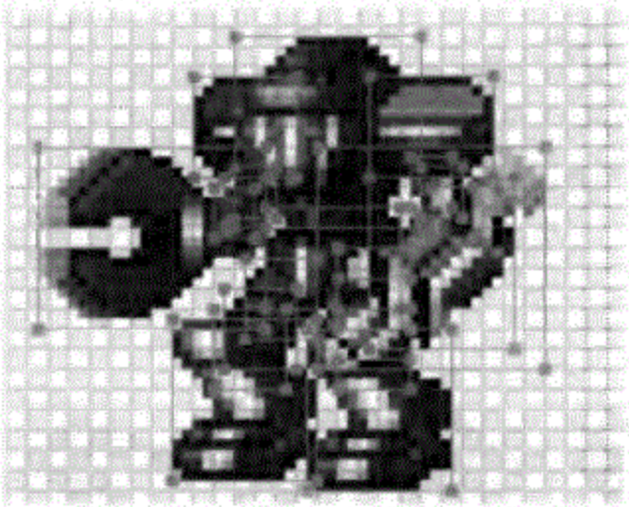


图 2-69

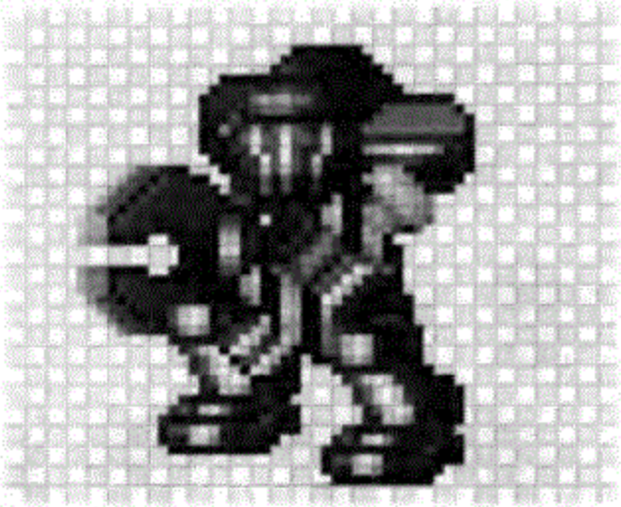


图 2-70

注意：像素人物行走动画绘制基本遵循人物动画的基本原理，总结为以下几点：

① 人物行走图分为“站立”、“抬左腿”、“抬右腿”共 3 帧，以 1-2-1-3-1-2-1-3 的顺序排列组成行走动画。

② 人物行走时站立帧身高最高，抬腿帧略矮（由腿的运动造成，上半身长度不变）。行走时有高低起伏变化。

③ 抬腿帧时描绘手臂的摆动与上半身的前后轻微晃动。

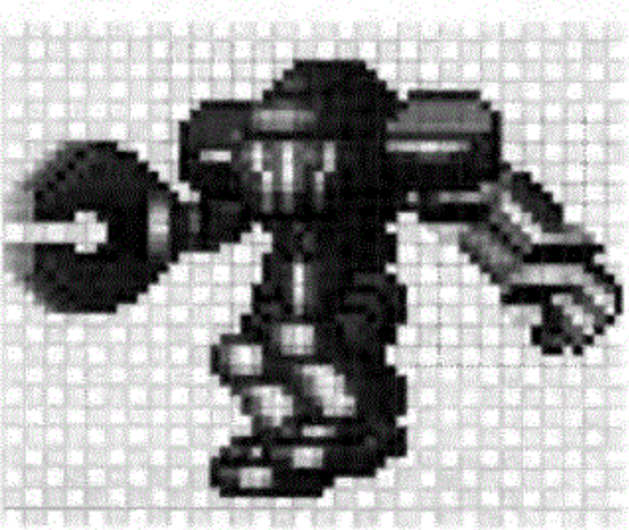


图 2-71

再次复制站立帧，按上述方法将复制出的帧调整为抬左腿的姿势，如图 2-71 所示。注意观察站立图，右腿的位置比左腿高一个像素，相对应的虚拟的“地面”也比左腿的“地面”（绿色辅助线）高一个像素，这次右腿不做垂直方向的位置变化。为做出摆臂动作，将左机械手臂（机械拳）进行 90° 旋转。90°、180° 旋转能保持像素图像素原貌，其他角度会使像素点发生变化而让图像模糊，如图 2-72 所示，在这种情况下就需要手动进行修改或重新绘制整个部件的位图，如图 2-73 中将 45° 手臂修改为水平手臂。

注意：例子中人物为机器人，所以在简单对各个部件进行调整产生新动作后几乎不需要做修改。而一般人物若不进行修改会显得僵硬，效果类似皮影戏。在之前绘制阶段时，对一些柔软的部位，如人的胳膊和腿可以使用和例子中机器人大腿类似的分节绘制以提高柔韧性和动作复杂程度。图 2-66 中为 FFT 中的人物，可以看到手臂与人物是分开绘制的，这是分层绘制法在游戏中的应用实例。而图 2-65 中的格斗游戏例图（春丽），则因为动作细腻复杂，无法以这种简单的分层绘制和拼贴的方法完成。

图 2-72

图 2-73

行走图的 3 个帧都绘制完毕后，复制站立帧。拖动帧之间的位置关系，按“站立—抬左腿—站立—抬右腿”的顺序排列。

单击“文件”→“导出向导”→“继续”→“继续”命令，进入“图像预览”窗口。在“动画”选项卡中可以调整每一帧的持续时间，这在 2D 游戏引擎中会用到。

在“图像预览”窗口中单击播放按钮查看机器人行走图的实际效果，在“动画”选项卡中调整每一帧的持续时间。这里我们选中第一帧按住 Shift 键并单击最后一帧以选中所有帧。在“帧延时”中输入每帧持续时间，检查效果并继续调试。

动作调整完毕后单击“确定”按钮关闭“图像预览”窗口。根据动作调整时遇到的问题，修改部分帧中的内容。完成后另存为 .PNG 图像。

提醒：持续时间，与 Flash 不同，Firework 的该功能主要用于 GIF 动画制作。一系列动作中并非每帧持续时间都是相同的，保持每帧持续时间的差异可以带来更好的动画效果。举例：有“刀举起”、“刀悬空”、“刀放下”3 帧画面，以“放下”—“悬空”—“举起”—“悬空”—“放下”的顺序模拟举刀后劈下的一系列动作。若每帧持续时间相同，只能看到刀做机械摆动。表现方法应为举起过程花费较长时间，劈下过程几乎在瞬间完成。各帧持续时间比例大致为“长”—“长”—“长”—“短”—“短”—“长”，其中每帧之间可能还有更细微的差别，最小单位为 1/100 s。复杂的动作就要经过长时间的调试才能有满意的效果，这些数据将在导入 2D 游戏引擎时被使用。在实际导入引擎后会发现在横版过关游戏中流畅的动作可能比之前的美工工作更重要，这也是像素横版过关游戏的制作比飞行射击游戏困难的原因之一。

复制并打开之前绘制的静态机器人杂兵文件（PNG）。同样以站立帧为起点，在不同文件中分别制作“攻击动作 1”、“攻击动作 2”、“被攻击 / 死亡”3 个动作。动作帧数并没有限制，帧越多则动作越细腻，然而动作的节奏感才是最重要的。制作过程中要修改或添加许多部件，比如机器人的背面、张开的爪子、爪子闭合产生的灰尘、必杀技的特效等，如图 2-74 所示，注意火焰特效中含有半透明成分。要在被攻击状态发生闪烁，在属性面板中添加滤镜调整色相 / 饱和度即可。

提醒：人物的死亡动画一般紧接着被攻击状态的动画，所以这两部分可以放在一起归类。被攻击时调用前半部分动画（到闪烁为止），被击败则调用整段动画。

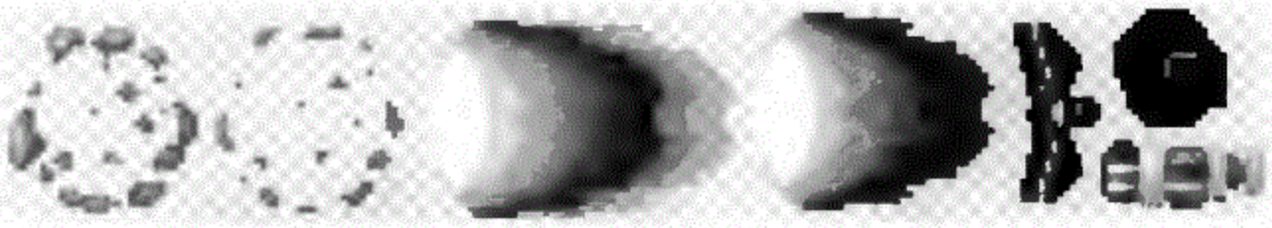


图 2-74

当人物的所有动作都调整完毕后就进入下一阶段，即切割和排列。使这些像素图可以被引擎识别。

（2）人物动作素材的排列和拼接

图 2-75

与像素场景相似，人物在导入到引擎之前要按照引擎标准进行重新排列，如图 2-75 所示。排列方式与引擎有关，如上一个例子中 FFT，人物的手可以与躯体分别导入。而大多数 2D 引擎不支持这类功能。下面我们用在上一个例子中绘制完成的机器人作为例子讲解人物动作素材的排列和拼接。

在 Firework 中打开制作完成的行走图 PNG 文件。此时人物的绘制阶段已经结束了，所以可以考虑在每一帧内将人物的各部分进行合并。框选工作区内构成人物的各个位图图像，右键合并所选。不过原来分层绘制的源文件一定要进行备份，因为像素素材修改频繁，可能在导入引擎后发现效果不佳而需要再次进行大幅度修改，况且帧面板中包含之前调整好的每帧持续时间，这些数据能对在引擎中调整动作有极大帮助。

打开“洋葱皮”工具观察行走图 3 帧图像重叠时的最大宽度，在这个例子中“抬起左腿”帧的宽度最大，涵盖了 3 帧重叠时的总宽度，达到 60 像素。

因为有 3 帧不同图像，所以我们将画布宽度调整为 60 × 3 即 180 像素，改变画布尺寸时向右方向延伸，相应地将网格调整为 60 × 60，若觉得不明显也可以用辅助线进行标示，注意之前绘制过程中留下的“地面”辅助线也继续保留，如图 2-76 所示。

辅助线确立完毕后，首先要进行基本的位置矫正。以宽度最大的人物姿势作为位移时的标准物体，由于上一步的原因使得宽度最大的人物动作必定能接触到其左侧或右侧辅助线。而在本例中“抬左腿动作”总宽度与辅助线间距等宽，正好能同时接触左右两边辅助线（因为是 3 个等宽区域，画布两侧边缘也可以当做辅助线），所以以抬左腿动作为水平移动的标准物体。

图 2-76

在抬左腿动作所在帧中，打开帧面板中“洋葱皮”工具里的“多帧编辑”功能，框选抬腿动作的同时也顺便框选了其他帧中的动作。水平移动使其位于画布最左侧，最左侧与画布边缘接触，最右侧与辅助线接触。此时其他帧内的人物动作也移动了相同的距离。关闭“多帧编辑”功能。

在帧与帧中切换时，观察每帧中人物位置与左、右两侧辅助线的位置。将它们一一复制到最上方帧或新建帧中（重复帧只需复制一个即可），按原帧顺序（动作播放顺序），每两根辅助线中摆放一个。根据它们在原帧中与辅助线的关系确定并调整水平方向（X 轴）位置，如图 2-77 所示。

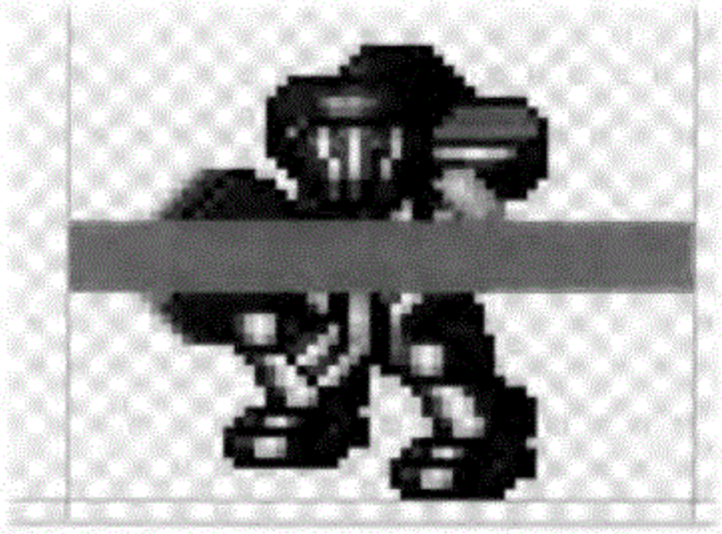


图 2-77

注意：调整过程的小技巧。与辅助线有接触的人物动作，粘贴到新位置后继续与辅助线接触即可。与辅助线无接触的人物动作，可以在其所在帧先用矩形工具绘制宽度与辅助线间距等宽的矩形，如图 2-77 所示。将矩形与人物一同复制粘贴到新位置后保持同时选中状态，根据矩形移动到正确位置，最后删除矩形即可。最后还有一个原理简单，且方便好用的高级技巧，即数像素点。

调整垂直方向（Y 轴）位置。以“地面”辅助线为参照确定垂直方向的位置，要以人物在原有帧中的 Y 轴位置作为参照而不是简单地将人物的最低点与“地面”紧贴。可以多拉几条辅助线作为参照或直接在属性面板中输入 Y 轴具体坐标。完成后另存图片，人物的行走图就排列完成了。

以上仅仅以人物行走图的 3 帧作为例子，实际情况下我们要将一个人物的所有动作拼接到一张图中去。因为之前每套动作都是以站立作为动作开始帧的，所以都包含站立帧。只要将所有帧导入一个文件后通过对齐站立帧来对齐所有帧即可。单位宽度则取决于打开“洋葱皮”工具后所有帧重叠产生图形的宽度。除此以外原理基本相同，如图 2-78 所示。

如果引擎对长度有所限制，则要在适当的时候换行。也有像 RPG Maker 等对导入

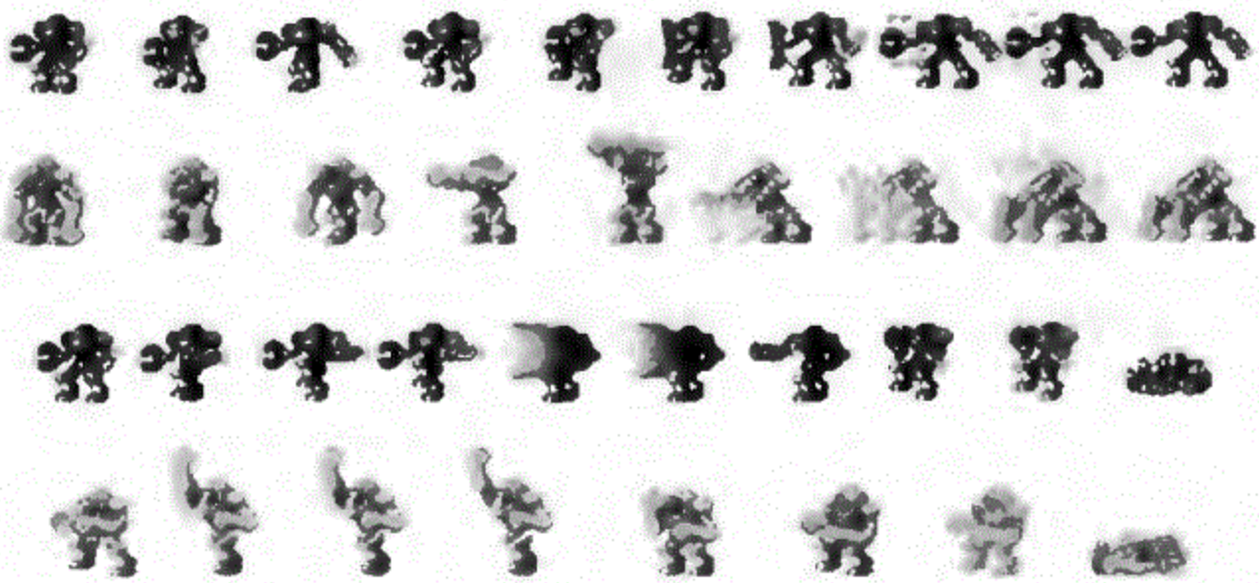


图 2-78

格式有严格要求的，如 RPG Maker 的人物行走图（RPG Maker XP 从 3×4 改变为 4×4，同时支持的分辨率也更大）。不过因为每个单元的大小是一样的，一旦排列过一次以后就可以通过辅助线的帮助很容易地重新整理位置。

注意：实际上有些引擎支持在引擎内调整帧的位置，也就是说我们只需要按照“最宽的单帧动作”的宽度和“最高的单帧动作”的长度来确定每个单元的长和宽，再随便地将所有动作帧分别放入这些单元格中进行排列即可。既不用对齐也不用计算，比我们的例子中轻松许多，然而这一切只是看上去很美，实际差别只是将一系列对齐工作放到引擎中进行而已。在引擎中进行对齐使用的手段单一，并且理所当然地不及专业的绘图软件便捷，过程几乎完全凭借眼睛和感觉以及手指的重复劳动，所以想要一劳永逸的话还是先认认真真地对齐后再导入引擎。引擎中调整帧位置的功能应当在实现“跳跃”、“滑铲”等动作时发挥作用。

4. 飞行射击游戏

之前的内容几乎都是在横版动作游戏中，因为飞行射击游戏中的动态效果相对较少，制作也相对容易。然而要制作一个生动的飞行射击，如图 2-79 所示，还是要运用到一些技巧的。

（1）动态尾焰效果

首先来看两台飞行射击游戏中的自机，见图 2-80、图 2-81。两台自机皆为机器人，因为是飞行射击游戏的自机，所以描绘的是它们的背影。向左或右侧移时有侧身动作，然而由于种种设计和绘制上的不完善，这两台机器人作为自机始终太过简陋。那么除了重新制作或者修改还有什么方法能对其进行美化呢？接下来要为它们添加上飞行背包的尾焰效果。

绘制两个圆角的长条矩形，一粗一细，分别进行复制。每 4 条一组，重叠关系如图 2-82 所示，然后将其重叠于自机机器人上方察看组合效果。调整位置直至效果满意。

新建空白帧，将背后尾焰较细的一组 3 个机器人连同尾焰一起复制到该空白帧。机器人连同尾焰对齐到原帧中粗尾焰机器人的位置。在“图像预览”窗口中单击播放按钮检查尾焰因反复在粗和细两种状态中切换造成的闪烁效果，如图 2-83 所示。

图 2-79

图 2-80

图 2-81

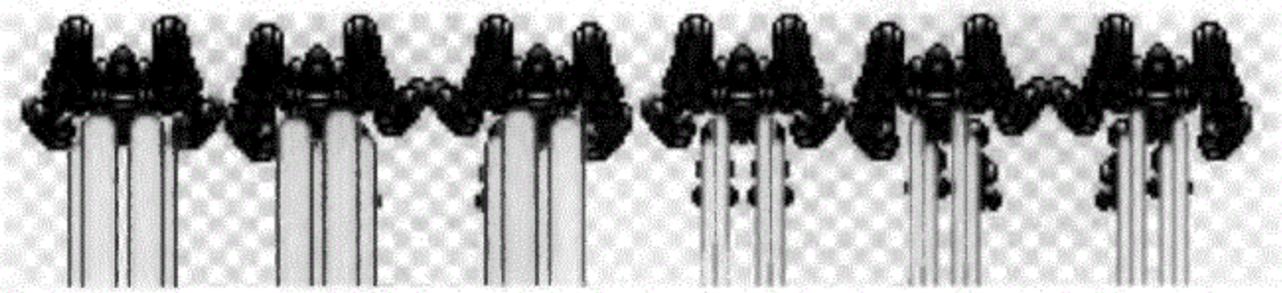


图 2-82

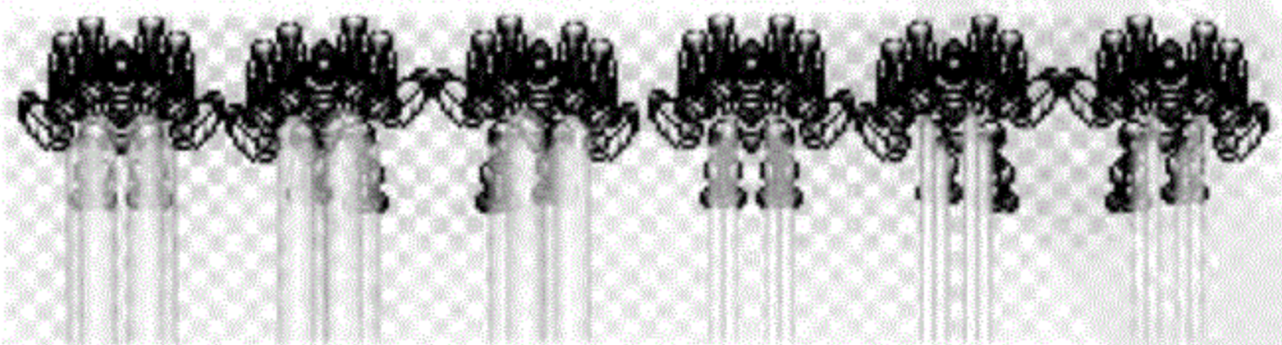


图 2-83

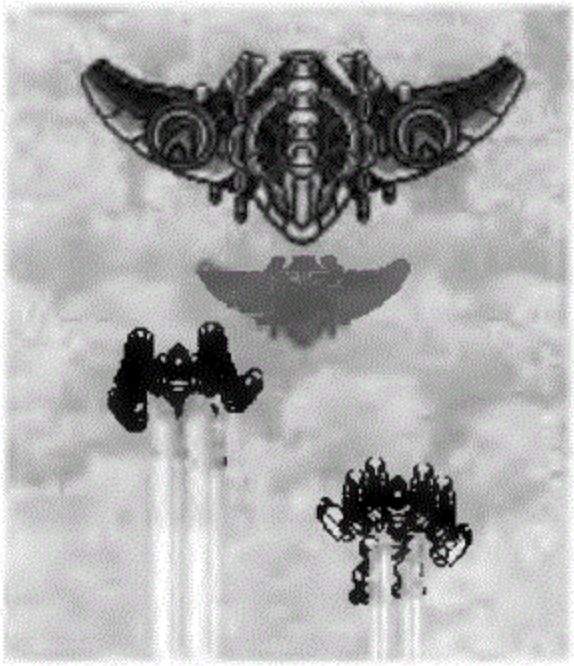


图 2-84

将尾焰延伸到一定长度（超出游戏画面高度），修改为半透明，如此一来尾焰就添加完成了。请看例图中的游戏画面模拟，如图 2-84 所示。

（2）BOSS 角色动态

飞行射击游戏中最有魅力的是大型的 BOSS 角色，见图 2-79。弹幕和 BOSS 几乎是飞行射击游戏的全部，所以下面我们来讲一些 BOSS 的制作技巧。

我们的目标是制作一个可活动，可进行局部破坏的 BOSS。BOSS 虽然同样是先分解为许多部件再进行制作，但与之前的人物制作不同的是，这个分开制作的流程一直持续到游戏制作完成。简单地说，所谓的这个 BOSS 实际是由几个不同的敌人拼凑重叠在一起的。每一个都有独自的火力、HP 以及被攻击动画，由于这个原因造成了可以对 BOSS 进行“局部破坏”的“假象”，见图 2-85 和图 2-86。通过仔细调整每个敌人的行动范围和行动模式，使它们从整体看上去更像一个正在活动的大型 BOSS，了解这几点后就可以更有针对性地设计 BOSS 了。

图 2-85

图 2-86

第 3 章　2D 游戏教学引擎的使用

3.1 游戏引擎基本概念

1. 需要游戏引擎的原因

引擎能在游戏制作过程中使程序员从许多基础建设的繁重体力劳动中解放出来（例如运算、调整坐标等），从而集中精力投入游戏的编辑过程，提高工作效率。游戏引擎在研发过程中使企划、程序、美术设计结合得更为紧密，工作分配更为合理，极大地提高了各工种的研发能动性，以避免重复劳动。在研发后期的调整阶段，其便利的调整方法尤为突出，它是游戏研发工程化的必然结果。

2. 对 2D 美术数据的灵活编辑

早期游戏研发的美工除了制作美术数据以外，很难更深入地参与游戏研发；而程序员通过程序编写体现的画面效果往往不能使美工满意，尤其在动画播放方面，此时调整和修改成为极其艰难和漫长的过程。现在，美术设计者可以通过编辑引擎对自己制作的数据进行直观地编辑直至满意为止，并且可以保存编辑的设置数据，而程序员只需调用编排这些设置数据就可以在游戏中体现出令人满意的画面效果。

3. 2D 游戏引擎理念下的美术数据的制作方法

（1）图像资源数据

图像资源是其他美术数据的根本，没有图像资源其他数据就无从谈起。在制作图像数据之前必须考虑到该数据将来是用于地图（之前提到的背景），还是用于动画（之前提到的物件），以便来规划图像数据的尺寸。

（2）单位资源数据

单位资源数据是将图像数据分割成若干个等分的矩形而生成的数据，并在之后的制作中能合理并且反复地利用与制作。

（3）地图资源数据

地图资源数据是根据游戏需要，用单位数据中的每个单位组合配置成游戏场景的配置数据。地图资源数据的大小由游戏本身的类型所决定，主要分为横版、纵版以及跟踪式。

（4）动画资源数据

动画资源数据是用单位数据中的每个单位组合配置成由数帧组成的动画数据。在游

戏制作完成后所表现的动画就是由这些数据组成。

3.2 引擎界面介绍

在以往的教学中，受条件的限制，通常的 2D 游戏作业只能停留在场景、角色、物件等几个独立的环节。在学校和游戏公司的不懈努力下，终于完成了 2D 游戏教学引擎的初步制作，并且在之后的时间里几度修改，才完成了大家现在所看到的最终版本。在本章中，将详细介绍 2D 游戏教学引擎的具体使用。

2D 游戏引擎的界面分为 4 个区域，呈品字形分布。界面中最上面一排是编辑菜单区域，编辑菜单区域下方的是编辑工具区域，左下侧是工程视图，右下侧是操作区域，如图 3-1 所示。

图 3-1

3.3 新建项目工程

① 单击编辑菜单中的“文件”会弹出一个下拉菜单。

② 在此菜单中选择“新建项目”命令，会弹出“创建新项目”对话框，如图 3-2 所示。

图 3-2

③ 在“创建新项目”对话框中，输入项目名称、指定新项目路径，然后单击“确定”按钮，这样一个新的项目工程就建好了。

④ 以上操作通过“文件”下方的按钮也能实现。

⑤ 完成以上操作后，单击编辑菜单中的按钮保存工程文件（文件格式为 *.iprj）。

注意：这个文件格式是该软件的工程文件，并不是它的输出文件格式。输出文件格式将在后面详细介绍。

⑥ 单击编辑菜单中的按钮可以打开已保存的工程文件。

3.4 创建工作目录

① 单击编辑菜单中的“工具”会弹出一个下拉菜单。

② 在此菜单中选择“添加资源”子菜单中的“目录”命令，弹出“创建目录”对话框，如图 3-3 所示。

③ 在“创建目录”对话框中，输入目录名称，然后单击“确定”按钮，这样一个新的目录就建好了。

④ 以上操作通过“文件”下方的按钮也能实现。

⑤ 完成以上操作后单击编辑菜单中的按钮保存工程文件。

提示：创建目录只是为了能更好地来管理项目工程中的各种资源。在游戏制作过程中，元素非常多、且杂乱，为了更好地整理思路，所以用不同的文件夹来存放不同类型的文件。

图 3-3

3.5 导入图像资源

① 单击编辑菜单中的 按钮打开其子菜单。

② 在弹出的子菜单中选择“图像资源”命令，打开“选择图像文件”对话框，如图 3-4 所示。

图 3-4

③ 打开所需的图像文件（可以打开的文件格式有 4 种：*.bmp、*.png、*.jpg、*.tga）。

④ 双击工程视图中的已建图像资源，此图像资源就会以窗口形式在操作区域中打开，如图 3-5 所示。此窗口的工具栏上有 3 个工具图标按钮：、、，可以将单位视图放大，可以将单位视图缩小，可以将单位视图复原到原始大小。

图 3-5

完成以上操作后单击编辑菜单中的按钮保存工程文件。

3.6 生成单位资源

① 在已经导入所需图像资源的前提下，单击编辑菜单中的按钮打开子菜单。

② 在弹出的子菜单中选择“单位资源”命令，打开“创建新单元”对话框，如图 3-6 所示。

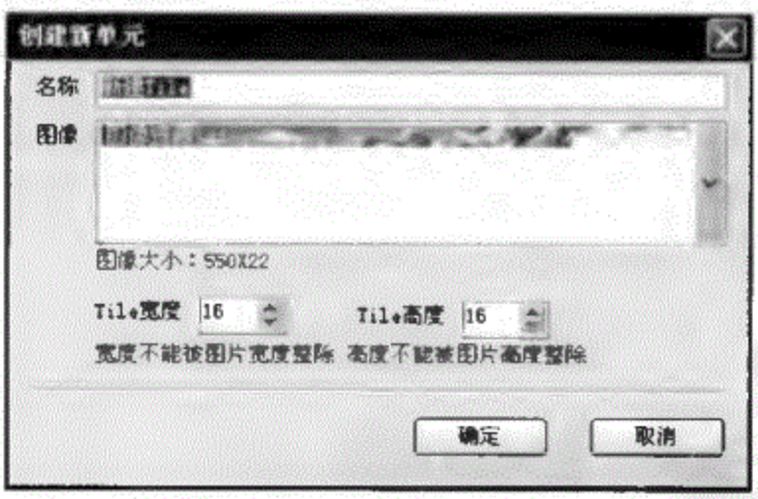


图 3-6

③ 在此对话框中输入单位名称、选择图像资源以及宽度和高度后，单击“确定”按钮成功创建一个新的单位资源。

注意：在“图像”显示框下面用绿色文字显示着目前图像的宽和高，例如“图像大小为 550×22”表示该图像宽 550 个像素、高 22 个像素。如果输入单位宽度和高度的数值不能被图像的宽度和高度整除，则在数值框下方会显示红色警告文字，如图 3-6 所示；

当输入数值能被整除时，红字会自动消失并会有绿色文字告诉用户当前图像被分割成了几个单位。

④ 双击工程视图中的已建单位资源，此单位资源就会以窗口形式在操作区域中打开，如图 3-7 所示。此窗口的工具栏上有 4 个工具图标按钮：、、、，可以将单位视图放大，可以将单位视图缩小，可以将单位视图复原到原始大小，按钮可以将当前窗口放在其他窗口的前面。在工具栏上方有一个名为“编辑”的下拉式菜单按钮，单击此按钮会出现“修改属性”和“导出图像资源”两个命令，选择“修改属性”命令可以对当前单位资源重新编辑，选择“导出图像资源”命令可以将当前单位资源所用的图像另外保存为 PNG 格式的文件。

图 3-7

提示：当输入单元的宽度和高度，并单击“确定”按钮后，再打开已经创建的单位资源的时候，可以看到在单位资源上有红框，这个红框就是能将图像资源切割成单位的工具，可以通过检查红框的大小来确认单位资源是否能分隔成所需要的尺寸。

⑤ 完成以上操作后单击编辑菜单中的按钮保存工程文件。

3.7 导入背景音乐和音效文件

① 单击编辑菜单中的按钮打开子菜单。

② 在弹出的子菜单中选择“背景音乐”命令，打开“选择背景音乐”对话框，如图 3-8 所示。

③ 打开所需的背景音乐文件（背景音乐的文件格式为 *.mp3）。

④ 再单击编辑菜单中的按钮，打开子菜单。

⑤ 在弹出的子菜单中单击“音效”命令，打开“选择音效”对话框。

图 3-8

⑥ 打开所需的音效文件（音效的文件格式为 *.wav）。

提示：虽然都是音乐文件，但是背景音乐的文件格式和音效文件的格式却是不同的，背景音乐的文件格式是 mp3 格式，而音效的格式是 wav 格式。

3.8 制作地图资源

① 在已经导入所需图像资源做成所需单位资源的前提下，单击编辑菜单中的 按钮打开子菜单。

② 在弹出的子菜单中选择“地图资源”命令，打开“创建新地图”对话框，如图 3-9 所示。

③ 在该对话框中输入地图名称、选择单位资源以及输入地图的宽度和高度后，单击“确定”按钮。

④ 双击工程视图中的已建地图资源，此地图资源就会以窗口形式在操作区域中打开。

⑤ 单击窗口面板中的 按钮，打开地图的单位资源。

⑥ 将打开的单位资源，用鼠标拖动的方式，拖至地图面板的空白处，就像搭积木一样，进行地图拼接。

⑦ 完成拼接，如图 3-10 所示。

图 3-9

图 3-10

⑧ 地图资源窗口的工具栏上有若干个工具图标按钮：、、、、、、、、、、、。

——可以取消上一步操作。

——可以恢复上一步操作。

——可以删除当前选定的单元。

——可以复制当前选定的单元。

——可以粘贴当前选定的单元。

——可以保存当前地图资源。

——可以打开关联的单位资源，从单位资源中将选中的单位拖入地图进行编辑。

——可以将当前选定的单元水平反转。

——可以将当前选定的单元垂直反转。

——可以在地图视图上添加网格。

——下拉式菜单按钮，可以修改地图尺寸大小，含“删除行”、“删除列”、“添加行”、“添加列”等功能。

——可以放大视图。

在工具栏上方有名为“查看”的下拉式按钮，单击“网格线颜色”命令可以更换网格线颜色。

⑨ 完成以上操作后，单击此窗口工具栏上的按钮可以保存此地图资源。

3.9 制作动画资源

① 在已经导入所需图像资源的前提下，单击编辑菜单中的按钮，打开子菜单。

② 在弹出的子菜单中选择“动画”命令，打开创建新动画对话框，如图 3-11 所示。

图 3-11

③ 在该对话框中输入动画名称、选择单位资源后单击“确定”按钮，即成功创建一个新的动画资源。

④ 双击工程视图中的已建动画资源，此动画资源就会以窗口形式在操作区域中打开。此窗口分为 3 个区域，呈品字形分布，上端是编辑工具栏，左下侧是动作列表，右下侧是动画帧编辑区域，如图 3-12 所示。一个动画资源中可以有若干个动作，每个动作由若干帧组成。

⑤ 动画资源窗口的工具栏上有若干个工具图标按钮：、、、、、、、、、、、、、、、、、、、、、、、、。

——新建动作。单击此按钮会弹出“设定动作名称”对话框，输入动作名称后单击“确定”按钮，一个新动作就会在动作列表中出现。

——可以在创建了动作的前提下，激活列表中的动作，单击此按钮添加动画帧。

——可以打开关联的单位资源，从单位资源中将选中的单位拖入动画帧进行动画编辑。

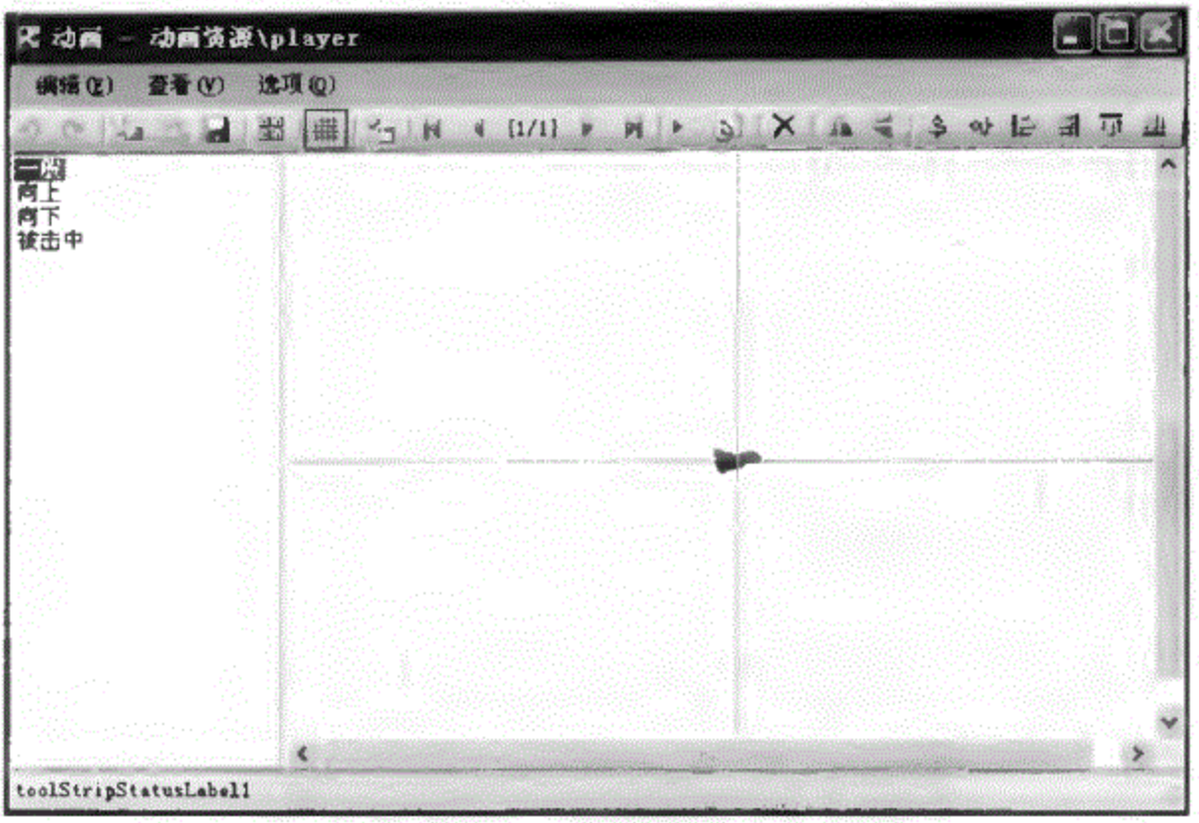


图 3-12

——可以取消上一步操作。

——可以恢复上一步操作。

——可以复制当前选定的单元。

——可以粘贴当前选定的单元。

——可以保存当前动画资源。

——可以在动画帧编辑视图上添加网格。

——可以移动到第一帧。

——可以移动到前一帧。

——可以移动到后一帧。

——可以移动到最后一帧。

——可以设定当前帧的持续时间。

——可以删除当前帧。

——可以水平反转选中的单位。

——可以垂直反转选中的单位。

——可以将选中的单位水平居中于当前动画帧。

——可以将选中的单位垂直居中于当前动画帧。

——可以将选中的若干单位左对齐。

——可以将选中的若干单位右对齐。

——可以将选中的若干单位上对齐。

——可以将选中的若干单位下对齐。

⑥ 单击工具栏上的“编辑”下拉菜单，从弹出的菜单中选择“删除选定动作”、“删除选定单位”等命令。

⑦ 单击工具栏上的“查看”下拉菜单，可以从弹出的菜单中选择更改网格线尺寸、更改网格线颜色、放大动画帧视图、缩小动画帧视图、预览当前动作的动画效果。

⑧ 如果发觉当前关联的单位不对，可打开“选项”下拉菜单，选择“更改单元”命令。

提示：(a) 可以使用 按钮和 按钮方便地将图案进行中心对齐，但是前提是前后图片的大小（至少是长宽比例）必须相同。

(b) 可以通过 按钮来调整前后帧之间的时间间隔，从而可以调节动画的节奏变化，形成不同节奏变化的动画。

⑨ 完成以上操作后，单击此窗口工具栏上的 按钮，保存此动画资源。

3.10 完成游戏物件定义

① 在已经建立动画资源的前提下，单击编辑菜单中的 按钮，打开子菜单。

② 在弹出的子菜单中选择“游戏物件”命令，打开“新建游戏物件定义”对话框，如图 3-13 所示。

图 3-13

③ 在该对话框中输入物件名称、选择需要使用的动画资源后，单击“确定”按钮，即成功创建一个新的物件定义。

④ 双击工程视图中的已建游戏物件定义，此定义会以窗口形式在操作区域中打开，如图 3-14 所示。

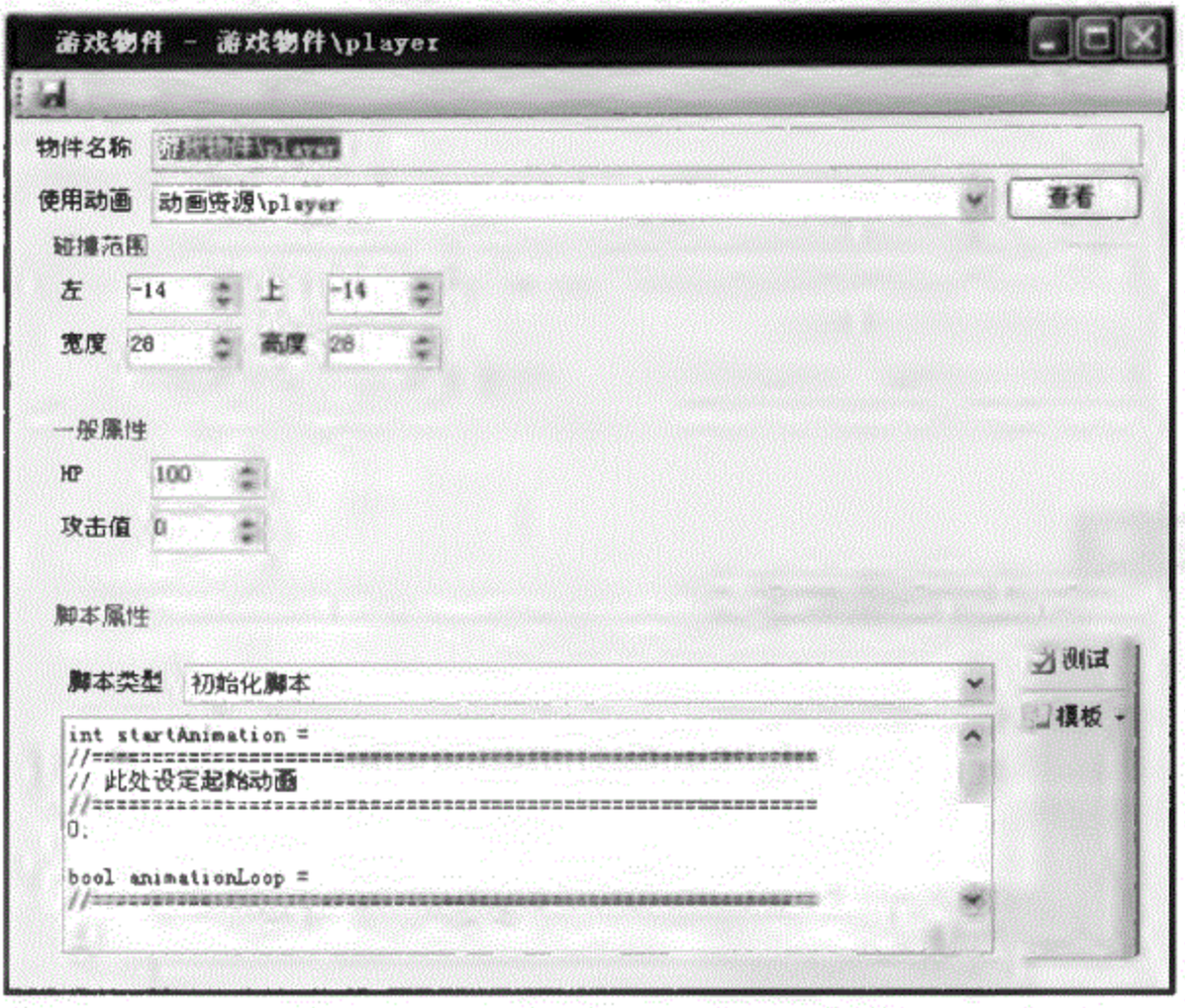


图 3-14

⑤ 游戏物件窗口分成“碰撞范围”、“一般属性”、“脚本属性”3 部分。

- 碰撞范围：指物件与物件相碰撞的判定范围，此范围以方形判定，此方形的大小设定方法为：先找到方形左上角顶点坐标（即输入左和上的像数值），然后输入此方形的宽度和高度的像数值。

提示：在面板中可以通过计算图案在坐标轴中方格的数量来确定碰撞范围的坐标值和大小，每个方格的长宽是 4×4，即：长为 4，宽为 4。

- 一般属性：输入物件的生命值（HP）和攻击值。
- 脚本属性：一般物件的脚本属性由“初始化脚本”、“碰撞测试脚本”、“运行脚本”和“被伤害脚本”4 个类型构成（但不是每个模块的物件属性都需要这 4 个类型的）。每个脚本类型都可以通过单击右侧的 模板 按钮选择模块类型进行编辑。编辑完后可以单击 测试 按钮测试是否通过。

⑥ 在物件定义的过程中如果需要查看相关联动画的图像视图，可以单击“使用动画”右侧的“查看”按钮进行查看。

完成以上操作后，单击此窗口工具栏上的按钮保存此物件定义。

3.11 血槽设定

在游戏界面中需要进行两种设定，一个为血槽设定，包括玩家血槽和敌人血槽；还有一种是 UI，即开始动画和结束动画的设置。

① 先单击编辑菜单中的按钮打开子菜单。

② 在弹出的子菜单中选择“血槽”命令，打开导入血槽图片的“获取字符串”窗口，如图 3-15 所示。

图 3-15

③ 在该窗口中的 label1 文本框中输入血槽，单击“确定”按钮，即成功创建一个新的血槽设定。

④ 双击工程视图中的已建游戏血槽设定，此定义会以窗口形式在操作区域中打开，如图 3-16 所示。

⑤ 血槽设置窗口分为 4 层设置：可选择的背景图片、可选择的前景图片、必须设置的血槽图片和位置设置。

⑥ 背景、前景和血槽图片都是在图像资源中预先导入完成的，在这里只要将图片选定即可。

⑦ 在场景定义中确定位置，并且在最后生成的文件中反复调试以确定最佳的位置。

⑧ 完成以上操作关闭“血槽设定”窗口，单击工具栏上的按钮保存此游戏场景定义。

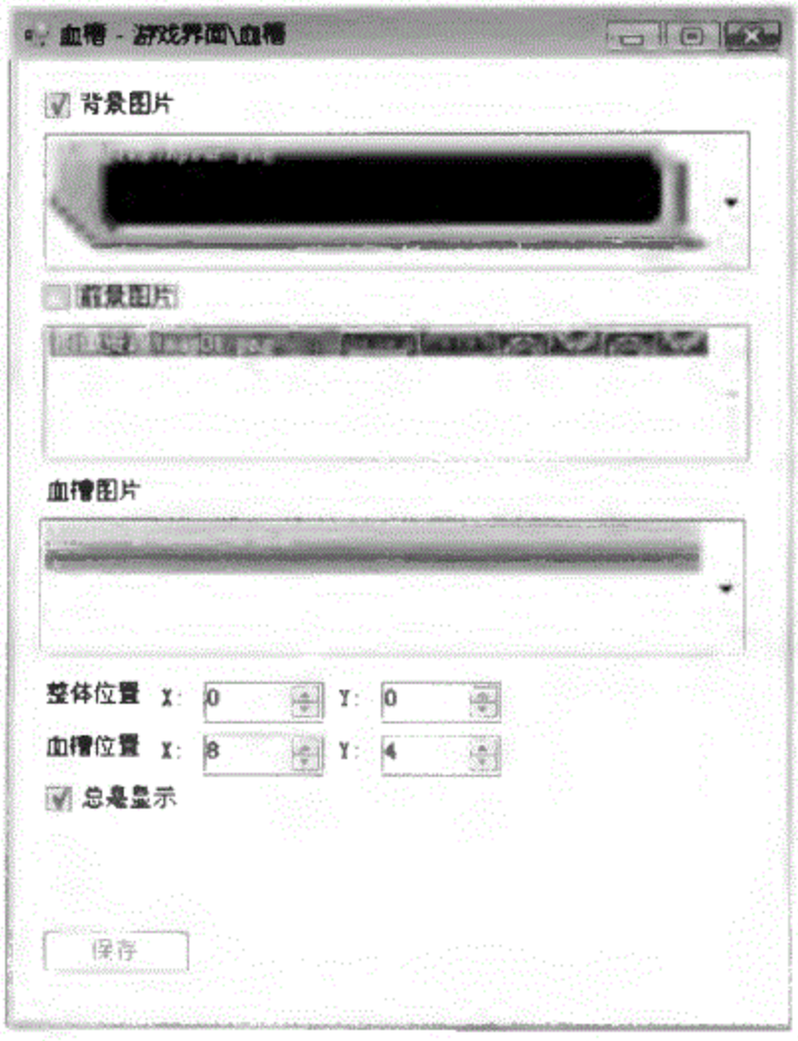


图 3-16

3.12 定义游戏 UI

① 先单击编辑菜单中的 按钮打开子菜单。

② 在弹出的子菜单中选择“游戏界面”命令，打开导入游戏界面动画窗口。

③ 在 UI 的名称文本框中输入 UI，单击“确定”按钮，即成功创建一个新的 UI 设定。

④ 双击工程视图中的已建游戏 UI 设定，此定义会以窗口形式在操作区域中打开。

⑤ 在 UI 编辑器中有两项选择，游戏开始动画和游戏结束动画。在动画资源已经预先制作完成的情况下，分别选择动画资源的开始动画和结束动画。

⑥ 完成以上操作关闭“UI 设定”窗口，单击工具栏上的 按钮保存此游戏场景定义。

3.13 完成游戏场景定义

① 在已经建立所需的游戏物件和地图资源的前提下单击编辑菜单中的 按钮打开子菜单。

② 在弹出的子菜单中选择“游戏场景”命令，打开“新建游戏物件定义”对话框，如图 3-17 所示。

图 3-17

③ 在该对话框中输入场景名称、选择需要使用的地图资源后，单击“确定”按钮，即成功创建一个新的场景定义。

④ 双击工程视图中的已建游戏场景定义，此定义会以窗口形式在操作区域中打开，如图 3-18 所示。

⑤ 将鼠标移至地图上再单击鼠标右键，从打开的菜单中选择已建的游戏物件并将其添入游戏场景，如图 3-19 所示，加入场景的物件可以用鼠标拖动。

图 3-18

⑥ 在已加入的物件上按鼠标右键打开菜单选择属性打开窗口，如图 3-20 所示，在该窗口中通过输入坐标数值可以改变此物件在场景中的位置；在“显示优先级”微调框中的输入数值越大该物件显示越向前；“类型”是指定该物件是中立方、友方，还是敌方。

⑦ 完成以上设置后，单击工具栏上的 按钮，可以显示各物件的名称和碰撞范围。

⑧ 单击工具栏上的 按钮，可以在场景视图上添加网格。

⑨ 单击工具栏上的 按钮，打开“场景脚本编辑器”窗口，如图 3-21 所示。

图 3-19

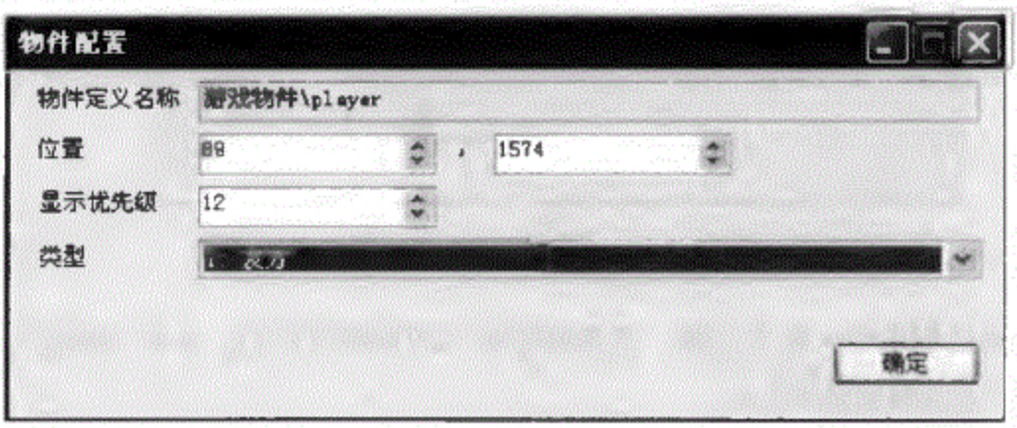


图 3-20

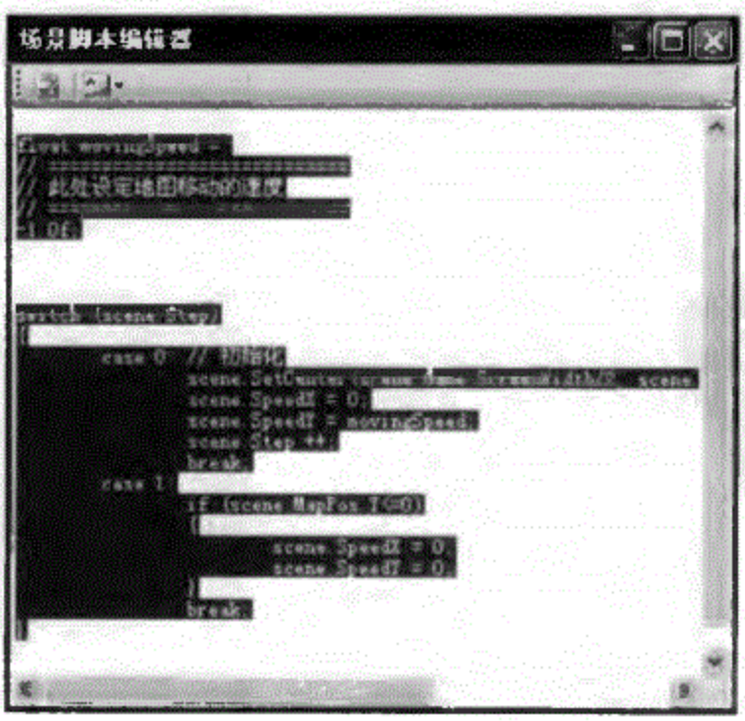


图 3-21

⑩ 在“场景脚本编辑器”窗口的工具栏上单击按钮选择场景模块，单击按钮测试脚本编译是否成功。

⑪ 在游戏场景设定窗口上的“场景界面”中选择前面设定好的UI，在右侧的背景音乐中选择之前设定好的背景音乐，如图3-22所示。

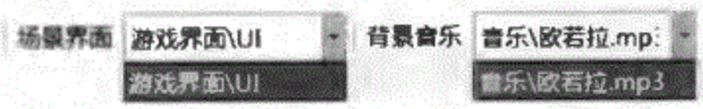


图3-22

⑫ 完成以上操作后关闭“场景脚本编辑器”窗口，单击“游戏场景定义”窗口工具栏上的按钮，保存此游戏场景定义。

3.14 定义游戏

① 单击编辑菜单中的按钮打开子菜单。

② 在弹出的子菜单中选择“游戏”命令，打开“新建游戏定义”对话框，如图3-23所示。

图3-23

③ 在该对话框中输入游戏名称、游戏窗口的宽度和高度（像数单位）、选择所需场景定义后，单击“确定”按钮成功创建一个新的游戏定义。

④ 双击工程视图中的已建游戏定义，此定义会以窗口形式在操作区域中打开，如图3-24所示。此窗口可以对上一步骤的设置进行修改。

⑤ 完成以上操作后，单击编辑菜单中的保存工程文件。

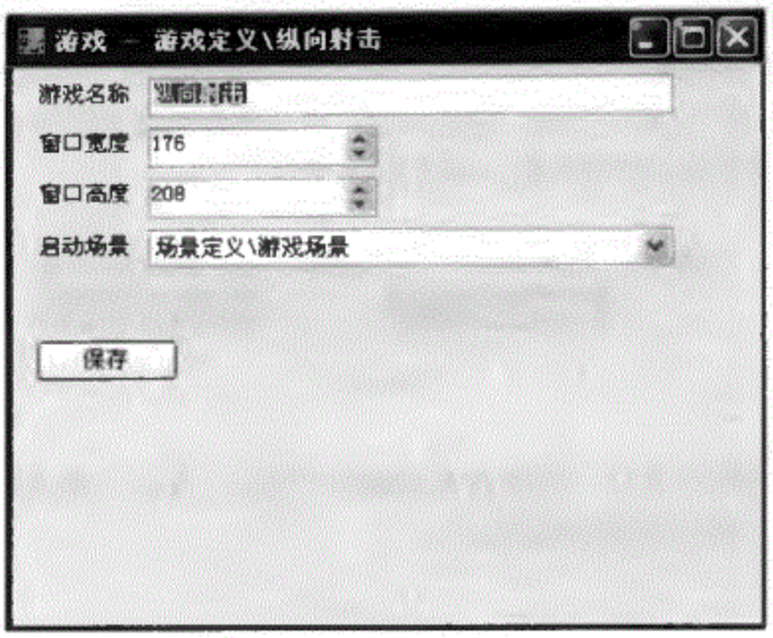


图 3-24

3.15 对已建工程进行编译

① 打开“工具”菜单，选择“编译配置管理器”命令，打开相应的对话框，如图 3-25

图 3-25

所示。

② 单击对话框左下角的“新建”按钮，然后在左上角输入配置名称，全选编译项目中的所有项目，最后单击“生成文件路径”文本框右端的“浏览”按钮选择生成路径。

③ 完成以上操作，关闭对话框后单击“工具”菜单，选择已做成的编译配置名称就进入编译状态，如图 3-26 所示。

图 3-26

④ 编译完成后就会在前面设定的文件路径处生成一个 *.bin 运行文件。

提示：（a）在这个 .bin 文件编译全部结束的时候，会提示编译完成，否则整个编译过程并没有结束，即使在编译过程中需要花费较多的时间等待，也不要着急。

（b）在编译的时候千万不要在还没有完成编译的时候就关闭面板，这样会导致软件的崩溃，这点非常重要，即使在编译的过程中发生错误也要耐心等待完成后再继续进行修改。

3.16 查看生成的 *.bin 运行文件

① 打开编辑菜单中的“查看”菜单，选择“游戏测试”命令，打开如图 3-27 所示的对话框。

② 选择已生成的 *.bin 文件，单击“打开”按钮就可以运行这个游戏了，如图 3-28 所示。

③ 在激活当前游戏测试窗口的情况下，按 Tab 键可以打开“按键设定”窗口，在此窗口中可对游戏操控键进行设定，如图 3-29 所示。

以上为一款 2D 游戏的基本制作过程，在这里只是大致介绍工具的使用情况，并没

图 3-27

图 3-28

图 3-29

有涉及具体的制作细节，在下一章节中，将使用实际案例，为大家介绍具体的制作过程，以及其中的制作细节。

第 4 章　2D 游戏教学引擎实例制作（一）

本章以一个纵向射击的实例介绍 2D 游戏教学引擎的实际使用。在制作之前，必须先绘制好以下几张 png 格式的像素图片资料：玩家、敌人、子弹、地图、爆炸、游戏开始、游戏结束、两个血槽背景和两个血槽图片，总共 11 张图片，如图 4-1 所示。具体绘制过程在前面章节已有介绍，这里不再赘述。接下来就以这个案例来进行讲解。

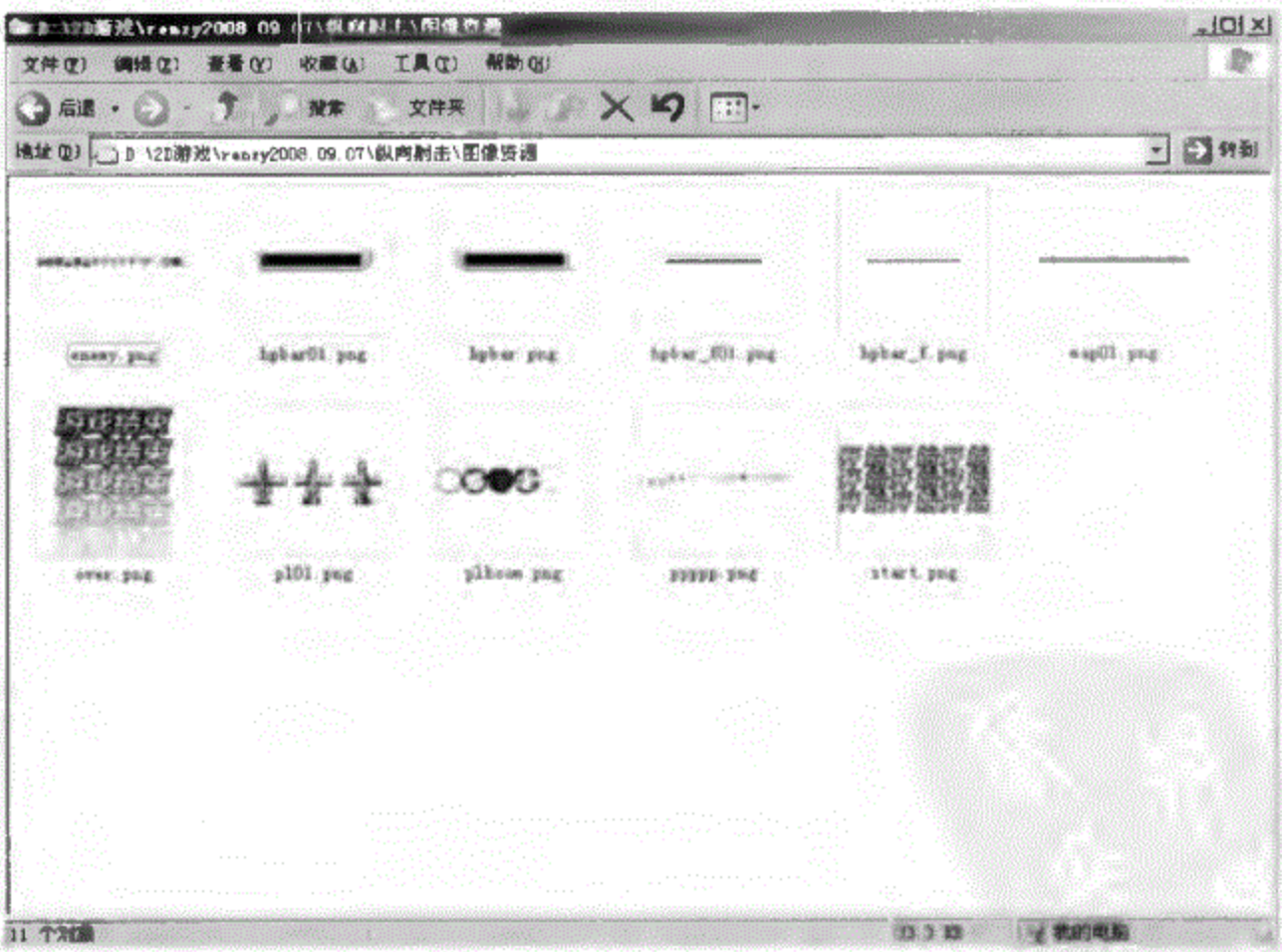


图 4-1

4.1 图像资料的导入

① 在打开 2D 游戏教学引擎后，开始进行项目的制作。在进行制作之初，首先要创建游戏项目。在打开的菜单中，选择“新建项目”命令，如图 4-2 所示。

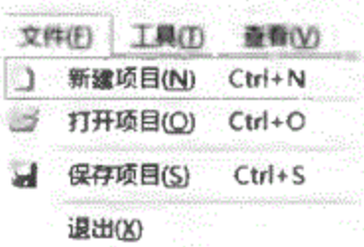


图 4-2

② 单击“新建项目”命令后，打开“创建新项目”对话框。在“创建新项目”对话框的“项目名称”文本框中输入“纵版射击”，在“项目路径”文本框中将项目保存在桌面，如图 4-3 所示。读者可以根据自己实际情况决定项目的放置路径。

③ 确定上述操作后，单击“确定”按钮，然后在左侧的工程视图中，可以看到已

图 4-3

经有名称为“纵版射击”的项目，如图 4-4 所示。在以后的制作过程中，需要反复地对制作完成的部分进行确认，在此不再重复说明。

④ 接下来，进行图像资源的导入。在导入之前，必须先创建工作目录，创建目录的目的是为了对不同的文件进行更好的归类。

图 4-4

注意：当然不归类也可以进行制作，只是在以后的制作中会出现条理不清的状况，所以不建议这样制作。

⑤ 单击编辑菜单中的“工具”会弹出一个下拉菜单，如图 4-5 所示，单击第一栏“目录”工具，系统会自动弹出“创建目录”窗口，如图 4-6 所示，在“目录名称”文本框中输入“图像文件”，表示该目录用于存放图像资源。

⑥ 单击“确定”按钮后，可以在工程视图中发现在“纵版射击”栏下出现了“图像文件”的文件夹，如图 4-7 所示。对于已建的文件夹，可以在选中的情况下，再单击该文件夹名称，即可对该文件夹重命名，将“图像文件”改为“图像资源”，如图 4-8 所示。

⑦ 在选中该文件夹的情况下，在工程视图中单击鼠标右键，这样也可以打开下拉

图 4-5

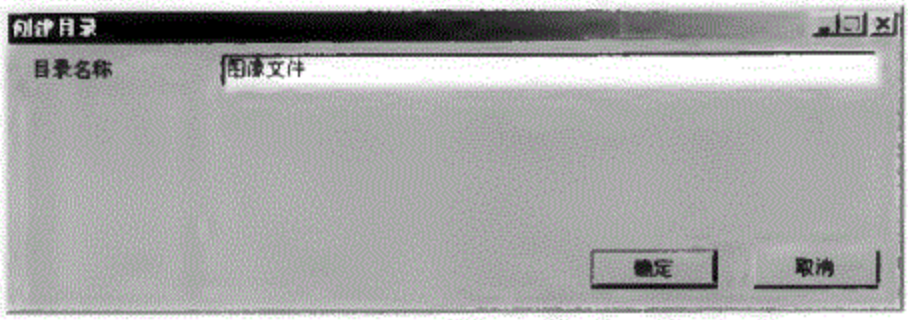


图 4-6

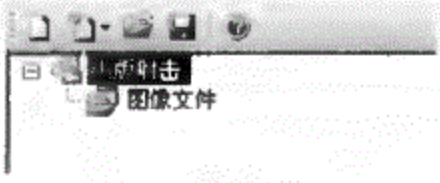


图 4-7

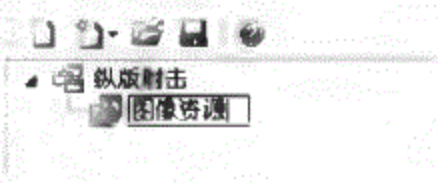


图 4-8

图 4-9

图 4-10

菜单，这个时候就要导入图像资源，如图 4-9 所示。在单击“图像资源”按钮后，系统会自动弹出导入图像资源的窗口，如图 4-10 所示。

提示：这些所需要的图片可以从本书配套光盘中调用。

⑧ 在导入图像资源的窗口中找到已经绘制完成的 enemy.png 图片后打开，可以在工程视图中发现，在“图像资源”文件夹中，已经导入了 enemy. png，如图 4-11 所示。

⑨ 继续使用这种导入方式，将其余 10 张 png 格式的图片也都导入到“图像资源”文件夹中，如图 4-12 所示。

图 4-11

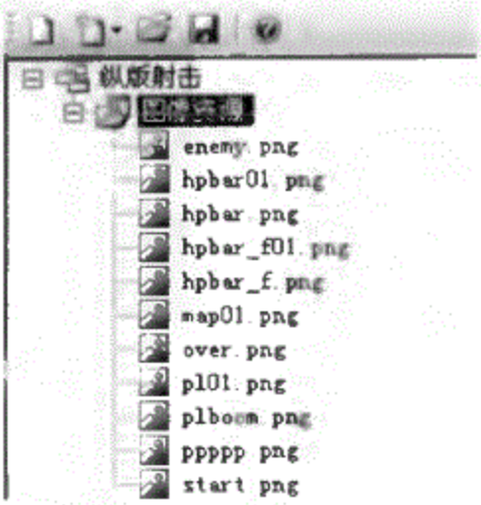


图 4-12

4.2 单位资源的制作

① 在将图像资源导入完成后，用创建“图像资源”文件夹一样的方式，创建“单位资源”文件夹，并且可以发现这个文件夹和“图像资源”文件夹呈并列关系，如图 4-13 所示。

图 4-13

图 4-14

注意：在这里会碰到一个常见错误，即在“图像资源”被选中的情况下创建新的目录，如图 4-14 所示，这样新建的目录就从属于“图像资源”目录之下了，如图 4-15 所示。遇到这种情况，只要将新的目录拖动到最上面的“纵版射击”项目，这个目录就会与原来的目录呈并列关系了。

② 在“单位资源”被选中、“图像资源”全部导入的情况下，在工程视图中单击右键，上次单击右键是导入图像资源，而这次是生成单位资源，如图 4-16 所示。

③ 完成“敌人”单位的制作。在单击“单位资源”按钮后，系统会弹出“创建新单位”对话框，在这个对话框中，首先输入单位资源名称“敌人”，其次在中间的“图像”图片选择栏中找到绘制“敌人”的图片，如图 4-17 所示。

图 4-15

图 4-16

图 4-17

④ 在前面的章节中，已经介绍过什么是单位，并且已经介绍如何去绘制图片，现在只需要按前面的方法重新操作即可。在单位宽度和高度数值框中，分别输入 22，如图 4-18 所示。

图 4-18

提示：在游戏公司中，引擎制作人员和绘制图片人员是分开的，那么引擎制作者在不知道单位大小的情况下，可以通过已知的图像大小和图像中元素的大小来计算出单位的大小。

⑤ 在单位资源创建完成后，单击“确定”按钮，这样就可以在工程视图中看到“单位资源”文件夹中已经存在了名为“敌人”的单位资源，如图 4-19 所示。

⑥ 使用这种方法将 enemy、pl01、plboom、start、over、map01、ppppp 这 7 个图像资源都生成单位资源，分别命名为敌人、主角、爆炸、游戏开始、游戏结束、地图和子弹，如图 4-20 所示。

图 4-19

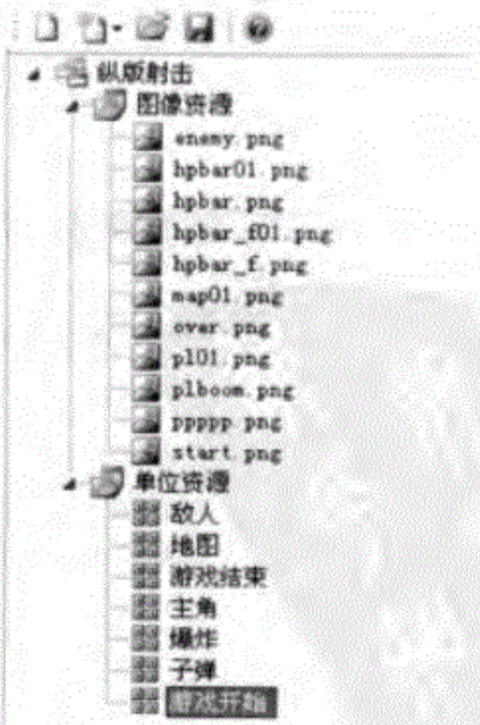


图 4-20

4.3 音乐文件的导入

① 在“纵版射击”下创建“音乐”文件夹，用来存放所有使用到的音乐文件，如图 4-21 所示。

② 在选中“音乐”文件夹的情况下，在工程视图的空白处单击鼠标右键，创建音效，如图 4-22 所示。

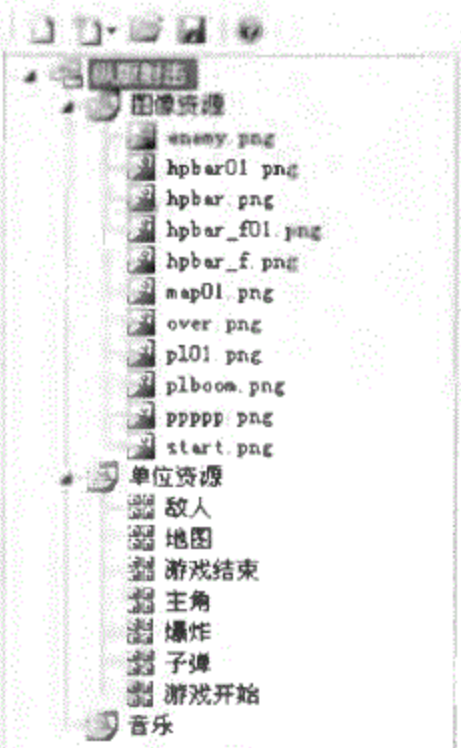


图 4-21

图 4-22

③ 当单击“音效”按钮后，就打开了“选择音效文件”对话框。在此对话框中选择 fire.wav 文件，单击“打开”按钮，如图 4-23 所示，然后就可以在工程面板中看到“音乐”文件夹中存在 fire.wav 文件，如图 4-24 所示。

④ 在选中“音乐”文件夹的情况下，在工程视图中单击鼠标右键，这次在弹出的菜单中选择“背景音乐”按钮，如图 4-25 所示，同样弹出“选择背景音乐”对话框，如图 4-26 所示，在此对话框中选择“欧若拉 .mp3”作为背景音乐，单击“打开”按钮后，就可以在工程视图中发现这个音乐文件了，如图 4-27 所示。

注意：虽然都是音乐文件，不过由于用途的不同，调用的类型也就不同，所以文件的格式也不同。在这里必须将不同用途的文件转换成相应的格式。

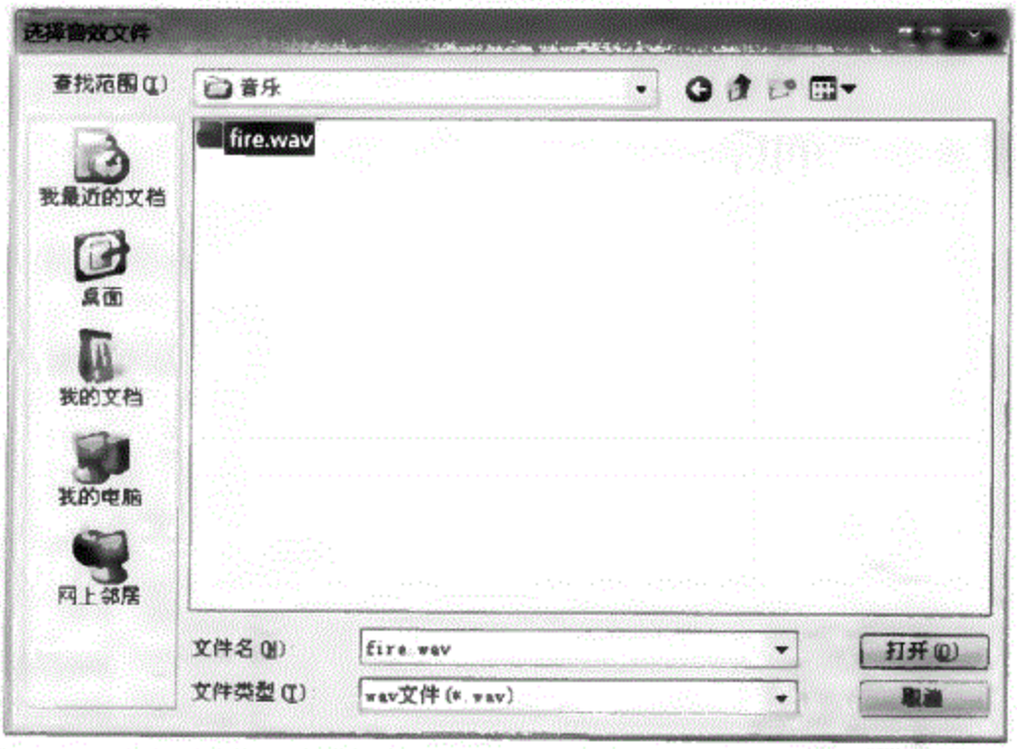


图 4-23

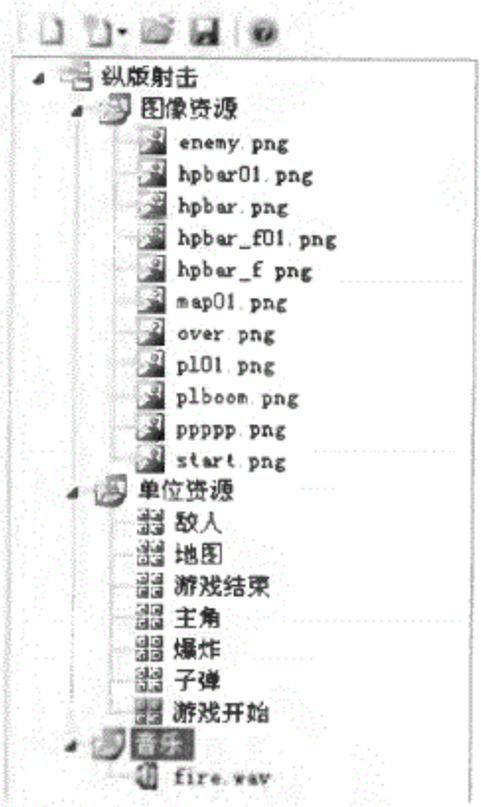


图 4-24

图 4-25


查找范围(I)
音乐
欧若拉.mp3
我最近的文档
桌面
我的文档
我的电脑
网上邻居
文件名(N)
文件类型(T)
MP3文件(*.mp3)
打开(O)
取消

图 4-26


图像资源
enemy.png
hpbar01.png
hpbar.png
hpbar_f01.png
hpbar_f.png
map01.png
over.png
pl01.png
plboom.png
ppppp.png
start.png
单位资源
敌人
地图
游戏结束
主角
爆炸
子弹
游戏开始
音乐
fire.wav
欧若拉.mp3

图 4-27

4.4 制作动画资源

① 创建“动画资源”文件夹，如图 4-28 所示。制作方法与之前一样，在此不再赘述。

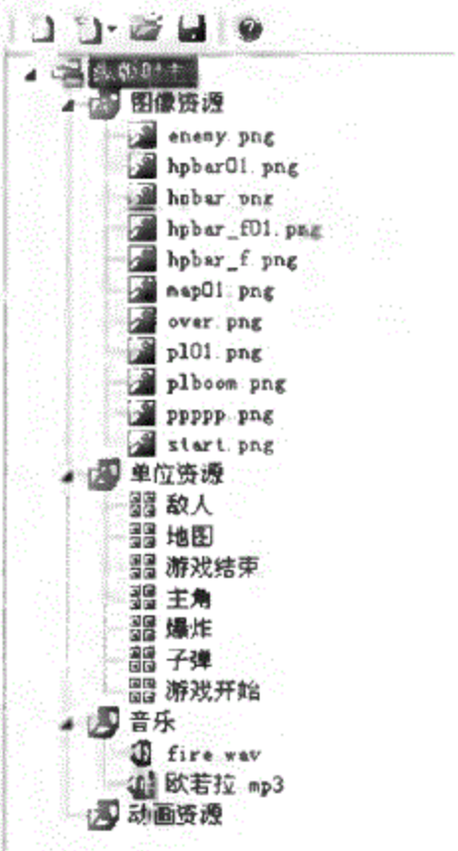


图 4-28

② 选中“动画”文件夹，在工程视图中单击鼠标右键，在弹出的下拉菜单中选择“动画”命令，即可打开“创建新动画”对话框。先制作敌人小兵的动画，在“动画名称”文本框中输入“敌人 1”；在“使用单位”列表框中，找到敌人的图片，如图 4-29 所示。

③ 单击“确定”按钮后，在工程视图中就可以看到已经建立的“敌人 1”。单击“敌人 1”动画资源，打开相应窗口，开始对“敌人 1”进行制作，如图 4-30 所示。

④ 单击工具栏中的“创建动作”按钮（按钮介绍见第 3 章），在弹出的“设定动作名称”窗口的文本框中输入“敌人”，如图 4-31 所示。单击“确定”按钮这时即可在动作创建窗口左侧的动作列表中看到已经输入的名为“敌人”的动作以及这个动作的序号，如图 4-32 所示。

⑤ 创建了动作以后，系统会经常跳出数据已被更改的对话框，如图 4-33 所示，这个时候只要单击“确定”按钮即可，然后系统会提示动画数据已保存成功，如图 4-34 所示。

提示：这两个对话框在制作动画中会经常遇到，只要如上述操作即可。

图 4-29

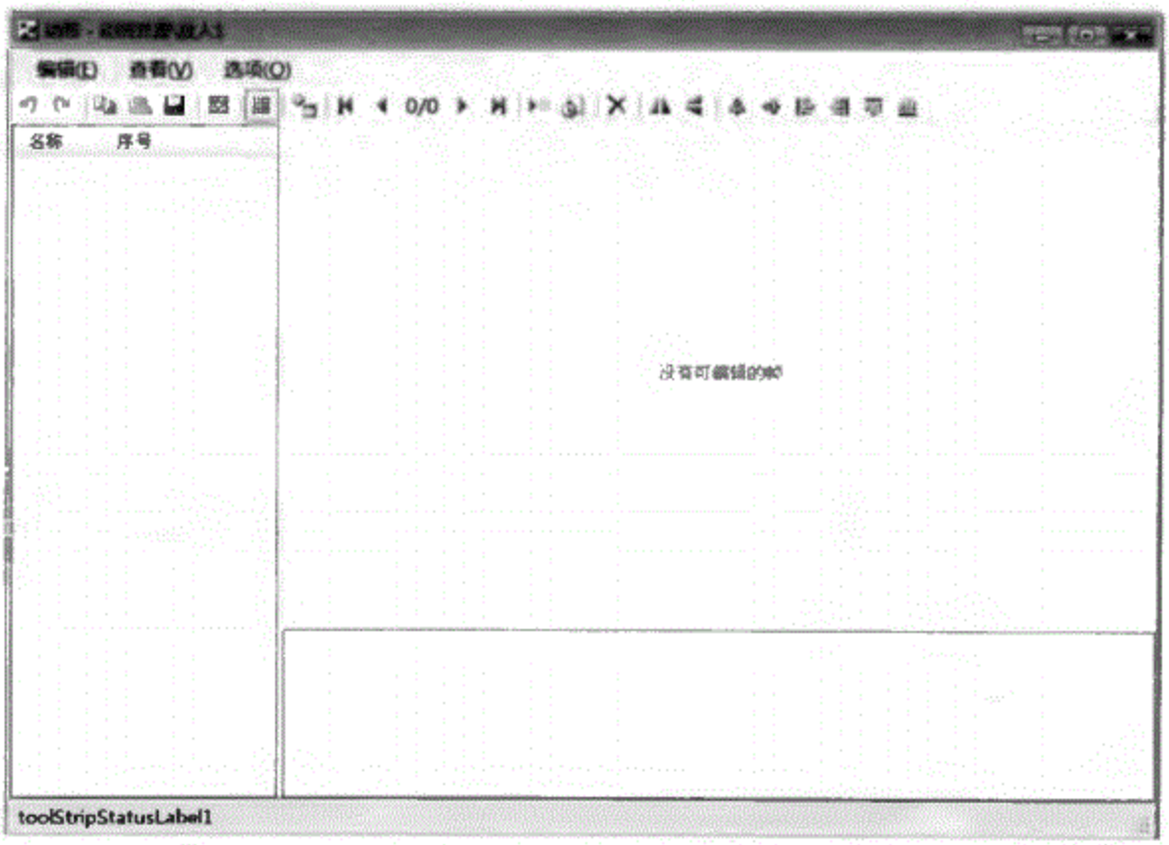


图 4-30

图 4-31

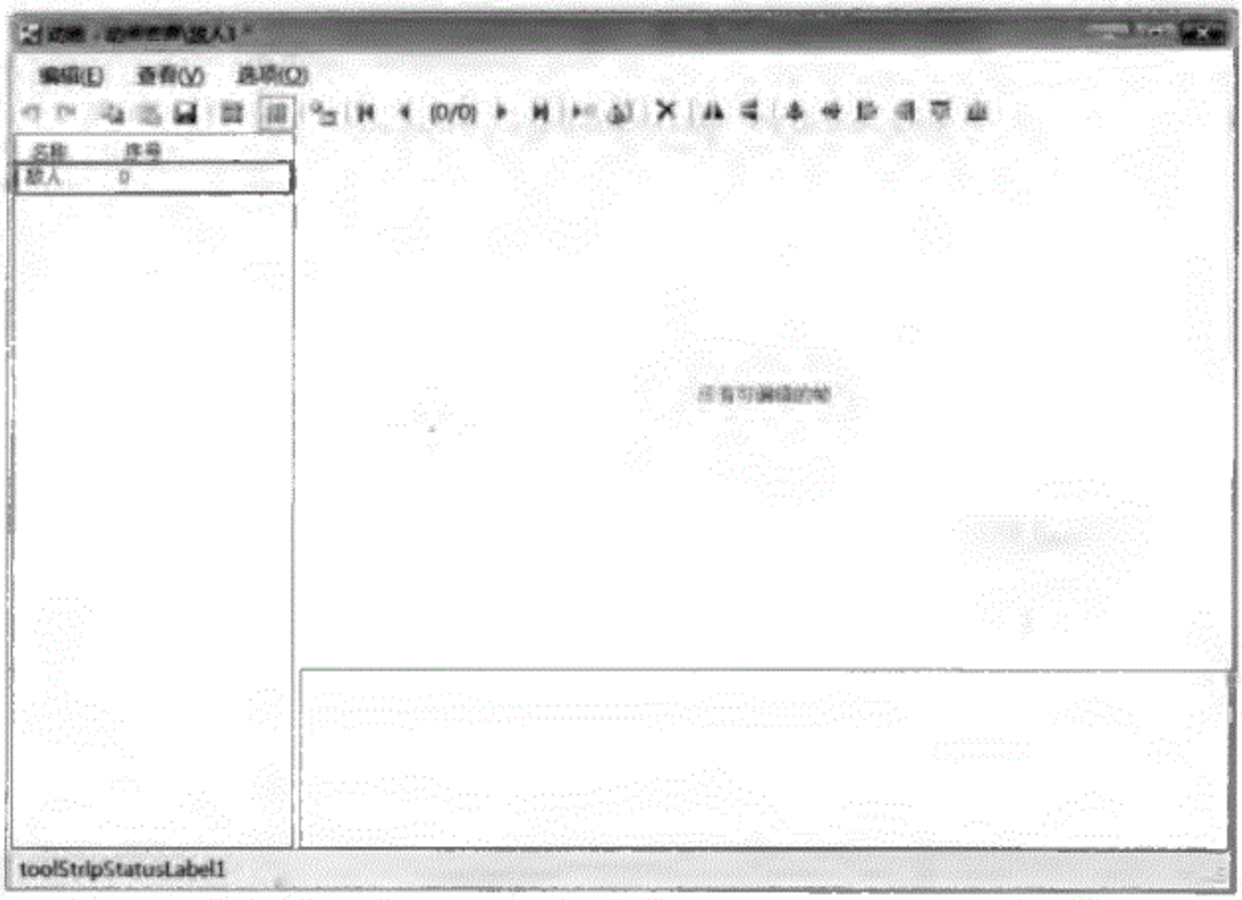


图 4-32

图 4-33

图 4-34

⑥ 在“敌人”这个动作中，使用创建帧工具（功能介绍见前一章）新建两个空白帧，如图 4-35 所示。在这两个帧上放置不同的图片，使之成为动画。

⑦ 在动画创建窗口的工具栏中，单击“单位资源”按钮，打开关联的单位资源窗口如图 4-36 所示，然后选中其中一个炮塔的单元，使用鼠标拖动将这个单位拖至动画帧编辑区域，如图 4-37 所示。

⑧ 因为动画的显示是在动画帧区域的中心位置播放的，所以要用 水平中心对齐和 垂直中心对齐两个工具，将炮塔置于动画帧的中心位置，如图 4-38 所示。在动画帧编辑区域下方的动画帧显示窗口中，单击第二张动画帧，再用同样的方法将另一个炮塔放置在第二张动画帧上，如图 4-39 所示。

⑨ 制作完成后，就可以通过预览工具，如图 4-40 所示，来观察动画效果，如图 4-41

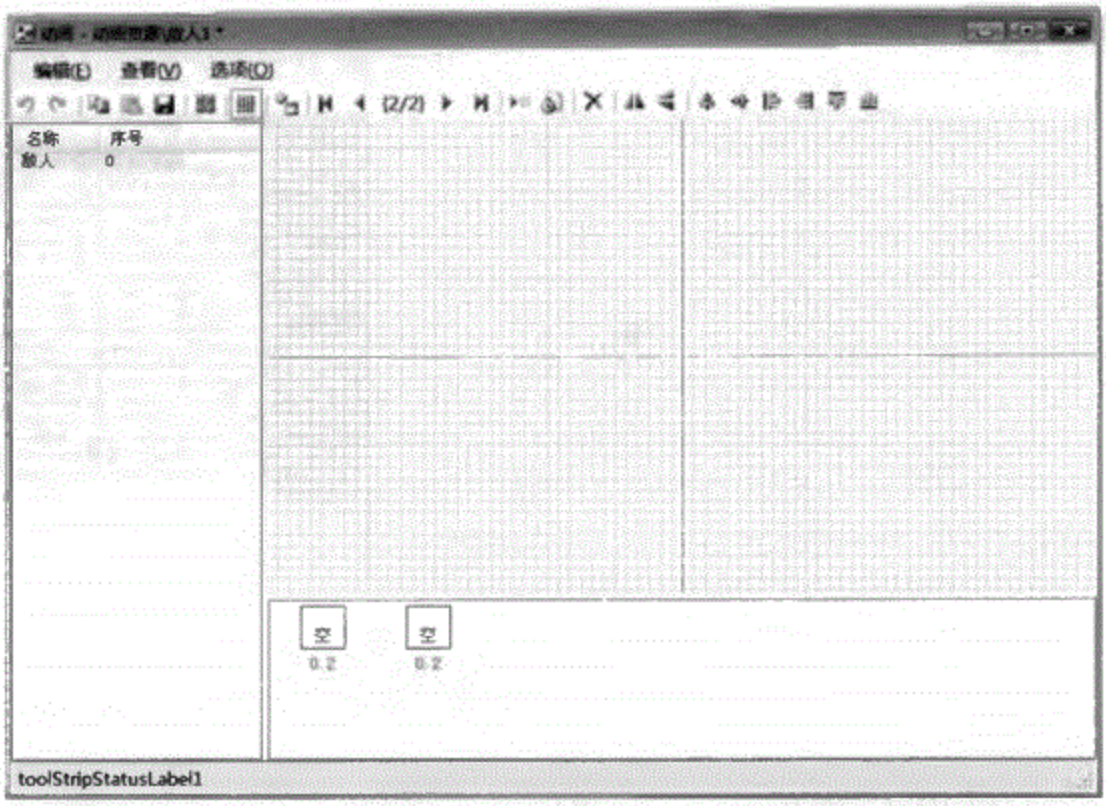


图 4-35

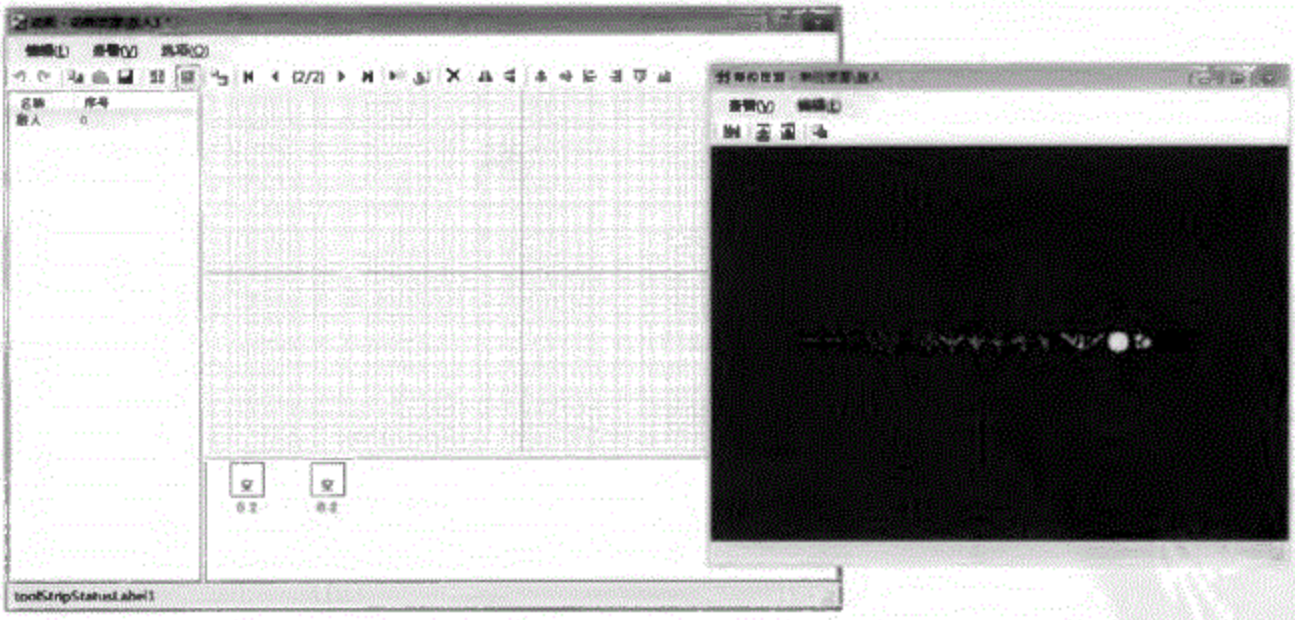


图 4-36

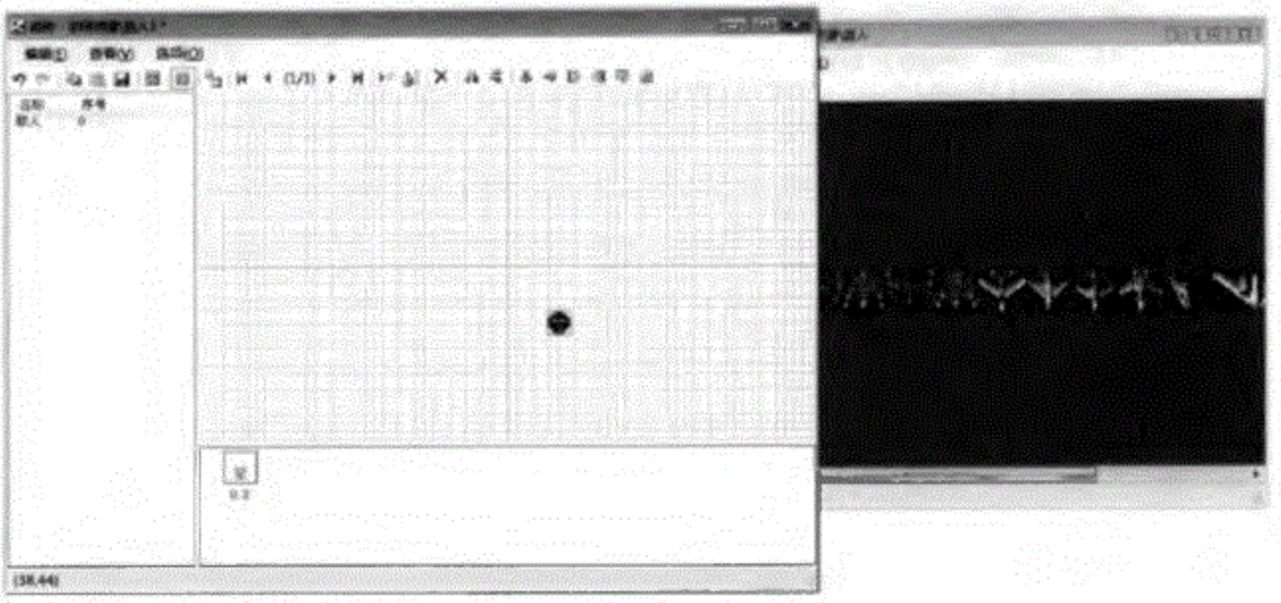


图 4-37

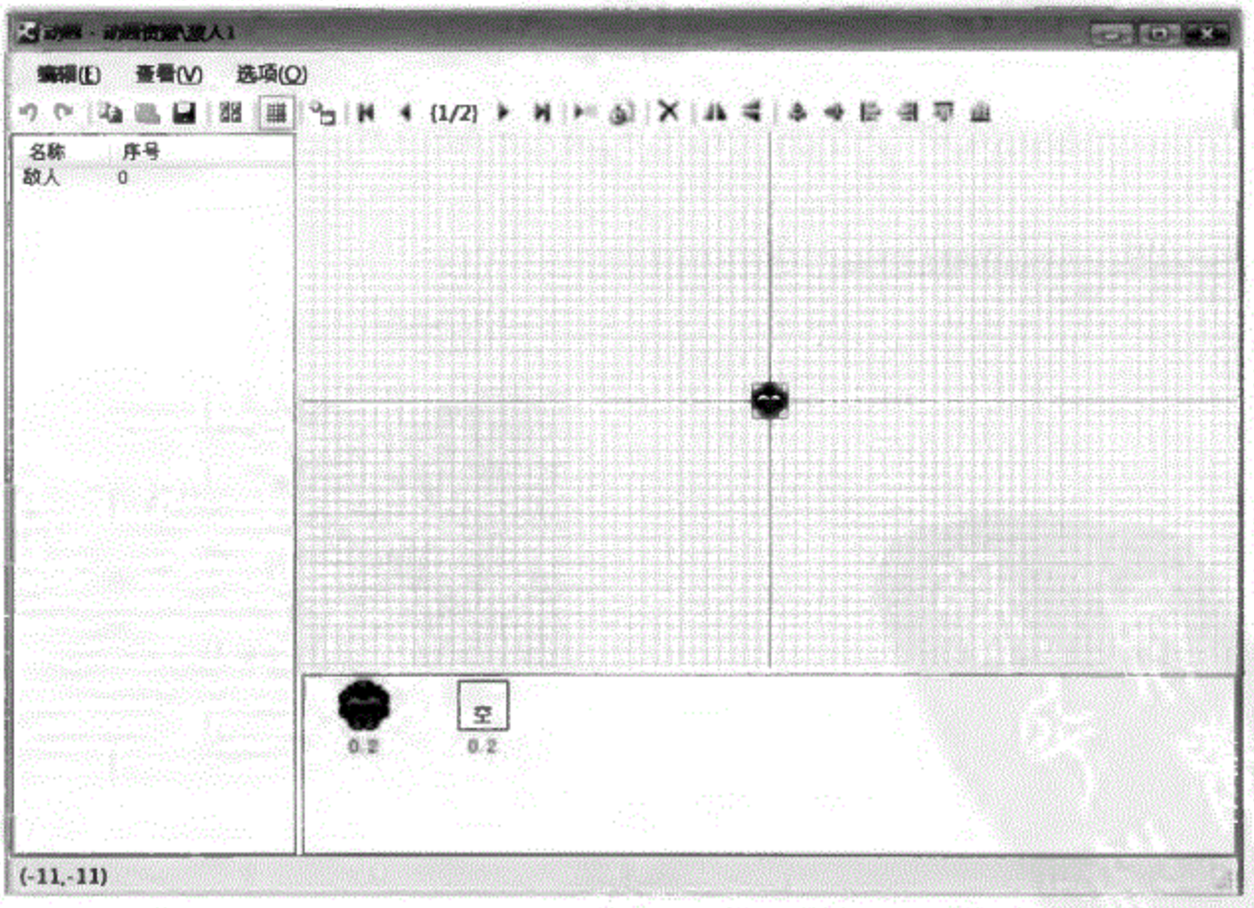


图 4-38

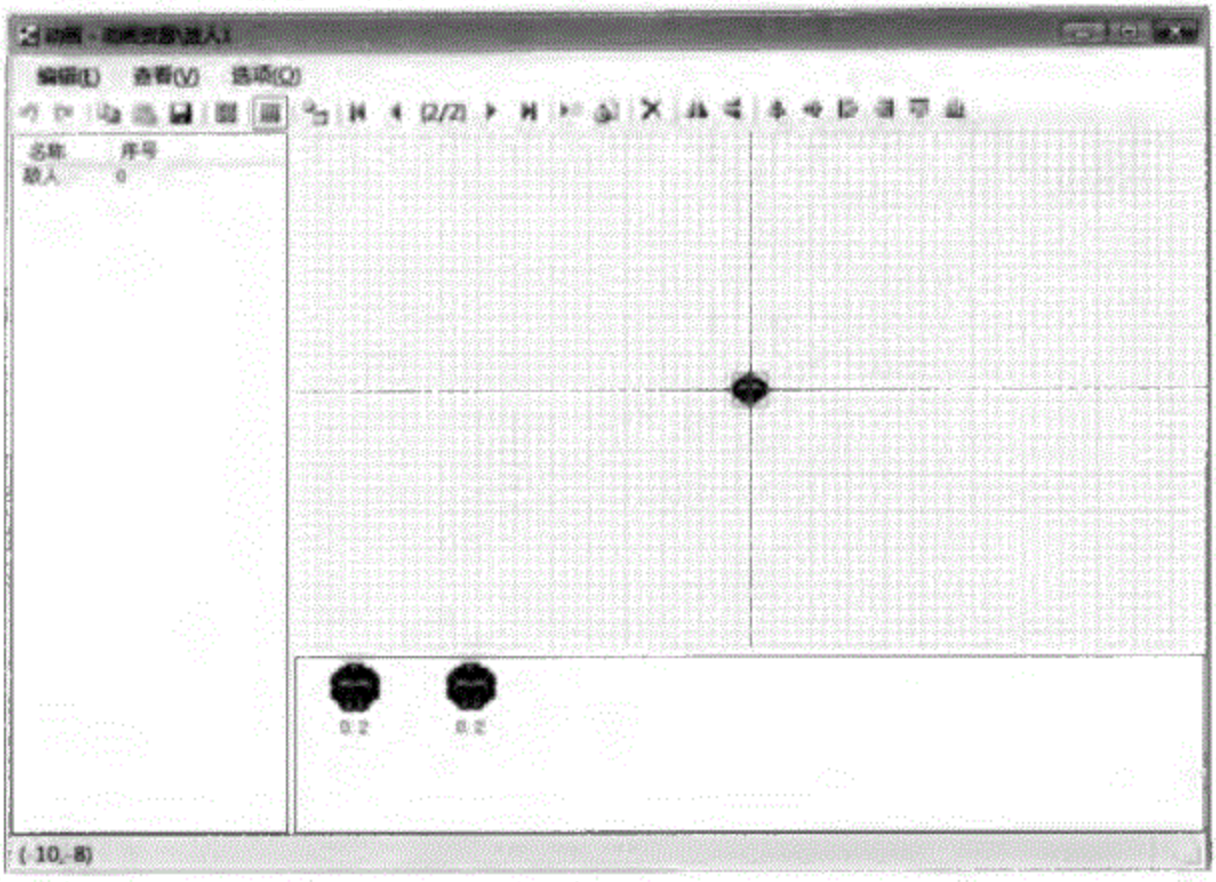


图 4-39

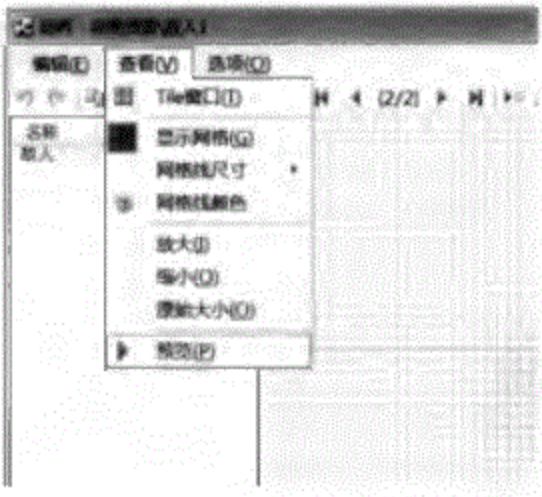


图 4-40

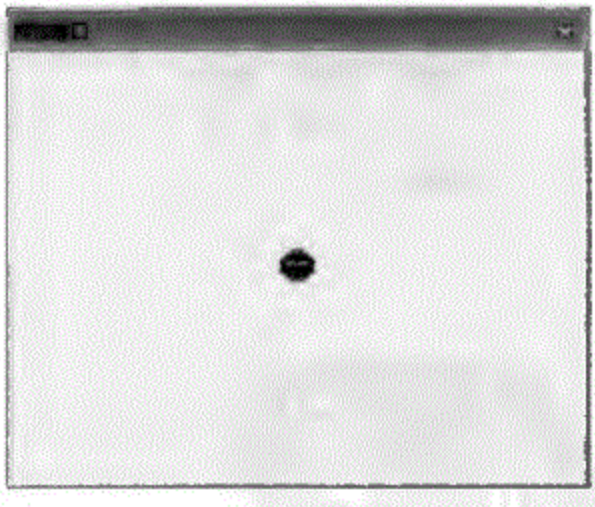


图 4-41

所示。

⑩ 按照以上步骤制作“敌人 2”，但是“敌人 2”是由上下两个单元构成的，从相关联的单元资源将“敌人 2”的两个单元分别拖入动画帧制作区域，如图 4-42 所示，然后使用垂直中心对齐工具将两个单元放置在中心线上，而垂直方向上的则是要手工将上下两个单元拼合在一起，如图 4-43 所示。

提示：在制作时，可以将图像放大或者缩小。在动画资源窗口中，菜单栏中的“查看”下拉菜单中的放大和缩小工具可以用来放大或缩小图片。如果需要还原成图片的原始大

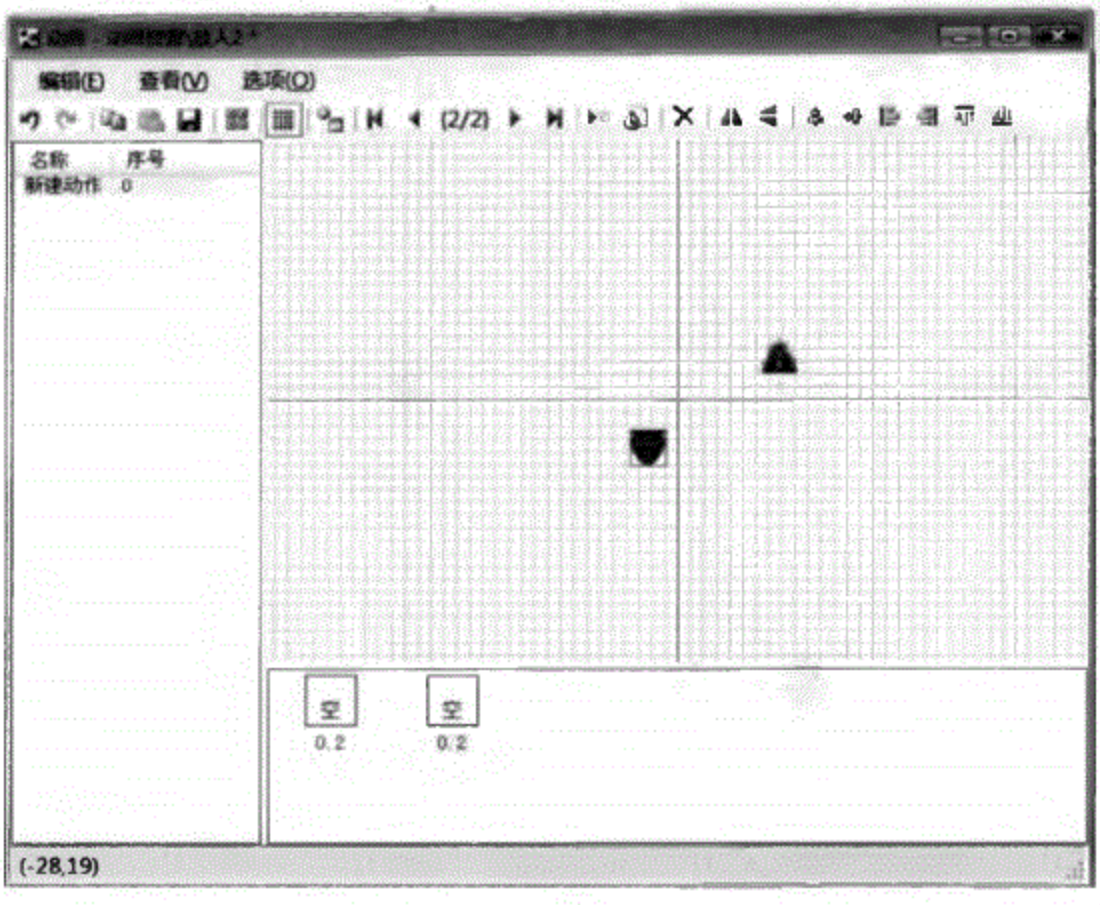

编辑(E)
查看(V)
选项(O)
(2/2)
名称
序号
新建动作 0
0.2
0.2
(-28,19)

图 4–42

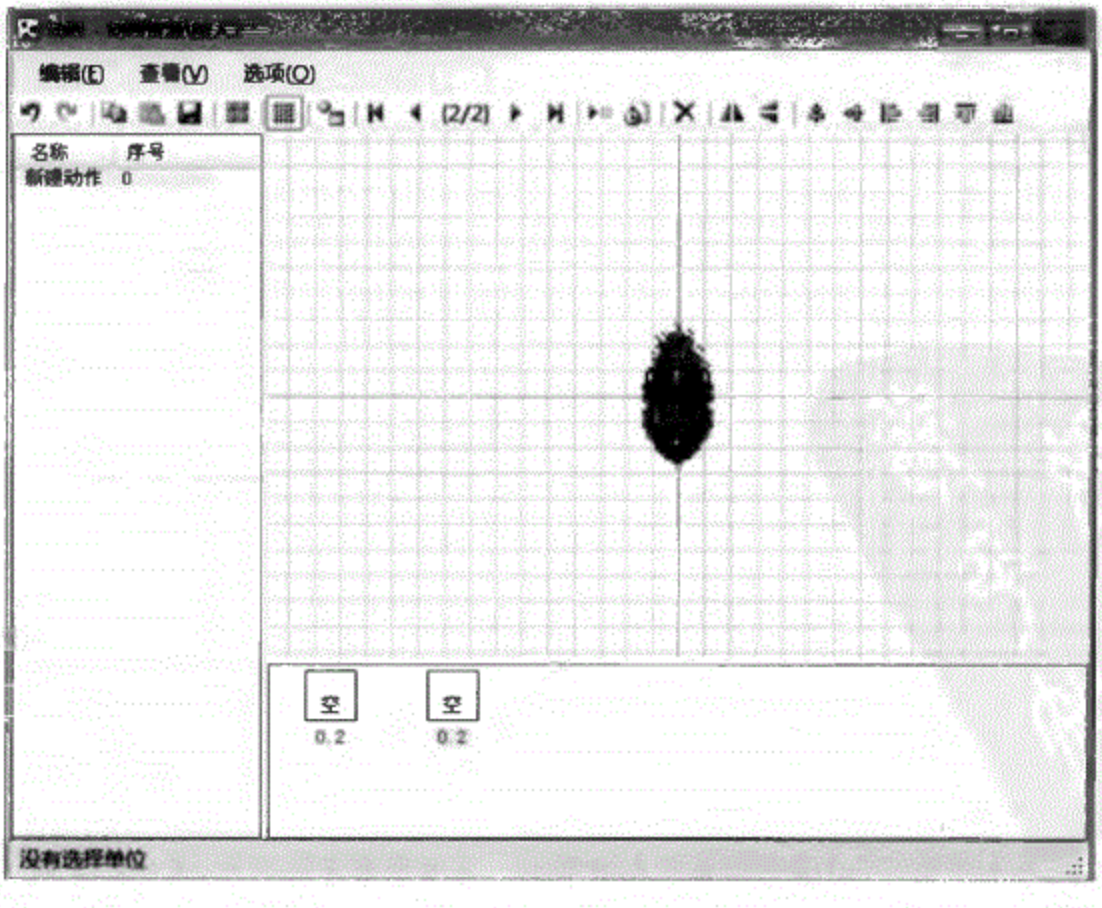

编辑(E)
查看(V)
选项(O)
(2/2)
名称
序号
新建动作 0
0.2
0.2
没有选择单位

图 4–43

小，只要单击菜单中的“原始大小”命令，就可以将图片还原成原始大小，如图 4-44 所示。

⑪ 使用同样的方法制作完成“敌人 2”的第二帧，如图 4-45 所示。

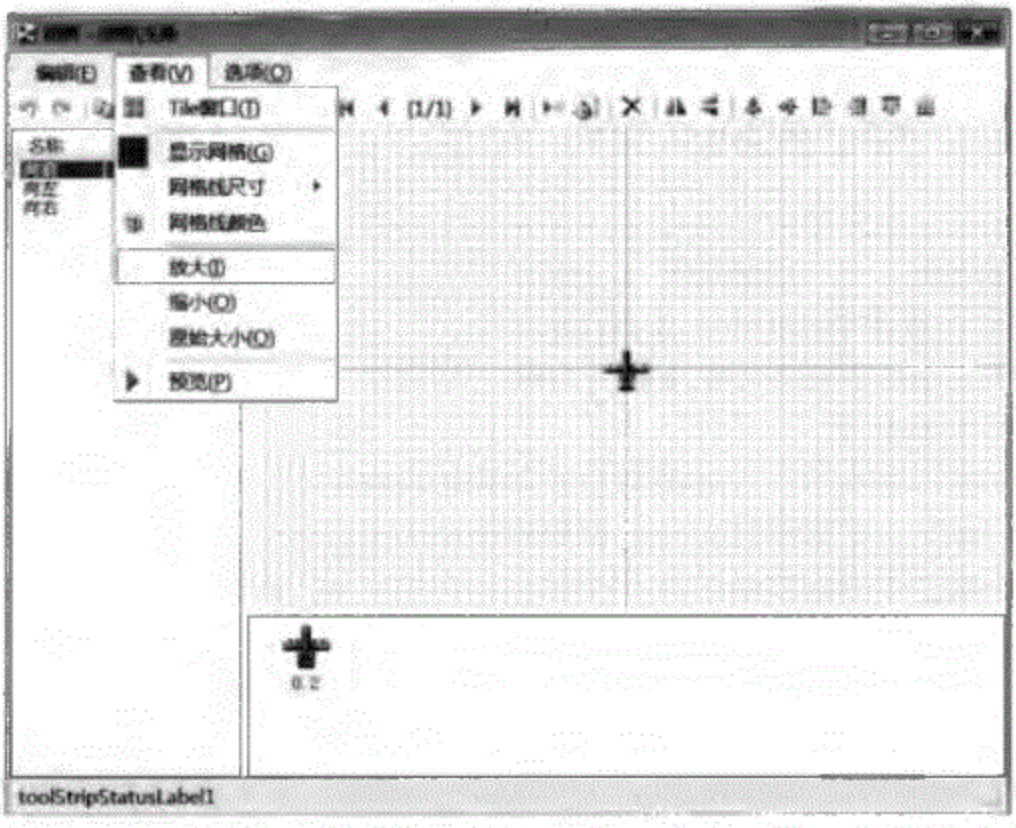


图 4-44

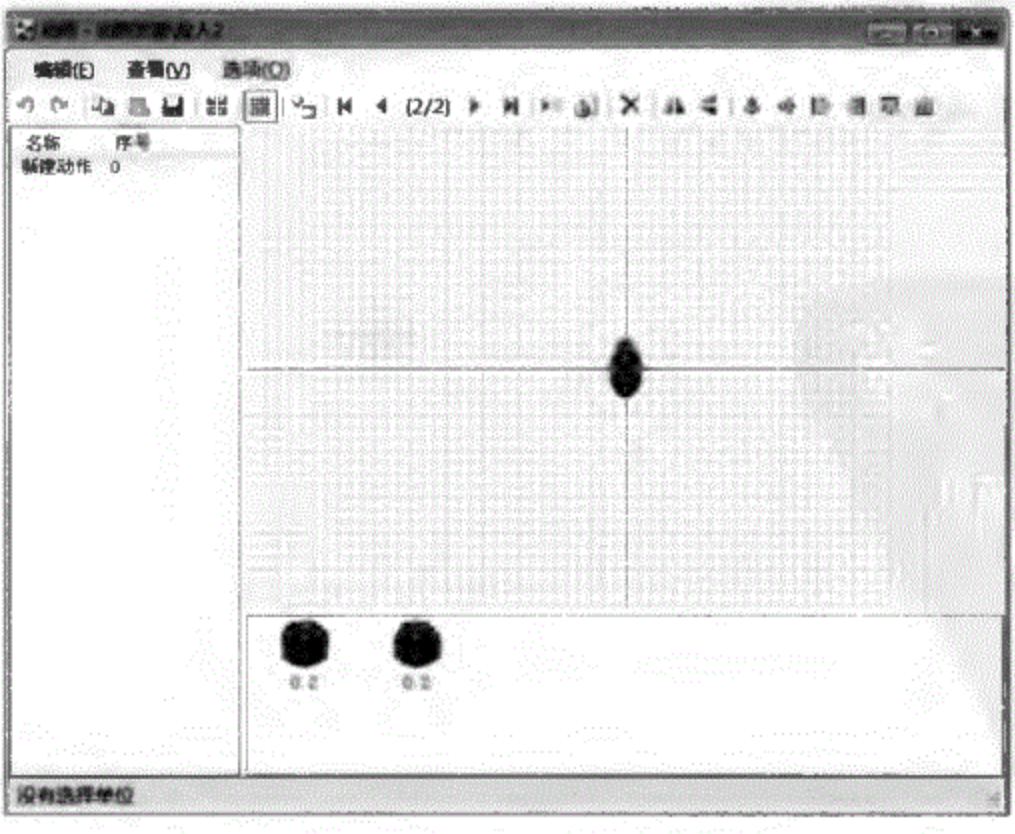


图 4-45

⑫ 用工具栏中的预览工具，检查“敌人 2”的动画效果，如图 4-46 所示。

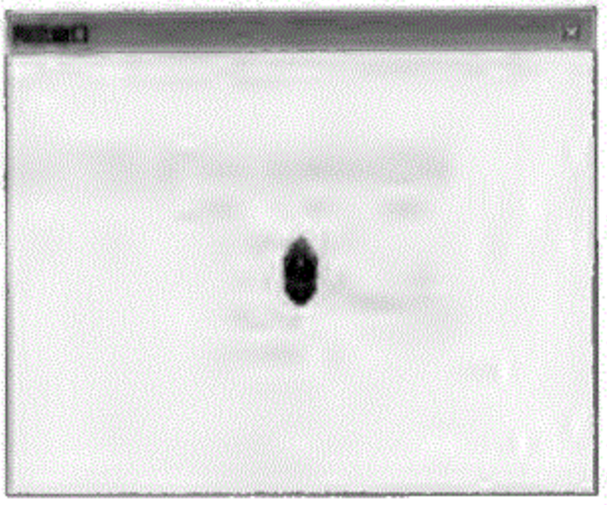


图 4-46

⑬ 将“玩家”用同样的方法导入到动画资源窗口中，这次要创建 4 个不同的动作。在前 3 个动作中，只要放入单帧即可，即“向前”、“向左”、“向右”，如图 4-47 至图 4-49 所示。

⑭ 在第 4 个即“受伤”动作中，要为玩家的受伤做出一闪一闪的特效，受伤的特效只需要创建两个帧即可，并且在第 1 帧中放入玩家正常的图片，第 2 帧为空帧即可，如图 4-50 所示。

⑮ 用预览工具检查，若发现受伤的飞机闪的频率过慢，则需要调整帧与帧之间的间隔时间，将闪动的频率加快。单击创建帧右侧的 按钮，对当前帧的持续时间进行重新设定。单击打开设定时间工具，系统弹出“获取字符串”窗口，在该窗口中将时间从 0.2 秒调至 0.1 秒钟，如图 4-51 所示；另一帧也改变为相同时间，如图 4-52 所示。

⑯ 使用相同方法制作完成“游戏开始”（见图 4-53）、“游戏结束”（见图 4-54）、“爆炸”（见图 4-55）、“玩家子弹”（见图 4-56）、“敌人子弹”（见图 4-57、图 4-58），如图 4-59 所示。

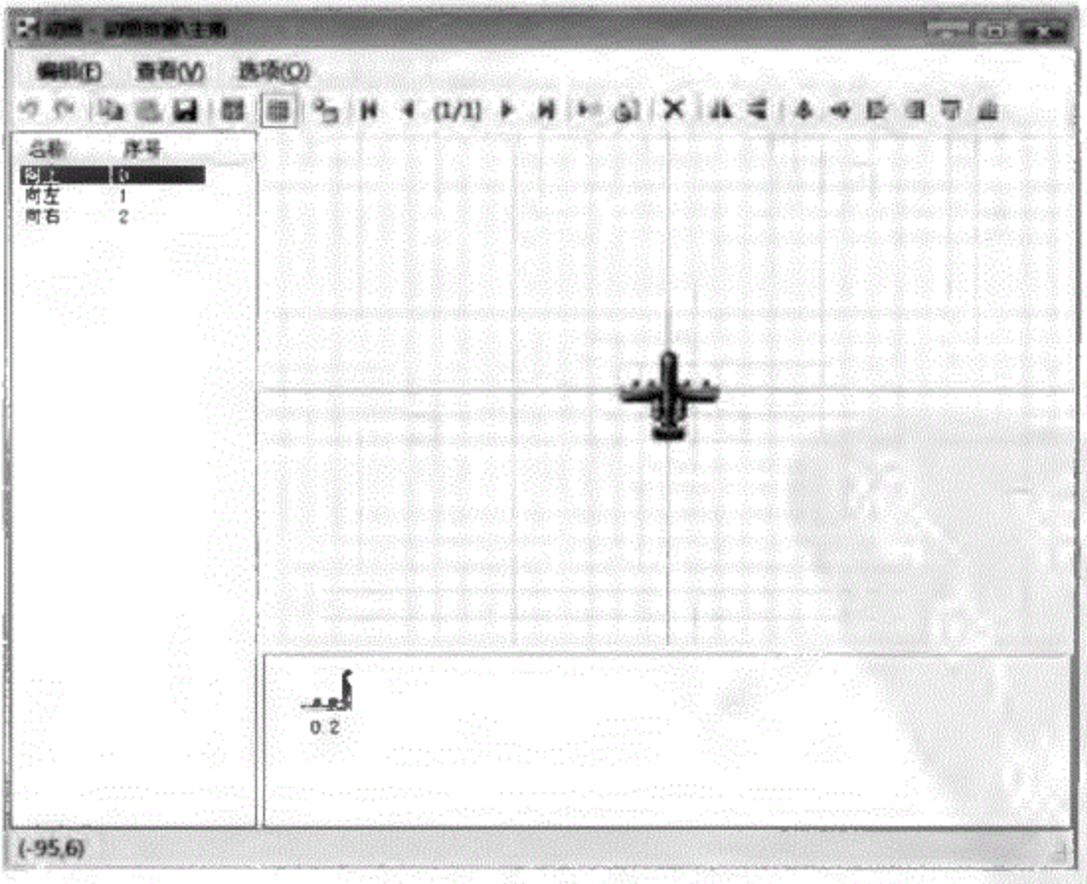


图 4-47

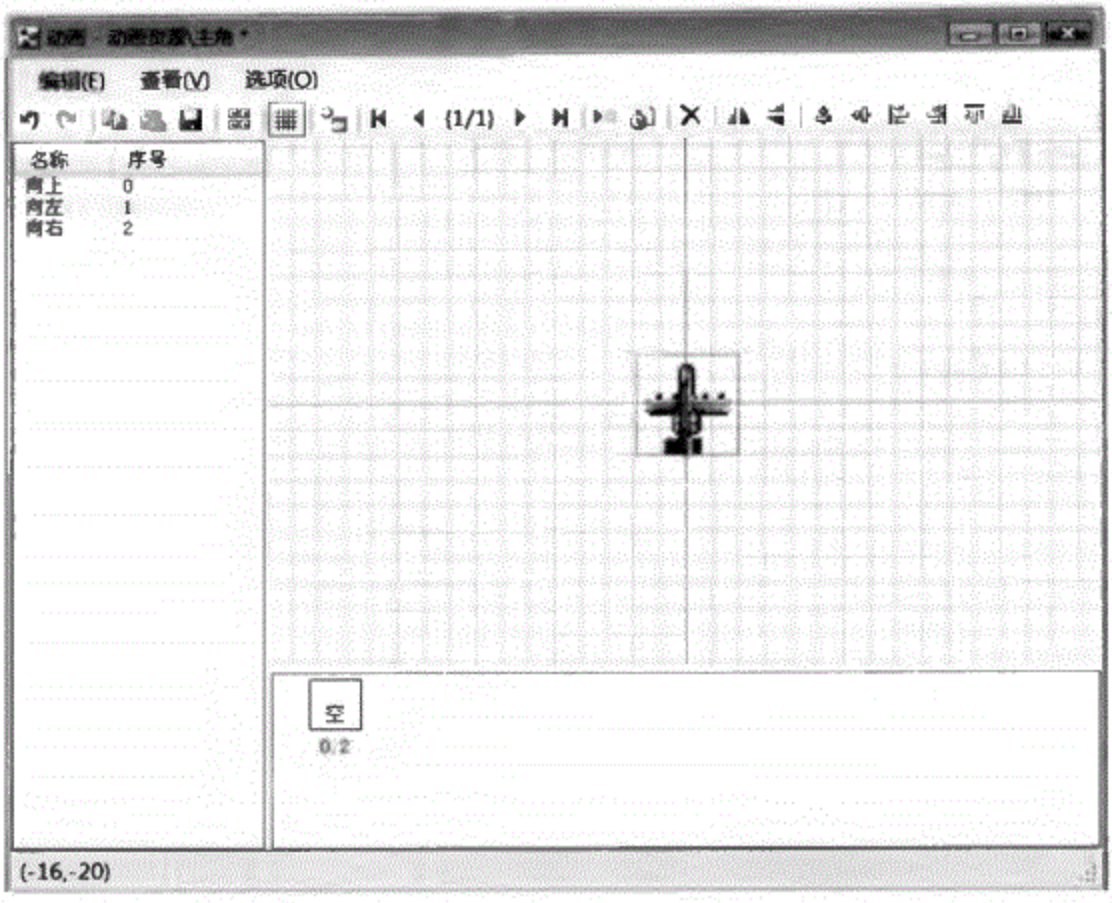

编辑(E)
查看(V)
选项(O)
(1/1)
名称
序号
向上
0
向左
1
向右
2
호
0.2
(-16,-20)

图 4–48

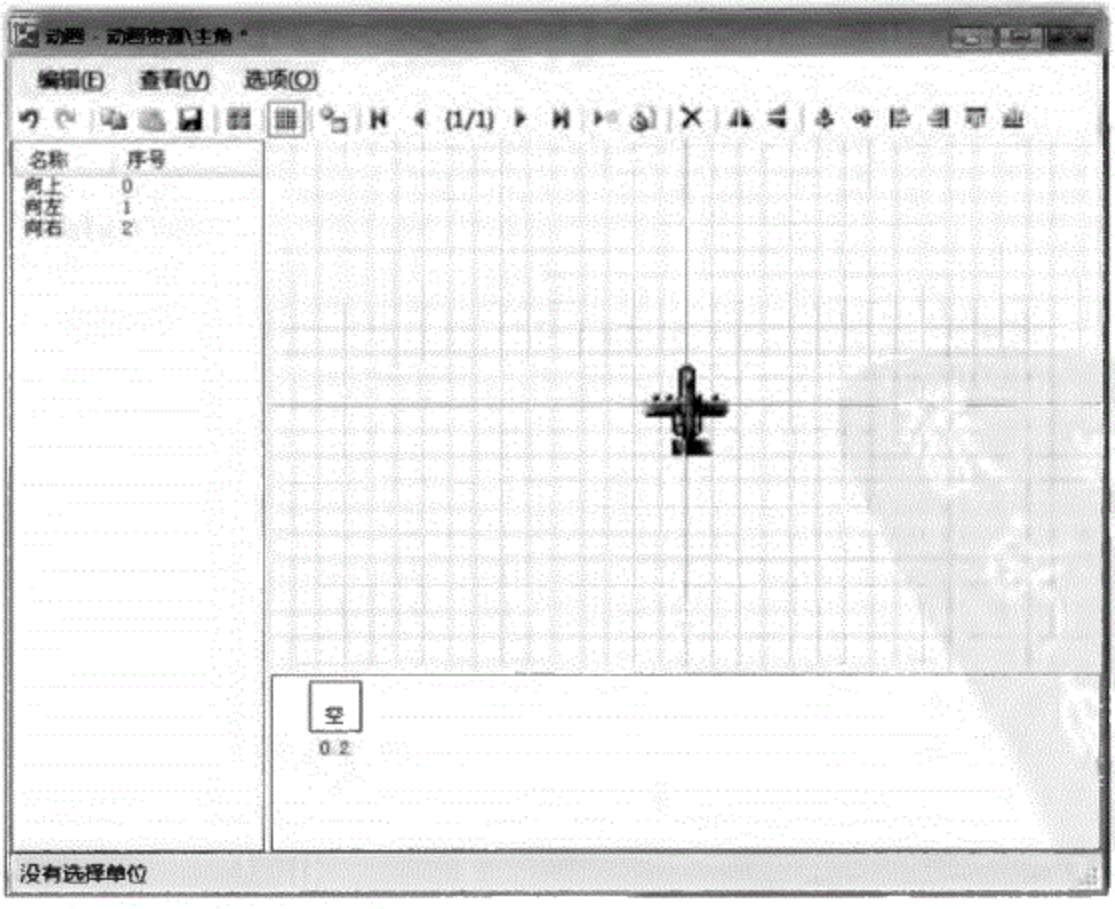

编辑(E)
查看(V)
选项(O)
(1/1)
名称
序号
向上
0
向左
1
向右
2
호
0.2
没有选择单位

图 4–49

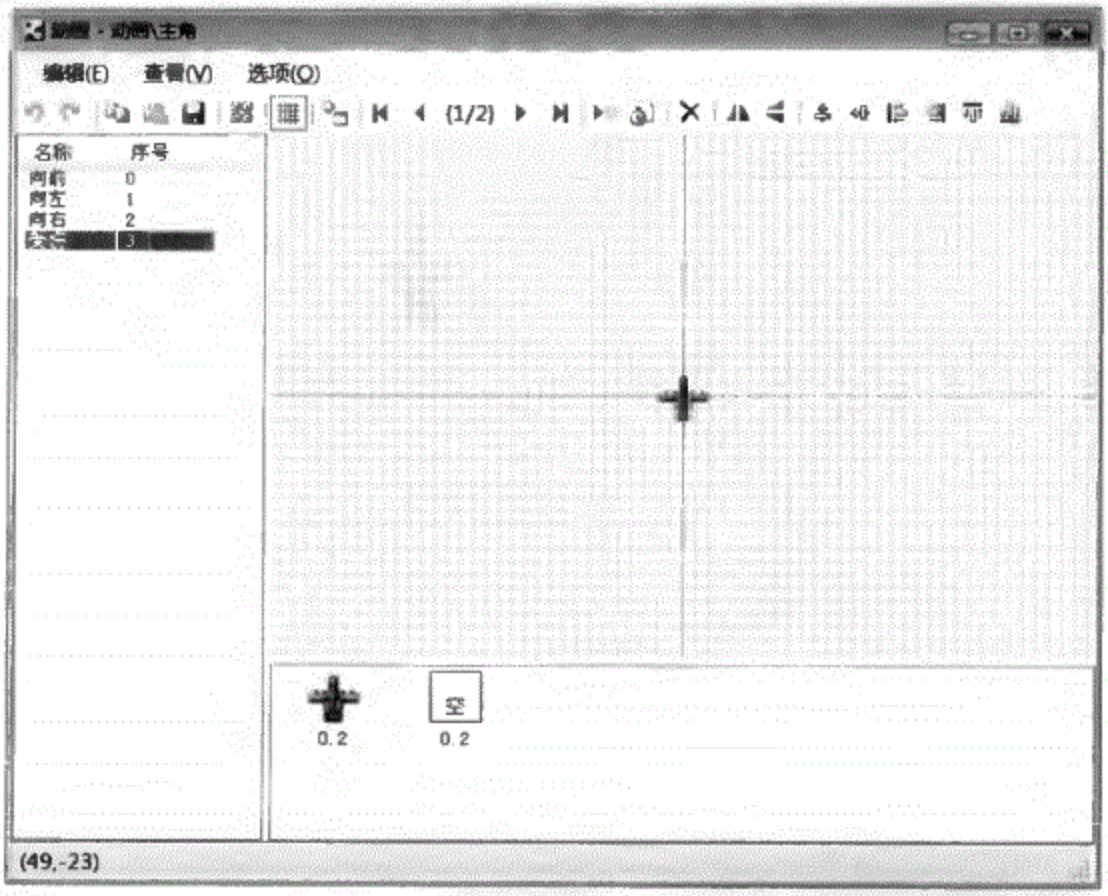


图 4-50

图 4-51

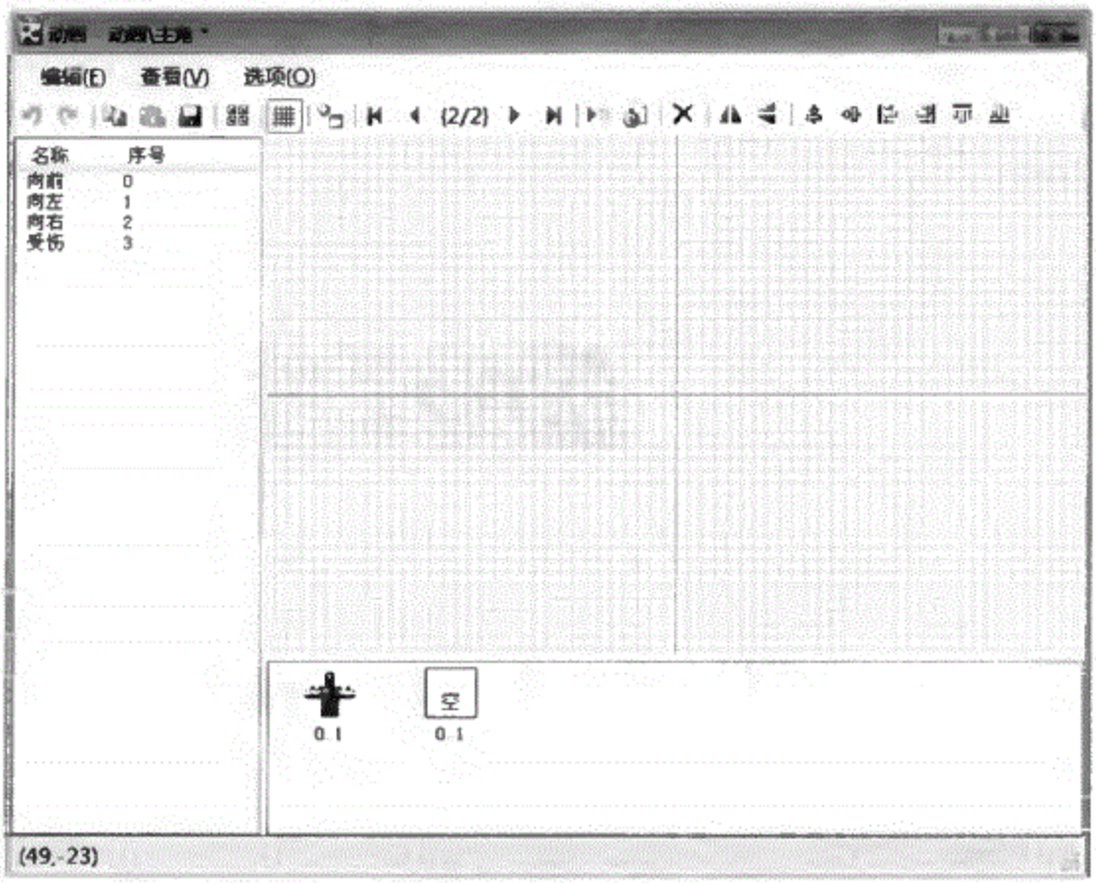

编辑(E)
查看(V)
选项(O)
(2/2)
名称 序号
向前 0
向左 1
向右 2
受伤 3
0.1
0.1
(49,-23)

图 4-52

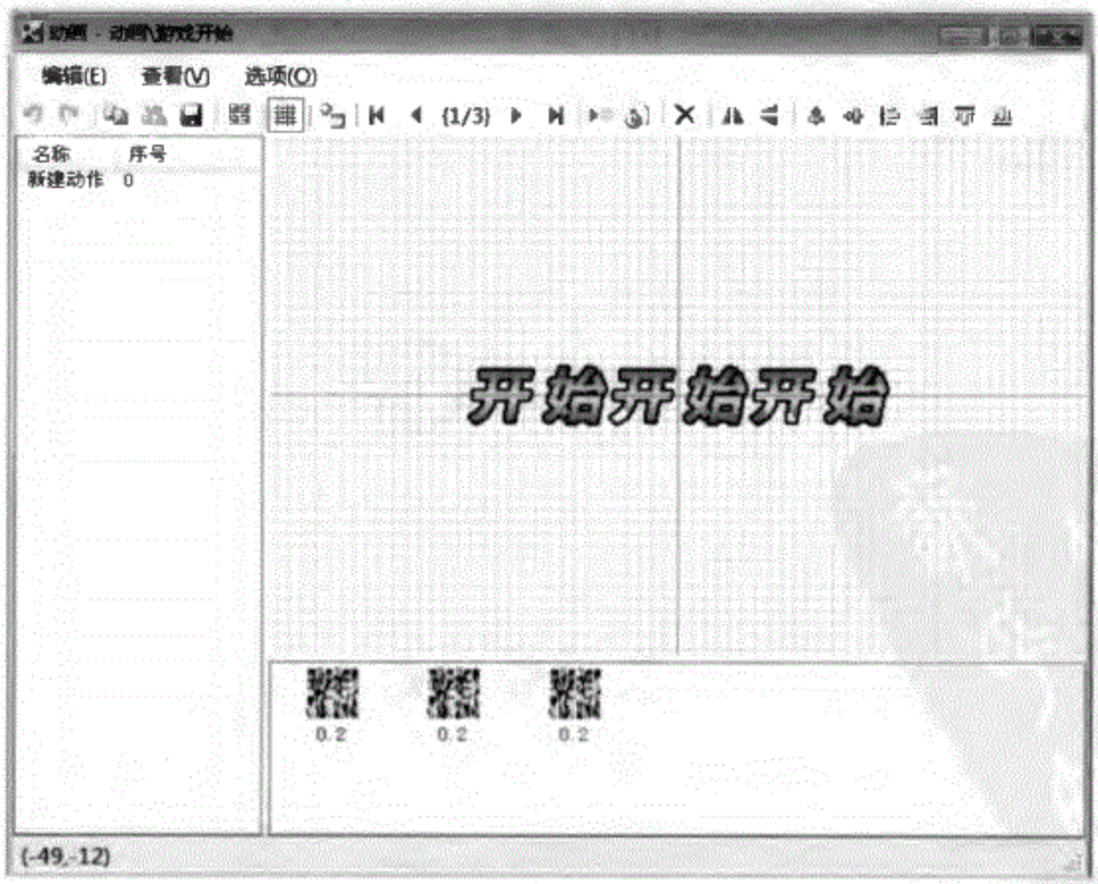

编辑(E)
查看(V)
选项(O)
(1/3)
名称 序号
新建动作 0
开始开始开始
0.2
0.2
0.2
(-49,-12)

图 4-53

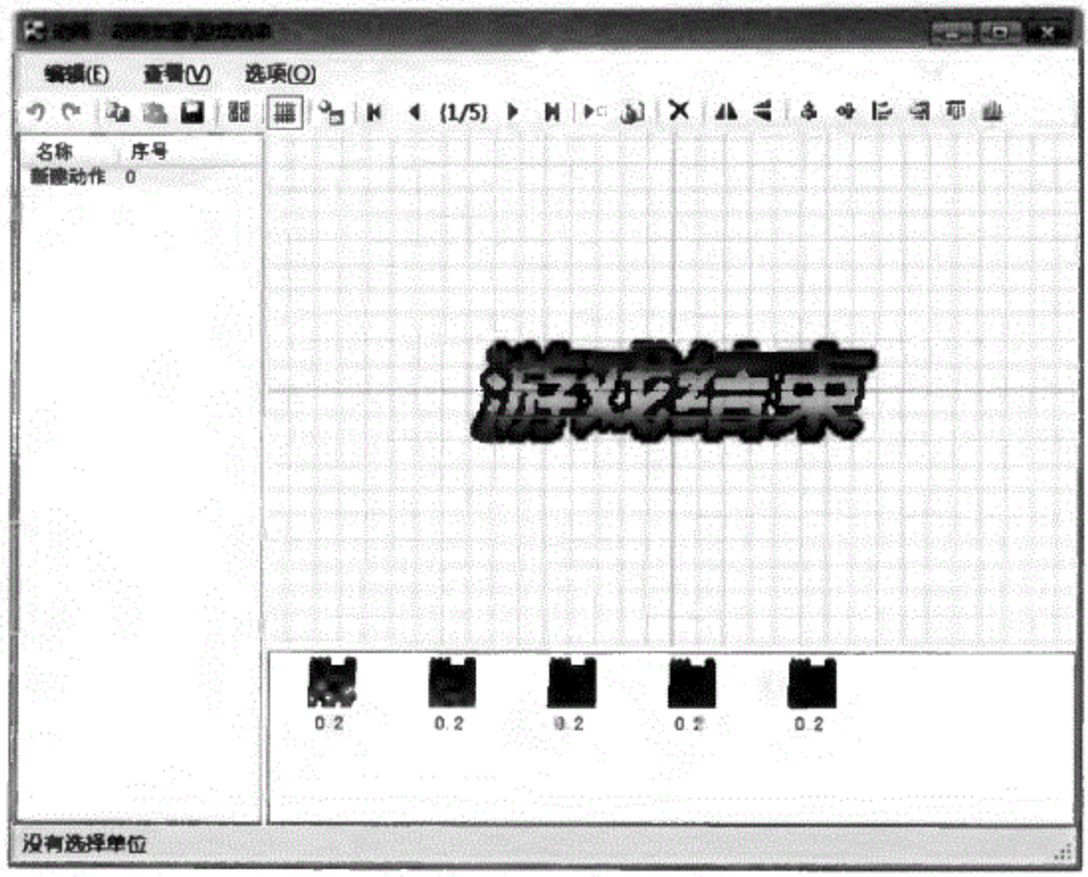

编辑(E) 查看(V) 选项(O)
(1/5)
名称 序号
新建动作 0
游戏结束
0.2 0.2 0.2 0.2 0.2
没有选择单位

图 4-54

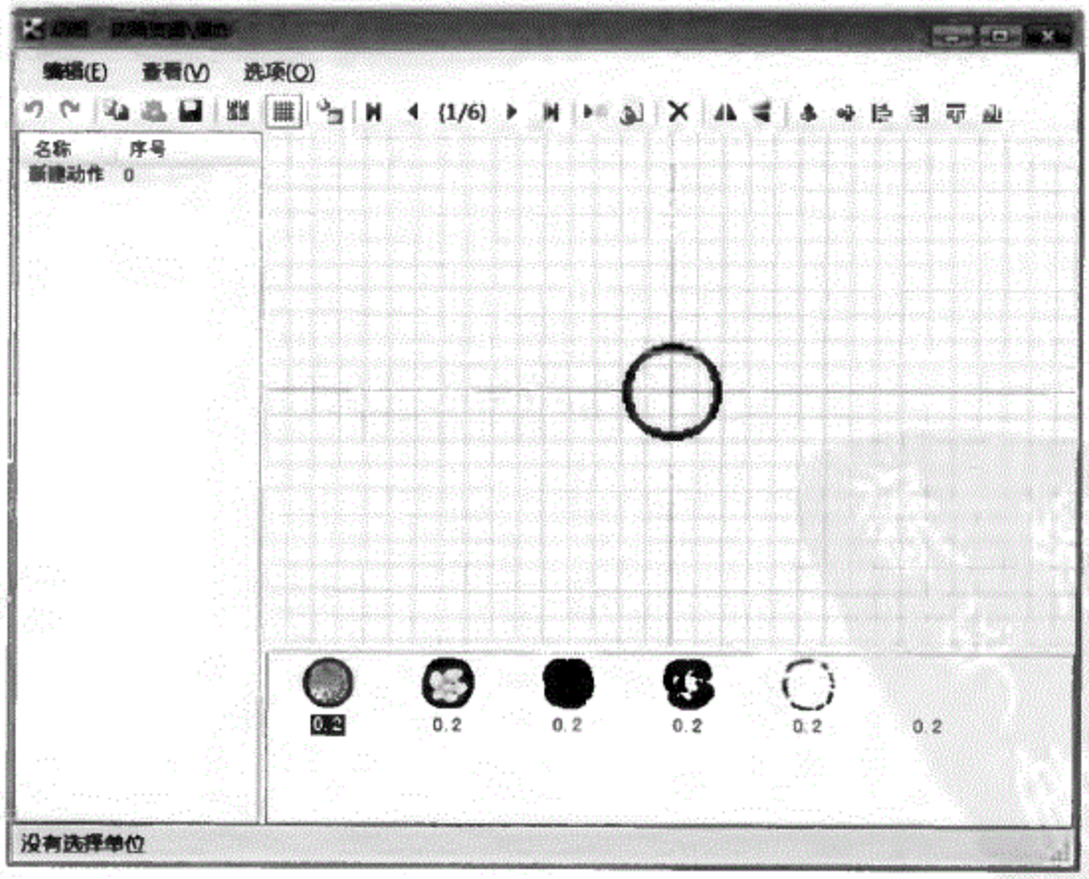

编辑(E) 查看(V) 选项(O)
(1/6)
名称 序号
新建动作 0
0.2 0.2 0.2 0.2 0.2 0.2
没有选择单位

图 4-55

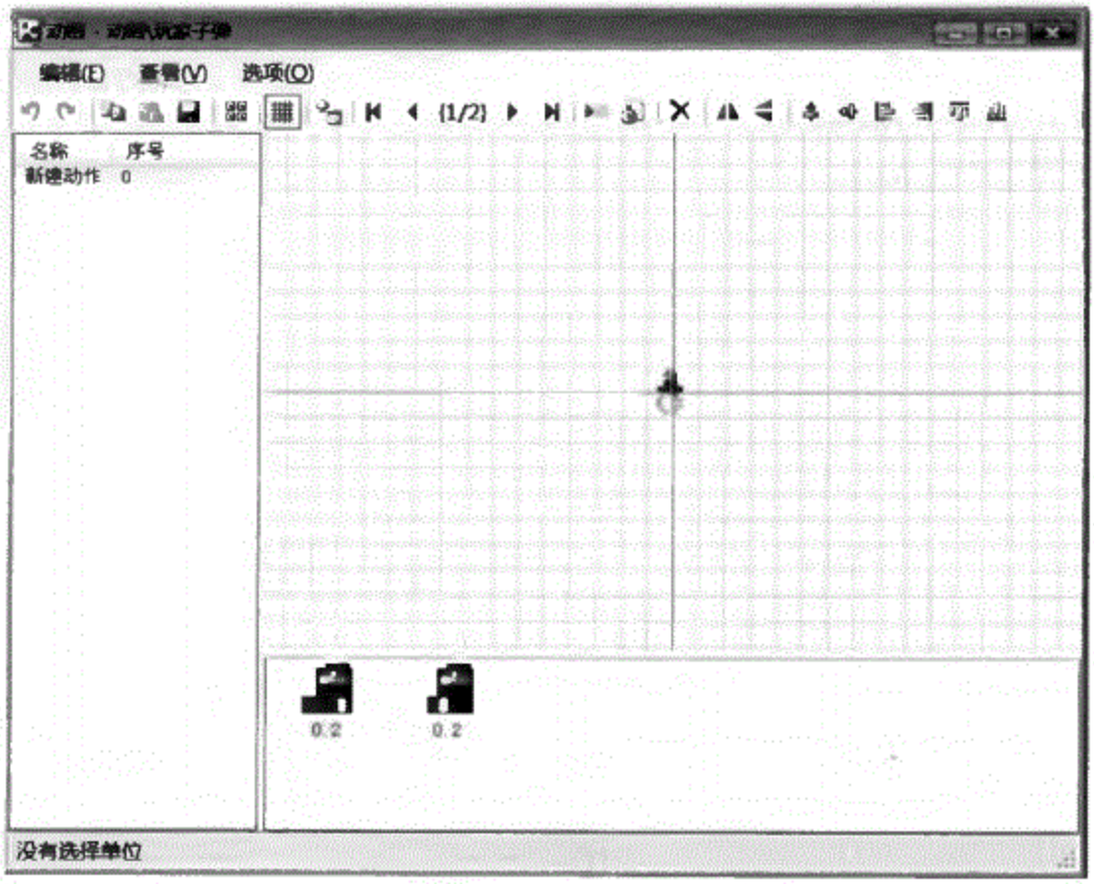

编辑(E)
查看(V)
选项(O)
(1/2)
名称
序号
新建动作 0
0.2
0.2
没有选择单位

图 4-56

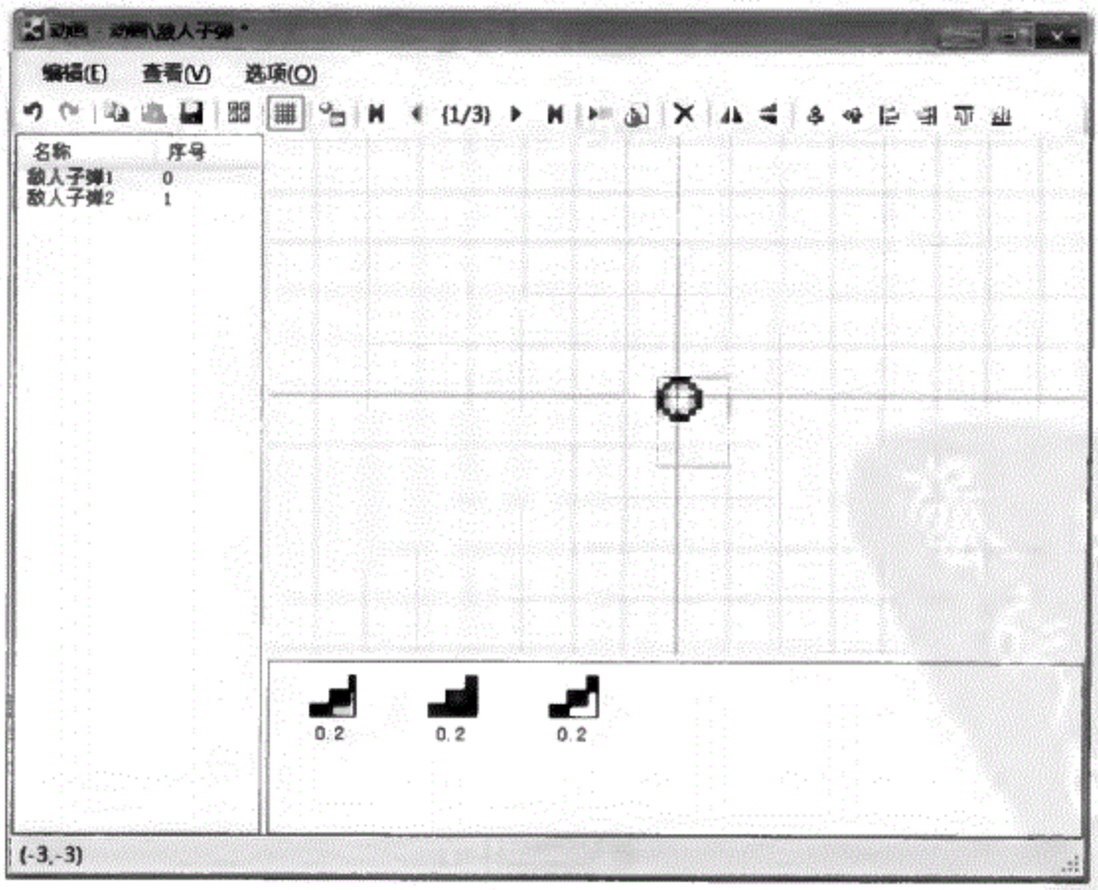

编辑(E)
查看(V)
选项(O)
(1/3)
名称
序号
敌人子弹1 0
敌人子弹2 1
0.2
0.2
0.2
(-3,-3)

图 4-57

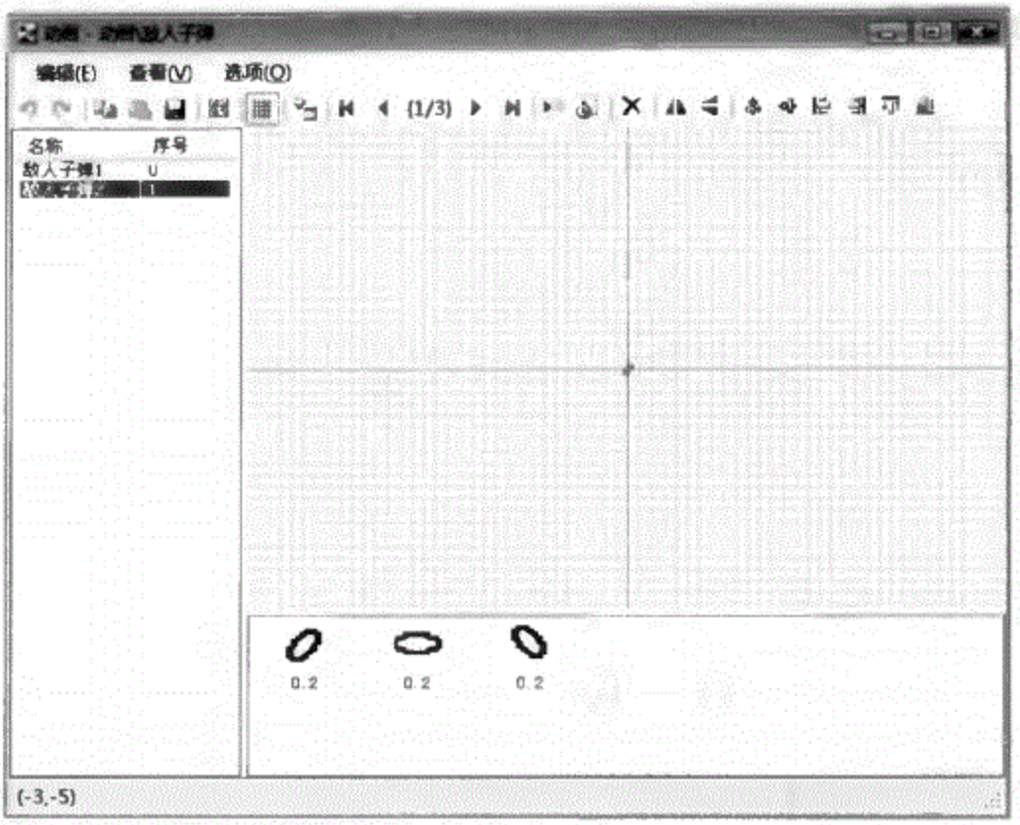

编辑(E)
查看(V)
选项(O)
(1/3)
名称
序号
敌人子弹1
0
0.2
0.2
0.2
(-3,-5)

图 4-58


纵版射击
图像资源
enemy.png
hpbar01.png
hpbar.png
hpbar_f01.png
hpbar_f.png
map01.png
over.png
pl01.png
plboom.png
ppppp.png
start.png
单位资源
敌人
地图
游戏结束
主角
爆炸
子弹
游戏开始
音乐
fire.wav
欧若拉.mp3
动画
敌人1
敌人2
游戏结束
主角
爆炸
游戏开始
玩家子弹
敌人子弹

图 4-59

4.5 游戏界面制作

① 创建“游戏界面”文件夹，如图 4-60 所示。在这个文件夹中，将放置血槽的制作文件和游戏开始、结束的动画所构成的 UI 文件。

图 4-60

② 首先创建游戏界面文件，如图 4-61 所示，单击相应按钮后系统弹出“新建 UI”窗口，在“UI 的名称”文本框中输入 UI 的名称“游戏界面”，如图 4-62 所示。

③ 单击“确定”按钮后，就可以在工程视图中看到名为“游戏界面”的 UI 文件，如图 4-63 所示。打开已经存在的 UI 文件，进入“UI 编辑器”窗口，如图 4-64 所示。

图 4-61

图 4-62

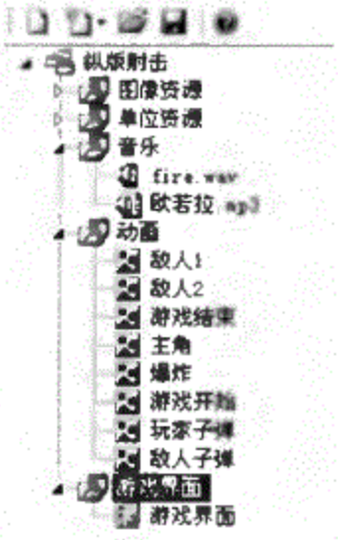


图 4-63

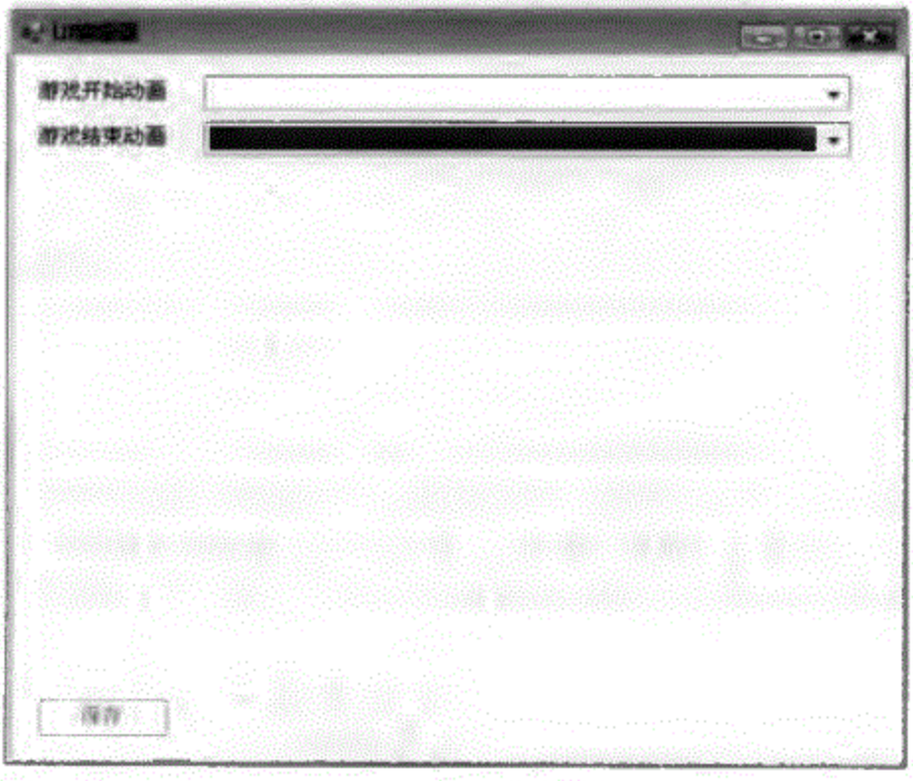


图 4-64

④ 在“UI 编辑器”窗口中有两个下拉列表框：“游戏开始动画”和“游戏结束动画”。在“游戏开始动画”下拉列表框中，选择“动画”文件夹中的“游戏开始”动画，如图 4-65 所示。在“游戏结束动画”下拉列表框中，选择“动画”文件夹中的“游戏结束”动画，如图 4-66 所示。

⑤ 制作血槽文件。首先制作玩家血槽，如图 4-67 所示，在弹出的“获取字符串”窗口的“Label 1”文本框中输入“玩家血槽”，如图 4-68 所示。

⑥ 单击“确定”按钮后，再单击工程视图中已建的名为“玩家血槽”的血槽制作文件，打开血槽制作窗口，如图 4-69 所示。在此窗口中先选择“背景图片”复选框，将背景图片变成可编辑状态，然后选择血槽背景图片，如图 4-70 所示；再在下面的血槽图片中选择蓝色血条作为玩家血条，如图 4-71 所示；最后选择“总是显示”复选框，因为玩家的血槽应该是一直显示的。

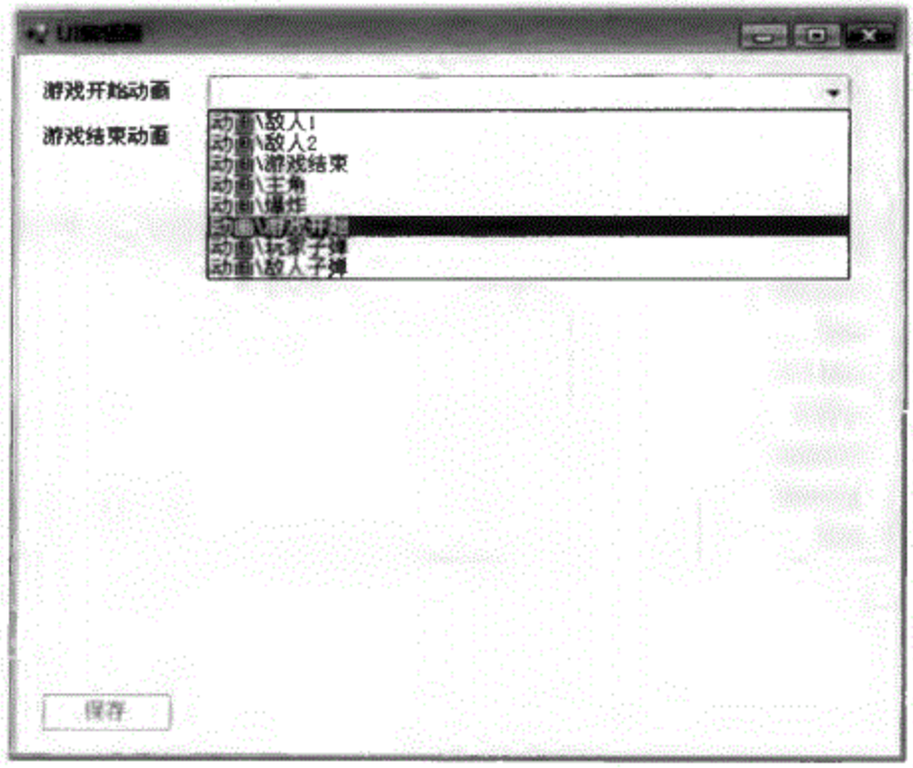

游戏开始动画
游戏结束动画
动画\敌人1
动画\敌人2
动画\游戏结束
动画\主角
动画\爆炸
动画\游戏开始
动画\玩家子弹
动画\敌人子弹
保存

图 4-65

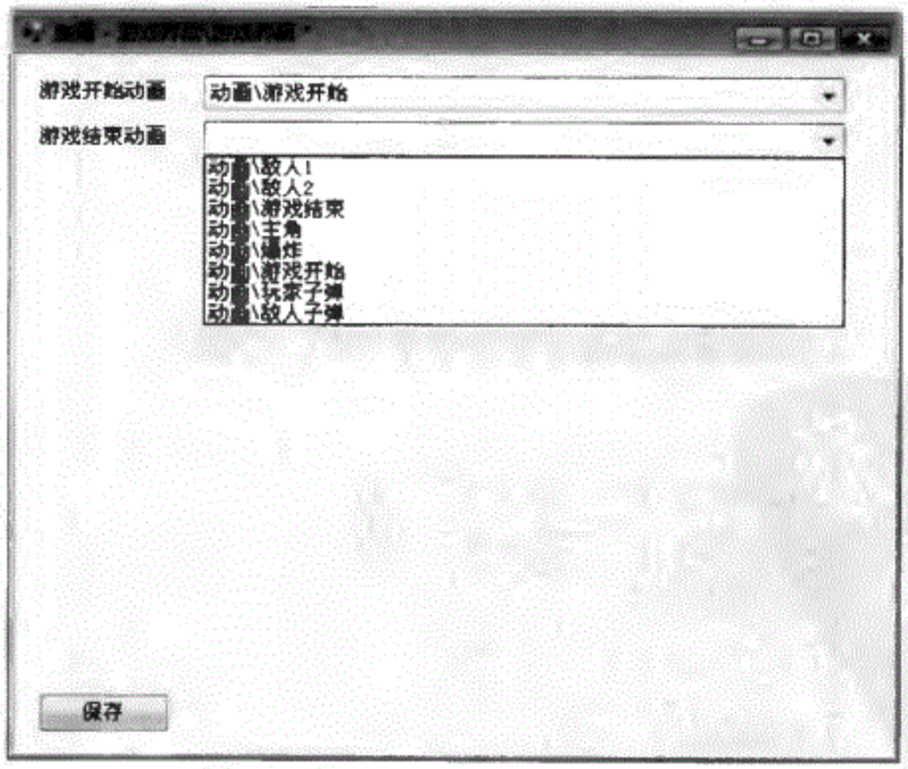

游戏开始动画
动画\游戏开始
游戏结束动画
动画\敌人1
动画\敌人2
动画\游戏结束
动画\主角
动画\爆炸
动画\游戏开始
动画\玩家子弹
动画\敌人子弹
保存

图 4-66

图 4-67

图 4-68

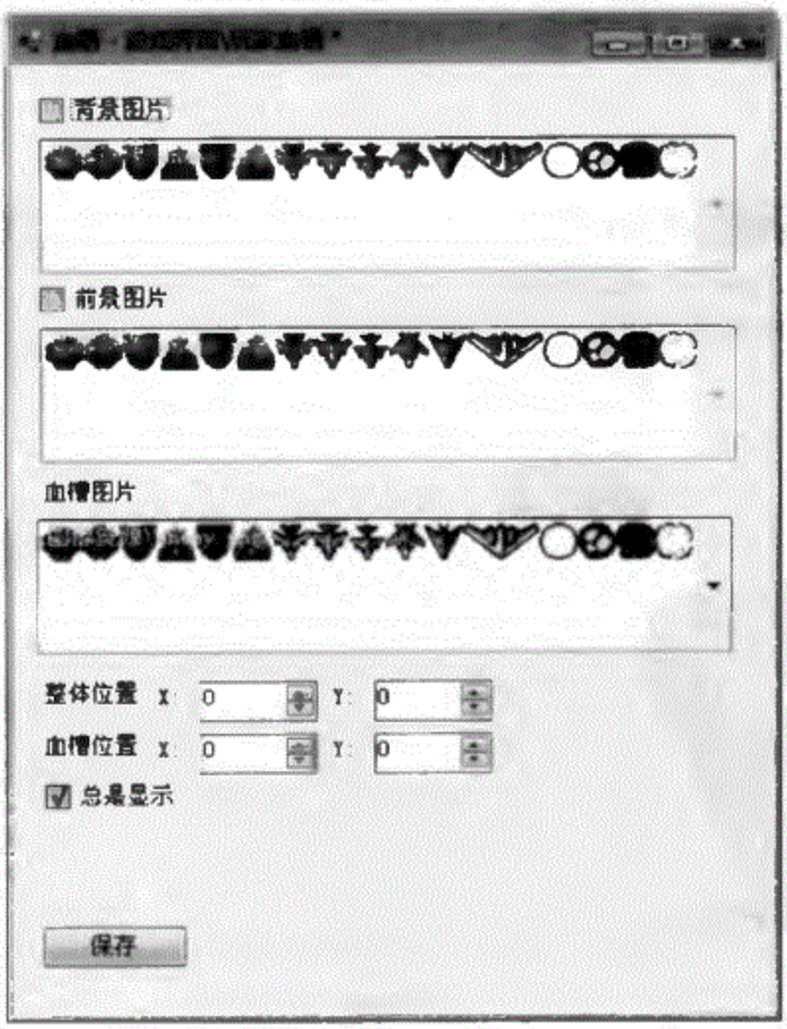


图 4-69

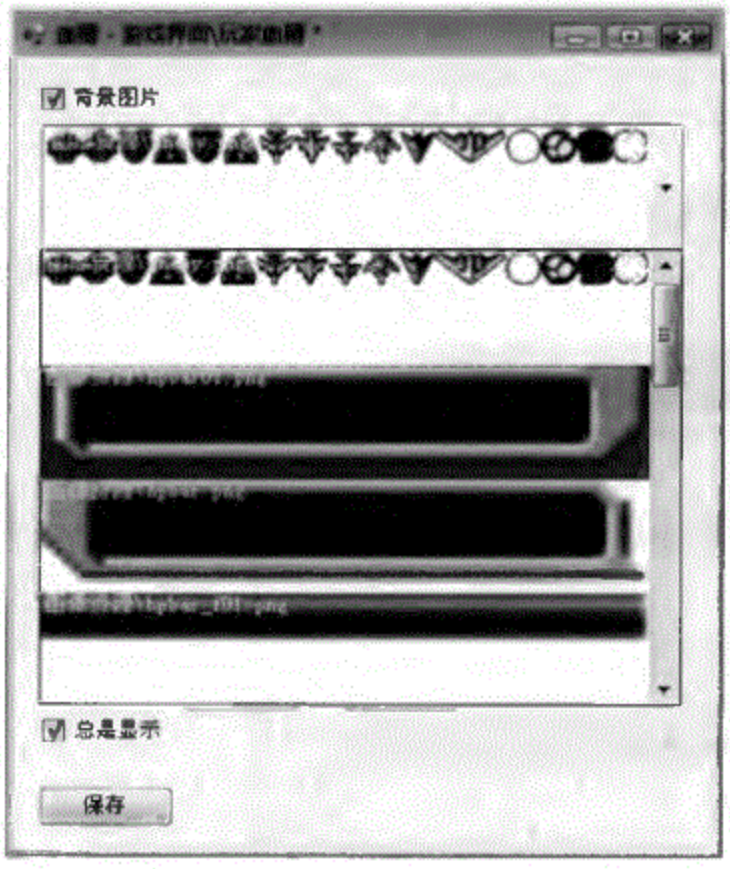


图 4-70

图 4-71

提示：血槽的位置在此时是不能确定的，只有在最后进行检查时逐步调整。

⑦ 用同样方法制作敌人的血槽，只是不选中“总是显示”复选框，因为敌人的血槽只在接触游戏主角时才显示，如图 4-72 所示。

图 4-72

4.6 地图的制作

① 新建“地图”文件夹，如图 4-73 所示，在工程视图中创建地图文件，如图 4-74 所示。

② 在弹出的“创建新地图”对话框中，为了区分不同的地图，将默认名“新建地图”改为“草原”，然后在“使用单位”列表框中找到已经导入到单位资源中的地图图片，最后设定整张地图的大小，由于是纵版的射击游戏，所以将地图的宽度设置为 176，地图高度设置为 1 760，如图 4-75 所示，然后单击“确定”按钮。

③ 在工程视图中打开已建的名为“草原”的地图文件，如图 4-76 所示，在这里可以看到整张地图已经被地图单位资源中的第 1 个单元所填充，然后使用地图单位资源的

图 4-73

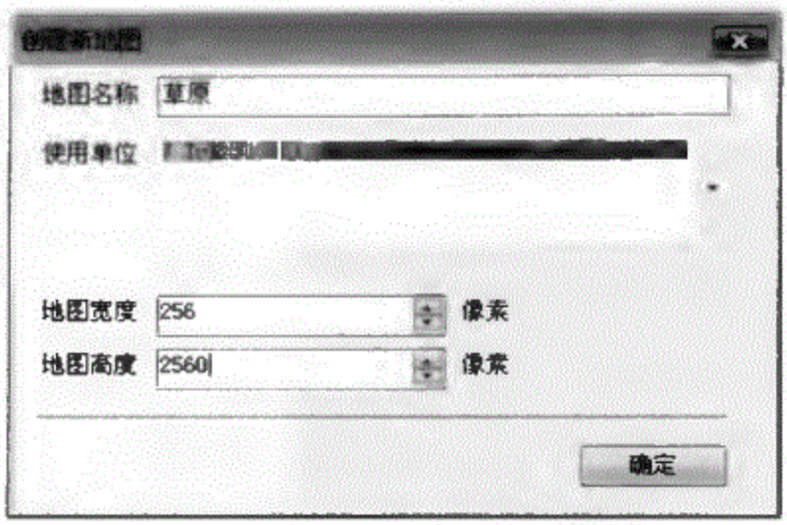


图 4-74

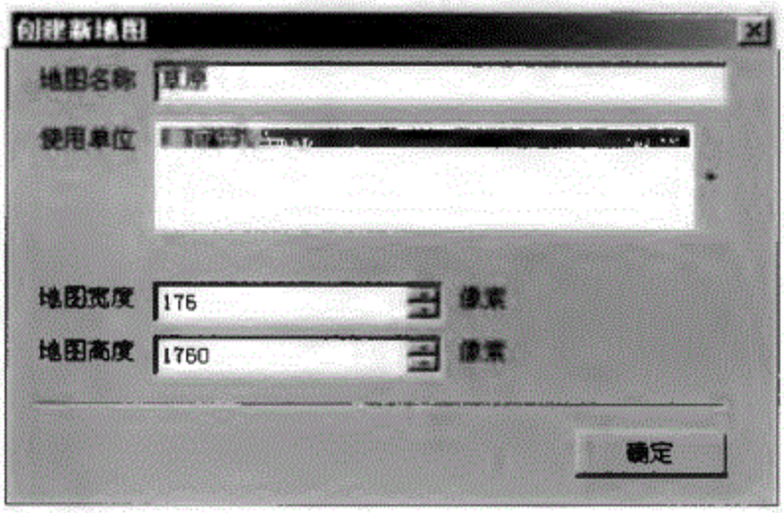


图 4-75

第 2 个单元和第 3 个单元，即另一种地板的纹样和树的图案，将局部的地图铺好，如图 4-77 所示。

提示：在制作过程中，可以使用第一单元铺满新建地图这一原则，将地图的底纹在绘制地图图片时，设置成第一单元。这样打开地图文件时，系统就会自动将第一单元铺好，省去制作者不少时间和精力。

④ 铺好局部地图后，将铺好的局部地图框选，然后使用鼠标拖动，将铺好的局部地图复制到下方未铺的地图上，如图 4-78 所示。

⑤ 最后将地图单位资源中其他的房屋、炮台都放置到地图中去，完成地图的制作，如图 4-79 所示。

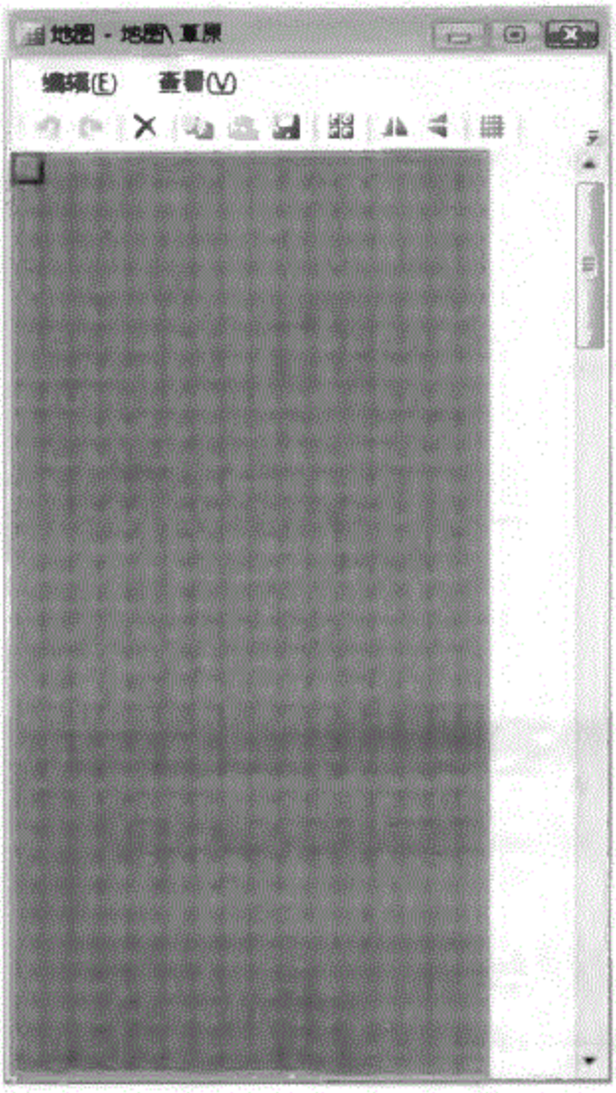


图 4-76

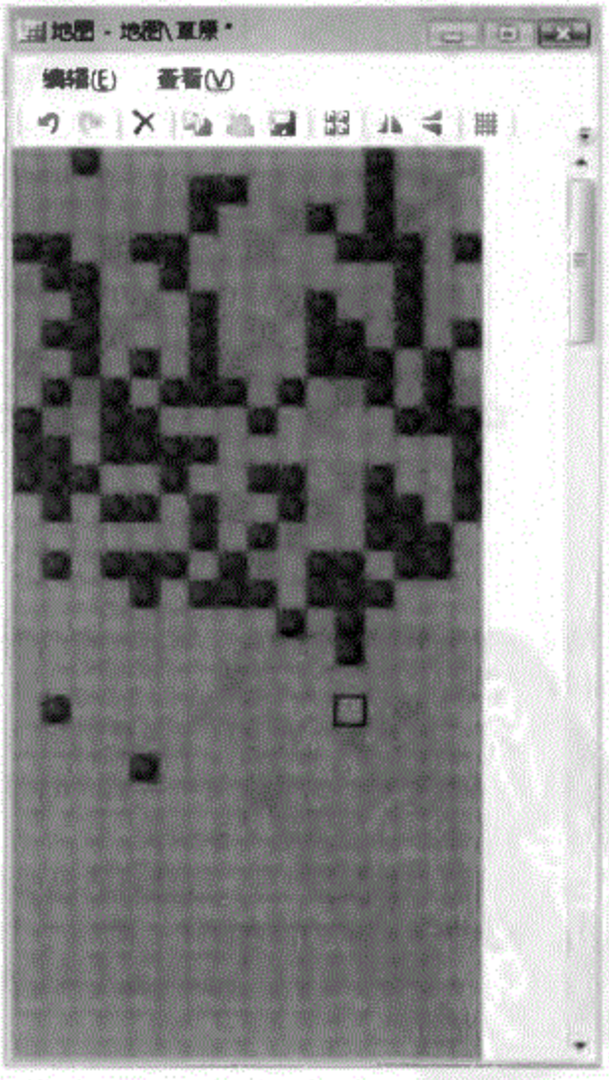


图 4-77

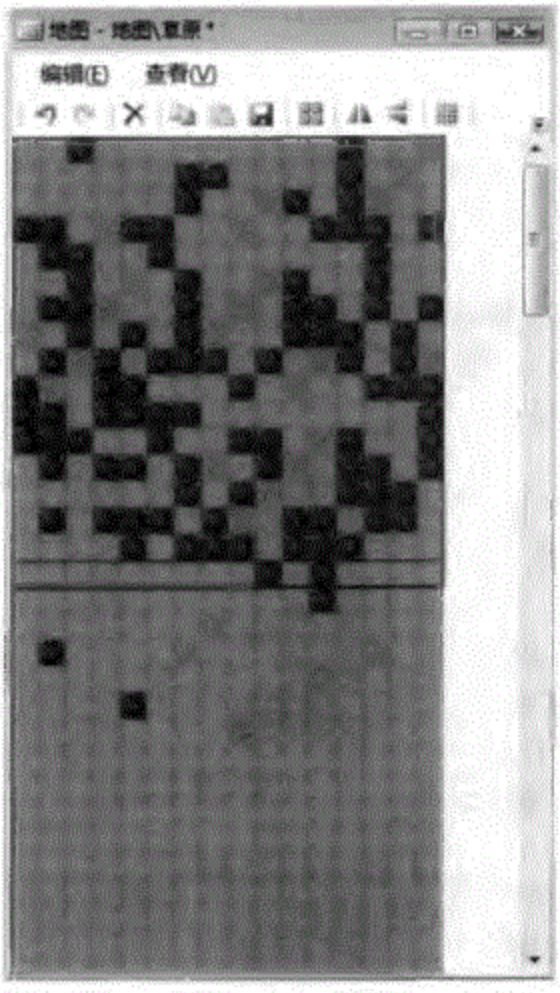


图 4-78

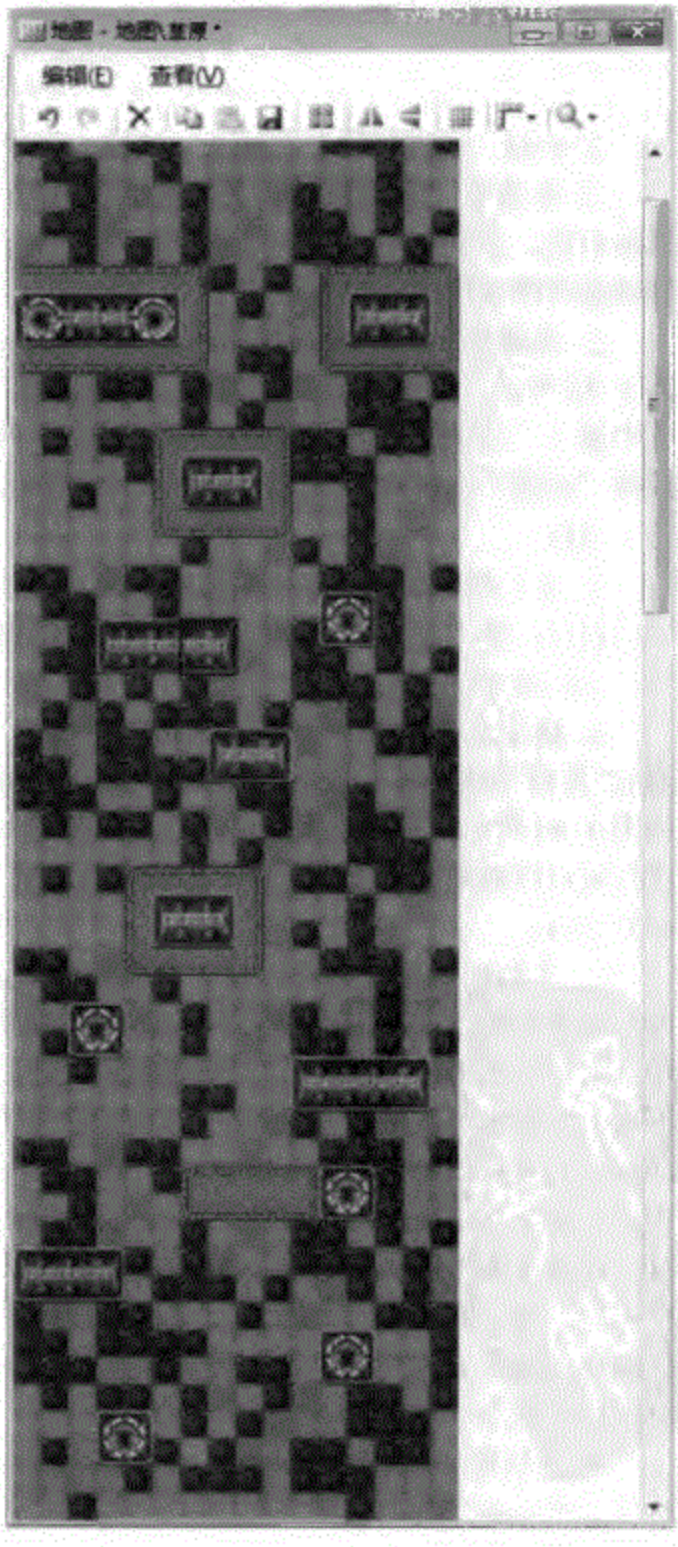


图 4-79

4.7 游戏物件的定义

① 游戏物件是整个游戏的核心，所有在游戏中移动和表现的物体和特效都必须使用游戏物件去定义。在本游戏中，需要定义“主角”、“敌人 1”、“敌人 2”、“玩家子弹”、“敌人子弹 1”、“敌人子弹 2”、“爆炸”这 7 个游戏物件。

② 创建“游戏物件”文件夹，如图 4-80 所示。

图 4-80

③ 在选中“游戏物件”文件夹的情况下，单击鼠标右键，在弹出的菜单中单击“游戏物件”命令，创建新的游戏物件，如图 4-81 所示。

④ 系统弹出“新建游戏物件定义”对话框，如图 4-82 所示。在这个对话框中的“物件名称”文本框中输入“主角”，在“使用动画”下拉列表框中，选择“动画”文件夹中的主角动画，完成后单击“确定”按钮。

⑤ 在工程视图中，即可检查已经建立的“主角”游戏物件。单击“主角”游戏物件，打开游戏物件窗口，如图 4-83 所示。

⑥ 确定主角的碰撞范围。单击“使用动画”右侧的“查看”按钮，将主角游戏物件相关联的动画打开，如图 4-84 所示。使用放大工具，将主角的图片放大，这样就可以清晰地看到图片在坐标轴中的位置，如图 4-85 所示。

图 4-81

⑦ 主角的碰撞范围可以通过主角的图案在动画窗口中坐标轴所占的方格的大小决定。在这个坐标轴中，它的正负值的方向和平时所用的数学坐标轴略有不同。数学坐标轴中向左是负的，向右是正的，这和动画中的坐标轴是一样的；但是数学坐标轴中向上是正的，向下是负的，这个和动画中的坐标轴却正好相反。在第 3 章中已经知道每个方格的长宽是以 7 个单位进行计算，因此这个图案中，可以计算出“主角”的碰撞范围的最左值为 -14，最上值为 -12，宽度为 28，高度为 26，如图 4-86 所示。

⑧ 将玩家的 HP 设定为 80，攻击值设定为 3，并且在这里选中“使用血槽”复选框，在其下方的下拉列表框中选择已经完成的玩家血槽，如图 4-87 所示。


物件名称
主角
使用动画
动画\主角
碰撞范围
左
-8
上
-8
宽度
16
高度
16
一般属性
HP
100
攻击值
16
确定
取消

图 4-82


物件名称
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-8
宽度
16
高度
16
一般属性
HP
100
使用血槽
使用道具栏
攻击值
16
脚本属性
脚本类型
初始化脚本
测试
模板

图 4-83

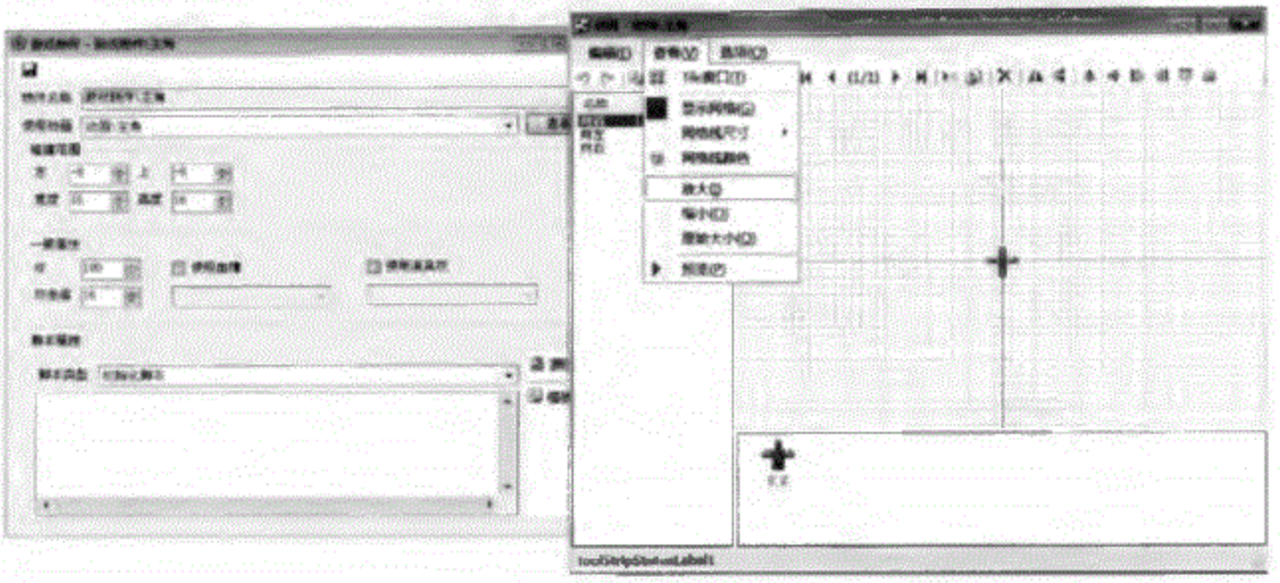


图 4-84

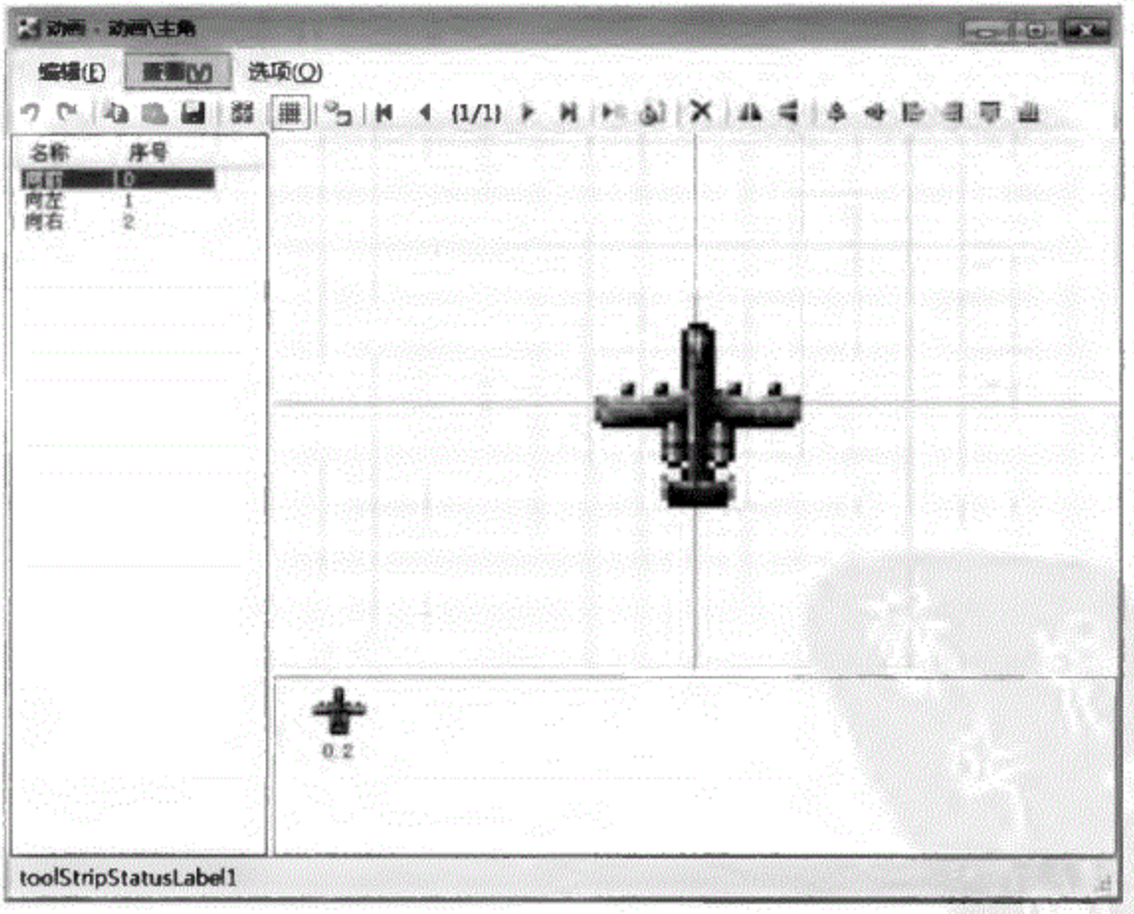

动画 - 动画\主角
编辑(E)
查看(V)
选项(O)
(1/1)
名称
序号
向下 0
向左 1
向右 2
0,2
toolStripStatusLabel1

图 4-85

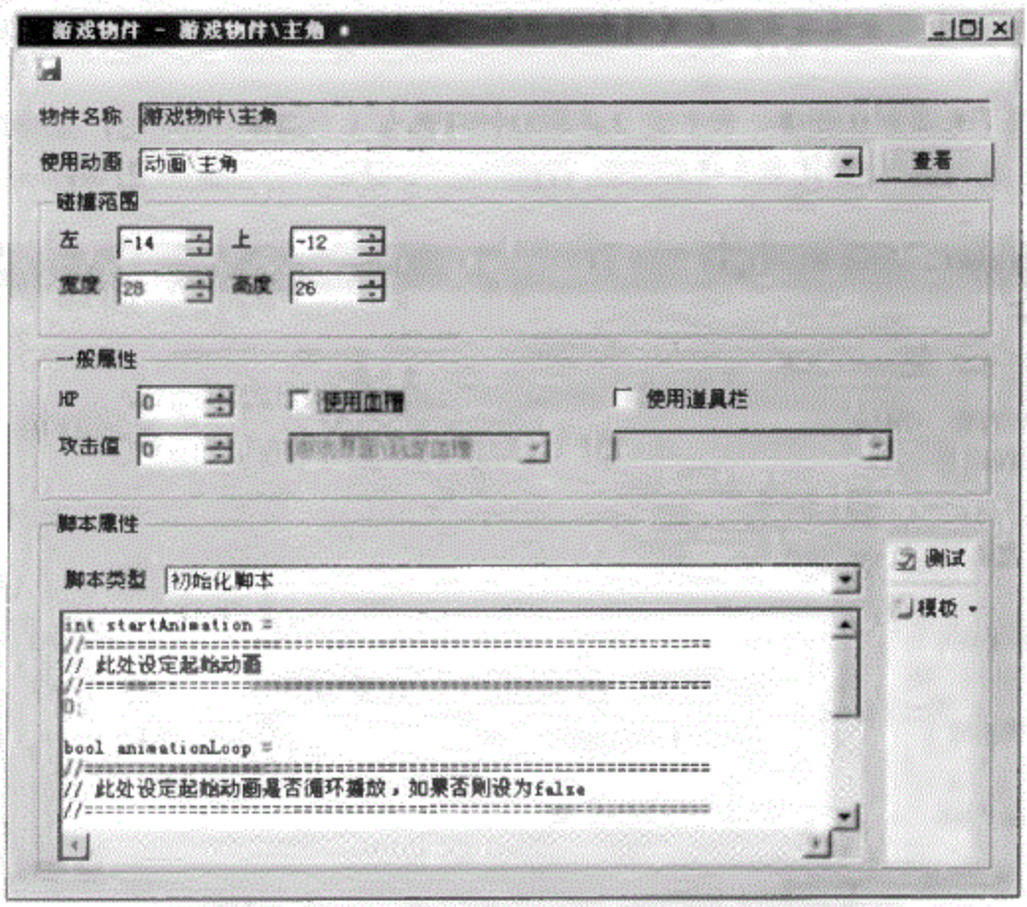

游戏物件 - 游戏物件\主角 *
物件名称 游戏物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -14
上 -12
宽度 28
高度 26
一般属性
HP 0
使用血槽
使用道具栏
攻击值 0
脚本属性
脚本类型 初始化脚本
测试
模板
int startAnimation =
//==========
// 此处设定起始动画
//==========
0;
bool animationLoop =
//==========
// 此处设定起始动画是否循环播放，如果否则设为false
//==========

图 4-86

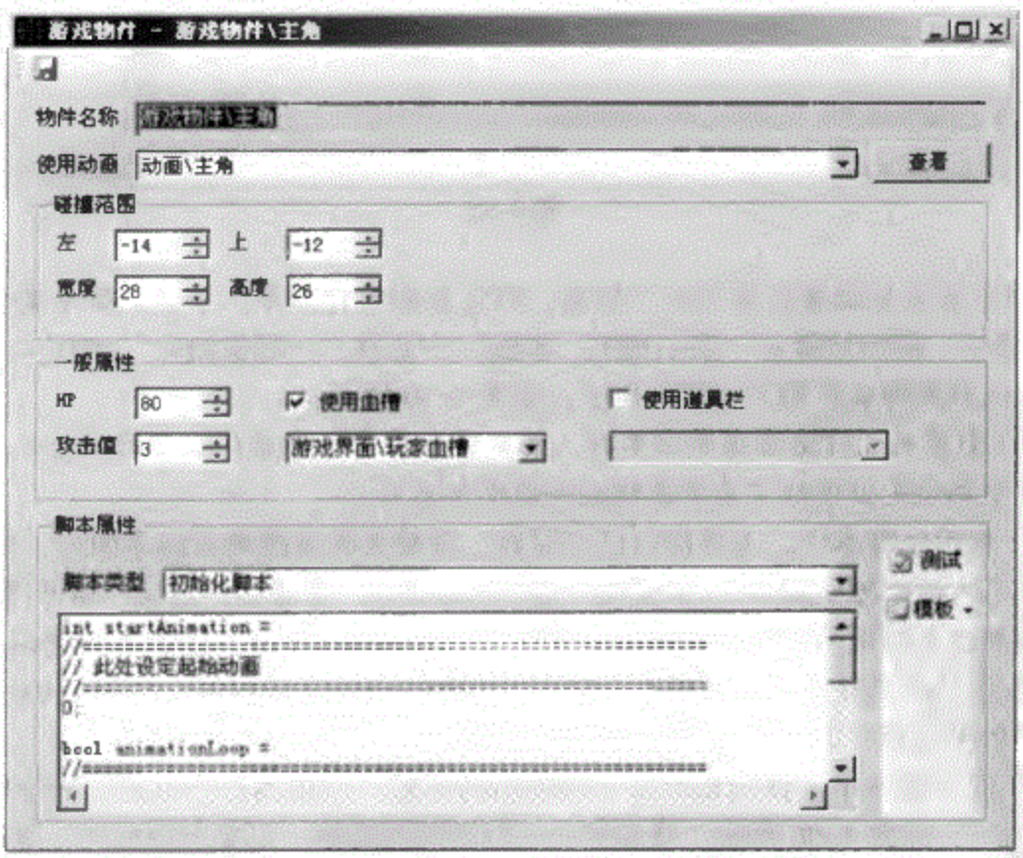

游戏物件 - 游戏物件\主角
物件名称 游戏物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -14
上 -12
宽度 28
高度 26
一般属性
HP 60
使用血槽
使用道具栏
攻击值 3
游戏界面\玩家血槽
脚本属性
脚本类型 初始化脚本
测试
模板
int startAnimation =
//==========
// 此处设定起始动画
//==========
0;
bool animationLoop =
//==========

图 4-87

提示：这个攻击值不是主角发射出的子弹的威力，而是指它本身接触到敌人时的攻击值。

⑨ 接下来将制作脚本。每个定义的游戏物件都有 4 个脚本：初始化、碰撞测试、运行和被伤害。它们分别代表了开始状态、碰撞判定、本身属性和伤害判定，如图 4-88 所示。

图 4-88

⑩ 接下来分别设置主角的 4 个脚本。首先是初始化脚本，单击“脚本类型”下拉列表框，选中“初始化脚本”，在右侧的“模版”中选择“一般初始化”，如图 4-89 所示，选择后可以看到脚本类型下的脚本程序，如图 4-90 所示。

提示：引擎的设计者在设计脚本时，已经替制作人员考虑到引擎的便捷性，所以在各个脚本中都预先设计好了若干套模板供制作者选用。

⑪ 在初始化脚本中，无需进行任何设置，直接选择碰撞测试脚本即可，如图 4-91 所示。在进入碰撞测试脚本后，从模板中选择所需要的“普通命中测试”即可，如图 4-92 所示，选择这个模板后就可以看到脚本类型下的命中脚本程序，如图 4-93 所示。

⑫ 然后进入最复杂的运行脚本。在脚本类型中选择运行脚本，如图 4-94 所示，在右侧的模板中选择“纵版射击”中的“主角运动”，如图 4-95 所示。

⑬ 在运行脚本中，首先设定第一项射击的音效。在提示栏下方的引号中输入“音乐 \\fire.wav”，如图 4-96 所示，这表示在运行游戏的时候，只要主角射击，系统都会调用这个音效。

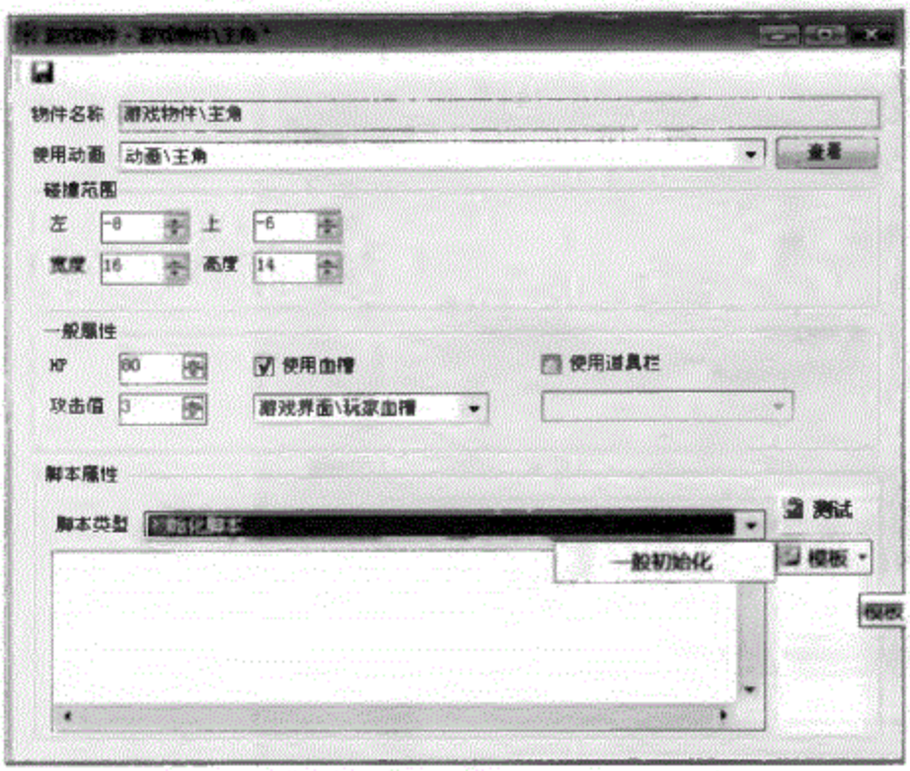

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
初始化脚本
测试
模板
一般初始化

图 4-89

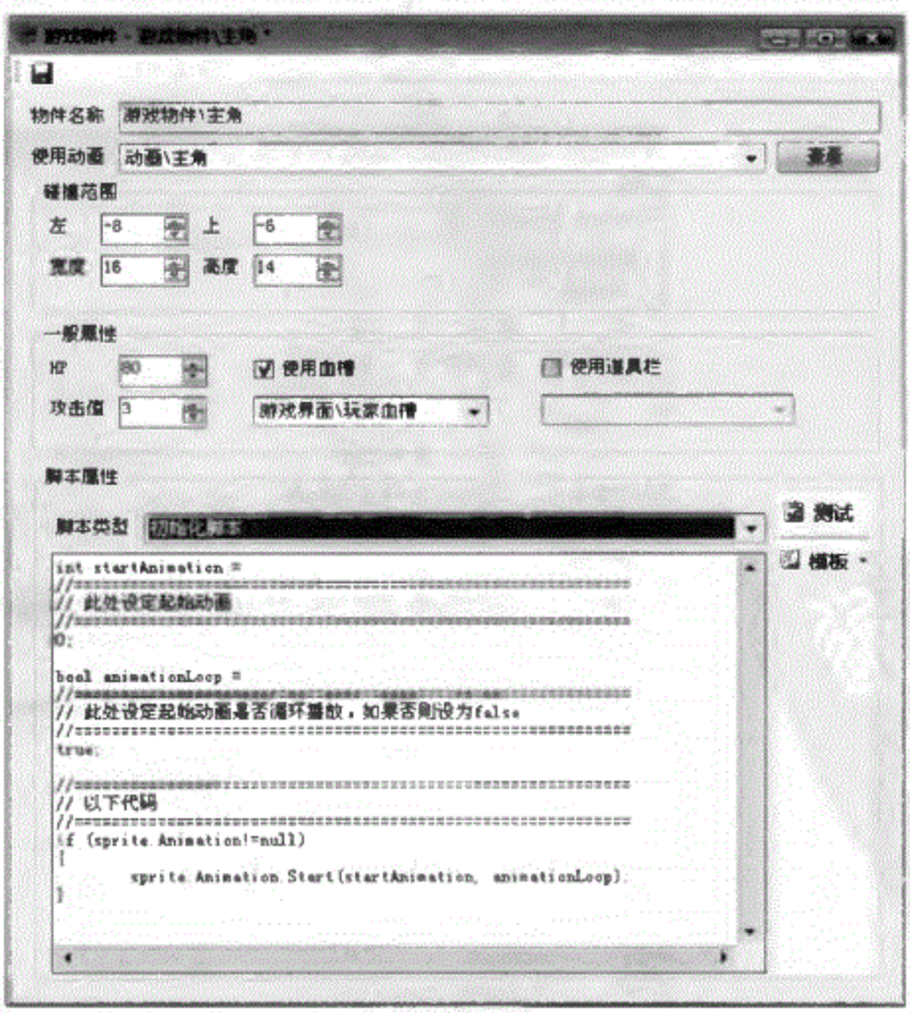

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
初始化脚本
测试
模板
int startAnimation =
//==
// 此处设定起始动画
//==
0;
bool animationLoop =
//==
// 此处设定起始动画是否循环播放，如果否则设为false
//==
true;
//==
// 以下代码
//==
if (sprite.Animation!=null)
{
sprite.Animation.Start(startAnimation, animationLoop);
}

图 4-90

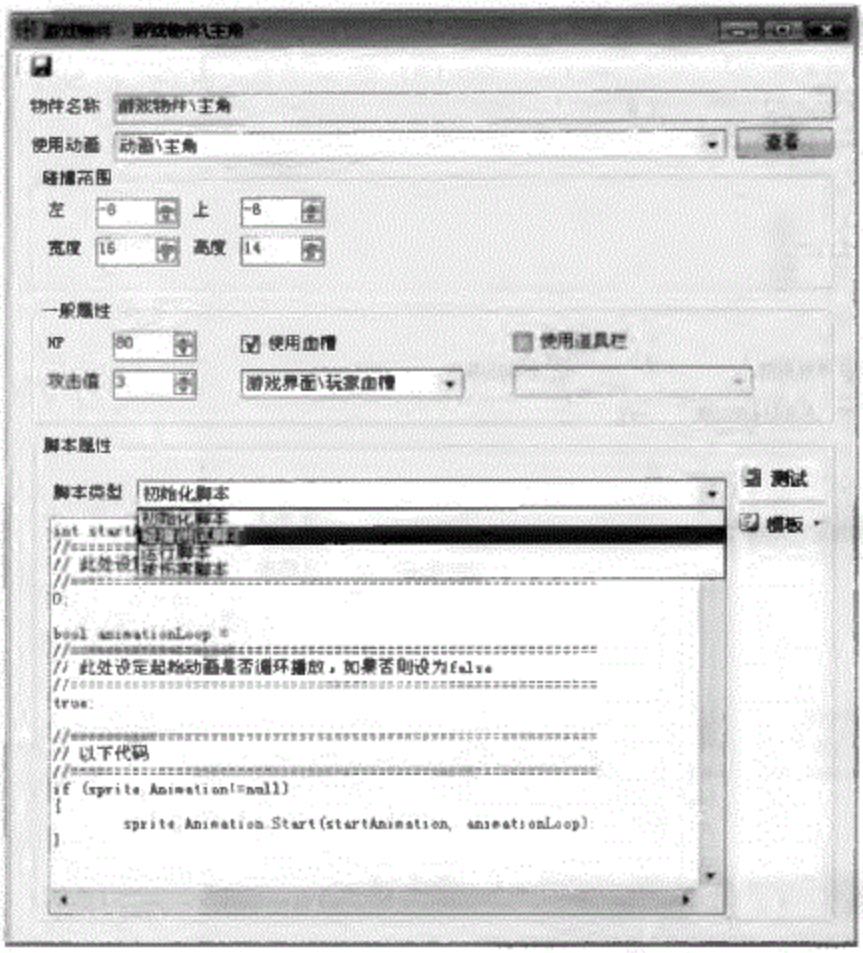

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-8
宽度
16
高度
14
一般属性
XP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
初始化脚本
测试
模板
运行脚本
// 此处设定起始动画是否循环播放，如果否则设为false
true;
// 以下代码
if (sprite.Animation!=null)
{
sprite.Animation.Start(startAnimation, animationLoop);
}

图 4-91

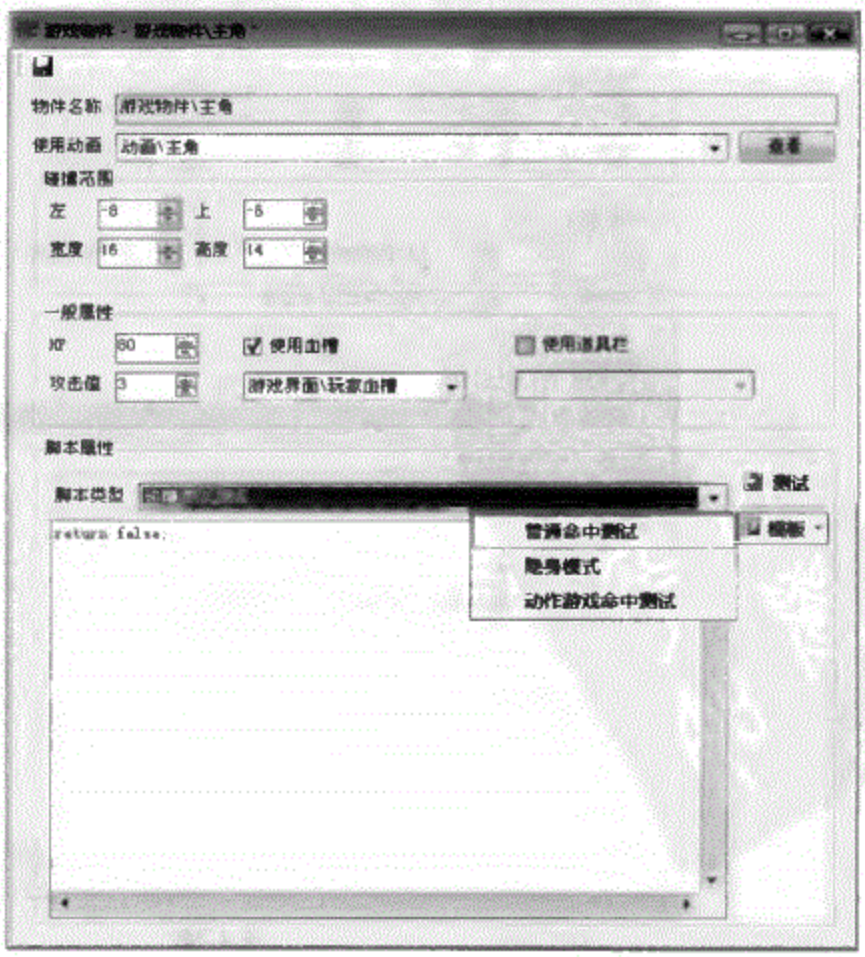

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-8
宽度
16
高度
14
一般属性
XP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
测试
模板
return false;
普通命中测试
隐身模式
动作游戏命中测试

图 4-92

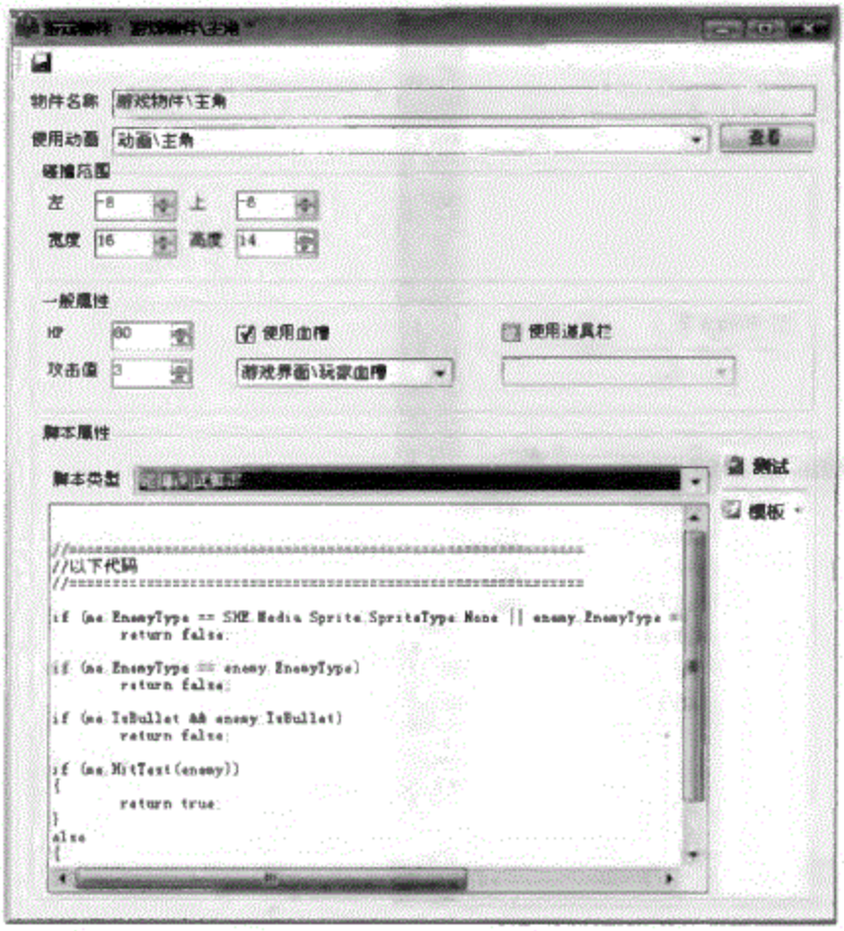

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
60
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
测试
模板
//以下代码
if (ne.EnemyType == SME.Media.Sprite.SpriteType.None || enemy.EnemyType ==
return false;
if (ne.EnemyType == enemy.EnemyType)
return false;
if (ne.IsBullet && enemy.IsBullet)
return false;
if (ne.HitTest(enemy))
{
return true;
}
else
{

图 4-93

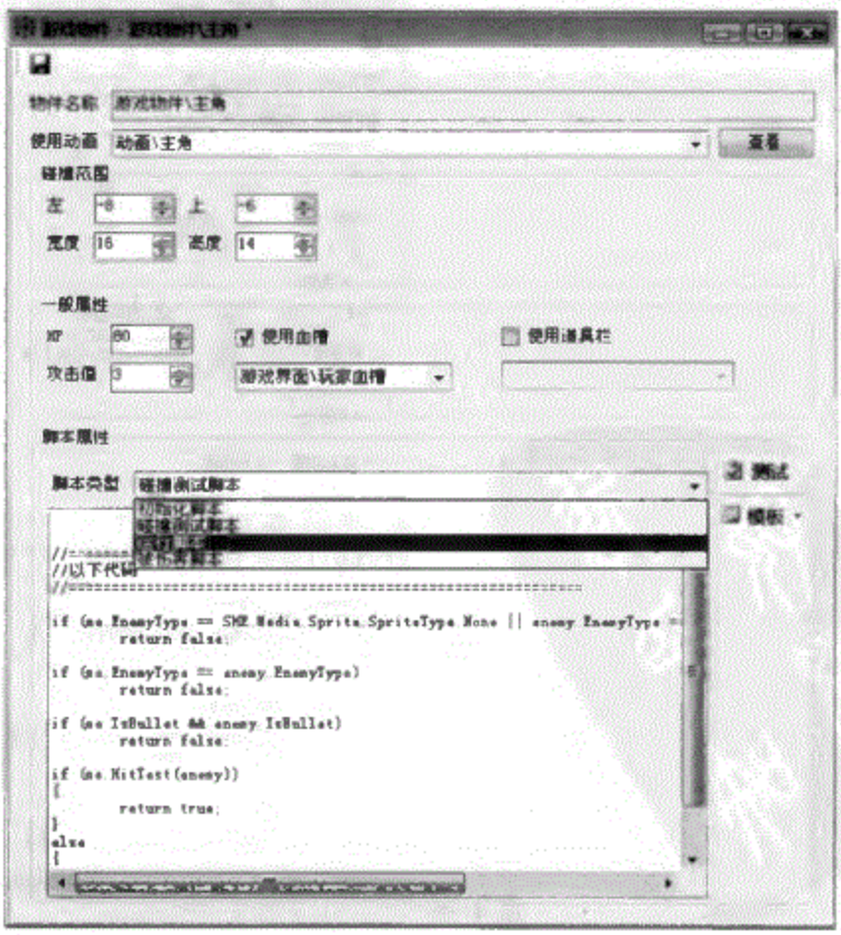

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
60
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
碰撞测试脚本
测试
模板
//以下代码
if (ne.EnemyType == SME.Media.Sprite.SpriteType.None || enemy.EnemyType ==
return false;
if (ne.EnemyType == enemy.EnemyType)
return false;
if (ne.IsBullet && enemy.IsBullet)
return false;
if (ne.HitTest(enemy))
{
return true;
}
else
{

图 4-94

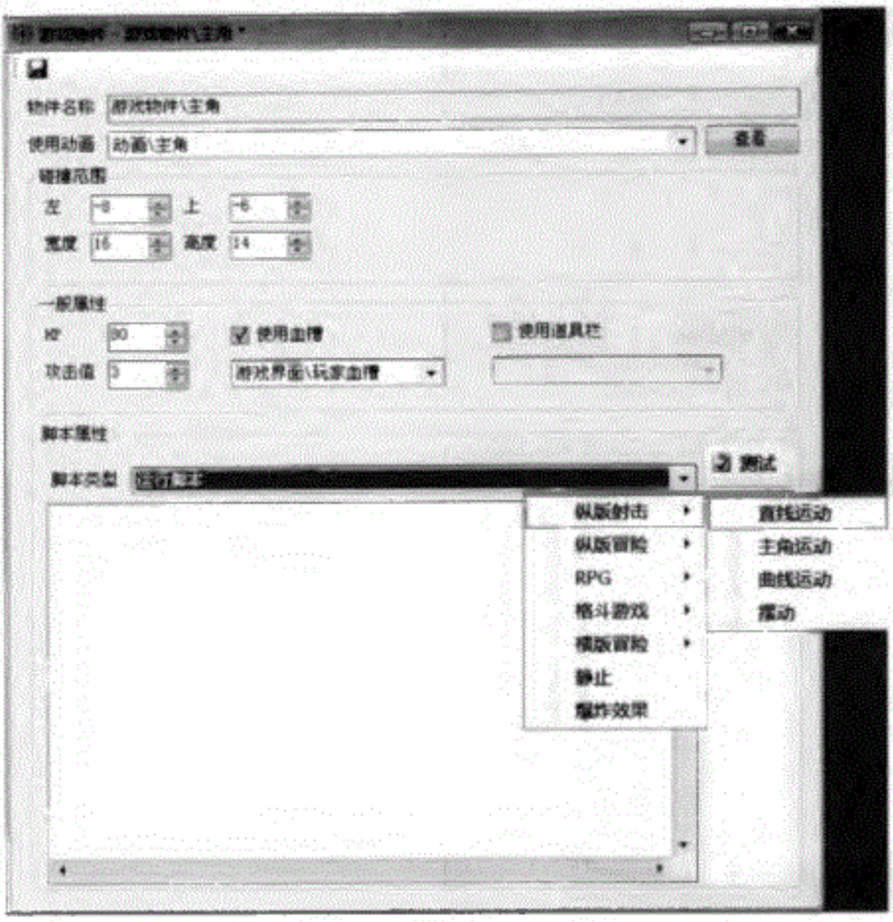

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
运行脚本
测试
纵版射击
纵版冒险
RPG
格斗游戏
横版冒险
静止
爆炸效果
直线运动
主角运动
曲线运动
摆动

图 4-95

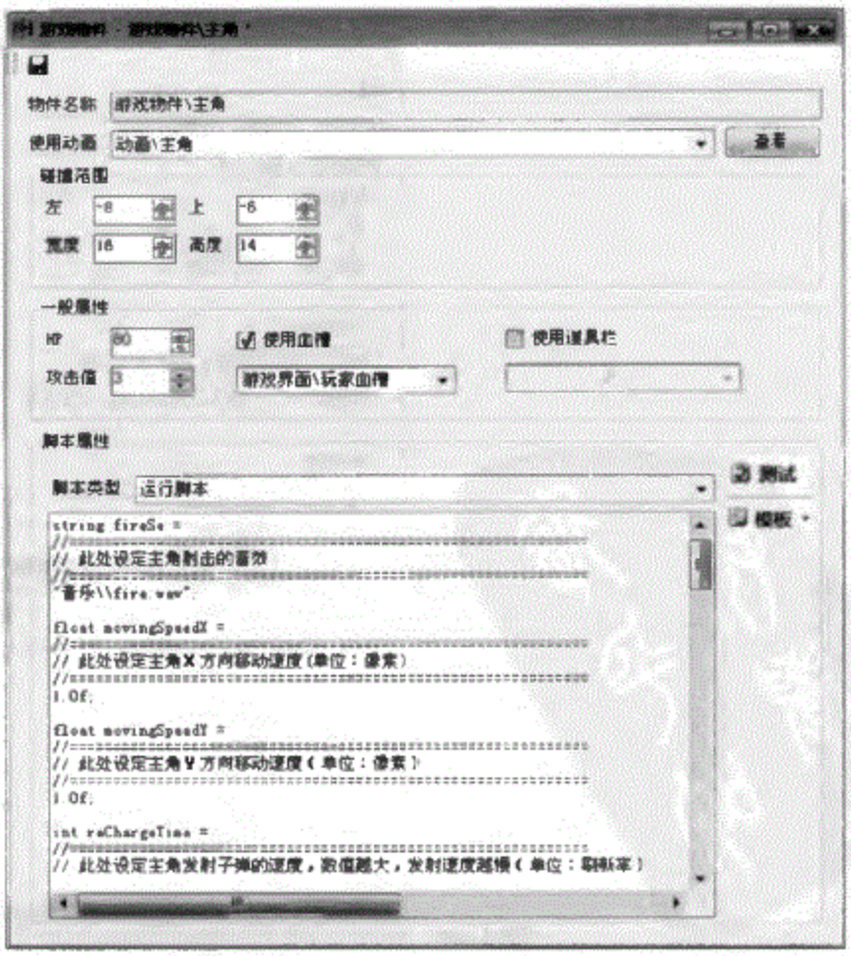

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
HP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
运行脚本
测试
模板
string fireSe =
//===
// 此处设定主角射击的音效
//===
"音乐\\fire.wav";
float movingSpeedX =
//===
// 此处设定主角X方向移动速度(单位：像素)
//===
1.0f;
float movingSpeedY =
//===
// 此处设定主角Y方向移动速度(单位：像素)
//===
1.0f;
int reChargeTime =
//===
// 此处设定主角发射子弹的速度，数值越大，发射速度越慢(单位：秒/发)

图 4-96

提示："文件夹名 \\ 文件名"是在游戏物件中调用其他文件的标准格式。在游戏物件中，只能调用音乐文件和游戏物件中的文件，其他的文件格式不在调用范围之内。

⑭ 调用子弹类型。在子弹类型的提示句下的引号中输入"游戏物件 \\ 玩家子弹"确定玩家所使用的子弹类型，如图 4-97 所示。

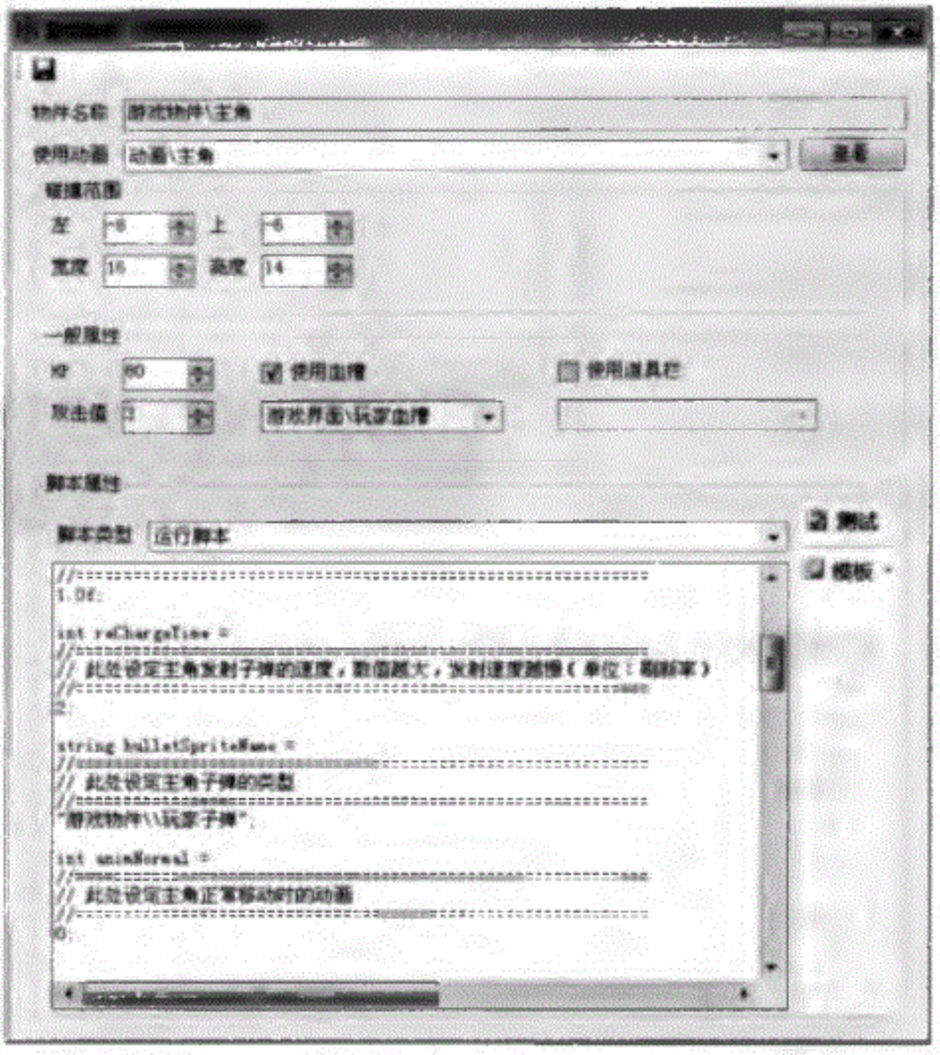


图 4-97

提示：虽然在游戏物件中还没有制作玩家的子弹物件，但这并不妨碍我们先将子弹给确定下来，以后只要用这个命名去指定子弹即可。键盘中有两种斜杠，千万不要用错，否则在游戏中将识别不出来。

⑮ 在子弹类型的下方，分别是左移动画、右移动画和被伤害动画，这个时候则要打开动画窗口进行观察。在观察中可以注意到，在之前制作"主角"动画的时候，分别做了"向前"、"向左"、"向右"和"受伤"4 个动作，在这 4 个动作右侧各有一个数字所代表的序号，游戏物件中的动画就是靠着这个序号和动画窗口中的动作相关联，如图 4-98 所示。

⑯ 完成之后，进入被伤害脚本的编辑，如图 4-99 所示。在被伤害脚本的模板中，选择"普通伤害"，如图 4-100 所示。在这个脚本模板中，只需要设定爆炸效果就可以了。在爆炸效果提示下的引号中输入"游戏物件 \\ 爆炸"，如图 4-101 所示。

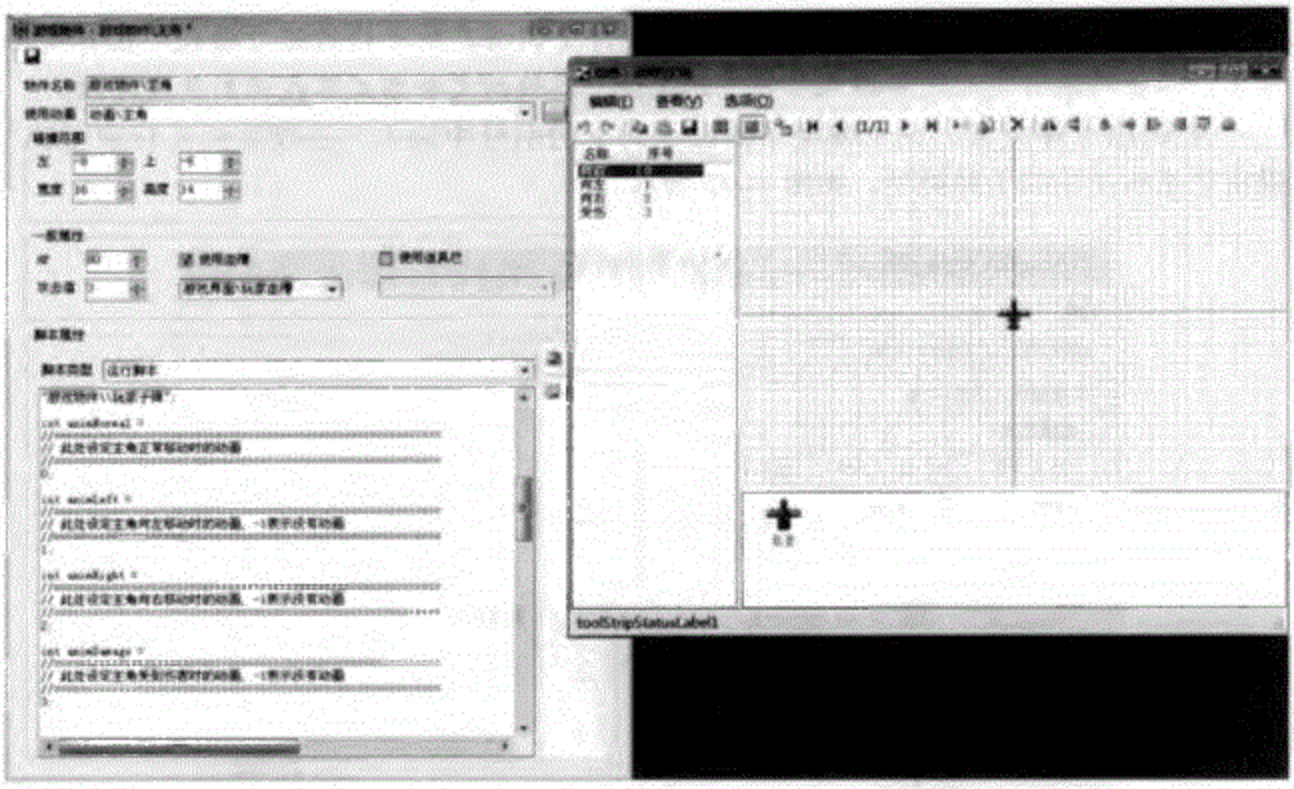


图 4-98

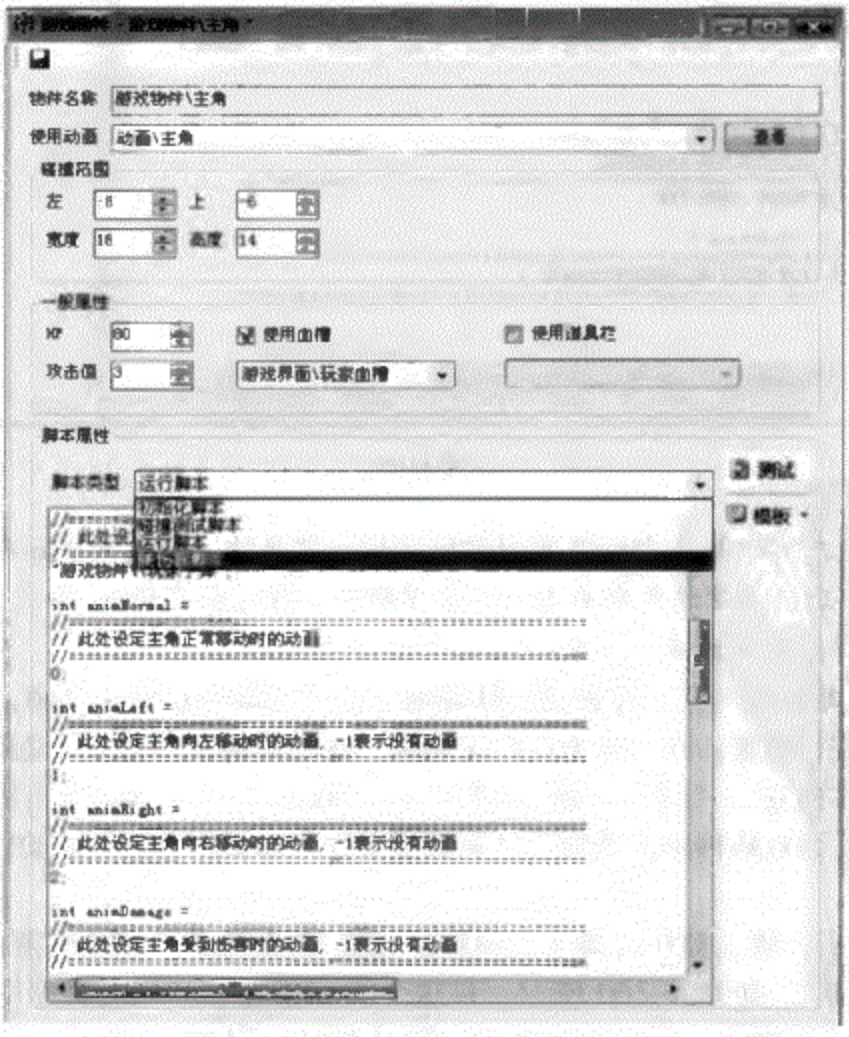

物件名称
游戏物件\主角
使用动画
动画\主角
查看
碰撞范围
左
-8
上
-6
宽度
16
高度
14
一般属性
XP
80
使用血槽
使用道具栏
攻击值
3
游戏界面\玩家血槽
脚本属性
脚本类型
运行脚本
测试
模板
初始化脚本
碰撞测试脚本
运行脚本
int aniNormal =
// 此处设定主角正常移动时的动画
0;
int aniLeft =
// 此处设定主角向左移动时的动画，-1表示没有动画
1;
int aniRight =
// 此处设定主角向右移动时的动画，-1表示没有动画
2;
int aniDamage =
// 此处设定主角受到伤害时的动画，-1表示没有动画

图 4-99

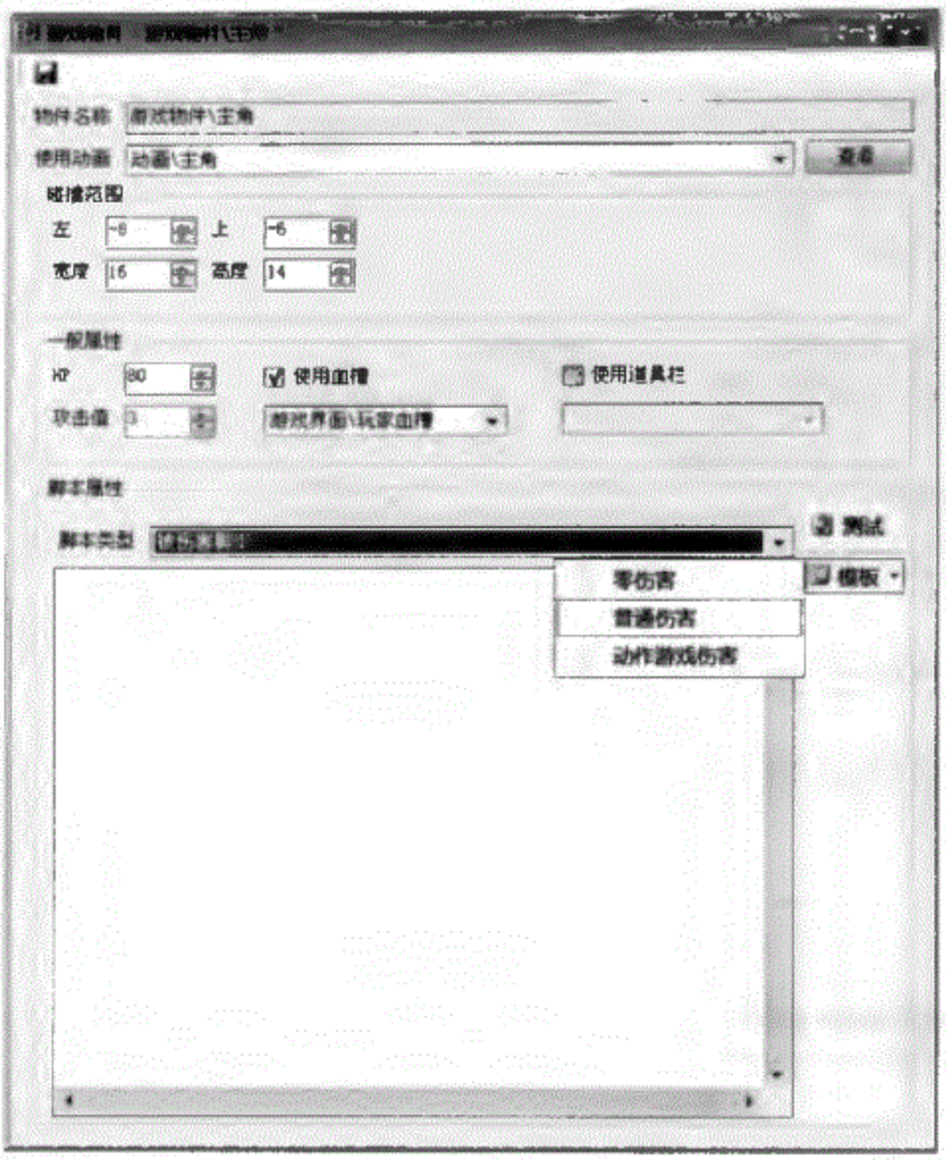


图 4-100

⑰ 这样，一个完整的游戏物件就制作完成了。在制作完成后，通常都需要测试新完成的游戏物件，来检测这个游戏物件是否编译正确。在模板的上方有一个测试按钮，单击这个按钮，如果编译正确，系统将弹出对话框告知测试正确；如果错误，则会弹出乱码，那么就必须回到 4 个脚本中仔细检查，如图 4-102 所示。

⑱ 在上述主角的游戏物件的制作过程中，已经涉及了另外两个游戏物件："爆炸"和"玩家子弹"。接下来就介绍"爆炸"和"玩家子弹"游戏物件的制作。

⑲ 先来制作"爆炸"的游戏物件。同样，在工程视图中创建游戏物件。在"新建游戏物件定义"对话框中的"物件名称"文本框中输入"爆炸"，在"使用动画"下拉列表框中选择"动画\爆炸"，单击"确定"按钮，如图 4-103 所示。

⑳ 由于"爆炸"不需要参与碰撞或者其他的操作，所以在这里，不需要设置碰撞范围、HP 和攻击值。在设置脚本类型中，爆炸因为不需要碰撞，也不需要进行其他的操作，所以在 4 种脚本类型中，只需要设置初始化脚本就可以了。在初始化脚本中，使用"一般初始化"模板，如图 4-104 所示，并在"脚本类型"文本框中将起始动画设置

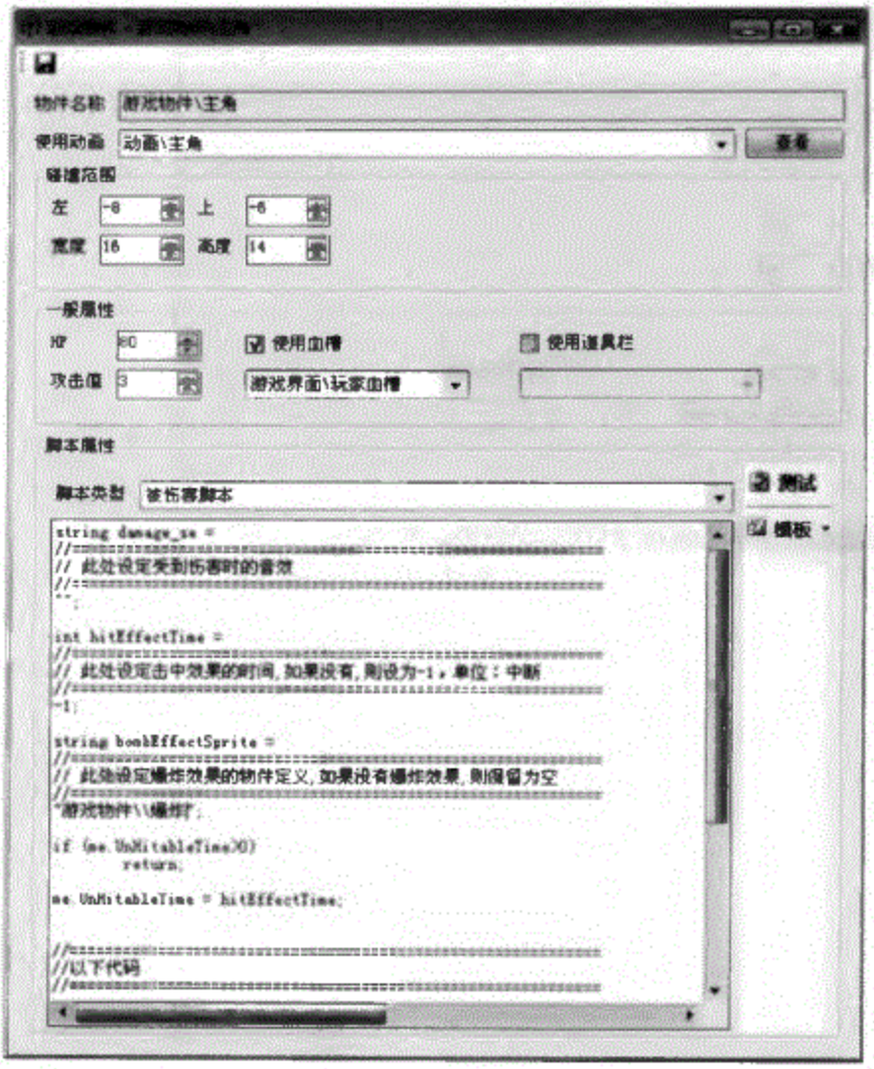

物件名称 游戏物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -8
上 -6
宽度 16
高度 14
一般属性
HP 50
使用血槽
使用道具栏
攻击值 3
游戏界面\玩家血槽
脚本属性
脚本类型 被伤害脚本
测试
模板
string damage_se =
// 此处设定受到伤害时的音效
int hitEffectTime =
// 此处设定击中效果的时间，如果没有，则设为-1，单位：中断
-1;
string bombEffectSprite =
// 此处设定爆炸效果的物件定义，如果没有爆炸效果，则保留为空
"游戏物件\\爆炸";
if (me.UnHitableTime>0)
return;
me.UnHitableTime = hitEffectTime;
//以下代码

图 4-101

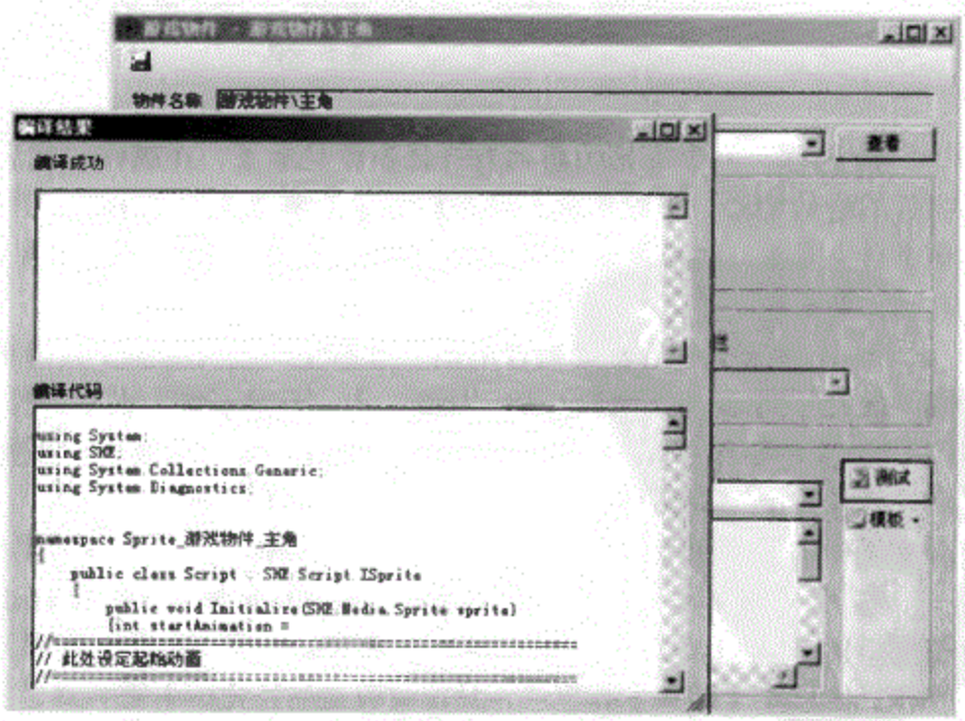

物件名称 游戏物件\主角
编译结果
编译成功
编译代码
查看
测试
模板
using System;
using SNE;
using System.Collections.Generic;
using System.Diagnostics;
namespace Sprite_游戏物件_主角
{
public class Script : SNE.Script.ISprite
{
public void Initialize(SNE.Media.Sprite sprite)
{int startAnimation =
// 此处设定起始动画

图 4-102

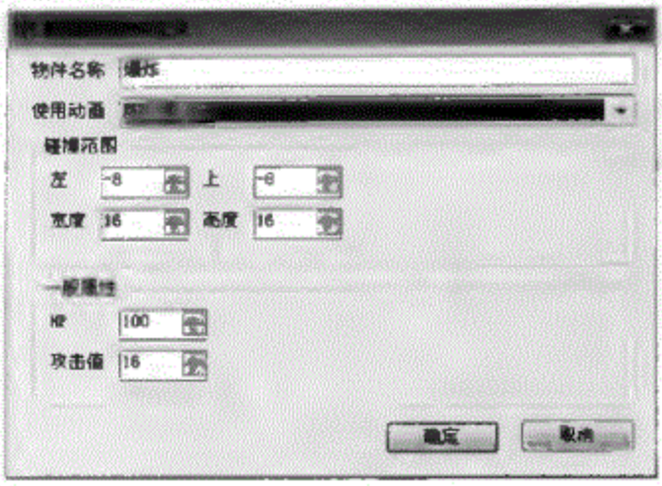


图 4-103

图 4-104

为不循环播放，如图 4-105 所示。

提示：在制作“玩家”游戏物件时，不用调整初始化脚本。在制作“爆炸”游戏物件时，因为“爆炸”的动画只要播放一次即可，不用反复播放。如果使用默认的 true 字符，爆炸的动画会一直在游戏中循环播放。

㉑ 然后制作“玩家子弹”游戏物件。同样在“新建游戏物件定义”对话框的文本框中输入“玩家子弹”，设定使用动画为“动画\玩家子弹”，如图 4-106 所示。

㉒ 接下来计算玩家子弹的碰撞范围。打开与“玩家子弹”游戏物件相关联的动画，如图 4-107 所示。从小方格的尺寸计算出“玩家子弹”的碰撞范围，左：-2，上：-4，

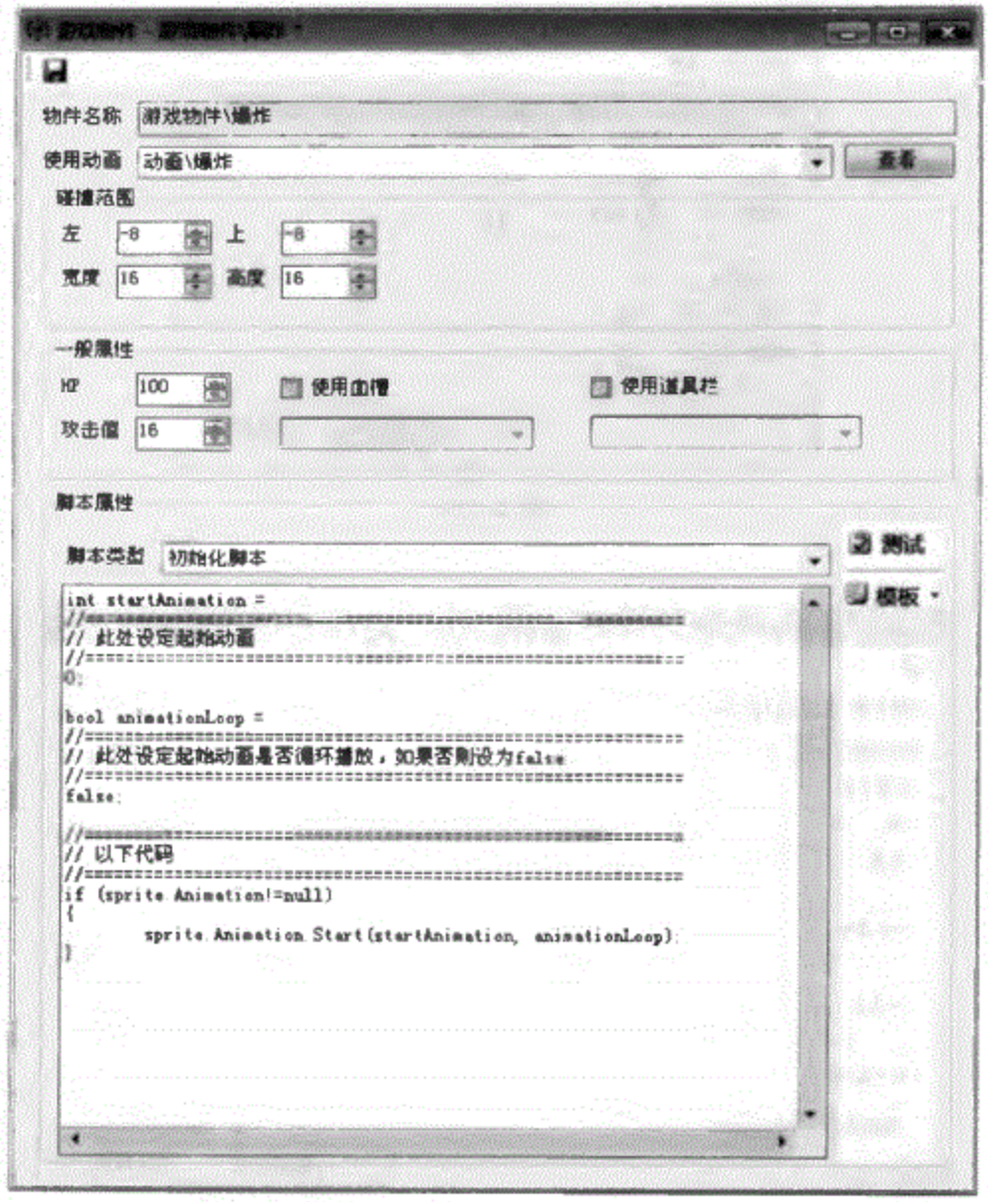

物件名称
游戏物件\爆炸
使用动画
动画\爆炸
查看
碰撞范围
左
-8
上
-8
宽度
16
高度
16
一般属性
HP
100
使用血槽
使用道具栏
攻击值
16
脚本属性
脚本类型
初始化脚本
测试
模板
int startAnimation =
// 此处设定起始动画
0;
bool animationLoop =
// 此处设定起始动画是否循环播放，如果否则设为false
false;
// 以下代码
if (sprite Animation!=null)
{
sprite Animation Start(startAnimation, animationLoop);
}

图 4-105


物件名称
玩家子弹
使用动画
碰撞范围
左
-8
上
-8
宽度
16
高度
16
一般属性
HP
100
攻击值
16
确定
取消

图 4-106

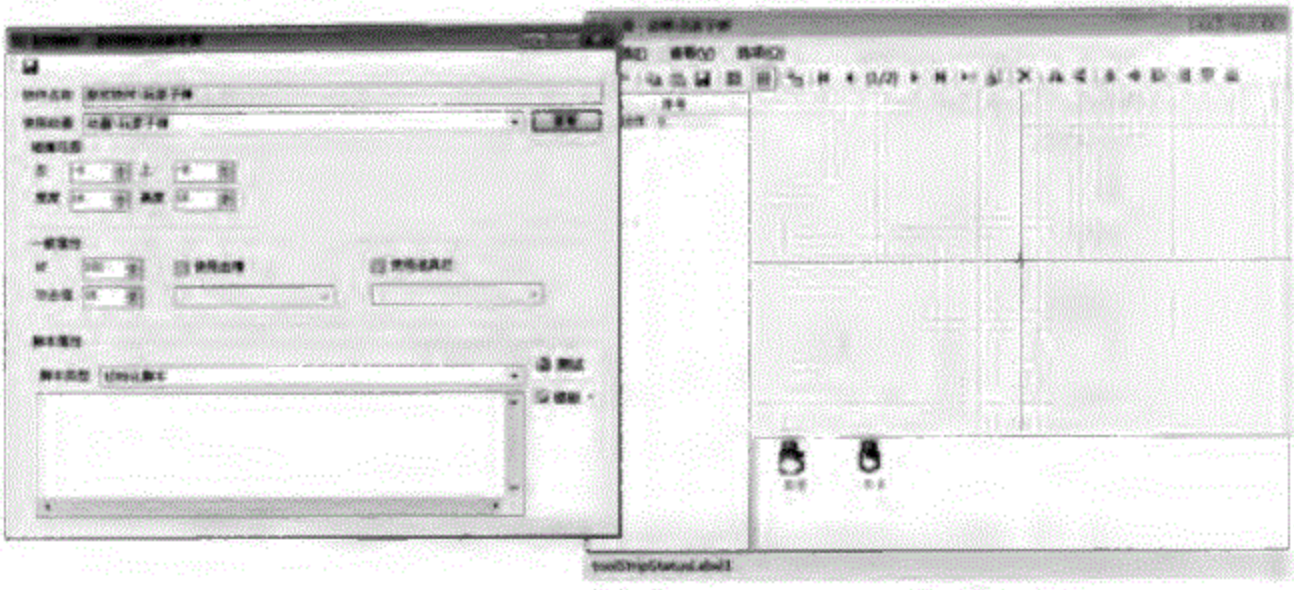


图 4-107

宽：4，高：4，并且将 HP 设为 1，攻击值设为 10，如图 4-108 所示。

㉓ 然后对“玩家子弹”设定 4 种脚本。初始化脚本还是为一般初始化，如图 4-109 所示；碰撞脚本设为普通命中测试，如图 4-110 所示；而运行脚本则是使用纵版射击的直线运动，如图 4-111 所示；最后的被伤害脚本用普通伤害脚本。

㉔ 其他的游戏物件都参照上述 3 个不同的游戏物件设置类型。敌人 1、敌人 2 参照玩家的脚本进行设置；“敌人子弹 1”、“敌人子弹 2”则参照玩家子弹的脚本进行设置。最终将所有的 7 个游戏物件设置完成，如图 4-112 所示。

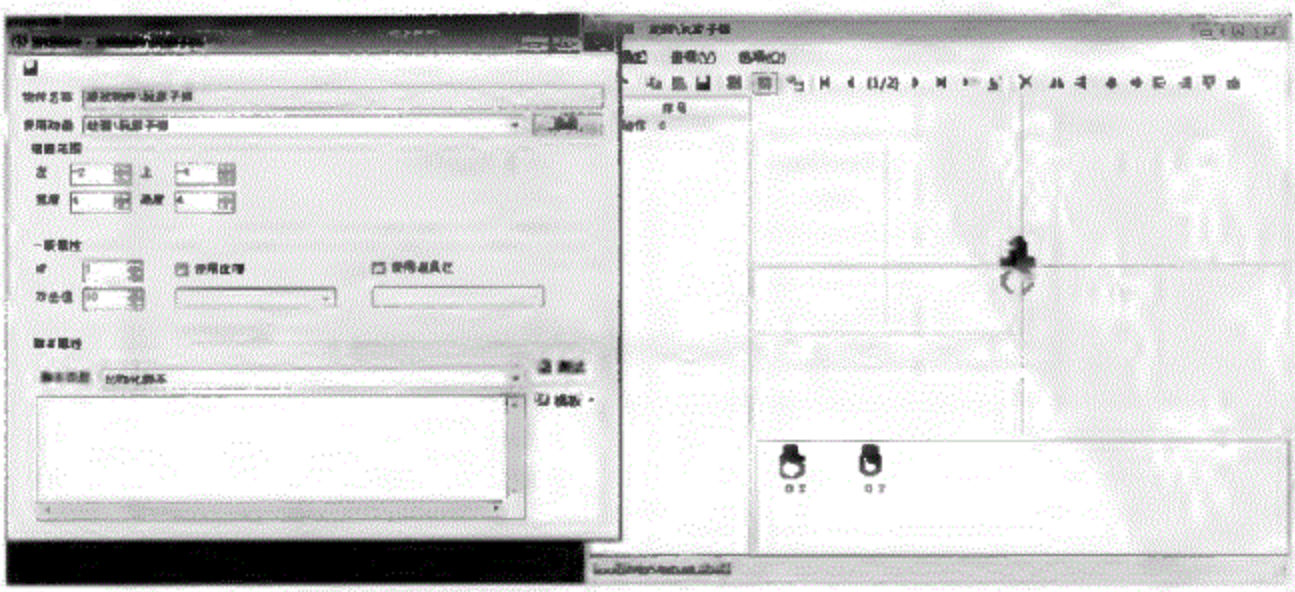


图 4-108

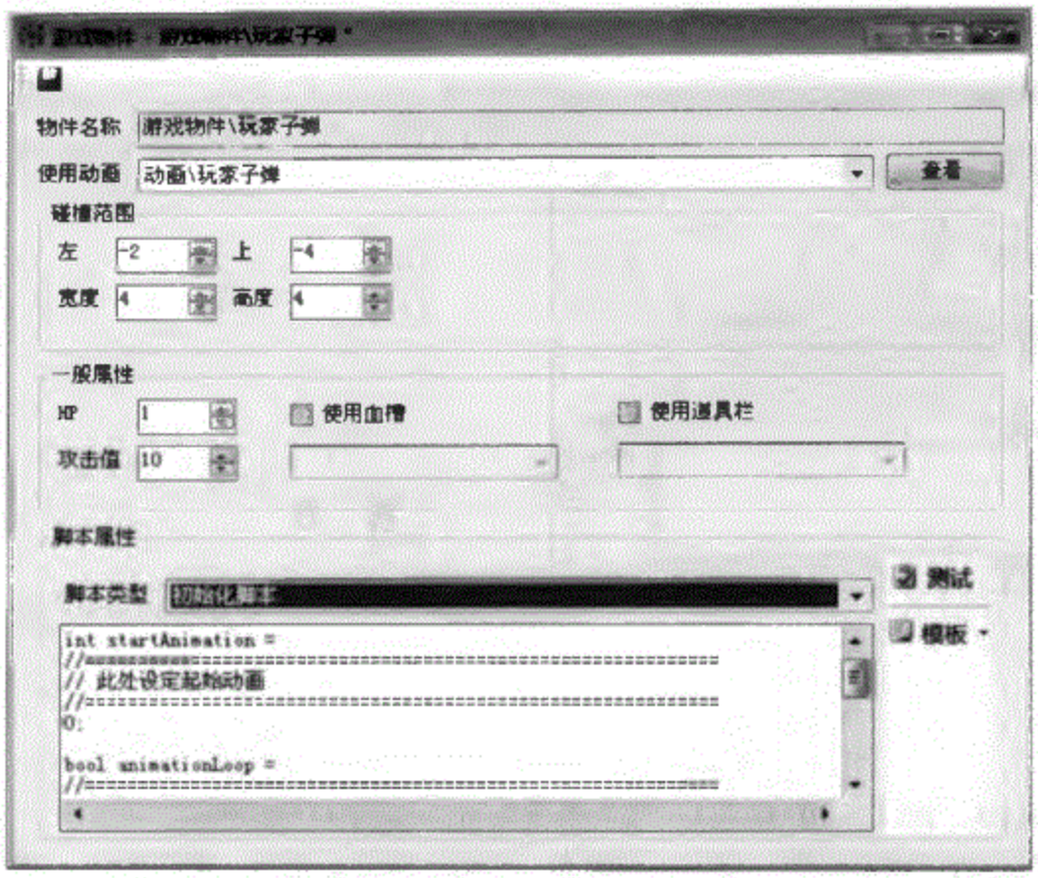

物件名称
游戏物件\玩家子弹
使用动画
动画\玩家子弹
查看
碰撞范围
左
-2
上
-4
宽度
4
高度
4
一般属性
HP
1
使用血槽
使用道具栏
攻击值
10
脚本属性
脚本类型
测试
模板
int startAnimation =
// 此处设定起始动画
0;
bool animationLoop =

图 4-109


物件名称
游戏物件\玩家子弹
使用动画
动画\玩家子弹
查看
碰撞范围
左
-2
上
-4
宽度
4
高度
4
一般属性
HP
1
使用血槽
使用道具栏
攻击值
10
脚本属性
测试
模板
return false;
普通命中测试
隐身模式
动作游戏命中测试

图 4-110

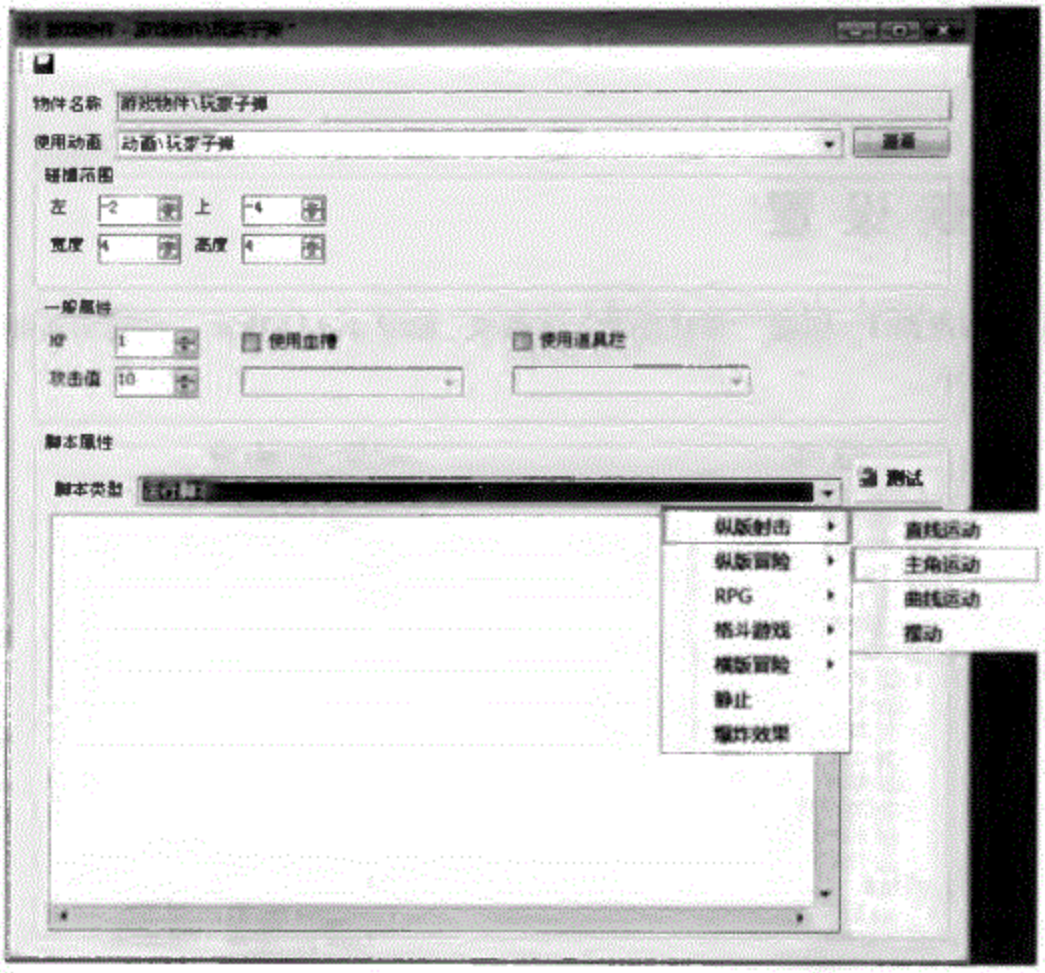

物件名称
游戏物件\玩家子弹
使用动画
动画\玩家子弹
碰撞范围
左
上
宽度
高度
一般属性
HP
使用血槽
使用道具栏
攻击值
脚本属性
脚本类型
测试
纵版射击
纵版冒险
RPG
格斗游戏
横版冒险
静止
爆炸效果
直线运动
主角运动
曲线运动
摆动

图 4-111


图像资源
单位资源
音乐
fire.wav
欧若拉.mp3
动画
敌人1
敌人2
游戏结束
主角
爆炸
游戏开始
玩家子弹
敌人子弹
地图
草原
游戏界面
游戏界面
玩家血槽
敌人血槽
游戏物件
主角
爆炸
玩家子弹
敌人1
敌人2
敌人子弹1
敌人子弹2

图 4-112

4.8 场景设置

① 在工程视图中，创建“游戏场景”文件夹，如图 4-113 所示。创建游戏场景文件，如图 4-114 所示。

图 4-113

图 4-114

② 在系统弹出的“新建游戏场景”对话框中，在“场景名称”文本框中输入“纵向射击”，并且调用场景“地图\草原”，如图 4-115 所示，单击“确定”按钮后就创建完成了一个游戏场景文件。

③ 单击打开“纵向射击”游戏场景文件，弹出游戏场景定义窗口，如图 4-116 所示。在这个游戏场景定义窗口中有地图文件“草原”，现在将“主角”游戏物件和“敌人 1”、

图 4-115

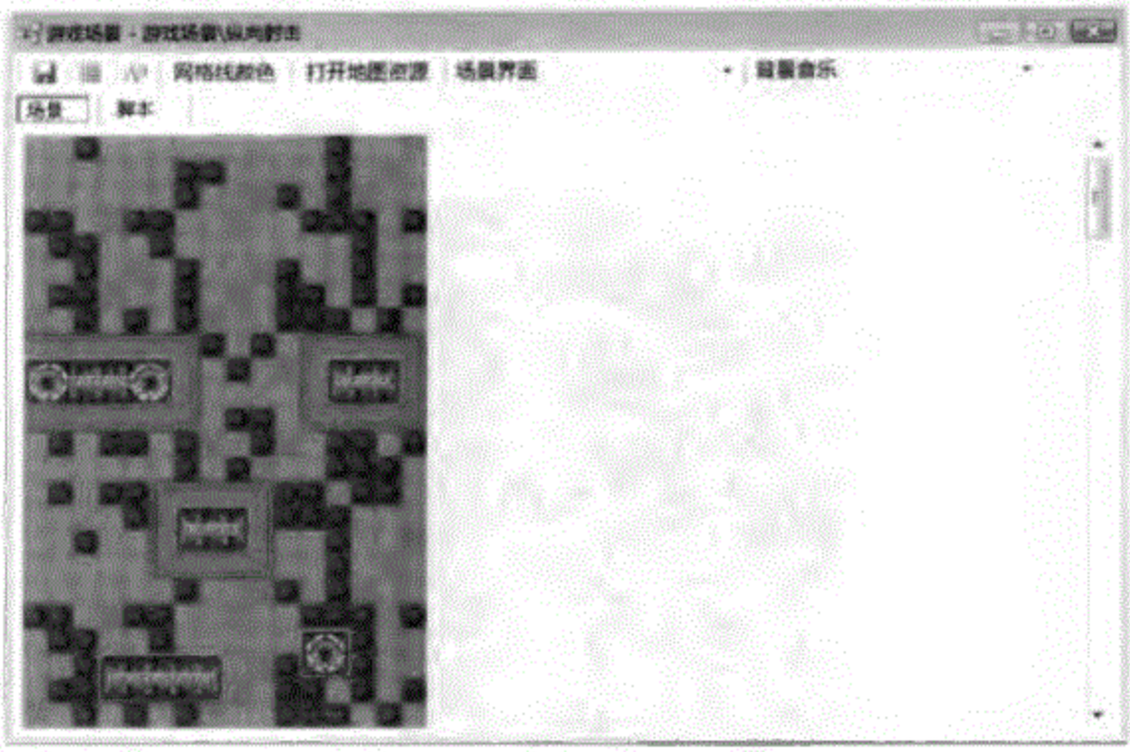


图 4-116

“敌人 2”游戏物件导入进去。

④ 由于地图是纵向的，并且按照一般习惯，玩家都是从下向上的过程。所以在导入游戏物件之初，先将地图拉到最下方，在地图的中间，用鼠标右键单击，在弹出的快捷菜单中单击“游戏物件\主角”，如图 4-117 所示。这个时候就可以在地图中下方看见“主角”游戏物件了，然后用同样方式将“敌人 1”和“敌人 2”游戏物件分别导入。

⑤ 然后使用框选方式选择玩家，右键弹出菜单，选择菜单中的“属性”命令，如图 4-118 所示，调出“物件配置”对话框，在该对话框的最下方的类型中将敌方属性改成友方属性，如图 4-119 所示。

⑥ 在游戏场景定义窗口中，将“场景界面”选择为“游戏界面\游戏界面”，将“背景音乐”选择为已经导入到音乐文件中的“音乐\欧若拉.mp3”，如图 4-120 所示。

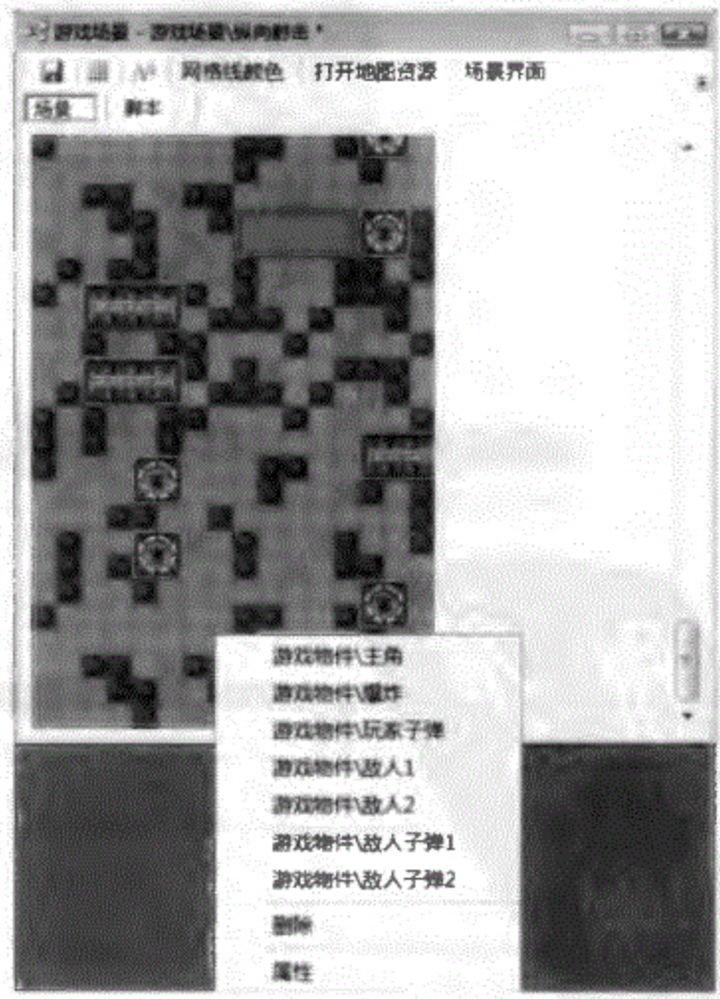


图 4-117

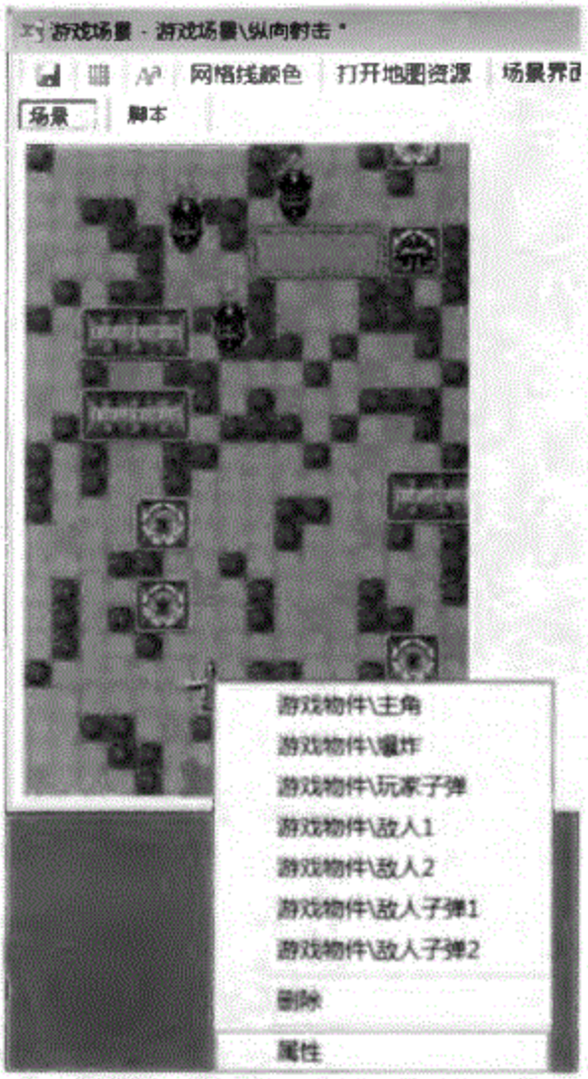

游戏场景 - 游戏场景\纵向射击 *
网格线颜色
打开地图资源
场景界面
场景
脚本
游戏物件\主角
游戏物件\爆炸
游戏物件\玩家子弹
游戏物件\敌人1
游戏物件\敌人2
游戏物件\敌人子弹1
游戏物件\敌人子弹2
删除
属性

图 4-118

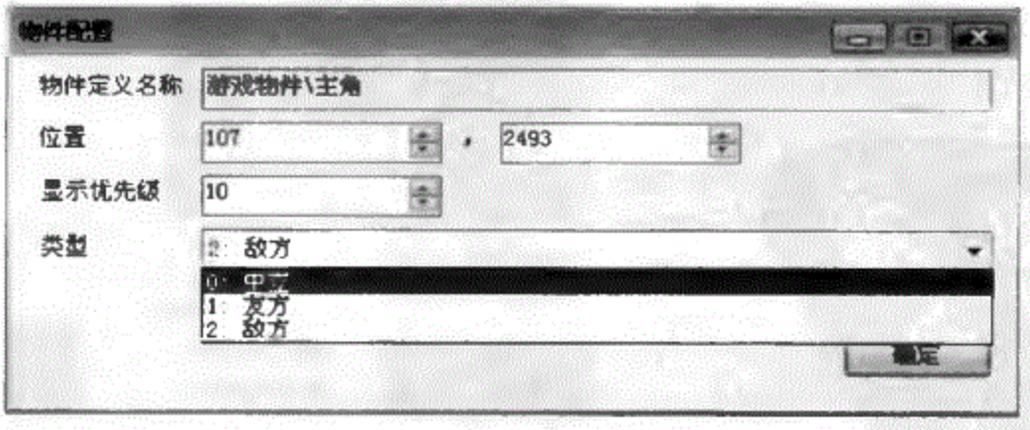

物件配置
物件定义名称
游戏物件\主角
位置
107
2493
显示优先级
10
类型
2: 敌方
0: 中立
1: 友方
2: 敌方
确定

图 4-119

图 4-120

⑦ 单击“脚本”按钮，在脚本下的模板工具中，选择“纵版射击 .script”，如图 4-121 所示。这个脚本的作用是为了确定游戏窗口在整个场景中的运动规律。

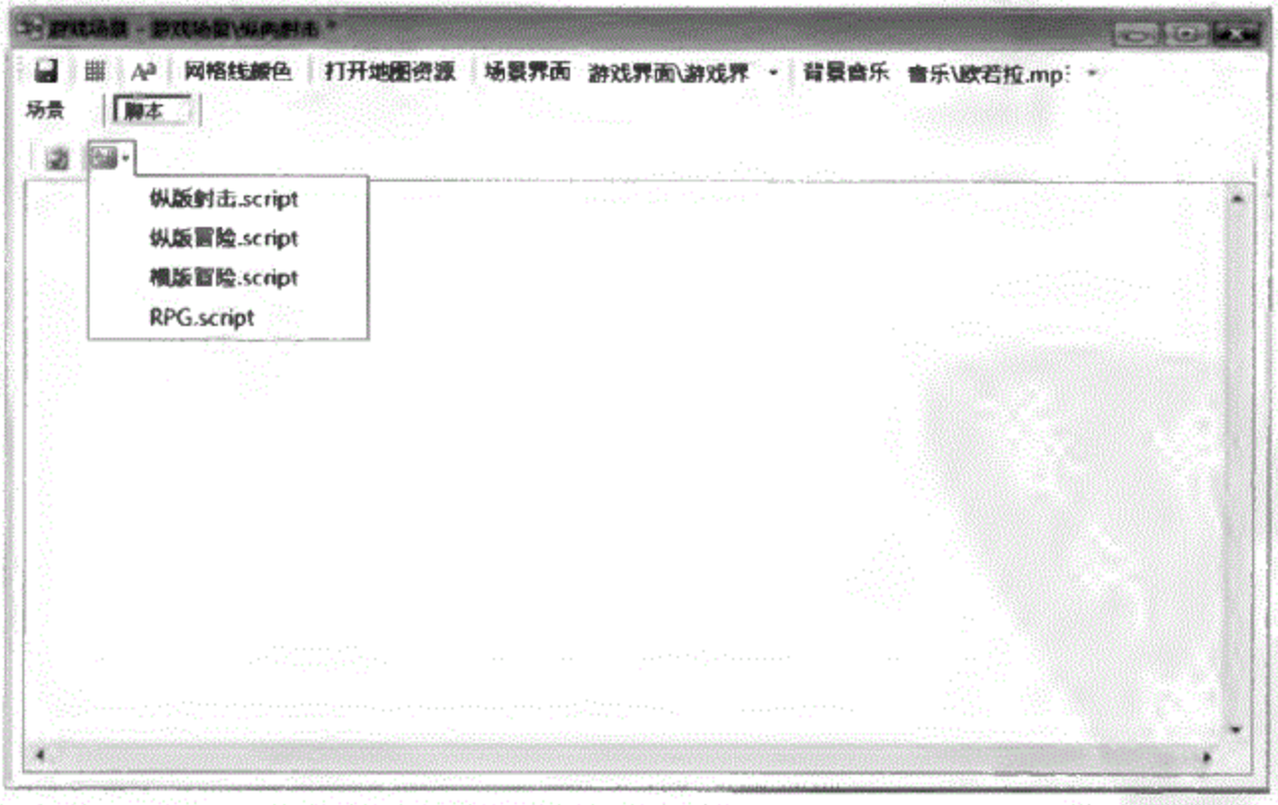


图 4-121

4.9 定义、编译并生成游戏

① 创建“游戏”文件夹，如图4-122所示，然后在工程视图中，创建“游戏”文件，如图4-123所示。

② 在“新建游戏定义”对话框的“游戏名称”文本框中输入“纵向射击”，游戏宽度设为176、高设为208，启动场景设为“游戏场景\纵向射击”，如图4-124所示。

提示：之所以将游戏窗口的宽和高设定为176×208，是因为这个数据是客户提供的

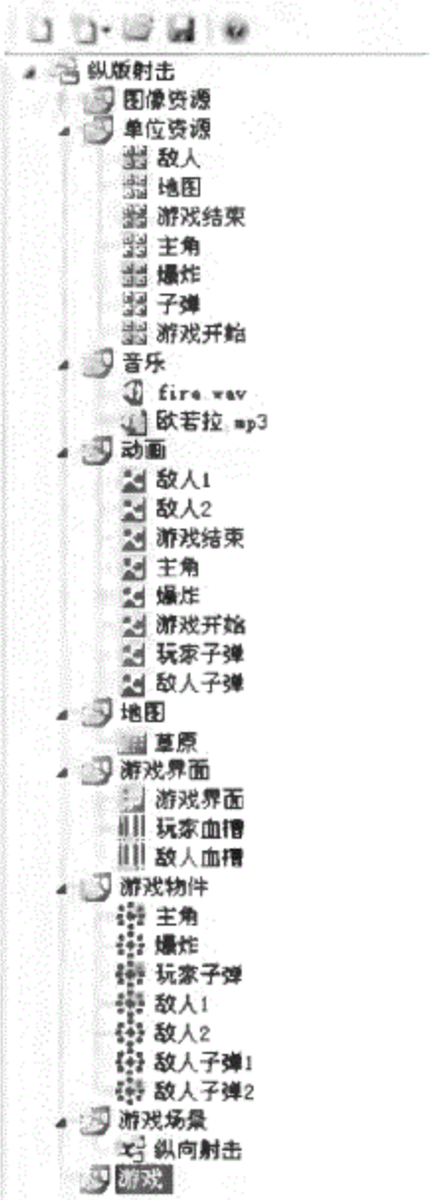


图4-122

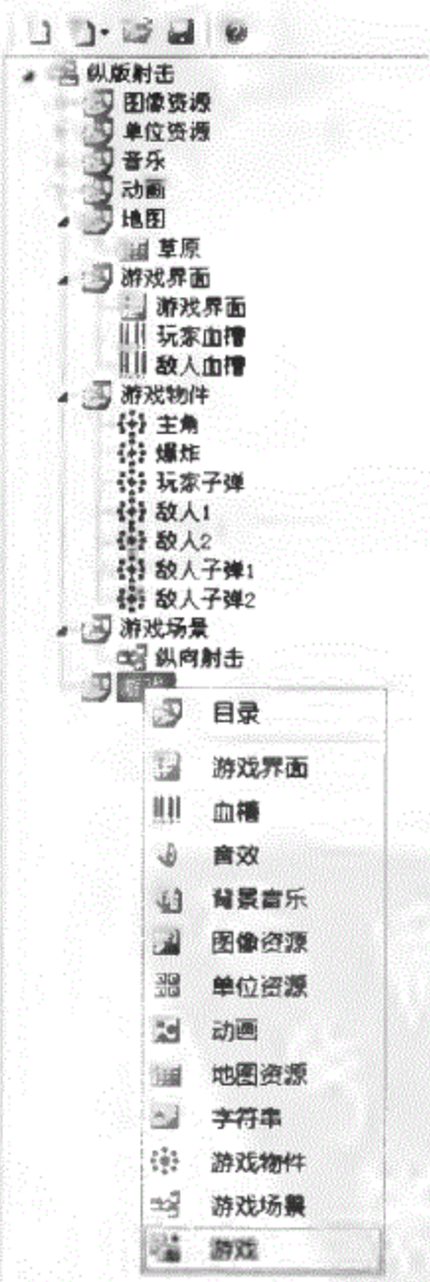


图4-123

图 4-124

诺基亚手机屏幕的尺寸，在此予以借用。

③ 单击“工具”下拉菜单中的“编译配置管理器”命令，如图 4-125 所示，打开“编译配置管理器”对话框，如图 4-126 所示，单击“新建”按钮，打开“新建编译配置”窗口，在“新建编辑配置的名称”文本框中输入“纵向射击”，如图 4-127 所示，最后单击“确定”按钮。

④ 在生成路径中确定新生成的游戏文件的位置，如图 4-128 所示，并将窗口中间的编译项目右侧的滚动条拉至最上方，选择“纵版射击”复选框，如图 4-129 所示，然后单击“关闭”按钮，关闭此对话框。

⑤ 单击“工具”→“编译”→“纵向射击”命令，对“纵向射击”配置进行编译，如图 4-130 所示。单击“纵向射击”命令后，系统自动弹出编译的“输出窗口”窗口。编译完成后关闭窗口，如图 4-131 所示。

至此，整个纵向射击游戏就制作完成了。如果需要检查这个游戏的效果，可以如前文所述使用“查看”下

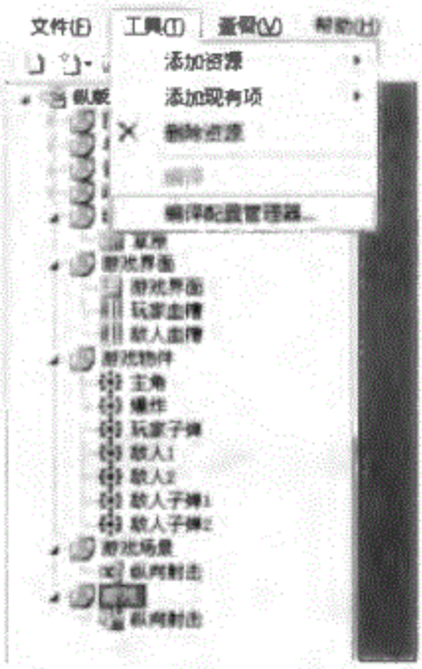


图 4-125

图 4-126

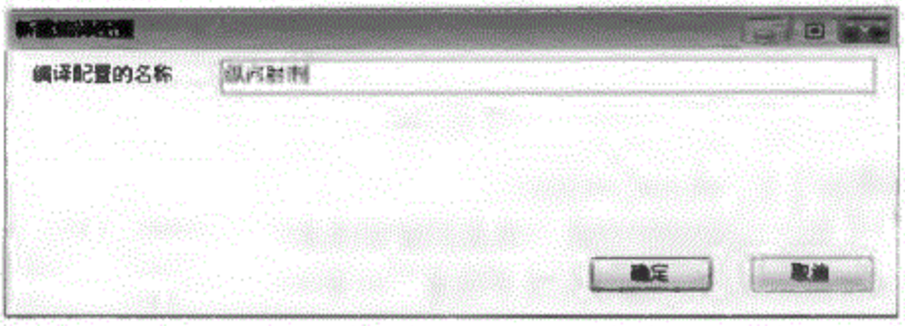

编译配置的名称
确定
取消

图 4-127

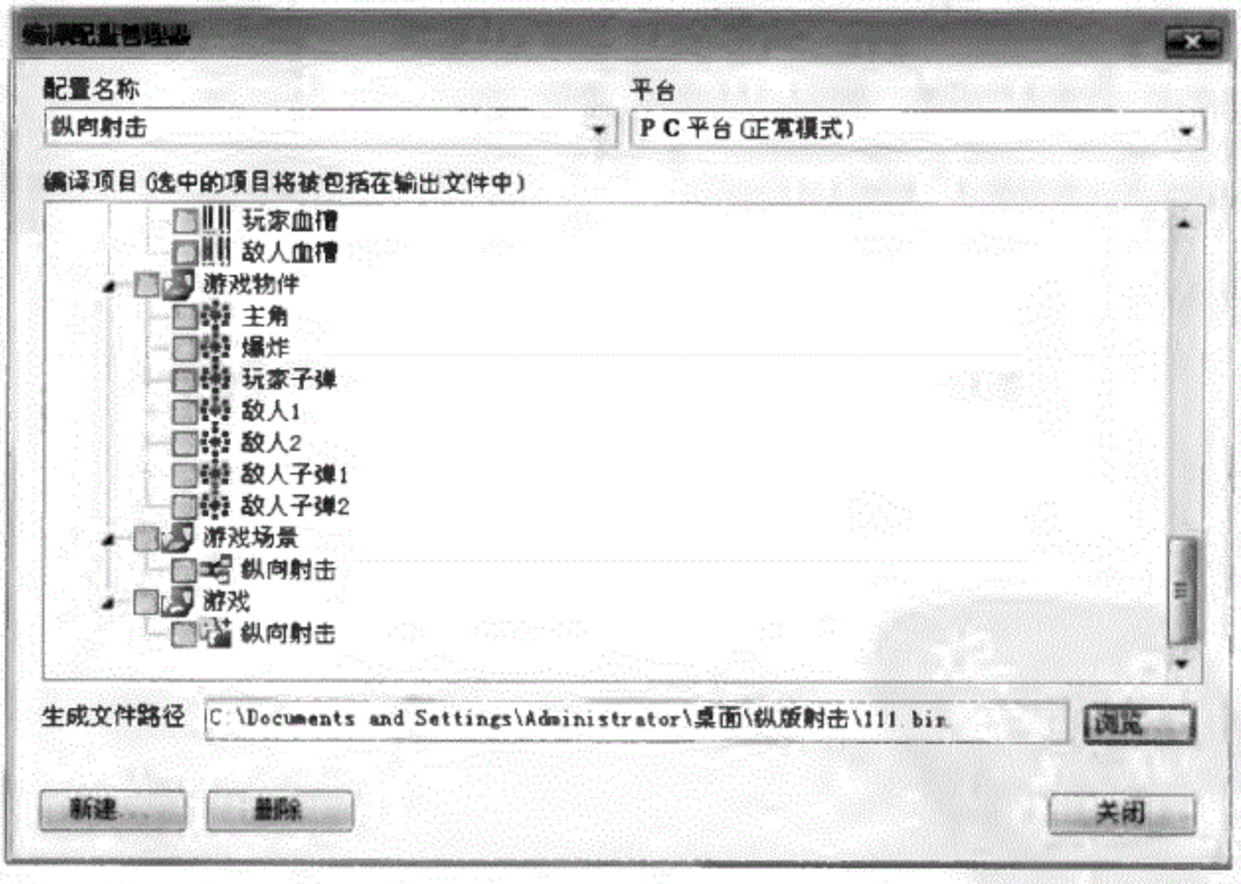

配置名称
纵向射击
平台
PC平台(正常模式)
编译项目(选中的项目将被包括在输出文件中)
玩家血槽
敌人血槽
游戏物件
主角
爆炸
玩家子弹
敌人1
敌人2
敌人子弹1
敌人子弹2
游戏场景
纵向射击
游戏
纵向射击
生成文件路径
C:\Documents and Settings\Administrator\桌面\纵版射击\111.bin
浏览
新建...
删除
关闭

图 4-128

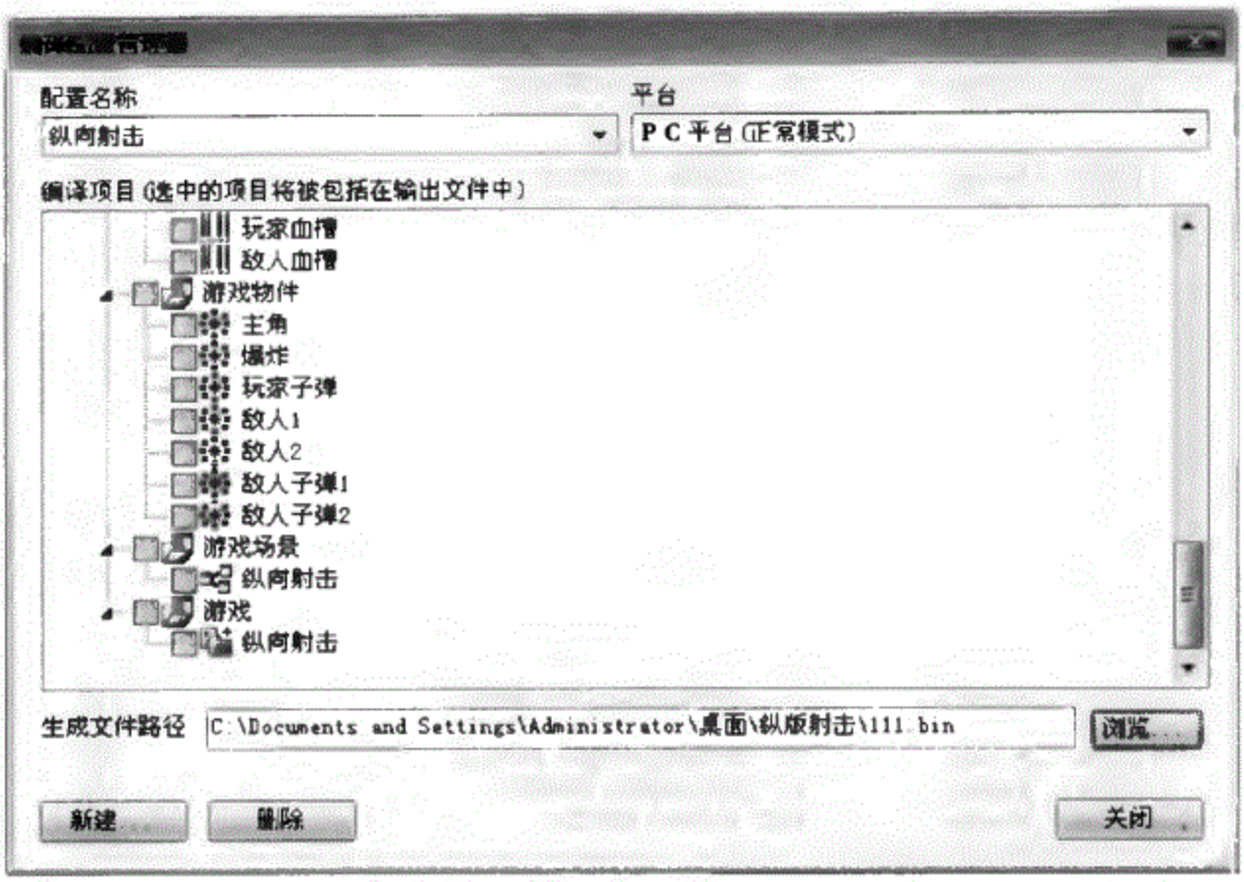

配置名称
纵向射击
平台
PC平台(正常模式)
编译项目(选中的项目将被包括在输出文件中)
玩家血槽
敌人血槽
游戏物件
主角
爆炸
玩家子弹
敌人1
敌人2
敌人子弹1
敌人子弹2
游戏场景
纵向射击
游戏
纵向射击
生成文件路径
C:\Documents and Settings\Administrator\桌面\纵版射击\111.bin
浏览...
新建...
删除
关闭

图 4-129


文件(F)
工具(T)
查看(V)
帮助(H)
添加资源
添加现有项
删除资源
编译
编译配置管理器...
所有配置
纵向射击

图 4-130

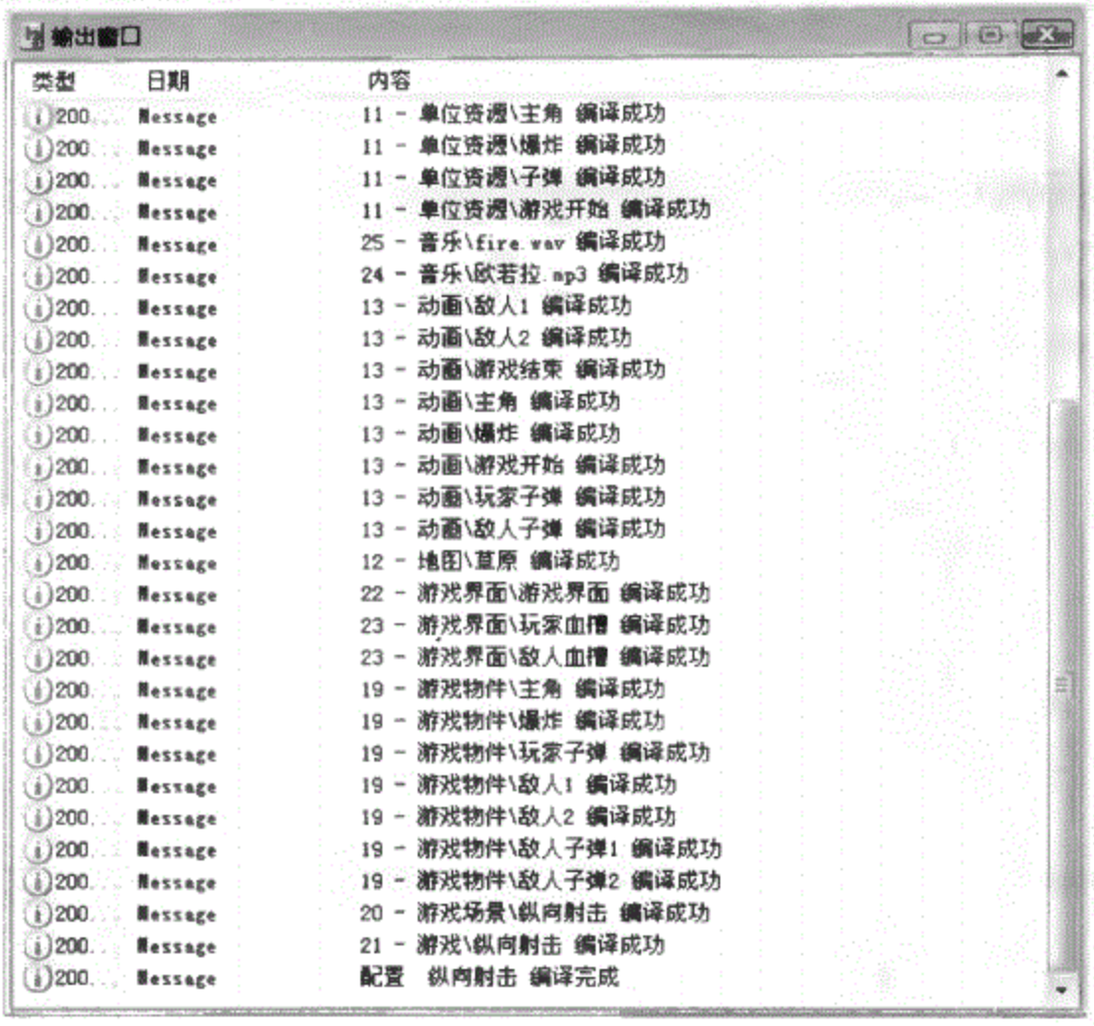


图 4-131

拉菜单中的“游戏测试”命令，如图 4-132 所示，来检查游戏的效果，并试玩由自己所制作完成的射击游戏，如图 4-133 所示，然后在回到工程文件中继续修改、调试，将整个游戏调整到最佳。

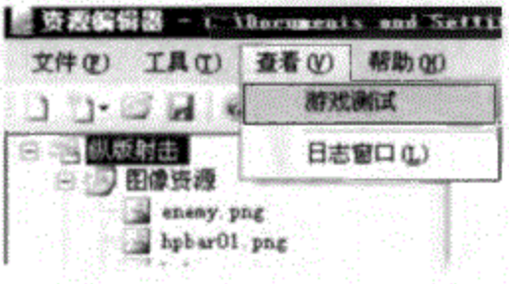


图 4-132

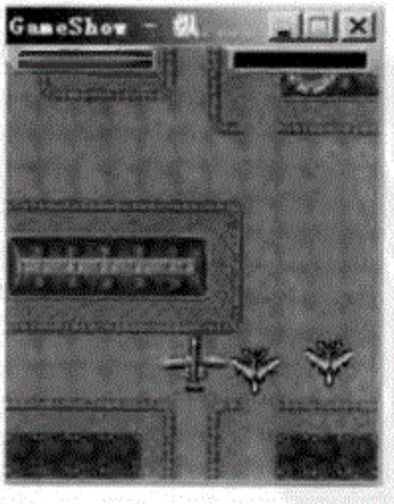


图 4-133

第 5 章　2D 游戏教学引擎实例制作（二）

在本章中，将继续以 2D 游戏教学引擎的另一个游戏种类“横版冒险类游戏”的制作来向大家介绍 2D 游戏教学引擎的使用。在制作之前，同上一章一样，必须先绘制好以下几张 png 格式的图片：“主角”、“杂兵 1”、“杂兵 2”、“BOSS”、“地图”以及其他游戏元件若干个；并且还有最后使用的血槽背景图片和血槽条；另外还有两个分别为 wav 格式和 mp3 格式的音乐文件作为背景音乐和音效文件。在“玩家”、“敌人 1”、“敌人 2”和“BOSS”的绘制过程中，必须分别为这些角色设计站立、行走、打斗、受伤和死亡的序列动画。玩家作为主要制作对象，则特别需要具体的细节和丰富的绘制。具体绘制过程在前面章节已有介绍，在此将不再展开介绍了。接下来就以这个案例来进行讲解。这个实例中有许多和上一个实例中相同的步骤和设置，为了节省篇幅，不在赘述。

5.1 图像资料的导入

① 和上一个实例相同，在打开 2D 游戏教学引擎，开始制作横版冒险类游戏之前，必须先创建游戏的目录。在“文件”菜单中，选择“新建项目”命令，如图 5-1 所示。

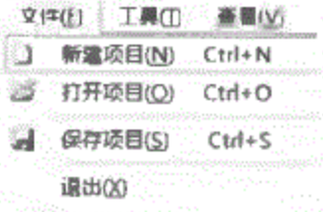


图 5-1

② 在打开“创建新项目”对话框后，在“项目名称”文本框中输入项目名称“横版冒险”，然后将项目保存在桌面上，（读者可以根据自己实际情况决定项目的存放路径），如图 5-2 所示。

③ 接下来，和上一个实例一样，开始图像文件的导入工

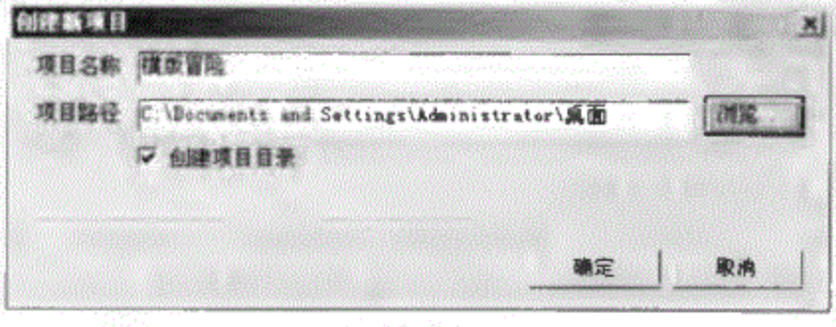


图 5-2

作。在这里就不再重复图像资源的导入制作过程。将图像资源完全导入至“图像资源”文件夹，如图 5-3 所示。

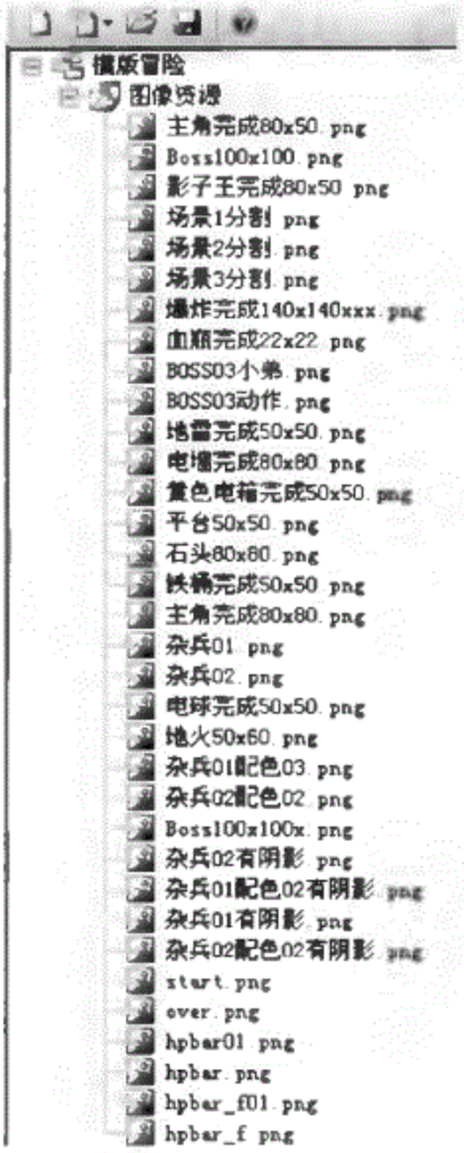


图 5-3

5.2 单位资源的制作

① 同样的，在完成图像资源的导入后，开始单位资源的制作，同样要创建名为“单位资源”的文件夹，如图 5-4 所示。

② 使用前面一个实例的方法，将所有的除了血槽图片以外的图片，包括“主角”、“杂兵”、“BOSS”、各种不同的障碍物、血瓶以及地图等图像资源，制作成单位大小合适的单位资源，并且统一放置在单位资源文件夹中，如图 5-5 所示。

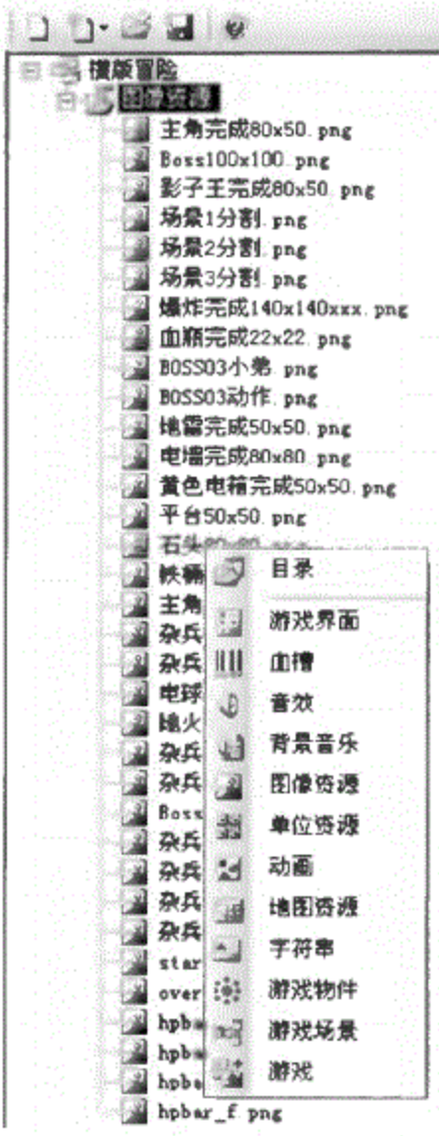


图 5-4

图 5-5

③ 在把图像资源分割成单位资源的时候，需要分别考虑主角、杂兵与 BOSS 之间的尺寸大小的比例。

5.3 音乐文件的导入

与上一章的音乐文件导入的过程相同。在创建完成名为音乐文件的文件夹后，使用上一章中的方法将名为“Fx13139_9mm1.wav”的文件导入为音效文件，将名为“BEYOND-03- 漆黑的空间 .mp3”导入为背景音乐，如图 5-6 所示。

图 5-6

5.4 制作动画资源

① 在创建完“动画资源”文件夹后，就要制作动画资源了，如图 5-7 所示。

② 首先还是来制作“主角”的动画资源。做为由玩家控制的角色，它的动画表现应该是最丰富的，所以在动画绘制过程中应该放置在最重要的位置，因此它的动画制作量在整个游戏中的动画制作中也是最大的，如图 5-8 所示。

③ 在玩家角色的动画制作中，正如大家所看到的那样，在绘制过程中，绘制者将“主角”的动作分别设定为：站、跑、刀、踢、跳、受伤和死亡 7 个动作。这样，在以后“主角”的游戏物件的定义中，就可以为不同的动作设定不同的动画，如图 5-9 所示。

④ 使用同上个实例相同的方法将其他的单位资源制作成动画资源。需要注意的是即使每个不同的敌人在单位资源中是在一起的，但是在制作动画的过程中却还是要将它们都分别制成单个的动画资源，如图 5-10 所示。


横版冒险
图像资源
单位资源
主角
boss
影子boss
场景1
场景2
场景3
爆炸
血瓶完成
BOSS03
BOSS03小弟
地雷完成
电墙完成
蓝色电箱完成
平台完成
石头
铁桶完成
真主角
杂兵01
杂兵02
电球完成
地火
杂兵01配色03
杂兵02配色02
boss x
zb1
zb2
zb3
zb4
开始
结束
音乐文件
Fx13139_9mm1.wav
BEYOND-03-漆黑的空间.mp3
动画资源

图 5-7

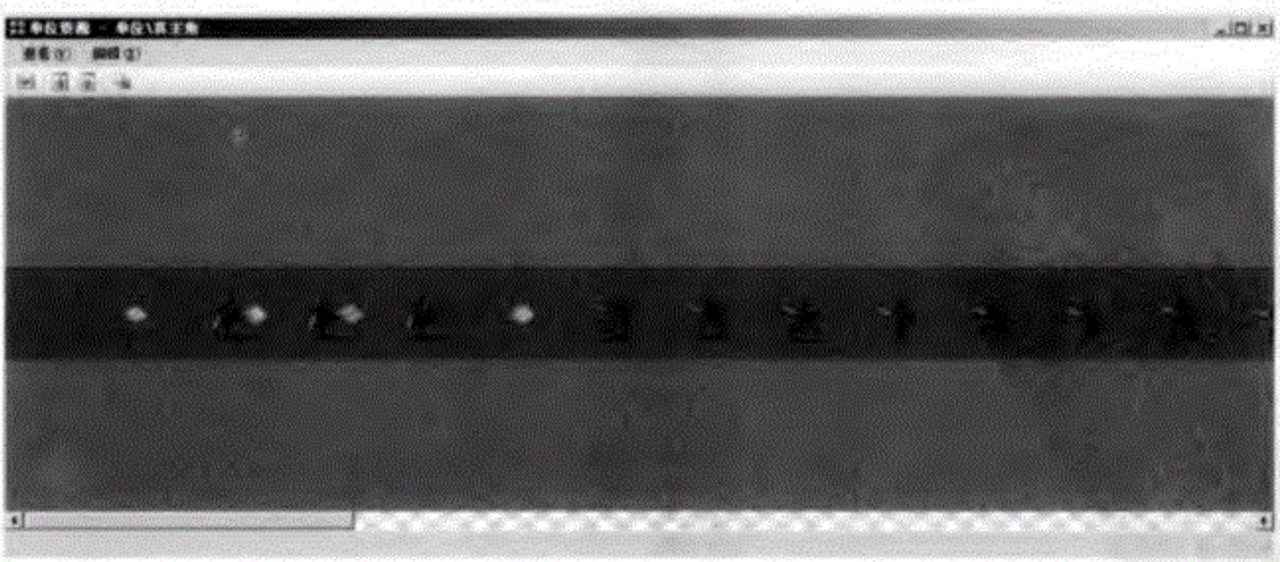


图 5-8

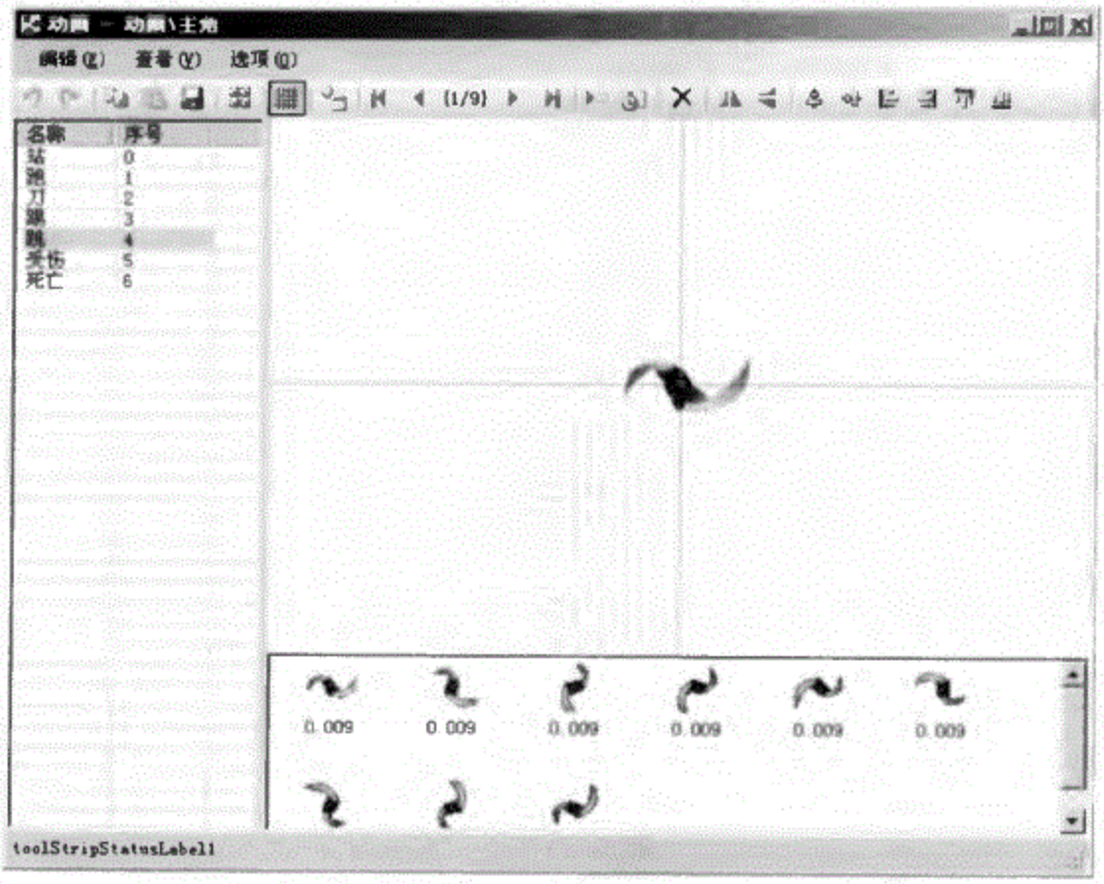

动画 - 动画\主角
编辑(E) 查看(V) 选项(O)
(1/9)
名称 序号
站 0
跑 1
刀 2
跳 3
跳 4
受伤 5
死亡 6
0.009
0.009
0.009
0.009
0.009
0.009
toolStripStatusLabel1

图 5-9


横版冒险
图像资源
单位资源
音乐文件
动画资源
主角
boss
影子boss
boss爆炸
血瓶完成
BOSS03
BOSS03小弟
地雷完成
电墙完成
黄色电箱完成
平台完成
石头
铁桶完成
杂兵01
杂兵02
电球完成
中爆炸
地火
杂兵01配色03
杂兵02配色02
开始
结束

图 5-10

⑤ 另外在横版冒险的动画资源的制作中，除了要制作“主角”、“杂兵”等角色动画外，还需要制作一些道具和障碍物的动画，这些是为了更好地增添游戏的可玩性。

5.5 游戏界面制作

① 同样，在创建完成“游戏界面”文件夹后，制作游戏的界面（即 UI）和游戏中角色的血槽，如图 5-11 所示。

图 5-11

② 在制作游戏界面中，将前面制作好的“动画\游戏开始”和“动画\游戏结束”分别导入进去，如图 5-12 所示。

③ 血槽的制作分为玩家血槽和敌人血槽两种，在上一章中已经谈过这个问题。使用上一章的方法将不同的背景图片和血条图片导入到血槽的制作窗口中，并且在制作中，需要注意的是玩家的血槽是始终显示的；而敌人的血槽可以不是始终显示的，如图 5-13 所示。

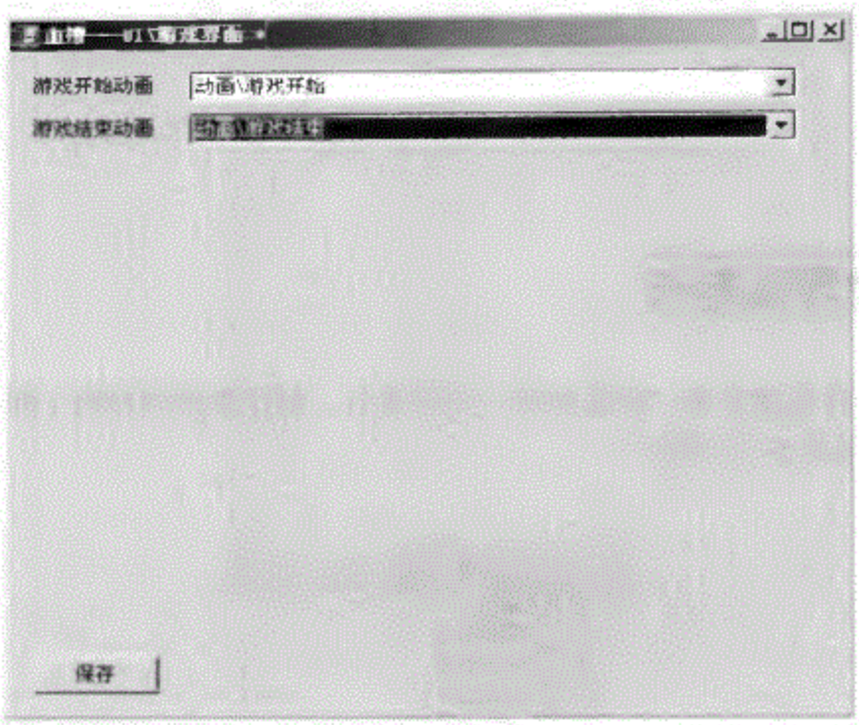

游戏开始动画
动画\游戏开始
游戏结束动画
保存

图 5-12

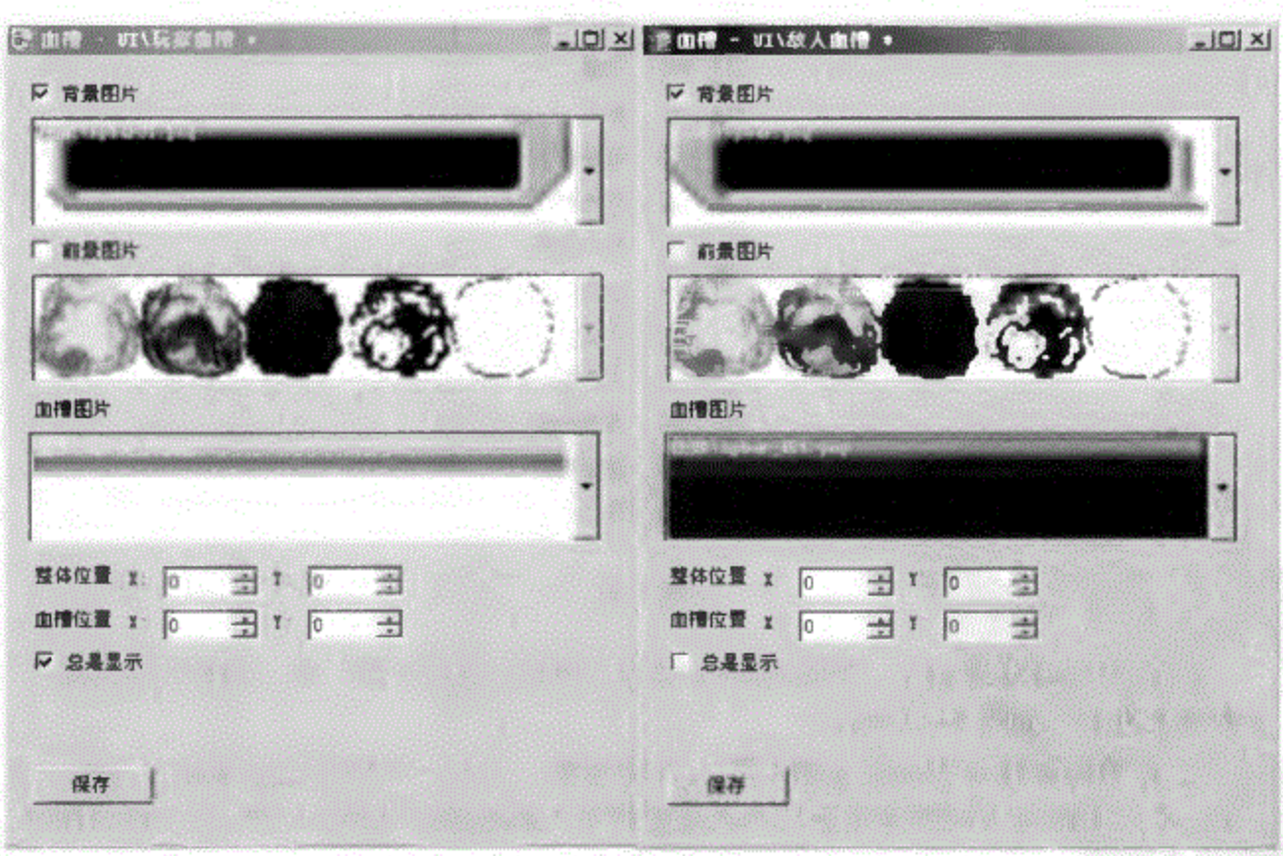

背景图片
前景图片
血槽图片
整体位置 X: 0 Y: 0
血槽位置 X: 0 Y: 0
总是显示
保存
背景图片
前景图片
血槽图片
整体位置 X 0 Y 0
血槽位置 X 0 Y 0
总是显示
保存

图 5-13

5.6 地图的制作

横版冒险类游戏地图的制作与纵版射击类游戏地图的制作的原理以及方法相同，只是纵版射击类游戏的地图是高且窄，即高度很高，而宽度相对很窄；而横版冒险类游戏的地图则正好相反，宽度很宽，高度相对很矮，如图 5-14 所示。

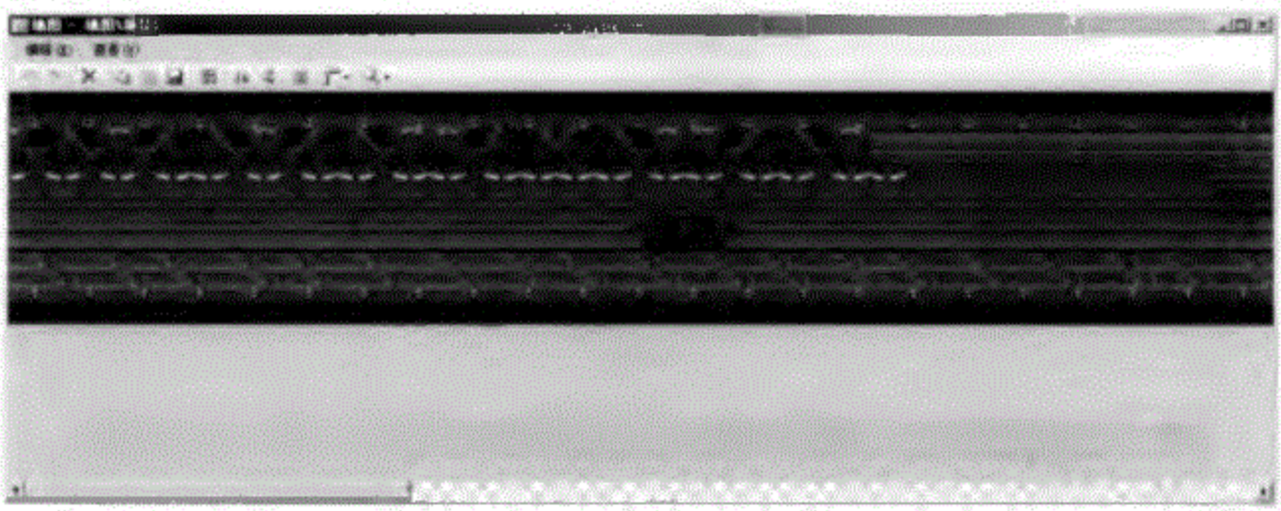


图 5-14

5.7 游戏物件的定义

由于横版冒险类游戏与纵版射击类游戏在游戏性上的不同，所以对于游戏物件的定义则可以说大相径庭。在这一节中，将会详细地介绍横版冒险类游戏的游戏物件定义。

① 按照惯例，同样还是先创建“物件”文件夹，如图 5-15 所示。

② 第一个先来定义玩家的游戏物件。在创建游戏物件窗口中，输入“物件名称”为“物件\主角”，“使用动画”为“动画\主角”。创建完成后再一次打开“主角”的游戏物件设置窗口。“碰撞范围”的设置和之前的纵版射击类游戏的游戏物件的设置是一样的，接下来的“一般属性”也可以按照自己的需要进行设置。只是在设置时，选中“使用血槽”复选框，并在“使用血槽”复选框下方的下拉列表框中选择“UI\玩家血槽”，如图 5-16 所示。

③ 接下来设定 4 种脚本。经过上一章对纵版射击类游戏的游戏物件的定义，读者

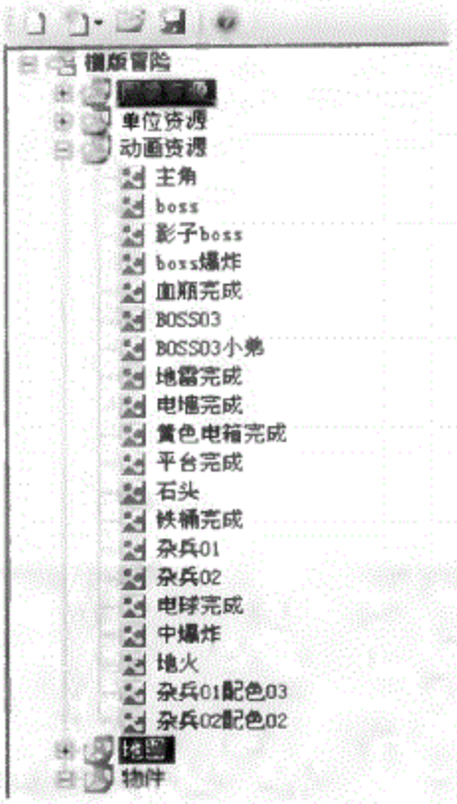

模版冒险
单位资源
动画资源
主角
boss
影子boss
boss爆炸
血瓶完成
BOSS03
BOSS03小弟
地雷完成
电墙完成
黄色电箱完成
平台完成
石头
铁桶完成
杂兵01
杂兵02
电球完成
中爆炸
地火
杂兵01配色03
杂兵02配色02
地图
物件

图 5-15


游戏物件 - 物件\主角 *
物件名称 物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -16
上 -20
宽度 30
高度 41
一般属性
HP 100
使用血槽
使用道具栏
攻击值 16
UI\玩家血槽
脚本属性
脚本类型 初始化脚本
测试
模板

图 5-16

对初始化脚本、碰撞测试脚本、运行脚本和被伤害脚本应该已经相当熟悉了，如图 5-17 所示。

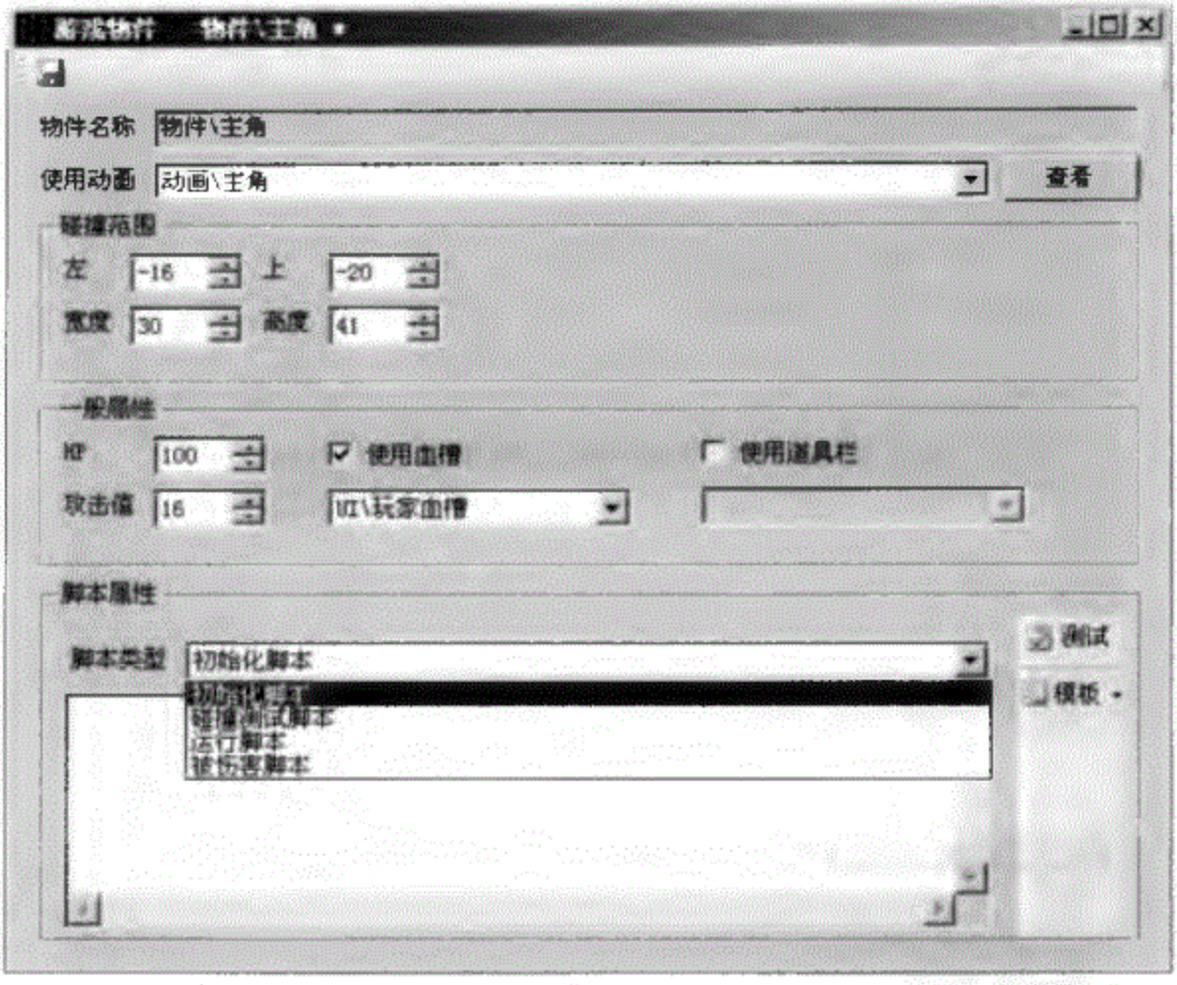


图 5-17

④ 在初始化脚本的设定中，还是一贯地使用一般初始化，如图 5-18 所示。

⑤ 在碰撞测试脚本的设定中，就不再使用上一章中用的那个“普通命中测试”了，而是使用“动作游戏命中测试”，这个脚本不用调节任何数值，直接设置下一个脚本即可，如图 5-19 所示。

⑥ 在整个游戏物件定义中最复杂也是最核心的大概就是运行脚本。在运行脚本中，对于玩家的控制，则使用的是模板中的横版冒险中的主角运动。在确定好基础的脚本模板后，还需要对其中的具体参数进行详细的设置，如图 5-20 所示。

⑦ 在对玩家的运行脚本进行设置过程中，先来进行音效的设置，在这里将已经导入到音乐文件中的音效文件与所需要使用的音效位置一一对应起来。对每个音效文件中都需要设置它的路径，例如，“音乐文件 //Fx13139_9mm1.wav”，如图 5-21 所示。


游戏物件 - 物件\主角 *
物件名称 物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -16 上 -20
宽度 30 高度 41
一般属性
HP 100 使用血槽 使用道具栏
攻击值 16 UI\玩家血槽
脚本属性
脚本类型
测试
模板
一般初始化

图 5-18


游戏物件 - 物件\主角 *
物件名称 物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -16 上 -20
宽度 30 高度 41
一般属性
HP 100 使用血槽 使用道具栏
攻击值 16 UI\玩家血槽
脚本属性
脚本类型
测试
模板
return false;
普通命中测试
隐身模式
动作游戏命中测试

图 5-19


物件名称 物件\主角
使用动画 动画\主角
查看
左 -16 上 -20
宽度 30 高度 41
一般属性
HP 100 使用血槽 使用道具栏
攻击值 16 UI\玩家血槽
脚本属性
脚本类型
测试
模板
纵版射击
纵版冒险
RPG
格斗游戏
静止
爆炸效果
敌人运动
道具
障碍物

图 5-20

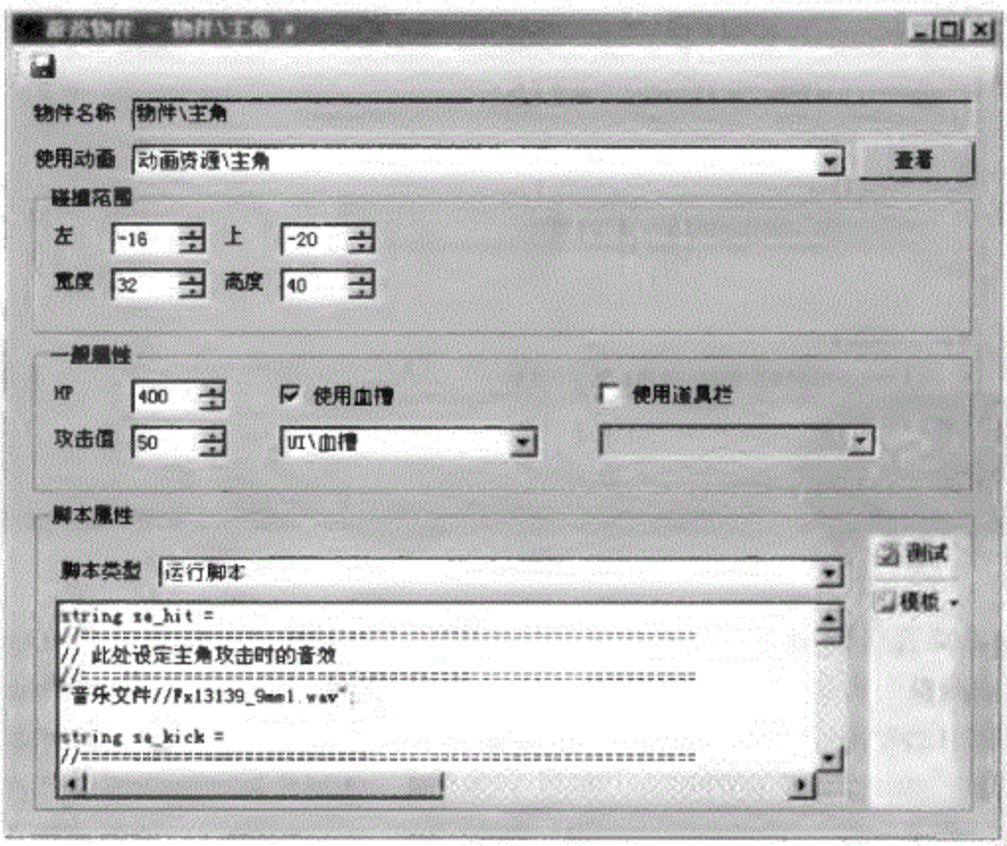

物件名称 物件\主角
使用动画 动画资源\主角
查看
碰撞范围
左 -16 上 -20
宽度 32 高度 40
一般属性
HP 400 使用血槽 使用道具栏
攻击值 50 UI\血槽
脚本属性
脚本类型 运行脚本
测试
模板
string se_hit =
//===
// 此处设定主角攻击时的音效
//===
"音乐文件//Fx13139_9mm1.wav";
string se_kick =
//===

图 5-21

⑧ 接下来进行速度的设置。需要设置的速度有移动速度、跳跃后落下的加速度、跳跃的初始速度和发生碰撞的速度。对于移动速度的设置，必须要确定玩家所控制的角色是敏捷的，即速度快；还是迟钝的，即速度慢。如果是速度快的，在制作过程中，一般将移动速度设置在 3.0 以上，而即使是速度慢的，也必须将移动速度设置为大于 1.0，因为如果移动速度为 1.0 或小于它，则正好等同于卷轴的移动速度，角色在窗口中的位置会保持固定不变。其他几个速度值也是如此，如图 5-22 所示。

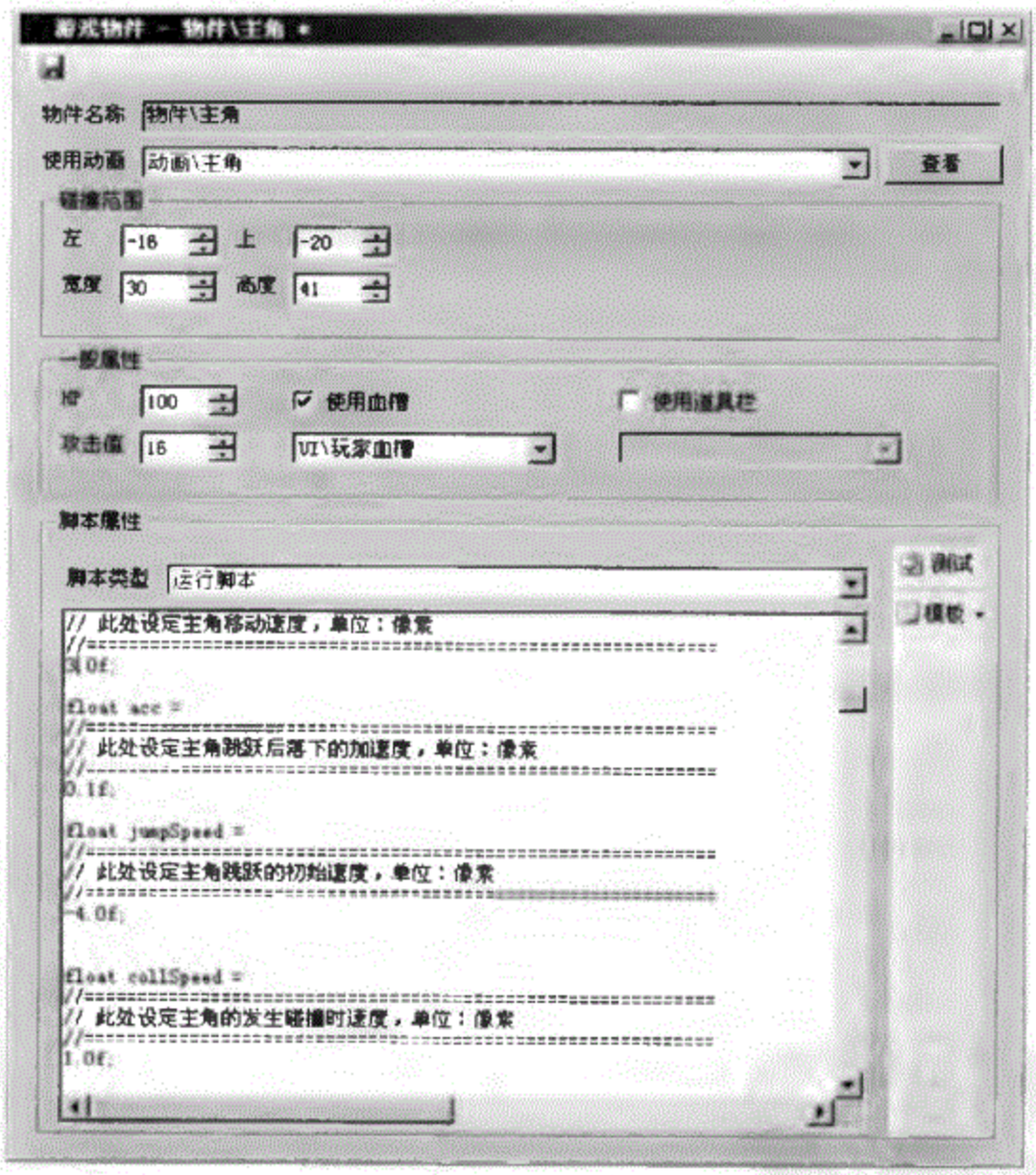


图 5-22

⑨ 接下来设置动画序号。通过上一章的制作，读者都已经知道这里的动画是需要和之前的动画窗口中的序列号相关联的。在这个时候，就需要打开玩家的动画制作窗口，来对照动画中动作的序号进行设置。以设定最基本的普通动画为例，所谓普通动画即没有人控制角色的时候所呈现的动画，即站立的动画，这时候找到动画窗口中站立动作的序号，将它填入，那么下面的移动动画、跳跃动画、攻击动画 1、攻击动画 2、被攻击动画和死亡动画，都可以这样去设置，如图 5-23 所示。

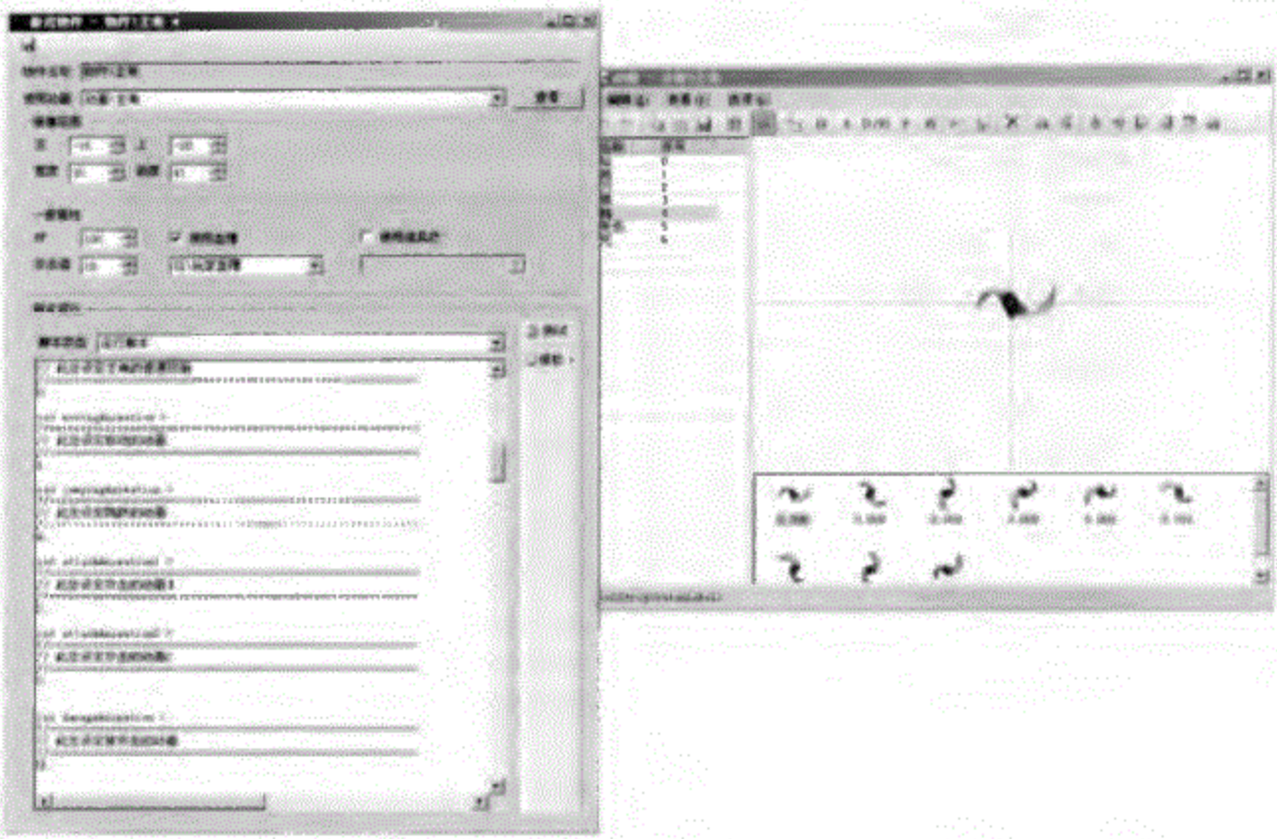


图 5-23

⑩ 在设置运行脚本的最后有一个设置地面的坐标，在这个设置项中，需要在地图中设置一个高度，作为玩家行走的地面，这个位置则需要在制作地图的时候就计算出来，并且经过观察已经生成的游戏，再去调节地面坐标的高低，如图 5-24 所示。

⑪ 最后设置被伤害脚本。在被伤害脚本的模板中选择动作游戏伤害模板。在这个模板里面，只有两个参数需要设置，而且都是与障碍物有关，与玩家的设置无关，这部分会在接下来对后面的障碍物的设置中进行讨论，如图 5-25 所示。

⑫ 这样，一个玩家的游戏物件设置便完成了。在之后的其他物件的设置中，将不再复述相同的地方，只讲解不同的地方，以缩短篇幅。

⑬ 接下来以 BOSS 为例来讲解横版冒险类游戏中敌人的设置。同样要创建名称为 BOSS 的游戏物件，并且使用的动画为动画资源中 BOSS 的动画。然后打开创建好的 BOSS 游戏物件，在“碰撞范围”和“一般属性”选项组中，使用以往的方法将碰撞范围计算出来，设置 HP 和攻击值，并选中“使用血槽”复选框，如图 5-26 所示。

⑭ 然后打开脚本设置，在设定初始脚本和碰撞测试脚本的模板中，都是用与玩家游戏物件相同的方法设置。在运行脚本中，则需要调用横版冒险的敌人运动，这里面的设定和玩家的运行脚本设置一样，制作时请参照玩家脚本的设置。在最后的被伤害脚本中，也一样使用动作游戏被伤害模板。这样 BOSS 游戏物件也设置完成了，并且这个设置可以套用到所有不同的敌人游戏物件中，如图 5-27 所示。

⑮ 在横版冒险类游戏中除了最基本的玩家类游戏物件和敌人类游戏物件外，还有另外两项，即障碍物类游戏物件和道具类游戏物件。接下来就以“电墙”为例来讲解障

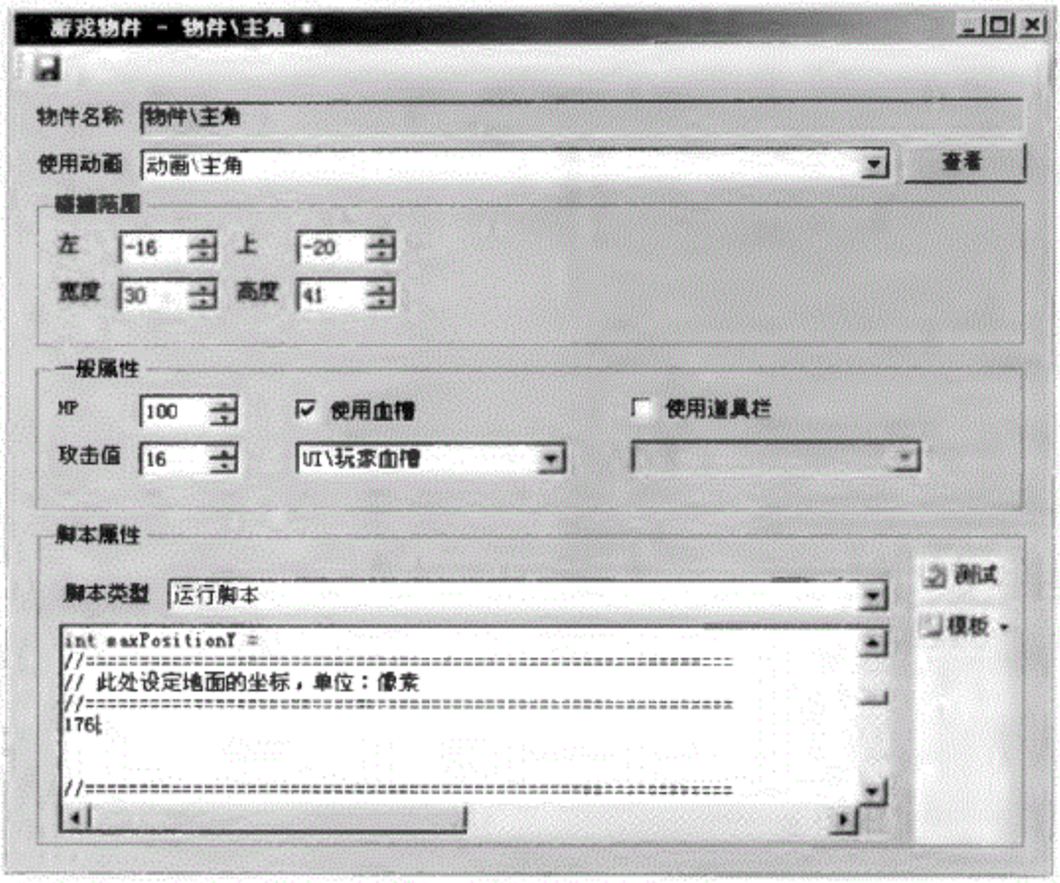

游戏物件 - 物件\主角 *
物件名称 物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -16 上 -20
宽度 30 高度 41
一般属性
HP 100 使用血槽 使用道具栏
攻击值 16 UI\玩家血槽
脚本属性
脚本类型 运行脚本
int maxPositionY =
// 此处设定地面的坐标，单位：像素
176;
测试
模板

图 5-24


游戏物件 - 物件\主角 *
物件名称 物件\主角
使用动画 动画\主角
查看
碰撞范围
左 -16 上 -20
宽度 30 高度 41
一般属性
HP 100 使用血槽 使用道具栏
攻击值 16 UI\玩家血槽
脚本属性
脚本类型
零伤害
普通伤害
动作游戏伤害
测试
模板

图 5-25


物件名称
物件\boss
使用动画
动画\boss
查看
左
-35
上
-28
宽度
63
高度
56
HP
1000
使用血槽
使用道具栏
攻击值
100
UI\敌人血槽
脚本类型
初始化脚本
测试
模板

图 5-26

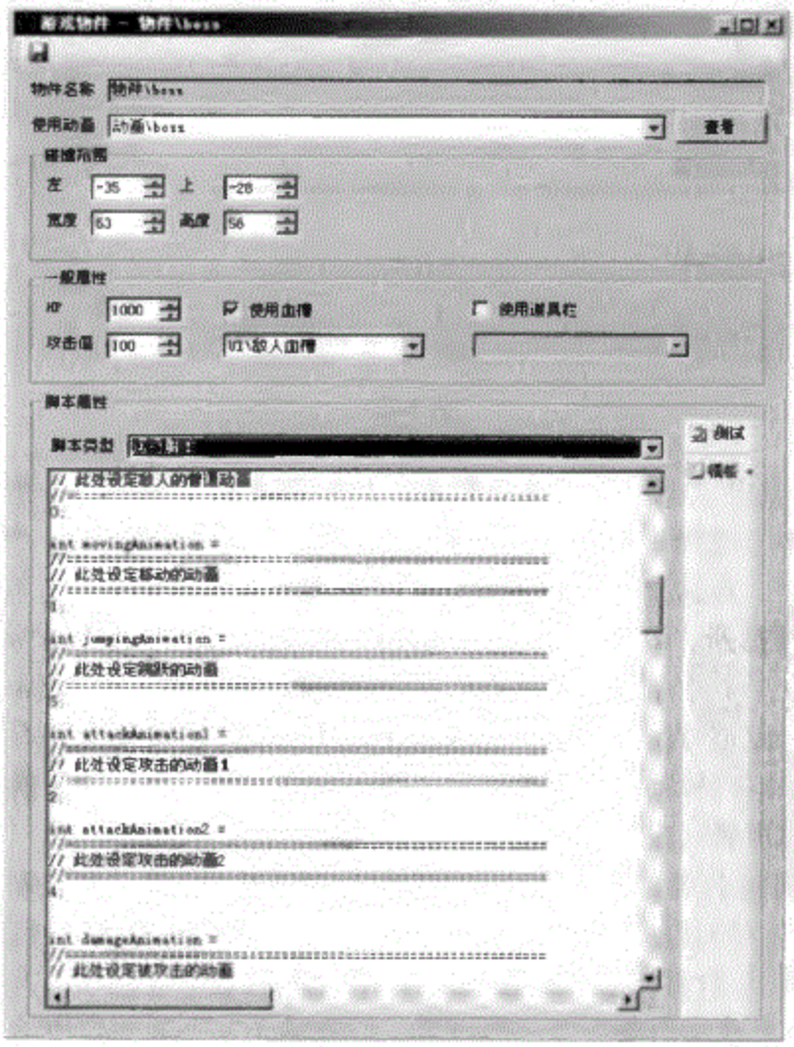

物件名称
物件\boss
使用动画
动画\boss
查看
左
-35
上
-28
宽度
63
高度
56
HP
1000
使用血槽
使用道具栏
攻击值
100
UI\敌人血槽
脚本类型
测试
模板
// 此处设定敌人的普通动画
0;
int movingAnimation =
// 此处设定移动的动画
1;
int jumpingAnimation =
// 此处设定跳跃的动画
5;
int attackAnimation1 =
// 此处设定攻击的动画1
2;
int attackAnimation2 =
// 此处设定攻击的动画2
4;
int damageAnimation =
// 此处设定被攻击的动画

图 5-27

碍物类游戏物件的设置。与之前一样，创建完成“电墙”游戏物件，再打开已经创建的“电墙”游戏物件的设置窗口，计算出碰撞范围。在预先的设想中，“电墙”是具有攻击力，只能摧毁而不能直接穿过的障碍物，所以在设定一般属性时，需要设定“电墙”的HP和攻击值，并且也要设定使用的血槽类型，如图5-28所示。不过如果设定的是一般的无攻击的障碍物的话，这里的攻击值则为0。

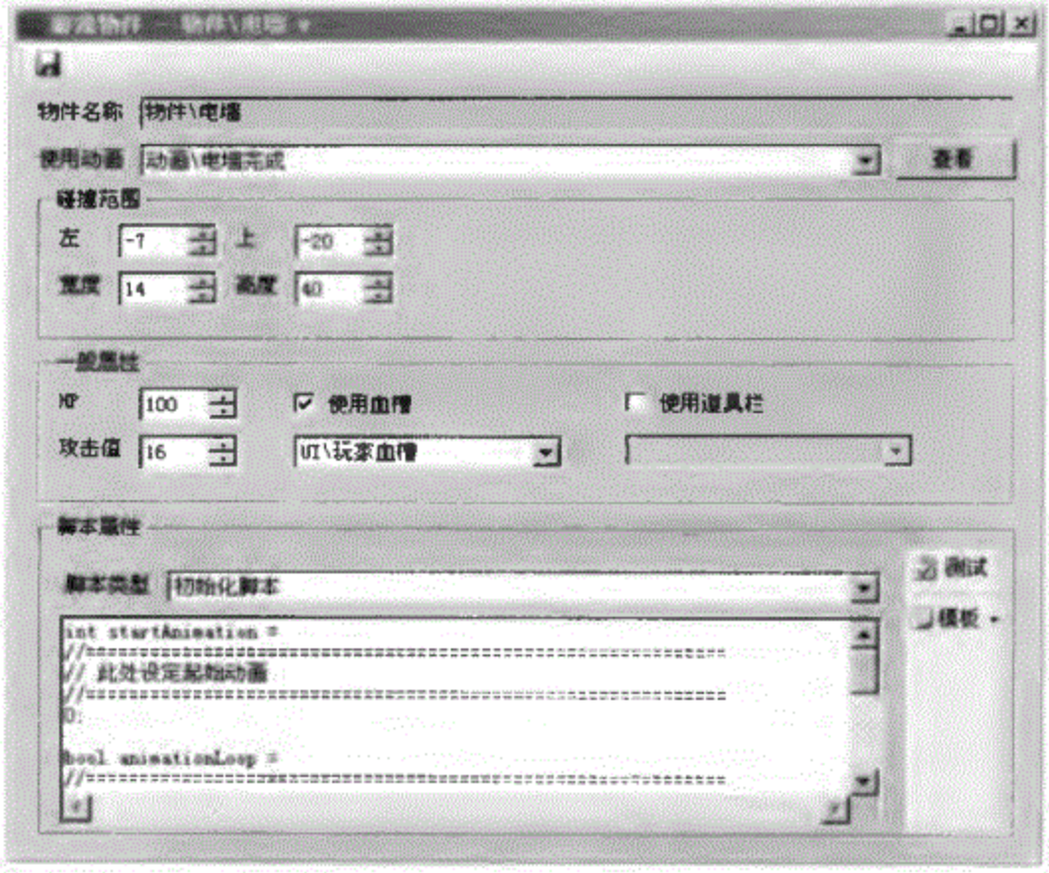


图5-28

⑯ 接下来是设定“电墙”的4个脚本。初始脚本总是选择“一般初始化”模板，在碰撞测试脚本中，选择“动作游戏命中测试”。在运行脚本中使用“横版冒险”类中的“障碍物”命令。在这个脚本中需要设定“电墙”被摧毁的动画，同样找到“电墙”的动画窗口，并且找到“电墙”破碎的动作序号，输入即可。最后设定是否能被摧毁，按照提示输入true字符，表示能被摧毁；能否穿越，按照提示输入false字符，表示不能直接穿过。最后设定被伤害脚本，在这里还是使用动作游戏伤害脚本，并且在这里需要设定是否是可以被打掉的路障，按照提示输入true字符，表示可以被摧毁。这样“障碍物”游戏物件也设置完成了，如图5-29所示。

⑰ 最后来设置道具类游戏物件，以能帮助玩家恢复HP的“血”来向大家说明道具类游戏物件的设置。同之前的操作一样，创建“血”游戏物件，再打开已经创建的“血”游戏物件的设置窗口，计算出它的碰撞范围。由于“血”是没有任何攻击力也不需要HP的，所以在这里的HP和攻击值都可以不用设置，同样由于它不需要血槽的显示，所以也无须选择“使用血槽”复选框，如图5-30所示。

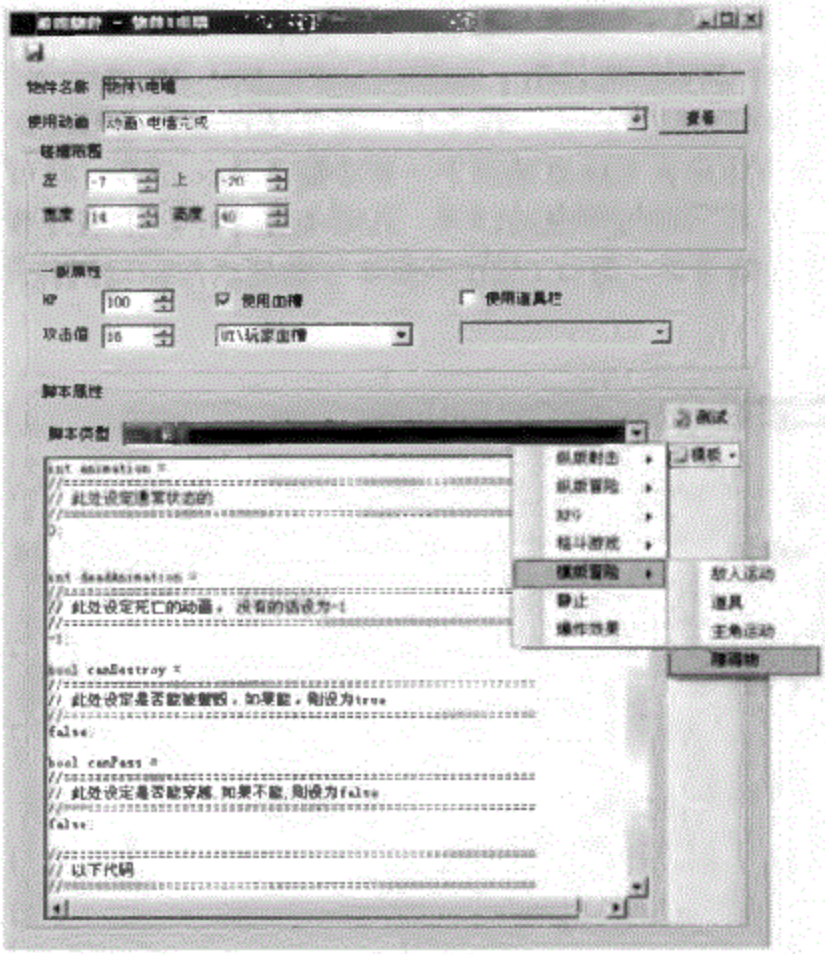

物件名称
物件\电锤
使用动画
动画\电锤完成
查看
碰撞范围
左
上
宽度
高度
一般属性
HP
使用血槽
使用道具栏
攻击值
脚本属性
脚本类型
测试
模板
int animation =
// 此处设定通常状态的
int deadAnimation =
// 此处设定死亡的动画，没有的话设为-1
bool canDestroy =
// 此处设定是否能被摧毁，如果能，则设为true
false;
bool canPass =
// 此处设定是否能穿越，如果不能，则设为false
false;
// 以下代码
射击游戏
格斗游戏
模板冒险
静止
操作效果
敌人活动
道具
主角活动
障碍物

图 5-29

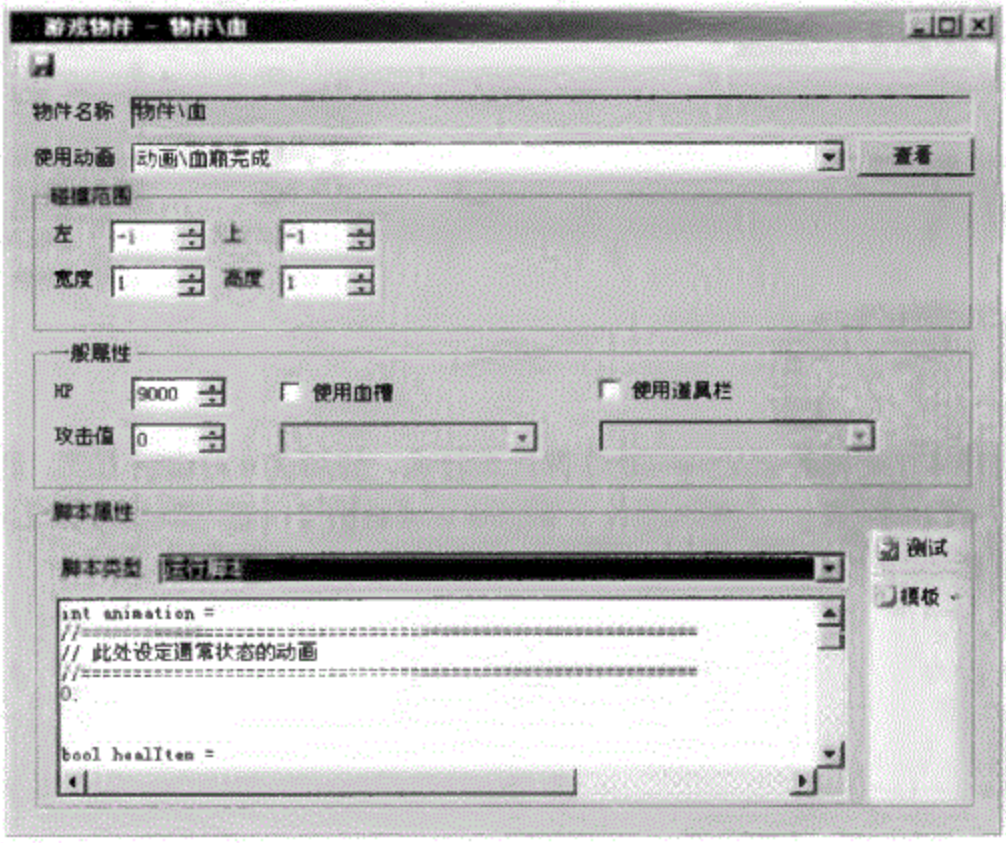

游戏物件 - 物件\血
物件名称
物件\血
使用动画
动画\血瓶完成
查看
碰撞范围
左
上
宽度
高度
一般属性
HP
9000
使用血槽
使用道具栏
攻击值
0
脚本属性
脚本类型
测试
模板
int animation =
// 此处设定通常状态的动画
0;
bool healItem =

图 5-30

⑱ 然后来设定“血”的 4 个脚本。初始脚本和碰撞测试脚本与“电墙”的设置相同，即初始脚本总是选择一般初始化模块；碰撞测试脚本选择动作游戏命中测试。在运行脚本中选择横版冒险的道具，在这个脚本中，需要选择该道具为补血道具还是无敌道具，只要在补血道具或者无敌道具的下一栏中输入 true 字符，就可以将该道具设定为相应道具，除了设定好何种道具以外，还需要在下一栏中输入数值，即回血量的多少或者无敌时间的多少。最后，被伤害脚本可使用动作游戏伤害模块，如图 5-31 所示。

图 5-31

⑲ 这样 4 个不同种类的游戏物件都设置完成，使用这 4 种设置方法，我们将所有的游戏物件都制作完成。当游戏物件全部制作完成后则可以进入游戏场景设定，将游戏物件与地图整合在一起，如图 5-32 所示。

图 5-32

5.8 场景设置

① 进行游戏场景的设置。对于游戏场景的设置，和上一章的制作方式是一样的。在确定好导入的地图文件后，将玩家和各种不同的敌人（包括杂兵和 BOSS），还有障碍物和道具等放入地图中，如图 5-33 所示。

② 这个时候不要忘记将主角的属性中的类型改为友方，如图 5-34 所示，不然的话，在运行游戏中主角和敌人将不会发生敌友判断。

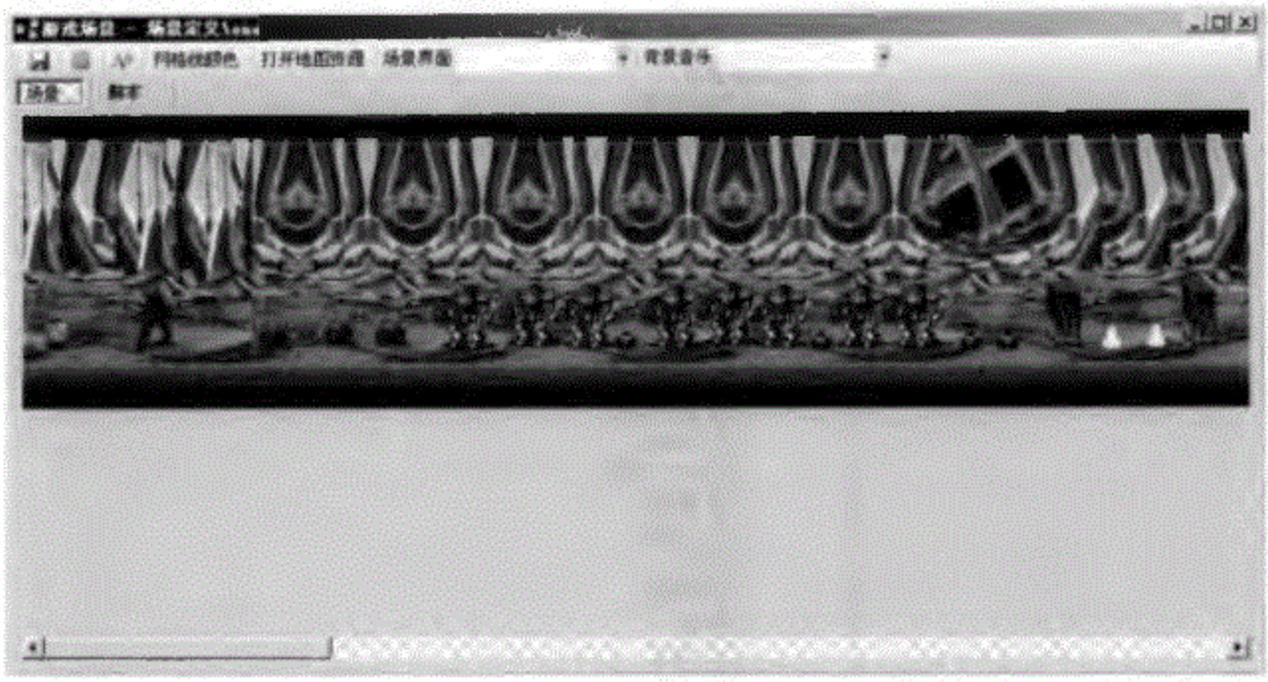


图 5-33

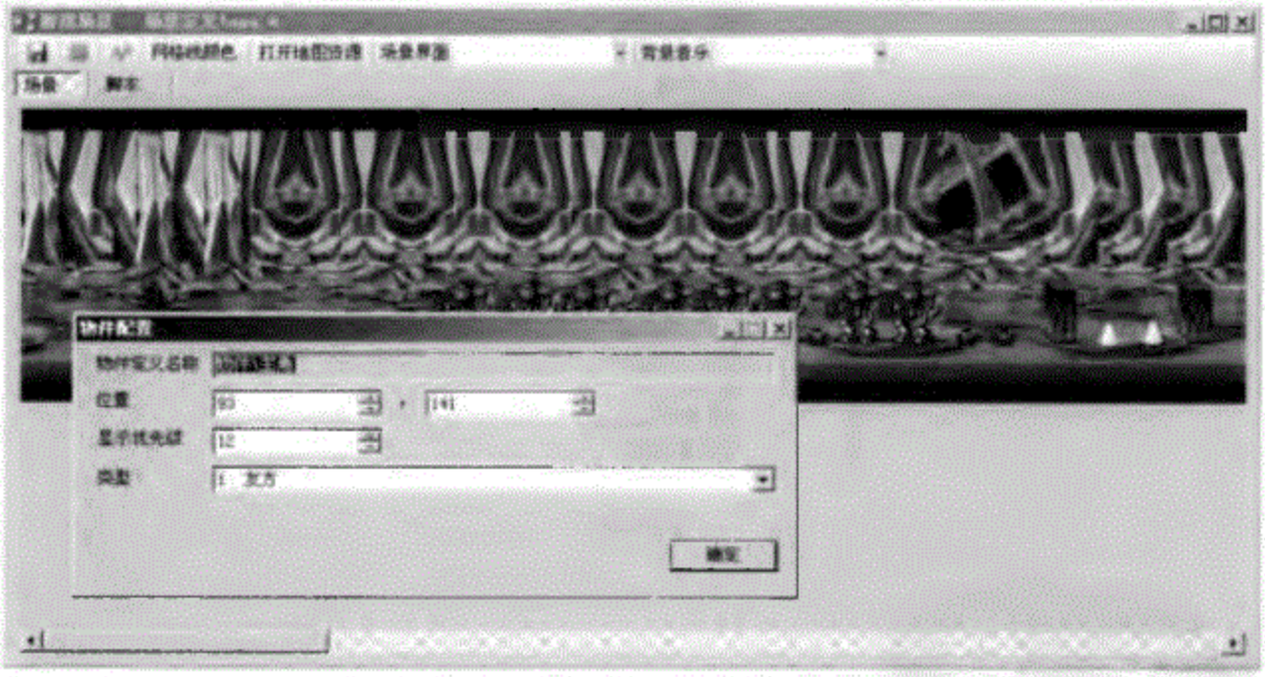


图 5-34

③ 将已经完成的场景界面调入，再将已经导入的背景音乐“BEYOND-03- 漆黑的空间 .mp3”调入到场景中，如图 5-35 所示。

④ 进入脚本编辑，调用“横版冒险 .script”，如图 5-36 所示，确定游戏窗口在整个场景中的运动规律。

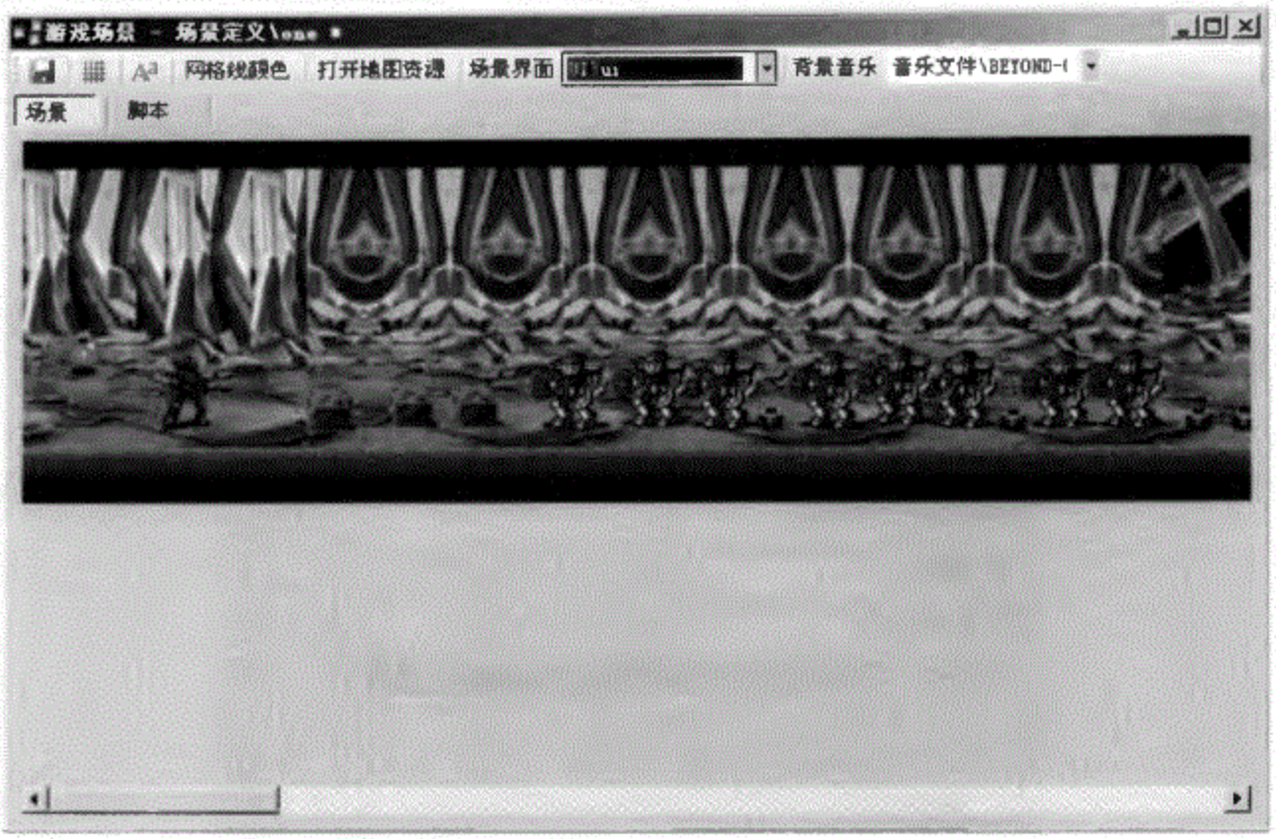

游戏场景 - 场景定义\one *
网格线颜色
打开地图资源
场景界面
UI\ui
背景音乐
音乐文件\BEYOND-(
场景
脚本

图 5-35


游戏场景 - 场景定义\one *
网格线颜色
打开地图资源
场景界面
UI\ui
背景音乐
音乐文件\BEYOND-(
场景
脚本
纵版射击.script
纵版冒险.script
横版冒险.script
RPG.script
scene.SetCenter(scene.Players[0].PositionX, scene.Players[0].PositionY);

图 5-36

5.9 定义、编译并生成游戏

① 接下来的制作和上一章对应的制作方法一样。要创建“游戏”文件，并且确定游戏的名称、游戏窗口的大小以及使用的游戏场景，如图 5-37 所示。

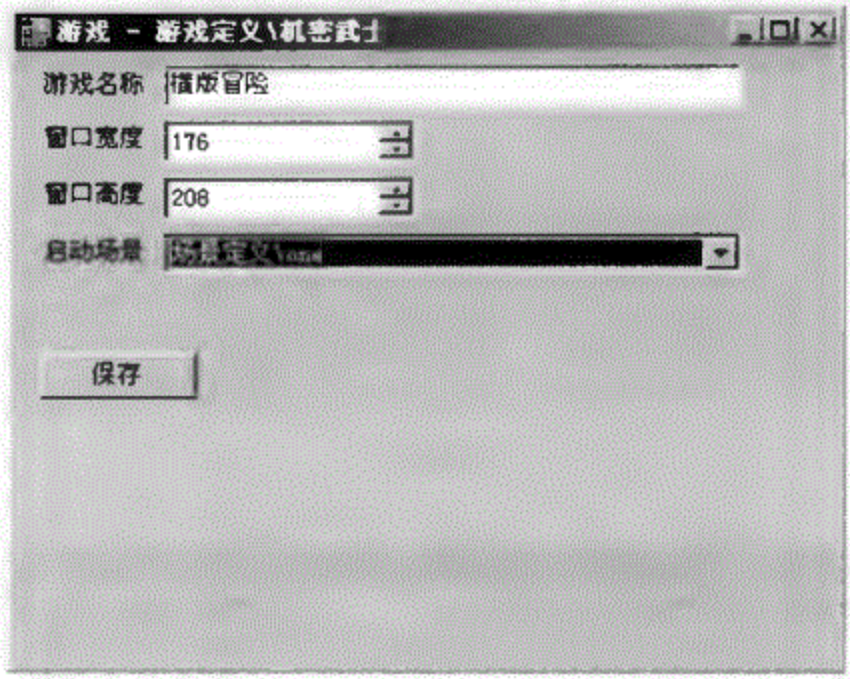


图 5-37

② 单击“工具”→“编译配置管理器”命令，打开“编译配置管理器”对话框，设置“配置名称”和编译项目，如图 5-38 所示，具体方法请参照上一章。

③ 完成“编译配置管理器”的设置后，就要开始进行编译生成了。单击“工具”→“编译”→“横版冒险”命令，打开相应窗口开始对横版冒险进行编译，直到编译完成后关闭窗口，如图 5-39 所示。

④ 这样这款横版冒险类的游戏就制作完成，可以来欣赏制作完成的作品了，如图 5-40 所示。

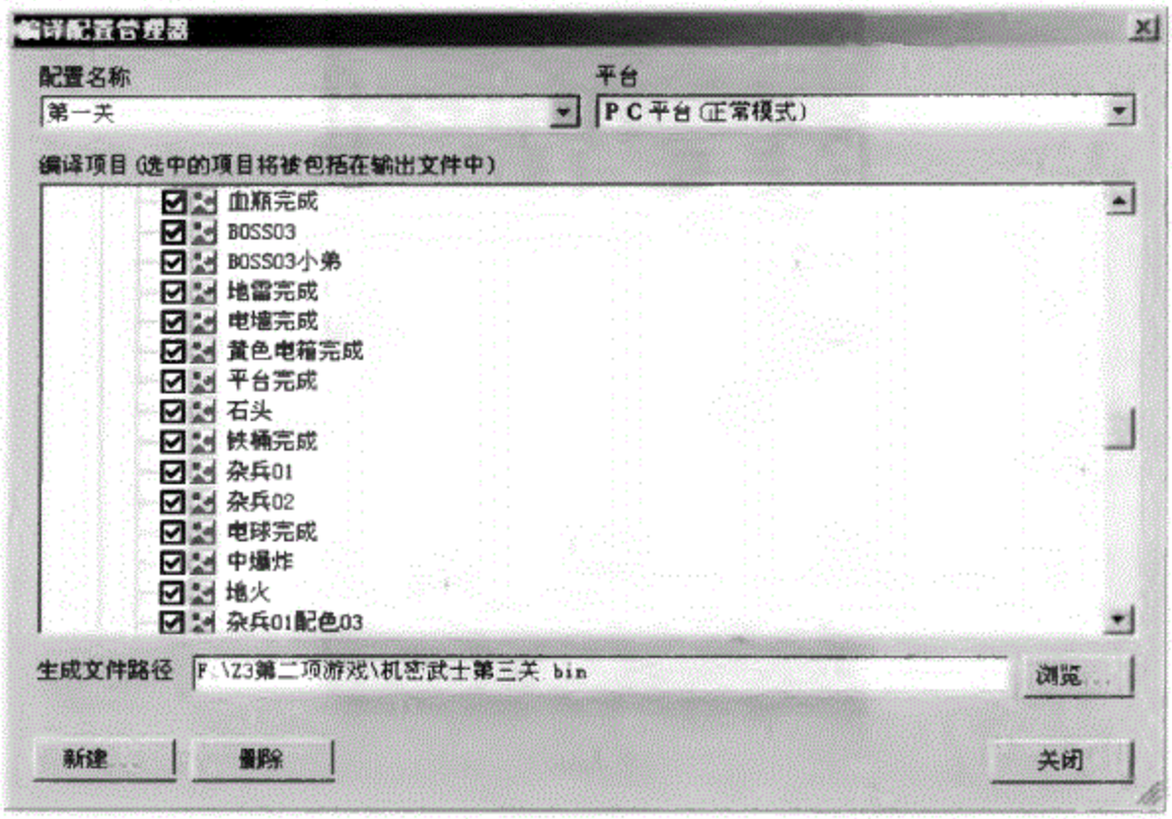

编译配置管理器
配置名称
第一关
平台
PC平台(正常模式)
编译项目(选中的项目将被包括在输出文件中)
血瓶完成
BOSS03
BOSS03小弟
地雷完成
电墙完成
黄色电箱完成
平台完成
石头
铁桶完成
杂兵01
杂兵02
电球完成
中爆炸
地火
杂兵01配色03
生成文件路径
F:\Z3第二项游戏\机密武士第三关\bin
浏览...
新建
删除
关闭

图 5-38


输出窗口
类型 日期 内容
200.. Message 19 - 物件\电箱 编译成功
200.. Message 19 - 物件\蓝电箱 编译成功
200.. Message 19 - 物件\zb1 编译成功
200.. Message 19 - 物件\zb0103 编译成功
200.. Message 19 - 物件\zb2 编译成功
200.. Message 19 - 物件\zb0202 编译成功
200.. Message 19 - 物件\平台小 编译成功
200.. Message 19 - 物件\平台小下 编译成功
200.. Message 19 - 物件\平台大 编译成功
200.. Message 19 - 物件\平台中 编译成功
200.. Message 19 - 物件\平台中下 编译成功
200.. Message 19 - 物件\zb1x 编译成功
200.. Message 19 - 物件\zb0103x 编译成功
200.. Message 19 - 物件\zb02x 编译成功
200.. Message 19 - 物件\zb0202x 编译成功
200.. Message 19 - 物件\BOSS03x 编译成功
200.. Message 19 - 物件\bossx 编译成功
200.. Message 19 - 物件\影子bossx 编译成功
200.. Message 19 - 物件\黄电墙 编译成功
200.. Message 19 - 物件\主角 编译成功
200.. Message 19 - 物件\boss 编译成功
200.. Message 20 - 场景定义\one 编译成功
200.. Message 21 - 游戏定义\机密武士 编译成功
200.. Message 23 - UI\玩家血槽 编译成功
200.. Message 23 - UI\敌人血槽 编译成功
200.. Message 22 - UI\ui 编译成功
200.. Message 24 - 音乐文件\BEYOND-03-漆黑的空间.mp3 编译成功
200.. Message 配置 第一关 编译完成

图 5-39

图 5-40

第 6 章　RPG Maker XP 工具使用

6.1 RPG Maker 简介

本章为大家介绍一款专门制作 RPG 游戏的游戏制作引擎——RPG Maker。

1. 什么是 RPG Maker

RPG Maker 是由日本 Enterbrain 公司开发的一款 RPG 游戏制作软件。它方便易用，是一款使用十分方便的游戏制作工具，并且采用了图形化的工作界面，即使一个业余玩家用它也能制作出一款简单的 RPG 游戏。RPG Maker XP（如图 6-1 所示）强化了画面的表现，另外还加入了 RGSS 脚本系统，也就是说，熟悉编程的玩家可以通过 RGSS 独立定义或更改游戏系统。

图 6-1

2. 系统配置

在安装完 RPG Maker XP 之后，首先简单介绍一下运行该软件所需要的系统配置，RPG Maker XP 系统要求如下：

最低配置，如表 6-1 所示。

表 6-1　最低配置

系统	Windows 98 / Windows 98 SE / Windows ME / Windows 2000 / Windows XP
CPU	Intel Pentium Ⅲ 800 MHz 以上
内存	128 MB 以上
显卡	分辨率 1024 × 768
声卡	兼容 DirectSound 声卡
硬盘	可用空间 100 MB 以上

推荐配置，如表 6-2 所示。

表 6-2　推荐配置

系统	Windows XP 中文版	显卡	分辨率 1024 × 768 以上
CPU	Intel Pentium 4 1.5 GHz 以上	声卡	兼容 DirectSound 声卡
内存	256 MB 以上	硬盘	可用空间 500 MB 以上

6.2 建立工程

在打开软件之后可以看到 RPG Maker XP 的菜单及工具条，如图 6-2 所示。

但是因为目前没有创建工程，所以工具栏的图标还是处于未激活状态。接下来就为读者介绍如何创建工程。

图 6-2

图 6-3

1. 新建工程

在 RPG Maker XP 系统中，是以工程为单位来制作游戏的。建立一个工程，就会在硬盘中新建一个文件夹，而游戏中的各种数据资料就以文件形式保存在其中。

单击“文件”→“新建工程”命令，如图 6-3 所示，在弹出的对话框中依照指示输入文件夹名和标题，并选择一个路径，输入完毕单击“确定”按钮，就建立了一个新工程，如图 6-4 所示。

新建文件夹内的 Game.rxproj 文件就是工程

图 6-4

文件。以后，只要双击这个文件就能直接打开该工程。可以把编辑中的工程在桌面上建立一个快捷方式，这样使用起来就更快捷方便。

图 6-5

2. 更改工程名

若要更改已建立工程的标题（游戏名），可先打开该工程，单击“游戏”→“更改标题”命令，则在弹出的对话框中输入新的标题，单击“确定”按钮就完成了标题的更改，如图 6-5 所示。

想要删除工程和更改文件夹名，可到 Windows 中找到该文件夹，直接进行相应操作即可。复制游戏工程文件夹，就可完成整个游戏的复制。

6.3 素材导入及素材规格

RPG Maker XP 可以使用其本身初始的图片和声音素材文件。

单击“工具”→“媒体库”命令，弹出相应的对话框，也可以按键盘上的快捷键 F11 打开“媒体库”对话框，在“媒体库”对话框中可以进行各种素材的导入与导出，如图 6-6 所示，向游戏文件夹里直接复制文件也可以，媒体库中还有图片预览功能，对素材不熟悉的话可以使用。

接下来就介绍一下媒体库中的图像素材与声音素材。

靠左侧一排的素材文件夹中所列出的是目前 RPG Maker 自带的一些图像与声音素材，其中 Graphic 开头的文件是图像素材，而以 Audio 开头的文件是声音素材。若需要使用自己绘制的图像素材，则需要单击右侧的“导入”按钮，导入自己绘制好的图像素材，但是各种类型的素材必须放在不同的文件夹中，以下就来做详细的介绍。

图 6-6

1. 图像素材

在用 RPG Maker 制作游戏时可以使用 PNG 文件和 JPG 文件，区别在于 PNG 图片带有透明通道，所以如果是需要导入 PNG 文件则必须为 32 位色彩。

（1）角色

角色是指地图上显示的角色图片。

一个角色使用一个文件，尺寸任意，以 4 个方向（下、左、右、上）×4 个步行样式合计 16 个样式的规定顺序排列。位图宽高的各 1/4 作为该角色的尺寸，如图 6-7 所示。

图 6-7

（2）战斗者

战斗者是指战斗画面中显示的角色图片。

图片尺寸任意，但注意最好不要超过 640 × 320 像素，如图 6-8 所示。

（3）动画

动画主要指在战斗画面中显示效果的动画图片，如图 6-9 所示。

由 5 张固定大小 192 × 192 的图片横向排列为一组，如有必要还可以纵向延长，成为一整个动画文件。文件的尺寸大小无限制，但因为显示动画速度的关系，最好不要用

图 6-8

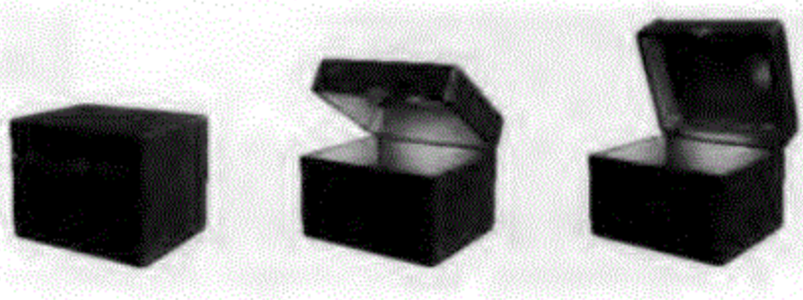


图 6-9

太大的图片。

（4）图块

图块指构成地图的地图元件，如图 6-10 所示。

由 8 张固定大小为 32 × 32 的元件图片横向排列为一组，如有必要还可以纵向延长，即成为一整个图块文件。文件的尺寸大小无限制，但因为显示地图速度的关系，最好不要用太大的图片。

（5）自动地图元件

自动地图元件是指边界自动生成的特殊地图元件，如图 6-11 所示。

图 6-10

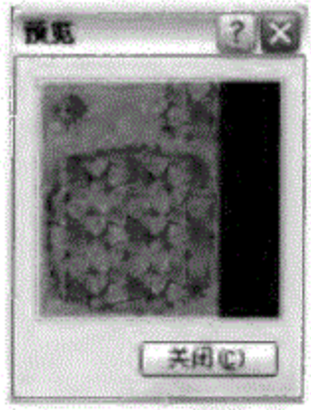


图 6-11

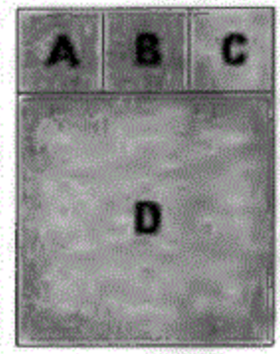


图 6-12

自动地图元件一般有 4 个基本样式，如图 6-12 所示：

- A 处为代表样式，指地图元件面板中显示的样式，在绘制单块的时候显示此效果。
- B 处为边界样式，指相同代表样式的自动地图元件紧邻放置在一起时，于边界侧作为边界线。
- C 处为四角边界样式。
- D 处为大范围边界带无边界中央区样式。

（6）远景

远景指地图远处显示的图片（远景）。

图片尺寸大小没有限制。但要符合网页壁纸的特点，制作成上下左右能连接的样式，如图 6-13 所示。

（7）雾

雾指地图前显示雾遮盖效果的图片。

图片尺寸大小无限制，但要符合网页壁纸的特点，制作成上下左右能连接的样式，如图 6-14 所示。

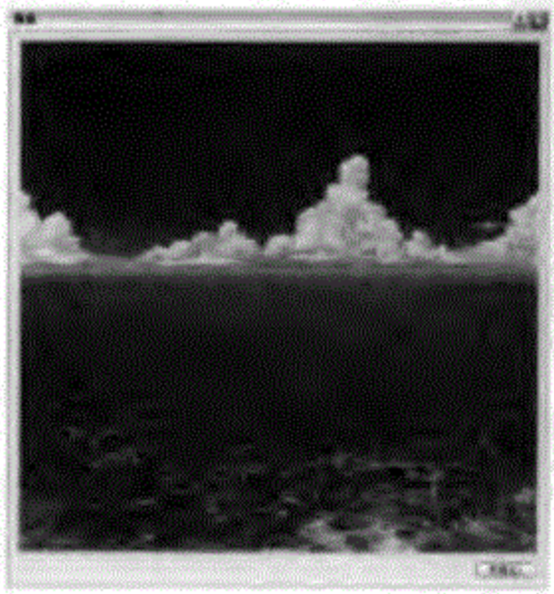


图 6-13

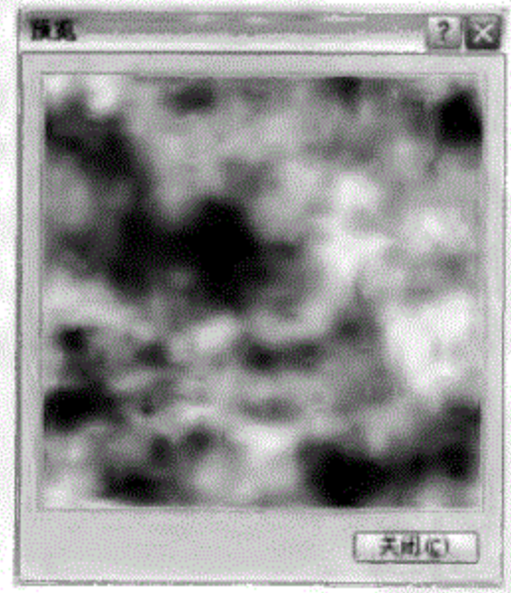


图 6-14

（8）战斗背景

战斗背景指战斗画面背景图片。图片尺寸为 640 × 320，如图 6-15 所示。

图 6-15

（9）图标

图标指技能和物品名称前显示的图标图片。图片尺寸为 24 × 24，如图 6-16 所示。

（10）标题

标题指标题画面图片。图片尺寸为 640 × 480，如图 6-17 所示。

（11）游戏结束

游戏结束指游戏结束画面图片。图片尺寸为 640 × 480，如图 6-18 所示。

（12）窗口皮肤

窗口皮肤指合成窗口画面的图片，如图 6-19 所示。

窗口皮肤是大小为 192 × 128 的图片。通常是使用 32 位元色彩的 PNG 文件，如图 6-20 所示。

图 6-16

A 处为窗口的背景。128 × 128 的样式，会按照实际窗口的大小而扩大缩小。为了严谨，窗口周围缩小了两像素大小。这是为了能自然地看见圆角形窗口，而且在一部分窗口中，背景部分只能以半透明显示。由于窗口为半透明，所以图片本身就没有做成半透明的必要了。

B 处为窗口的边框及箭头。四角的 16 × 16 边框照这样显示，剩下的边框（边框的一部分）在窗口中按 16 像素的设计形状连接。箭头则作为窗口内容滚动的图标使用。

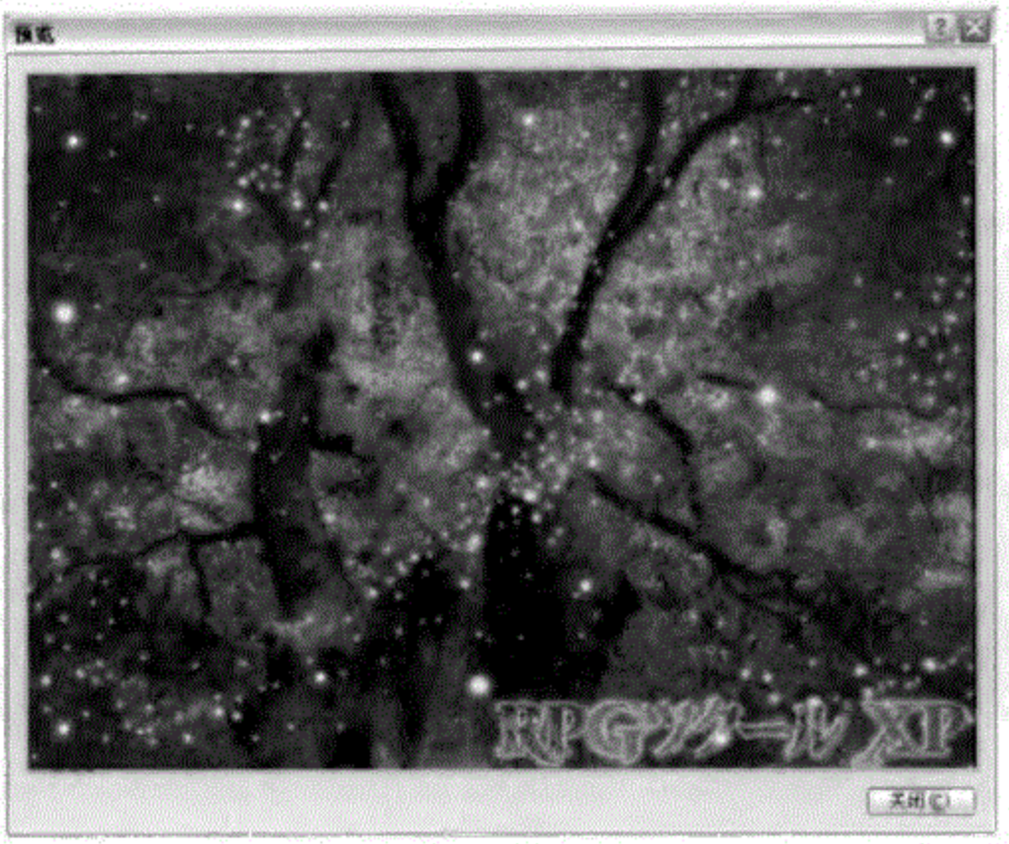


图 6-17

图 6-18

图 6-19

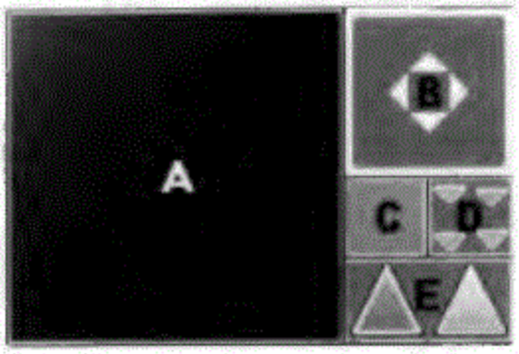


图 6-20

C 处为指令光标。在窗口内表现选择项目时使用。周围的两像素按长宽扩大缩小，剩下的按光标的大小平面显示。

D 处为暂停标记。在消息窗口中表示等待按钮输入的状态下使用。为 16×16 含 4 种图形的动画。

E 处为箭头光标。在战斗画面时选择角色和敌人时使用。显示为 32×32 的两种样式的交替。这不是窗口的一部分，但放在这个文件夹中会很方便。

（13）图片

图片指游戏中事件所使用的图片，图片尺寸任意。

（14）切换效果

切换效果指游戏运行时指定画面切换效果的图片。图片尺寸必须是 640×480，灰色阶 256 色的 PNG 文件。按号码从小到大的顺序进行画面的切换，如图 6-21 所示。

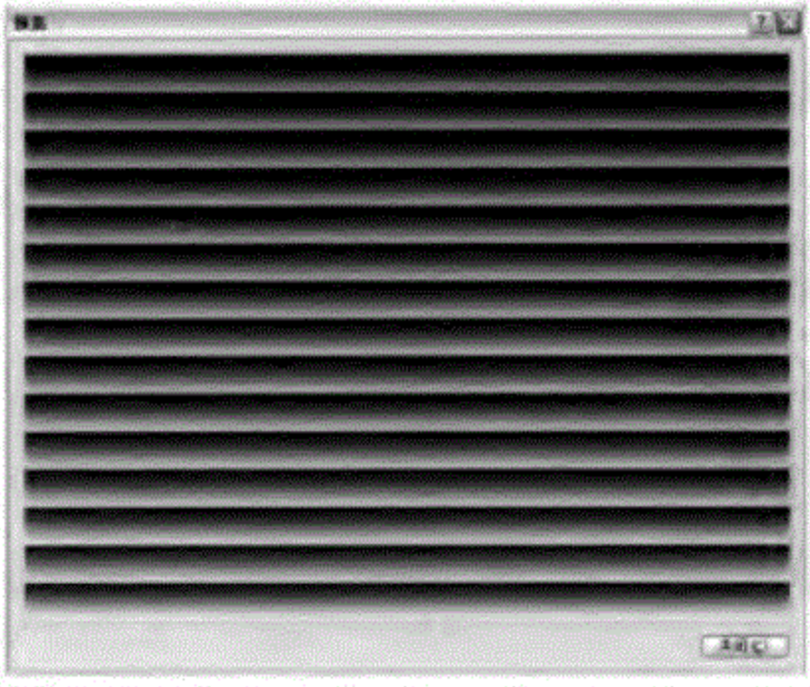


图 6-21

2. 声音素材

（1）格式类型

可以使用 MID、OGG、WMA、MP3、WAV 这 5 种类型的音频文件。简单介绍一下各个类型声音文件的特点。

MID 格式是以 DirectMusic 演奏的 MIDI 文件，文件大小是所有声音类文件最小的。背景音乐播放中，MIDI 数据中有控制重复播放的标记，当乐曲播放到最后会自动重播。

OGG 是含有音质和压缩率均优良的音频压缩格式 OGG Vorbis 数据的文件。但是，RPG Maker XP 中不对应流再生，OGG 文件要全部读取后才开始播放，不适合播放时间较长的乐曲。

WMA 是 Windows Media Player 使用的音频压缩格式，以 DirectShow 播放，和 OGG 不同，对应流再生（一边读取数据一边再生流），所以能播放长时间的乐曲。

MP3 读者都比较熟悉，它是一种普及率高的音频压缩格式，可以 DirectShow 播放，特点和 WMA 类似。

WAV 是 Windows 标准的音频格式。通常的 WAV 格式无太大的压缩比率，甚至有无压缩的格式，所以说 WAV 文件所占用的磁盘是比较大的，音质也较好。

（2）RPG Maker XP 中的声音素材内容

- BGM（Audio/BGM）。背景音乐（BackGround Music），主要使用 MID 文件。
- BGS（Audio/BGS）。背景声音（BackGround Sound），主要使用 OGG 文件。
- ME（Audio/ME）。效果音乐（Music Effect），主要使用 MID 文件。
- SE（Audio/SE）。效果声音（Sound Effect），主要使用 OGG 文件。

6.4 地图功能

在游戏中主角移动的舞台称为地图。游戏是由很多地图构成的，各个地图间可以随着事件的处理来回移动。例如主角在进入大门后，原地图就会消失，而转移到室内的另一个地图里，如图 6-22 所示，因此通过这样的衔接方式便能将整个游戏联系在一起。

地图看起来就像一整张图画，其实在前两章介绍 2D 游戏制作引擎的时候就已经讲到制作地图的方法和拼接地图的技巧，地图实际上和拼图玩具的小块一样是由许多小部件组合而成的。构成地图的最小部件称为地图元件。可以对地图元件设定各种不同的属性，比如人物能否通行等。

表现特定的场所就需要一组整理在一起的许多地图元件，并对不同元件设定不同的属性，这就称为图块。各个地图可以选择使用不同的图块，然后用该图块中的地图元件组合配置成一张地图，如图 6-23 所示。在图块中，还可以设定战斗时显示的背景（战斗背景）图形，遭遇敌人时会自动使用该背景。

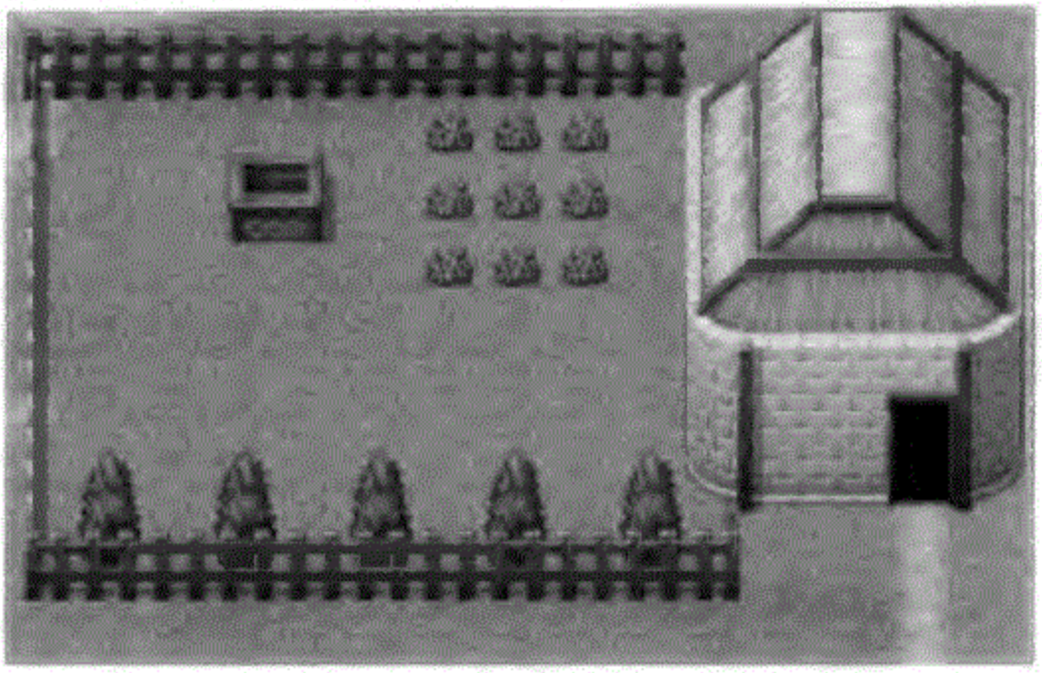


图 6-22

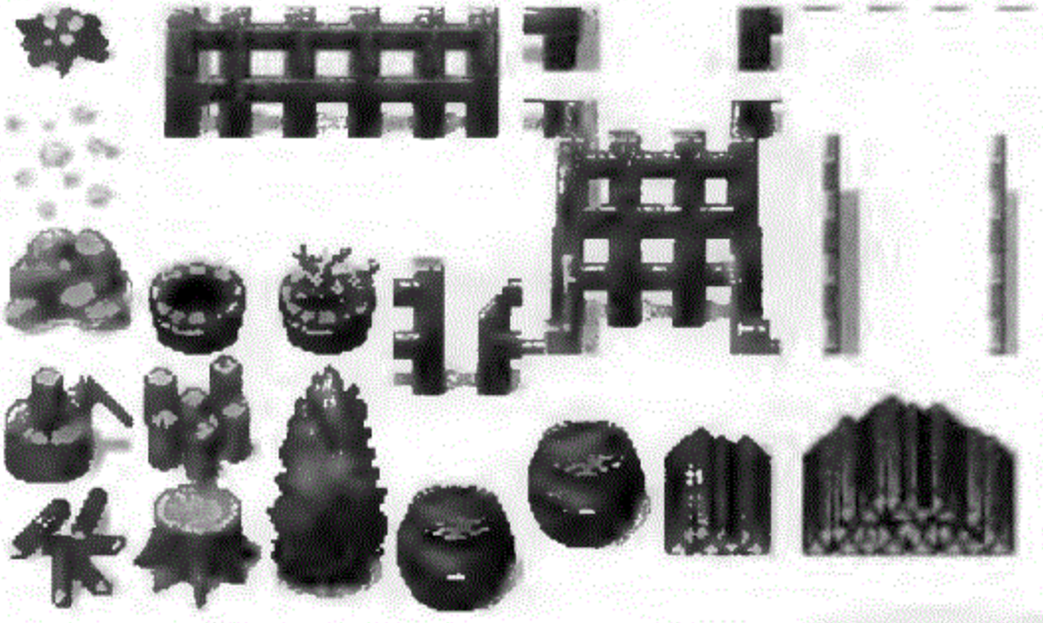


图 6-23

1. 制作地图

新地图的制作和多个地图的管理，是在主窗口左下方的地图树状目录里进行的，如图 6-24 所示。

首先，在地图树状目录里选择一个地图名称，按下鼠标右键，在弹出的快捷菜单中选择“新建地图”命令，如图 6-25 所示，便会弹出“制作地图 -ID：002”对话框，设定好地图名称、背景、地图大小、遇敌率等必要信息后单击“确定”按钮，就完成了新地图的制作，如图 6-26 所示。

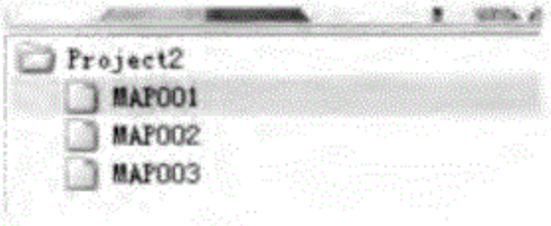


图 6-24

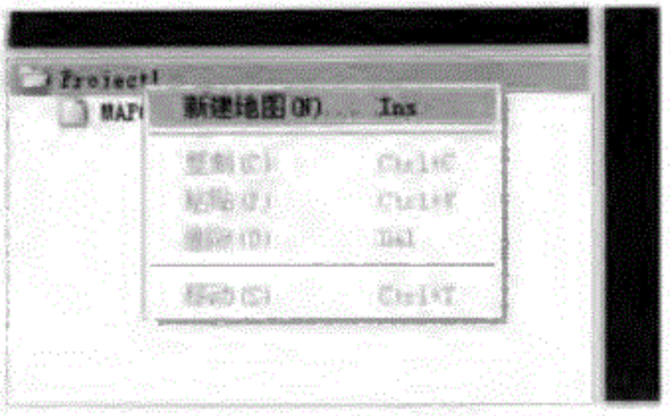


图 6-25

图 6-26

图 6-27

在这里简单说明制作地图对话框中的几个参数，“名称”为显示在地图目录中的地图名称；“背景”是图像素材中制作者准备好的地图图块资源；“宽”和“高”是以小图块的大小为计量单位的；选中“自动切换 BGM”复选框，可以选择进入该地图中自动播放的背景音乐；选中“自动切换 BGS”复选框，可以选择进入该地图中自动播放的背景音效，比如流水声，或是下雨声等，如图 6-27 所示。

在制作地图对话框中比较重要的参数便是“遇敌率”选项组，它直接影响到玩家在游戏中的进程，在“队伍”上单击鼠标右键，在弹出的菜单中选择“编辑”命令，可以打开“遇敌率”对话框，选择地图中随机出现的敌兵队伍，

如图 6-28 所示。

地图设定完毕，最初选择的地图下方就会建立一个新的子地图，而上方的地图就叫母地图，如图 6-29 所示，这和 Windows 文件夹的目录结构相同。这个结构并不直接影响游戏的运行，比如城市的建筑物内部地图做成该城市整体地图的子地图，这样对多个地图的管理就会更加容易。地图可以通过鼠标直接拖动来改变目录结构。

图 6-28

图 6-29

2. 绘制地图

在 RPG Maker XP 中绘制地图就必须要讲下层的概念，地图是由 3 层各种不同的地图元件配置而成的（地图元件即地图图片中的图块），可以向各层任意绘制各种地图元件。一般来说第 1 层是地面，第 2 层是栅栏和建筑物或是山坡，第 3 层是窗户和烟囱，按照这个标准，地图就能比较合理地制作完成。

在绘制地图时对层的选择，可以在工具栏上选择 ，对应的快捷键为 F5、F6 和 F7，在主菜单“模式”中也可以对这三个层进行选择，如图 6-30 所示。

在“视图”菜单中还可以进行“暗淡显示其它层”、“当前及下一层”或是“全部层”等设定，如图 6-31 所示。

编辑地图时需要用到的图块，必须从主窗口左边的地图元件面板中选择。在地图元件面板中选择一个地图元件（图块），也可以同时选择多个地图元件进行绘制。选择了地图元件，就可以用绘图工具在右侧的绘图区域中自由绘制了，如图 6-32 所示。

图 6-30

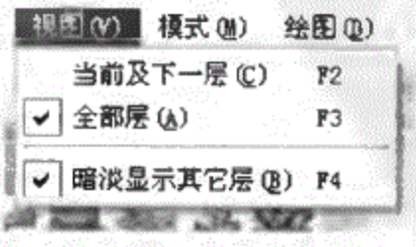


图 6-31

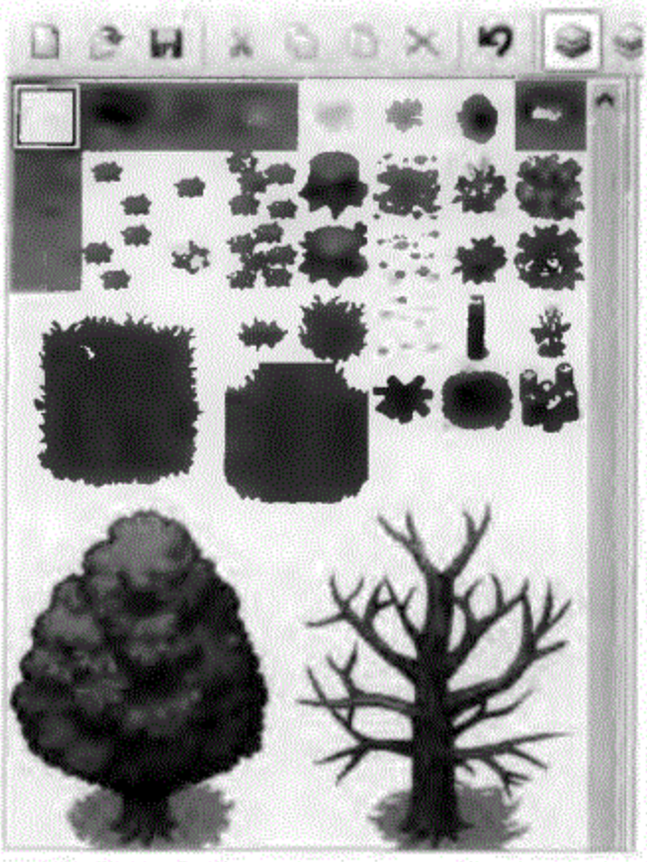


图 6-32

在绘制地图时所使用的绘图工具，在工具栏中可以找到相应的图标，也可以在主菜单“绘图”中选择。接下来为大家介绍各个绘图工具的具体功能。

- 铅笔：是最常用的绘图工具。在绘图区域上单击和拖动可将目前选取的地图元件配置到地图上。
- 四边形：拖动鼠标会形成一个四边形，将目前选取的地图元件填满该四边形区域。
- 椭圆：由拖动对角线形成一个四边形缩起的圆形，将目前选取的地图元件填满该区域。
- 填充：在单击处的上下左右连续相同的地图元件全部置换为目前选取的地图元件。
- 选择：为地图的剪切和复制等选取一个编辑范围。这种情况下单击鼠标右键会显示一个弹出菜单，就可以进行该范围的剪切和复制等操作。另外，选取范围的复制等操作包括全部 3 层的所有对象。

在绘制地图时，选中任何绘图工具（除了“选择”工具）时，在绘图区域某处单击鼠标右键，地图元件面板中白线框选的地图元件就会自动切换为该处所配置的地图元件。这个功能类似于 Photoshop 中的吸管工具。使用吸管功能会省略从地图元件面板中选择地图元件的步骤，从而大大提升作业效率。

3. 自动地图元件

在 RPG Maker XP 中，地图元件面板中默认最上面的一排地图元件为自动地图元件，它是一种特殊的地图元件，如图 6-33 所示。

在绘制地图时，使用自动地图原件可以节约不少时间，它在实际排列中会依照边界自动调节，如图 6-34 和图 6-35 所示。

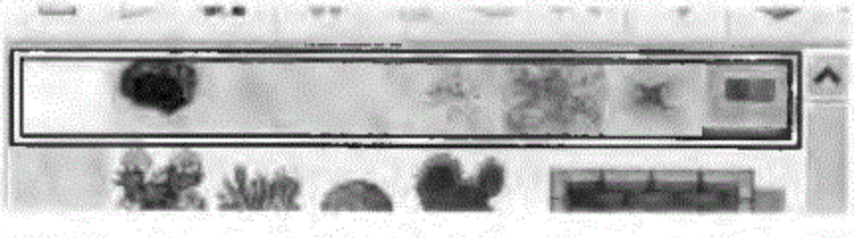


图 6-33

图 6-34

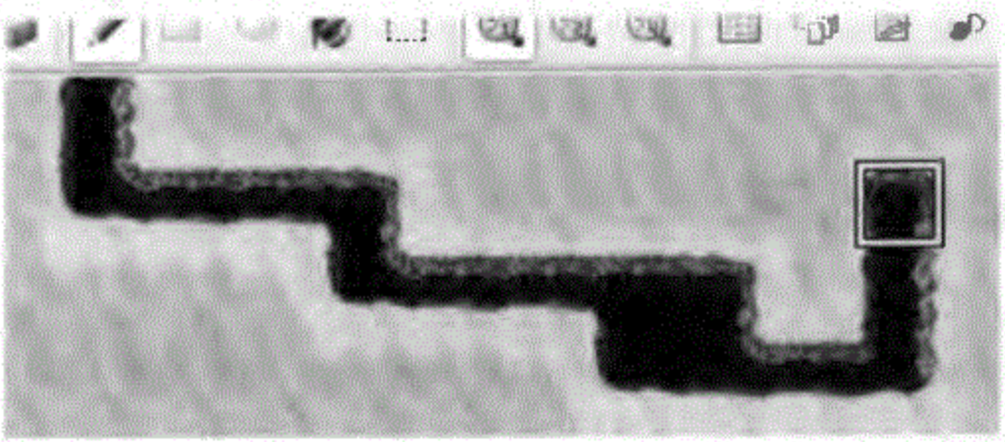


图 6-35

在绘制地图时可以选择地图元件面板中的自动地图图块，然后对其双击，就会弹出“自动展开元件”对话框，用户可以直接选择边界调整的式样，如图 6-36 所示。

图 6-36

如果不需要开启自动地图元件的功能，在绘制地图时按住 Shift 键绘制地图时，自动地图元件的功能就会失效。

6.5 事件处理

在解释事件之前，首先需要了解的是游戏是以何种方式驱动的，怎样才能将自己绘制的美术资源应用到游戏中，使其产生与玩家之间的互动。这些问题程序员就可以解决，但是对于一个擅长美术且没有学过编程的人来讲，要自己动手制作一款游戏似乎不可能，不过 RPG Maker 提供了这个方法，它将程序制作人员编写好的程序打包做成事件，以图形化以及窗口的形式让所有不会编程的玩家也能实现游戏的互动。

比如游戏中在角色走到地图中一定的位置，地图便进行切换，在拿到钥匙之后开启大门，这些都是通过事件所实现的，如图 6-37 所示。接下来为大家介绍事件的具体种类。

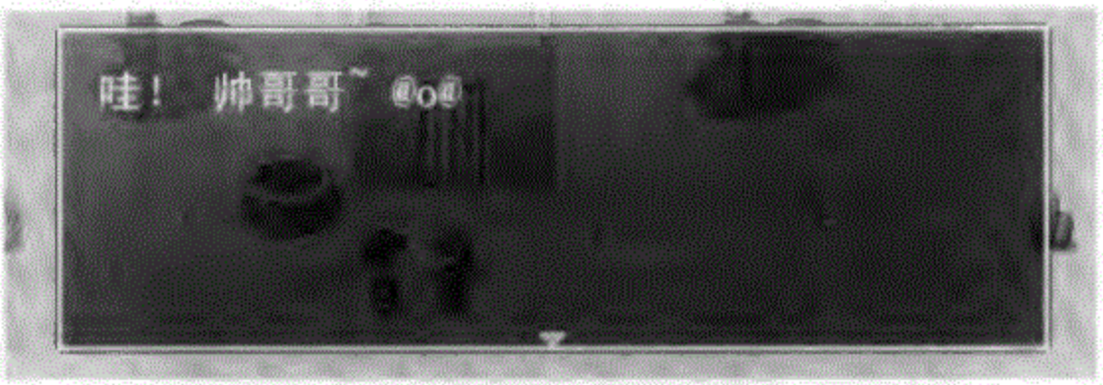


图 6-37

事件包含了“地图事件”、“战斗事件”以及“公共事件”。

1. 地图事件

地图事件就是在地图画面上添加并且运行的事件，包括场所的移动，人物的对话等。在添加地图事件之前，首先要选择事件模式，可以在工具栏上单击“事件”按钮 ，也可以在主菜单“模式”菜单中选择“事件”命令，如图 6-38 所示。

事件模式打开之后就可以在地图编辑区域上双击或是在地图上单击鼠标右键选择“新建事件”命令，新建一个地图事件，如图 6-39 所示。

图 6-38

图 6-39

在地图事件中，一个事件能设立多个不同内容的处理。这种设定称为事件页，如图6-40所示。

图6-40

各事件页内容的出现条件有两个开关、一个变量和一个独立开关。满足指定条件的事件页有多个的话，会执行号码最大的事件页的内容。

在游戏过程中，开关有ON和OFF两个状态。系统默认的开关状态为OFF，它用来判断事件是否能够被触发，例如在游戏中，首先要获得钥匙，才能够打开下一道门，那么在打开门这个事件中首先就应该用一个“获得钥匙”的开关来作为判断条件，一旦获得了钥匙，“获得钥匙”开关便要切换成ON，这样，在“切换场景”事件中，如果“获得钥匙”开关为ON，那么便可以执行开门这个事件，如图6-41所示。

变量与开关一样可以作为事件开始的前提条件，但是它的功能更强大，在整个游戏中，变量都是可以随时修改的数据（最大为99 999 999），例如它可以记录已经完成的任务数量信息，当任务完成一件，便可以让已完成任务数加一，若完成5个任务，便能够进入下一个场景，如图6-42所示。

图 6-41

图 6-42

独立开关是引发个体事件的特殊开关。不会给其他事件造成影响，仅为该事件使用，例如管理“打开了宝箱“的信息。

在编辑事件窗口的下方是“事件开始条件”，如图 6-43 所示，在这里总共有 5 个单选按钮。

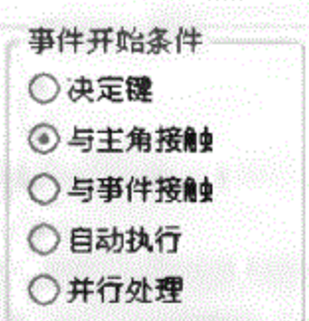


图 6-43

“决定键”是指在主角与事件接触的状态下，按下决定键（C 键）时事件开始执行。记住必须要主角在事件接触的情况下，例如与村民之间发生对话或是调查宝箱等基本事件会使用这种方式。

“与主角接触”相对于“决定键”来讲要方便很多，当主角以方向按钮向事件接触时事件便能够直接开始执行。如果事件没有添加图片，主角与事件能够重叠的话，那么重叠时开始执行事件，例如场所移动事件等。

“与事件接触”其实类似“与主角接触”，有些事件可以设置让它自己来移动，当事件移动到与主角接触时也能开始执行事件。例如怪物在地图上来回移动，接触到主角时触发战斗的事件就会使用这种方式。

选择“自动执行”单选按钮，则在游戏运行时事件会立即自动执行。如果事件的出现条件满足的情况下，事件会反复执行，导致游戏死机，所以在使用这种事件需要适当转换开关和独立开关，必须控制事件的出现条件，比如移动到特定地图发生强制事件时

会使用这种方式。

并行处理则是在事件出现期间，周期性地执行事件内容。并行处理的事件与通常的性质有所不同，在其他事件执行的同时并行执行该事件，使用时必须格外注意，因为每隔一小段时间它便会执行一次。

在编辑窗口的右方是“执行内容”，如图 6-44 所示。

双击列表中的符号◆，或是在◆处单击鼠标右键，在弹出的菜单中选择“插入”命令，如图 6-45 所示。

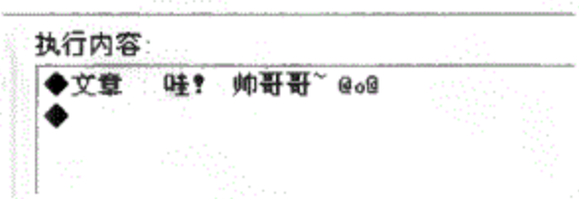


图 6-44

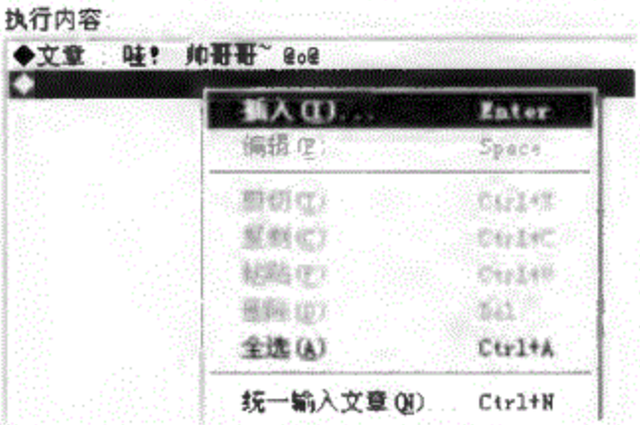


图 6-45

就能够打开“事件指令”对话框，如图 6-46 所示，在这里单击标有该事件指令名称的按钮，制作者就可以选择所要的事件指令。

打开“事件指令”对话框之后会发觉“事件指令”对话框中共有 3 个选项卡，每个选项卡上面的指令都有所不同。

第一个选项卡上比较多的是一些游戏中的逻辑操作，例如开关操作、变量操作、条件分歧、循环、中断事件处理等。

第二个选项卡上更多是一些画面与声音的处理事件，例如画面操作有画面卷动、显示动画、更改透明状态等，声音操作有演奏背景音乐、演奏背景声音等，如图 6-47 所示。

第三个选项卡上基本都是一些游戏进行中或是战斗中的一些文字处理，例如增减 HP、增减 SP、增减 EXP 等，如图 6-48 所示。

图 6-46

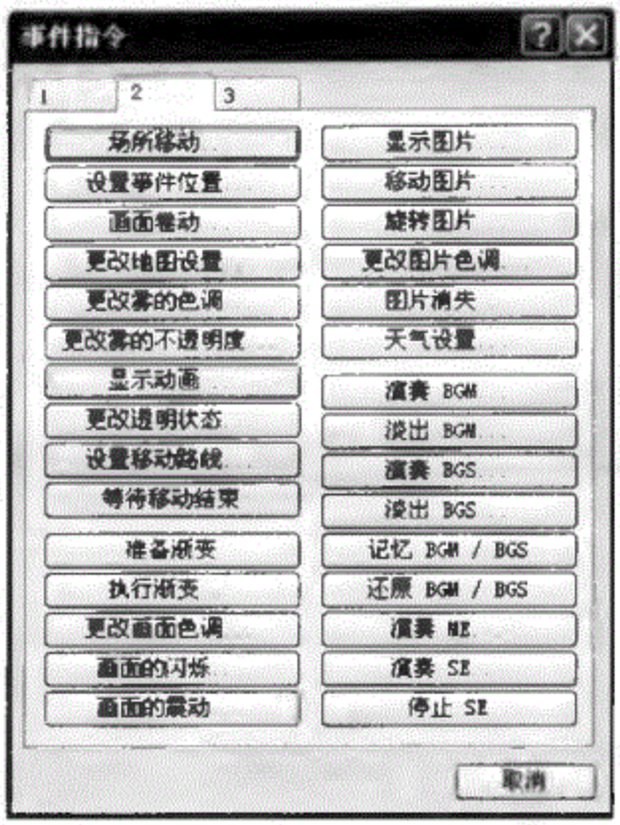


图 6-47

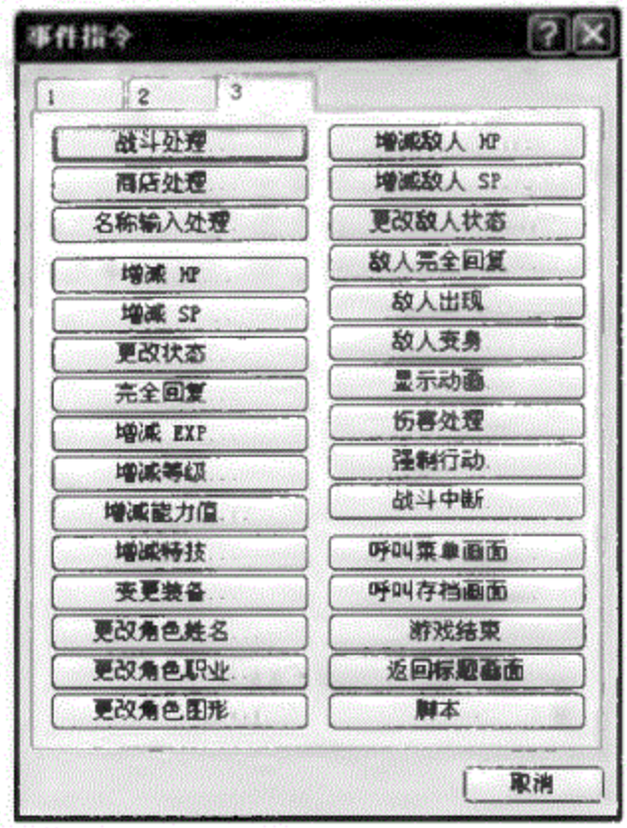


图 6-48

事件指令的具体用法会在下一章的 RPG Maker 实例制作中做详细的介绍。

完成必要的设定后单击“确定”按钮，在◆符号的位置就会插入一个事件指令，然后便可以在后面插入新的指令。一个事件页中可以有多条事件指令，这样整个游戏才更有逻辑性。

在事件指令列表中，选择一个事件指令单击鼠标右键会弹出快捷菜单，应用此菜单能进行复制粘贴等操作，用此方法可以来更改事件的叠放次序以及复制多个事件的操作。

实际中事件的执行内容是由登录于事件执行列表中的事件指令来定义的，执行时从上到下依次执行。事件指令有“显示文章”、“场所移动”、“战斗处理”等各种不同的指令，变化丰富、多种多样共计有 90 种之多。

执行内容的编辑方法种类事件都是共同的。

2. 战斗事件

战斗事件就是在战斗画面中运行的事件。比如“敌人在 HP 值达到规定值以下时变身”的处理就是使用的战斗事件，可在数据库“队伍”中设定。公共事件可以从地图事件和战斗事件中调用，为特殊情况的处理，是通用的事件，可在数据库“公共事件”中设定。

战斗事件也有事件页。经常按号码顺序检查事件页，有满足条件的事件页就立即执行。战斗事件的出现条件为回合数、敌人的 HP、角色的 HP 和开关这 4 种，也可指定多个。

6.6 数据库应用

游戏中使用的各种数据集合在一起称为数据库。编辑数据库的内容，可在“工具”菜单中选择“数据库”命令，在弹出的对话框的分页中进行切换，就可以编辑各种不同种类的数据。

1. 详细说明

各种数据的说明如表 6-3 所示。

表 6-3 数据说明

角色	在游戏中被玩家操作的人物角色的数据
职业	装备的武器、防具和能够学习的特技等，决定角色特征的数据
特技	主要是消耗 SP 发动特别的效果，是技能和魔法等的数据
物品	回复剂和事件物品等，除装备以外的物品的数据
武器	具有攻击力等属性的武器的数据，作为物品的一种来使用
防具	具有物理防御和魔法防御等属性的防具的数据，作为物品的一种来使用
敌人	触发战斗和事件战斗与主角敌对的敌人角色的数据
队伍	敌人的组合是游戏中实际出现的形态
状态	对角色的能力和行动造成各种各样的影响，是健康状况和状态的数据
动画	武器和特技的视觉效果等使用的动画的数据
图块	设定绘制地图时所用图块的地图元件的数据
公共事件	在整个游戏中通用的各种可调用事件的数据
系统	初期同伴、属性、SE 和用语等各种各样的数据

2. 编辑数据项目

数据库各分页的画面都是左右分开的，左侧为数据项目的 ID（管理号）和名称目录，右侧显示为该数据项目的内容。编辑数据项目时，单击左侧进行目录选择，再编辑右侧显示的各设定项目。各设定项目的具体含义请参照帮助提示。

在数据库窗口中按 F4、F5 键，就能在各个数据项目中上下选择。在有很多数据要设定特定项目的情况下，使用该操作能大大提高工作效率。

在数据项目目录中单击右键会弹出快捷菜单，可以通过此菜单进行复制和粘贴等操作。在这里选择“批量复制”，会出现指定批量复制项目数的对话框，在此对话框中设定要复制的项目数，这将会在从其他工程中转移数据的情况下使用。

3. 更改最大值

要更改数据项目的数量，单击目录下面的“更改最大值”按钮，会显示更改最大值

对话框，设定一个从 1~999 的数值。

增加最大值会增加游戏运行时必需的内存容量，数据的读写速度也会变慢，最好不要无意义地增加最大值。

如果减少了最大值，那么超过这个数量的项目会被全部删除。

关于数据库的内容此处就简单做了一下说明，之后的 RPG 实例制作会对数据库做更详细的剖析。

6.7 脚本简介

1. 什么是脚本

脚本也称为程序，掌管游戏运行的简单程序语言称为脚本。事件指令其实不是程序，只有脚本系统解读后才能运行。

脚本的编辑，是针对游戏系统的高级用户提供的功能，难度很高。如果只是制作普通游戏，就没有必要了解。最初的脚本系统完全可以制作不错的游戏，只有在制作者对默认脚本感到不满意的时候才进行编辑。

2. 脚本编辑器

在“工具”菜单中选择“脚本编辑器”命令，会弹出“脚本编辑器”对话框。

运行像 RPG 这类大规模的游戏需要非常多的脚本程序，所以把全部脚本程序分为多个适当的单位来管理是很重要的，RPG Maker XP 中把这个单位称为组，“脚本编辑器”对话框左侧显示的就是组列表。

脚本编辑器和数据库有相似的设计以便于操作。同数据库一样，按 F4、F5 键就能在各个组中上下选择，这里还增加了一个 F6 键，可以把光标当前位置的单词复制为该脚本组的名称。

3. 脚本的使用方法

脚本除了能在脚本编辑器中直接编辑以外还有以下 3 种使用方法：

- 在事件指令“脚本”中使用。
- 在事件指令“条件分歧”的条件中使用。
- 作为“移动路线”内的指令使用。

比如，调用加入了独立脚本的事件指令的情况就能使用到脚本。这样或许还能设计出各种各样有趣的使用方法。

第 7 章　RPG Maker XP 实例制作

通过前面的章节大家已经能够初步了解 RPG Maker XP 的功能和使用方法，接下去就要使用学到的这些内容制作一个短小精悍的 RPG 实例。在开始制作之前，先来了解一下使用 RPG Maker XP 制作一个 RPG 的基本流程。

7.1 概要及制作流程

1. 制作概要

首先要了解“游戏引擎”这个概念，它直接控制着剧情、关卡、美工、音乐、操作等内容，同时它扮演着中场发动机的角色，把游戏中的所有元素捆绑在一起，在后台指挥它们同时、有序地工作。RPG Maker 系列就相当于是一个界面友好的游戏引擎，预设了很多常用事件模式，即使完全不懂程序的制作者也能制作出 RPG。

摆脱了编程工作的束缚，使用者在使用 RPG Maker 的过程中主要扮演的是编剧、美工和关卡设计师的角色。具体到 RPG 游戏中，主要分为地图状态和战斗状态，这两种状态下有各自的角色、敌人、背景、事件和演出效果，战斗时还有各种数据、技能和道具（另外还有菜单、音乐、音效等，不过对大多数人来说自己制作音乐比较困难），这些都是需要进行制作和配置的。虽然 RPG Maker 为不擅长美术的使用者提供了丰富的素材库，另外对各种数据、技能和道具也进行了预设，但是想要真正地创作 RPG，还需要自己动手。明确了需要完成的工作后就可以拟定出制作计划了。

2. 简要制作流程

（1）队友的募集

因为工作量巨大，需要消耗大量的精力和时间。如果不是制作小品级别的 RPG 的话，一个和谐的团队是必不可少的。当然，下面要进行制作的实例即使一个人也能轻松完成。

（2）游戏策划

一个好的剧本是必不可少的。相信希望自制 RPG 的朋友们有相当一部分早已在心中拟定了一个丰富多彩的冒险世界，并苦于没有机会将其搬上游戏舞台。现在 RPG Maker 就可以帮你实现这个梦想，其实并不一定需要长篇巨著，短小精悍也未尝不可。PSP 游戏《勇者 30》就是一个号称在 30 秒内拯救世界的 RPG 游戏。

当然除了剧本外，这个阶段最重要的就是拟定制作计划。从下一个阶段开始就是实际制作了，制作者若不想到时候手忙脚乱的话请认真拟定计划。如果无从着手的话可以

先看下面的内容进行参考。

（3）建立工程

在“文件”菜单中选择“新建工程”命令，可建立一个全新的游戏工程。从这一步开始就宣告了RPG制作计划的正式启动，以后所编辑的游戏数据都会储存在该工程文件夹中，管理好这个工程并不断进行扩充和完善就是RPG Maker的基本功能。

（4）媒体库的管理

在工程文件夹中有游戏所需要使用的图片等素材。原创的素材需要通过“导入”到工程中才能在游戏中使用。全部使用原创素材的话，扩充素材库也将是最耗费时间精力的部分。

（5）地图的制作

RPG Maker的主窗口就是地图的编辑器，可见其重要程度。编辑的方式就是“拼地图”，这个过程的工作类似于关卡设计师的工作。地图效果的好坏很大一部分是取决于地图素材的制作。

（6）数据库的编辑

这里面包含了所有角色、敌人、物件、技能、地图等元素的编辑工作。除了脚本外，在RPG Maker内进行的工作大多是在数据库内完成的。

（7）事件的制作

事件可分为“地图事件”、“战斗事件”和“公共事件”3种，地图事件是在主窗口中的地图编辑区中直接进行编辑的，公共事件与战斗事件则是在数据库中设定的。游戏世界通过事件与玩家进行互动，小到对话、大到剧情演出都通过事件来完成。

（8）脚本编辑

如果不满足于游戏提供的现成系统，就要通过脚本来进行修改。制作者如果精通脚本，不但能在RPG Maker中创造出想要的游戏系统，甚至连其他类型的游戏都能通过RPG Maker来制作。不过脚本需要进行漫长而系统地进行学习才能驾驭，修改脚本丰富游戏系统的过程也会使游戏出现各种BUG，严重的会使游戏无法运行，需要“除虫”。因为使用程序预置的脚本就能够制作出不错的游戏了，具体脚本的使用方法会在下章中为读者做介绍。

（9）测试

这里将测试写在流程的最后，但实际上测试是贯穿于整个制作过程的。测试的过程相当于对已经制作完成的半成品进行“试玩”。

3. 剧情构成

简单介绍一下该游戏基本构思，游戏的名称为Rescue，描述了在公元776年，由于战乱，社会变得混沌、动荡。主角若零来到沦陷的洛阳城，开始了她成为救世主的征程……

该游戏主要有4类角色组成，分别是：主角若零、在战斗后会加入我方的敌军士兵、普通敌兵以及最终BOSS——魔熊。

7.2 素材准备

1. 前期准备工作

这里的前期准备不是指企划工作而是指把各种素材导入 RPG Maker 前的准备工作，也就是素材的制作和编辑，如图 7-1 所示。

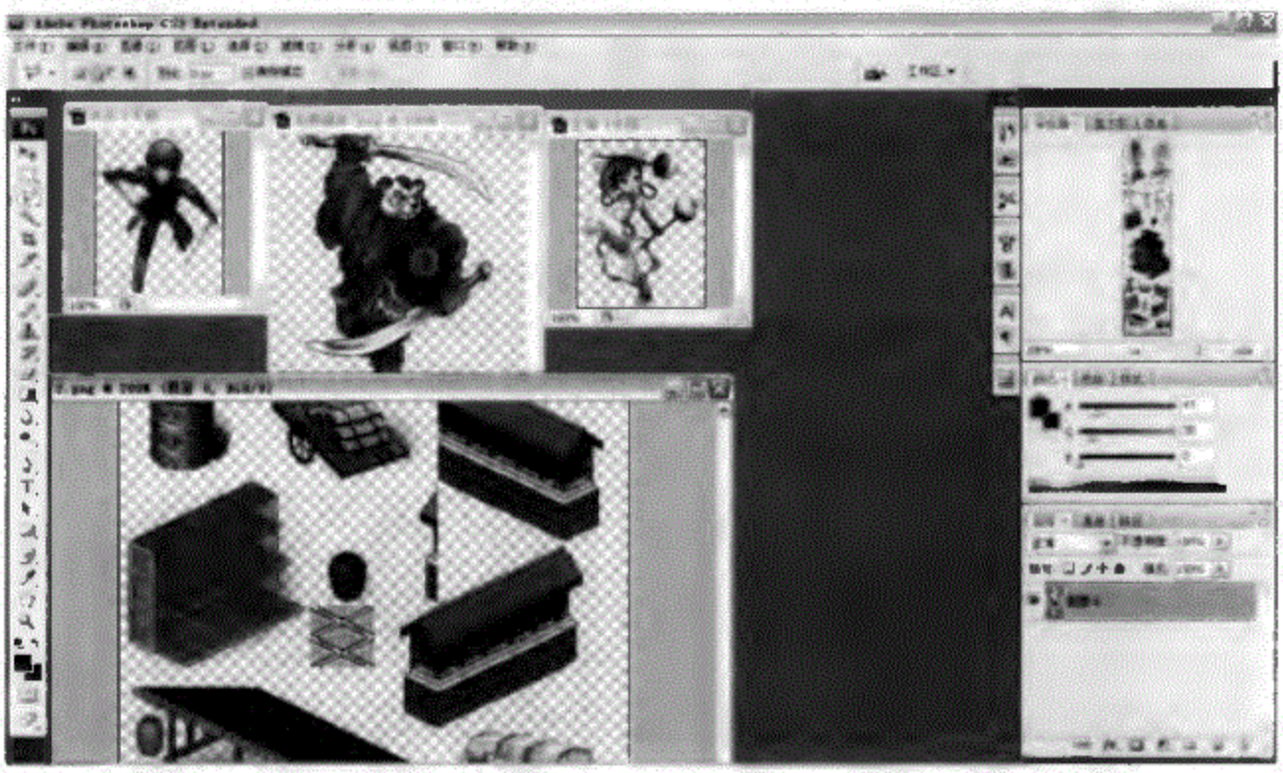


图 7-1

素材包括图片、声音、动画等，其中最主要是图片部分。需要说明的是，虽然 RPG Maker 支持俯视角游戏，但是如果把地图单元制作成 45°角的话，同样也可以运行。45°角游戏也被称为 2.5D 游戏或者是假 3D 游戏，也是 2D 游戏中比较流行的一种游戏形式。

导入素材前要按 RPG Maker 要求把它们进行切割和排列。当然，如果绘制的时候就有意识地按照格式来进行将更省力。各个版本的 RPG Maker 要求不同，一般越后面出的版本支持的图片分辨率也越大，同时不同版本的素材种类也会有一定增减，比如为了减轻使用者的工作量，从 RPG Maker XP 开始，预设脚本中取消了战斗时我方角色的战斗动画（不过依然有许多发烧友通过修改脚本还原了带有动态效果的战斗方式）。

下面列出的基本的图片素材格式都是以 RPG Maker XP 为基准的。该内容为之前基础教学中格式要求的补充说明。声音素材的格式在之前的基础教学中有详细说明。

（1）角色行走图

图像格式为 PNG 或 JPG。其中 PNG 文件则必须为 32 位元色彩（Alpha Channel）。

每个角色使用一张图片。角色的尺寸没有限制，图片宽高的各 1/4，即素材图片大小的 1/16 就是角色的大小，如图 7-2 所示。

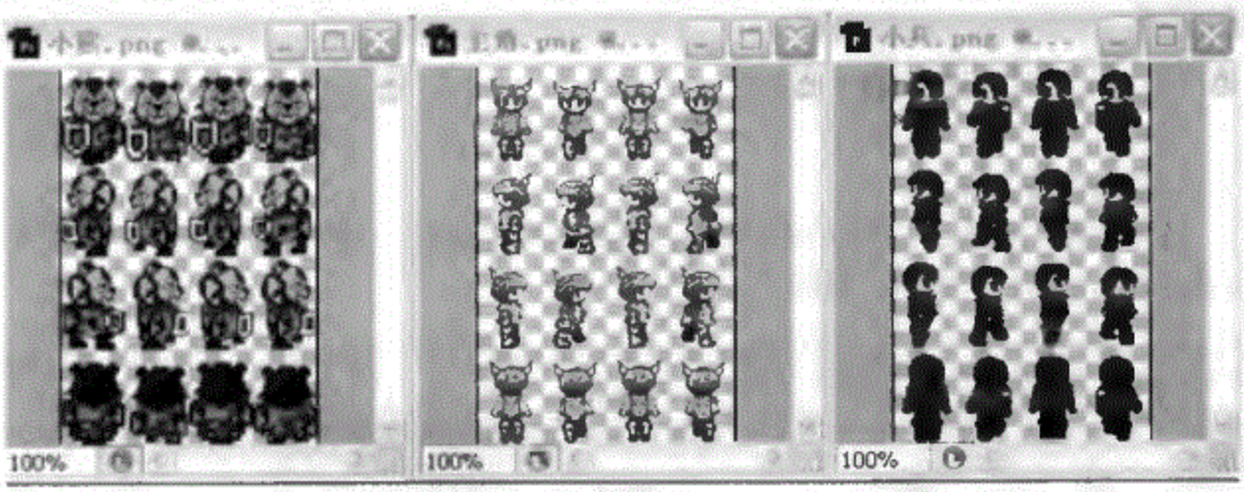


图 7-2

将图片平均分为 4 行 4 列共 16 格。1~4 行分别为角色向下、左、右、上 4 个方向的行走的逐帧。既然分 4 列就说明每个方向的行走动画都是 4 帧。其中第一列 4 帧由上至下分别作为下、左、右、上 4 个方向的站立图使用，如图 7-3、图 7-4 所示。一般第一和第三帧是相同的。

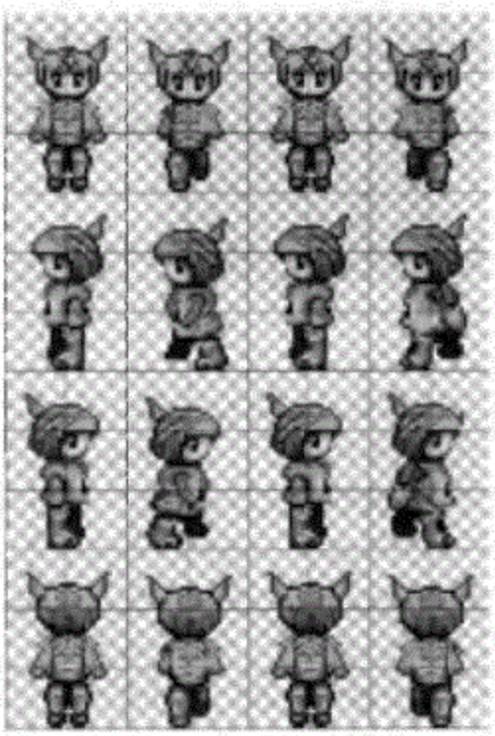


图 7-3

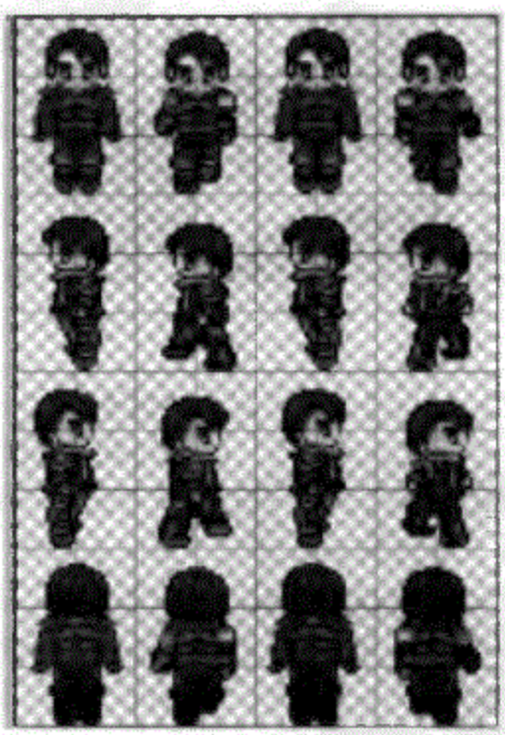


图 7-4

为了精确定位，绘制和排列时可以使用网格和辅助线，在 Photoshop 中选择“编辑”→“首选项”→“参考线、网格和切片”命令，打开相应面板，进行网格和辅助线的设置，如图 7-5 所示。

图 7-5

（2）角色战斗图

在这里绘制了 3 个角色的战斗图（战斗时使用的图片）格式为 PNG 或 JPG，如图 7-6、图 7-7、图 7-8 所示。其中 PNG 格式的文件必须为 32 位元色彩（Alpha Channel）。

图 7-6

图 7-7

图 7-8

RPG Maker XP 开始将战斗从 FF（Final Fantasy）式改为 DQ（Dragon Quest）式，角色战斗图缩水为一副静态图片，这样虽然方便，但这也是一个为人诟病的改动。

一个角色使用一张图片。图片尺寸上限为 640 × 320，敌我双方通用。这里主角和敌兵使用 128 × 170 像素，如图 7-9 所示；BOSS 使用 256 × 256 像素，如图 7-10 所示。

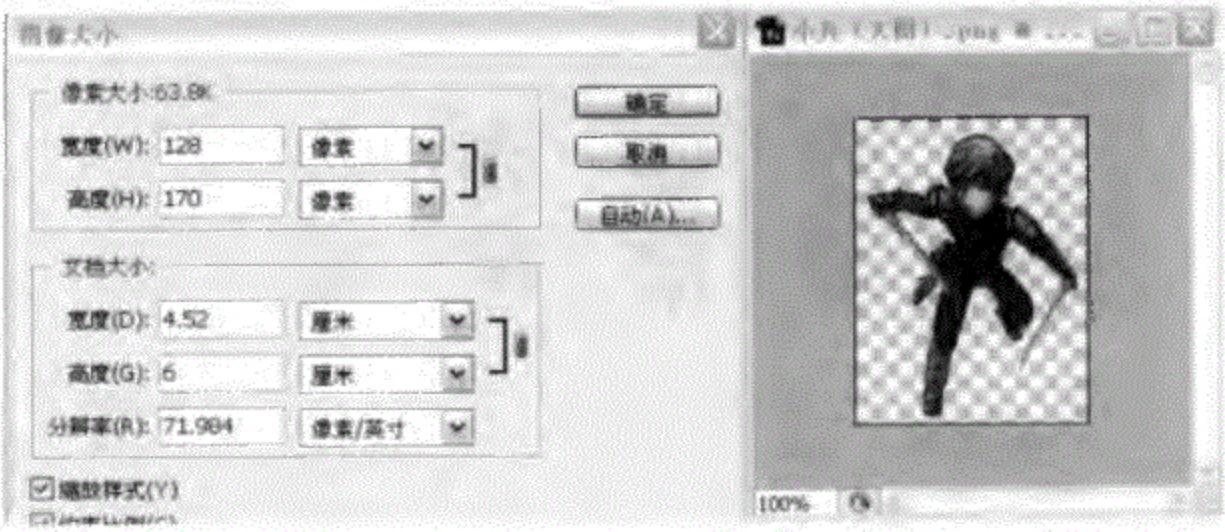


图 7-9

图 7-10

（3）图块

图块更通俗的叫法是地图，但是图块除了地图外也包含各种物件。图块由大小相同、内容不重复的方形元素紧密排列而成，地图的编辑工作实际上就是将图块中方形元素的内容复制并重新排列。

在 RPG Maker XP 中每一个方形最小单位的尺寸固定为 32 × 32 像素。每张图块的尺寸固定为横向 8 个方形即 256 像素；纵向最少一个方形即 32 像素，无上限。虽然一个图块可以无限地添加方形的元素，不过为了制作方便，一般将不同环境的元素配制在不同的图块上（比如沙漠和森林）。

首先绘制第一张地图，地图为古代城外的风格。在绘制的时候，要严格按照 32 × 32 像素一个单元的标准将绘制好的素材合理排列，如图 7-11 所示，再绘制一张城内地图，方法和之前相同，如图 7-12 所示。

对于游戏中结构复杂的建筑物，可以使用三维软件制作并且选好角度渲染再到 Photoshop 做像素的后期处理，如图 7-13 所示。

图 7-11

图 7-12

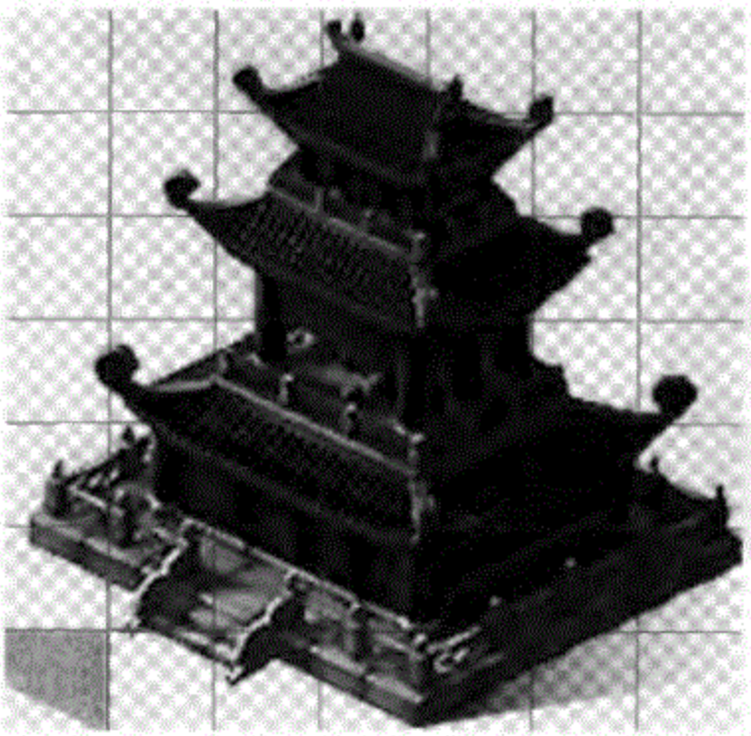


图 7-13

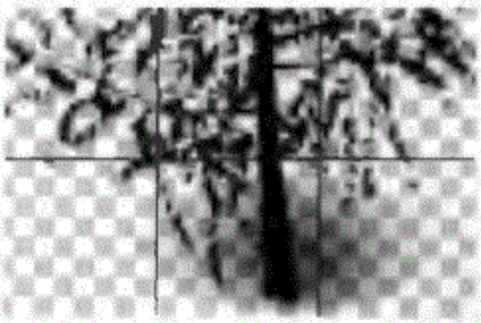


图 7-14

同时叠在上层的图像元素给它增加一些阴影来加强画面效果，例如建筑物，树木等，如图 7-14、图 7-15 所示。

在这里草地为最底层的地面元素，由于需要表现出斜 45°的画面效果，所以各个角度都要考虑到，如图 7-16 所示。

地图周围使用栅栏隔开以做点缀，如图 7-17 所示。

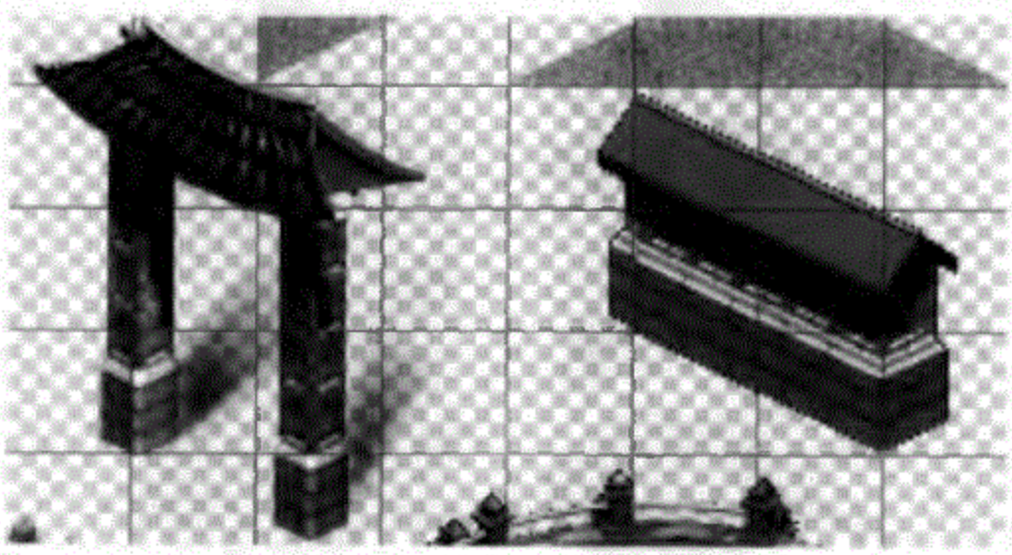


图 7-15

图 7-16

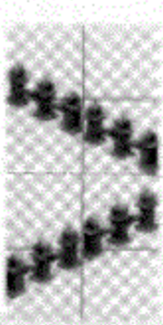


图 7-17

第一张地图绘制完成，紧接着绘制一张城内的地图，还是按照 32 × 32 像素的图像标准来制作，如图 7-18 所示。

在之前的构思中，这个场景中会有商人，所以为了配合剧情，制作了一个客栈，同样是使用了三维转二维的手法，如图 7-19 所示。

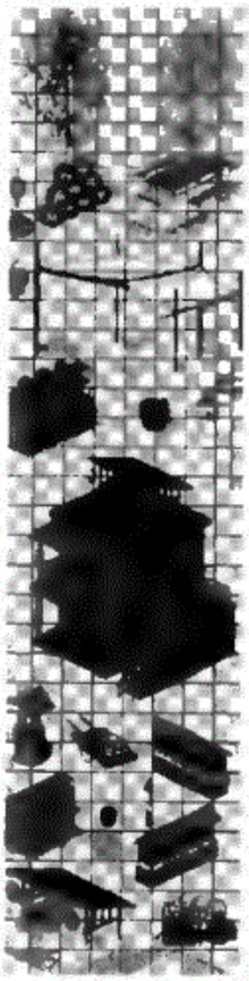


图 7-18

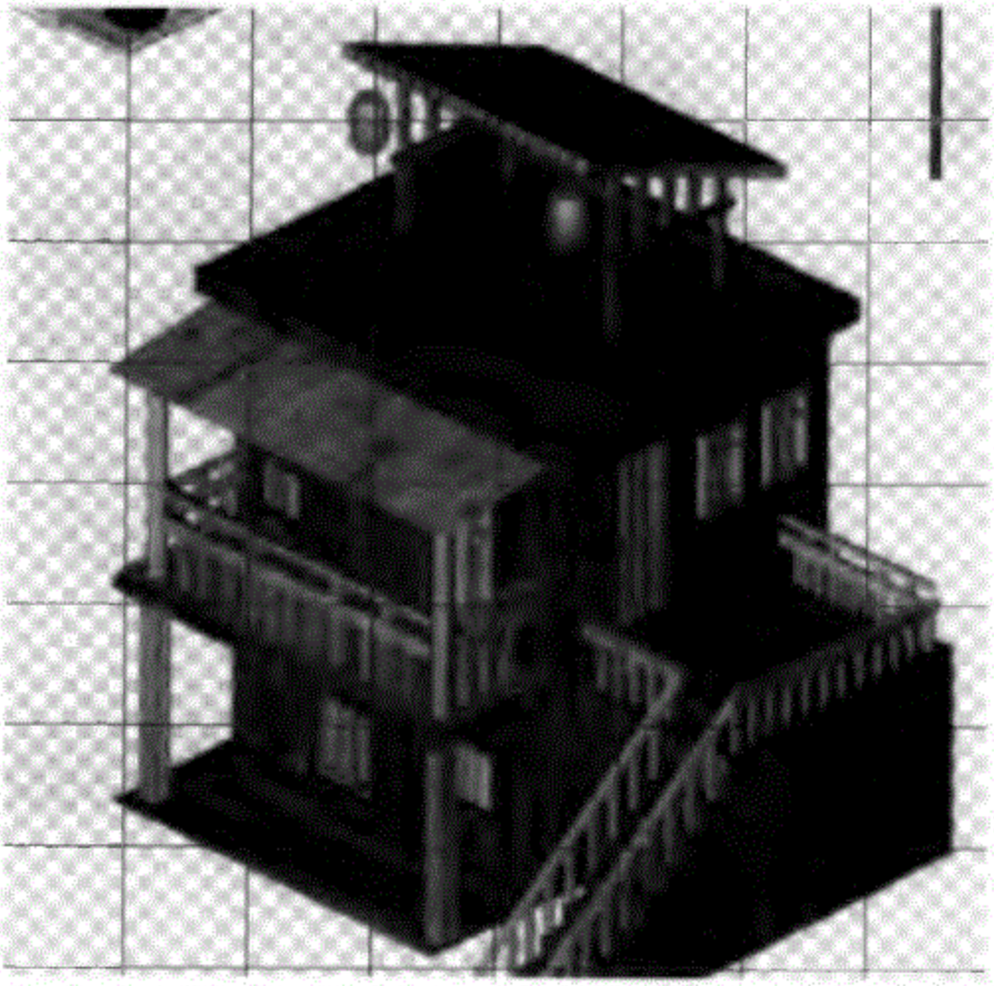


图 7-19

小物件及结构复杂的建筑物依然可以使用三维转二维的方法，如图 7-20、图 7-21 所示。

但是，要注意渲染出来的图像素材还是需要在 Photoshop 中进行再加工（颜色大小及像素调整），如图 7-22 所示。

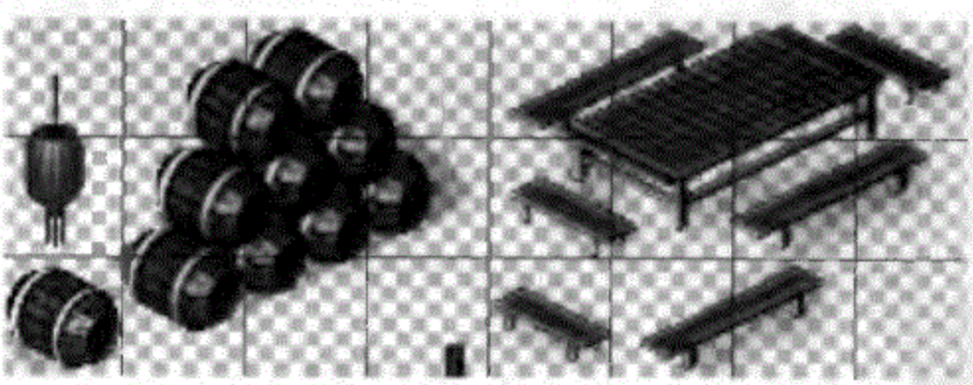


图 7-20

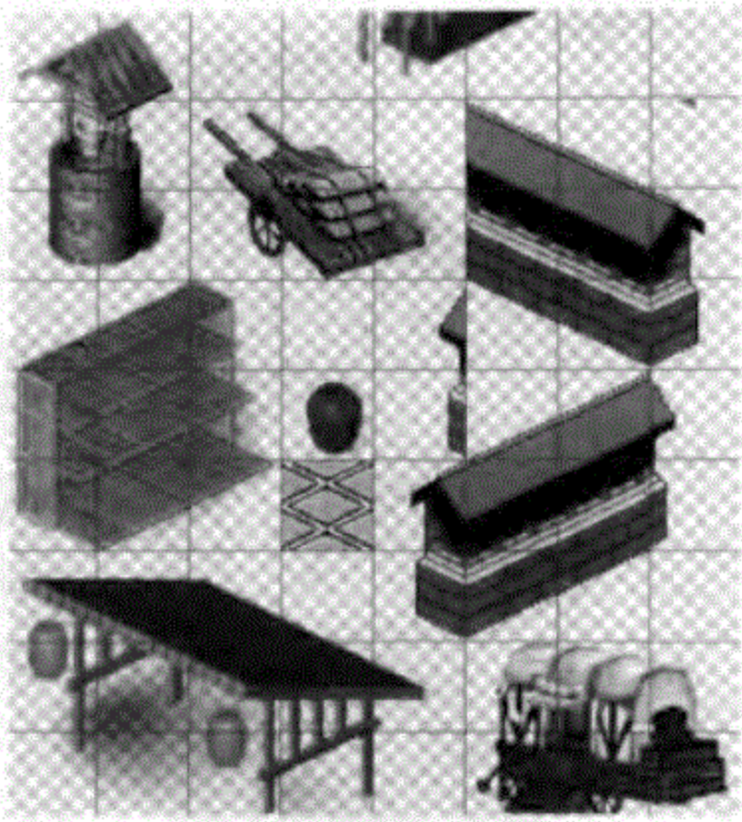


图 7-21

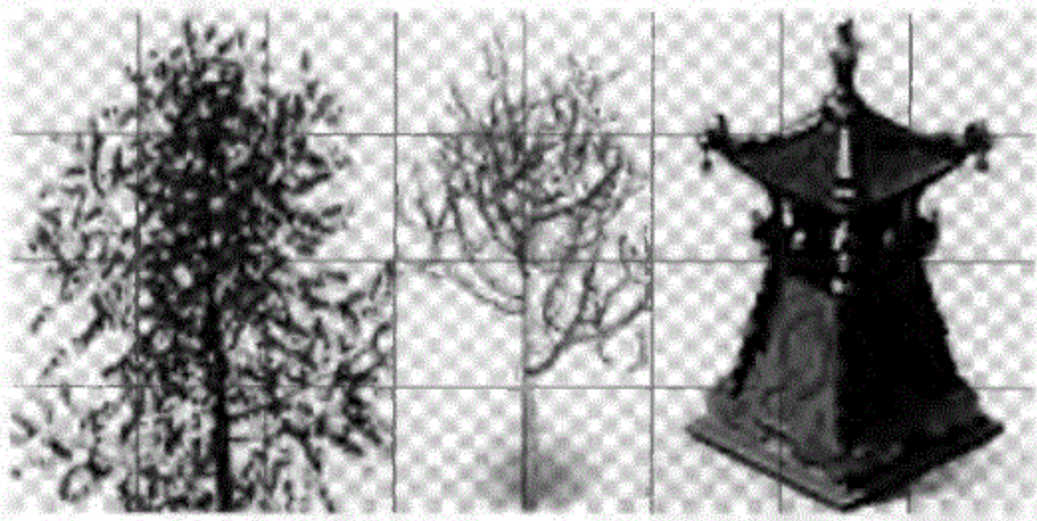


图 7-22

地面可使用四方连续的砖块。周围各个角度的元素都要准备齐全，如图 7-23 所示。

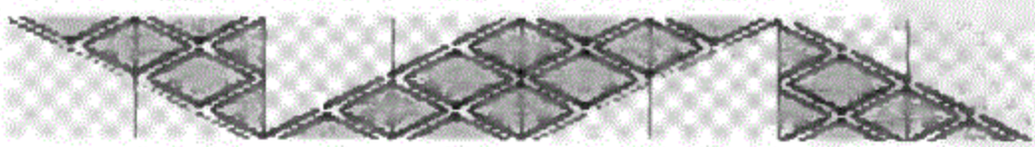


图 7-23

（4）自动地图元件

之前为大家介绍过自动地图元件，它是一种特殊图块，横向 3 个方块元素，96 像素；纵向 4 个方块元素，128 像素。

第一行分 3 个单独区域，剩下 9×9 的方形元素（96×96 像素）为另一个封闭区域。自动地图元件在编辑时能自动选取元素拼合为封闭区域（比如海），详细排列的原理请翻阅前文相关内容。

由于本次实例地图制作成斜 45°、2.5D 类型，所以没有使用到自动地图元件。

（5）战斗状态下的背景

战斗背景绘制了两张，分别为城外的和城内的，如图 7-24、图 7-25 所示。

图 7-24

图 7-25

绘制时要注意透视和景深，尺寸为 640 × 320，如图 7-26 所示。

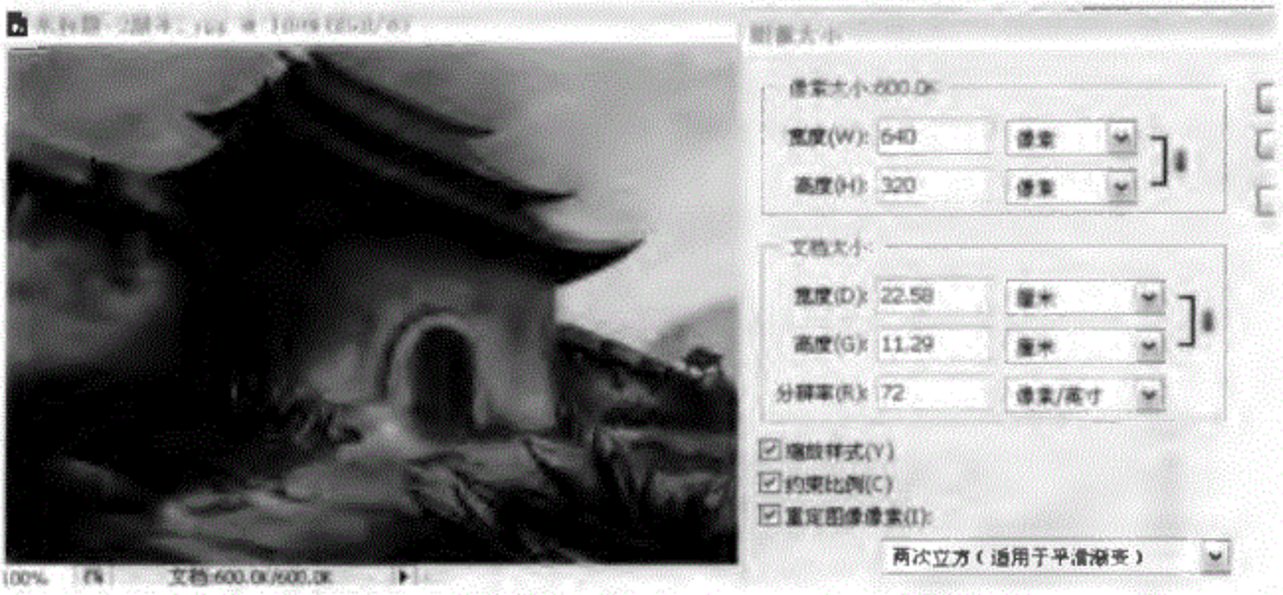


图 7-26

完成了准备工作并积累了一部分素材后，下面就进入 RPG Maker XP 软件中进行实例的制作。

2. 建立工程

之前都是在制作像素图等游戏素材，从建立工程起就开始真正“制作游戏”了，而之前准备的素材和今后扩充的素材都将放在工程文件夹里的各个子目录下面。

单击“文件”→“新建工程”命令，如图 7-27 所示，或直接按“新建”按钮新建工程，弹出“新建工程”对话框。在该对话框中的“文件夹名”文本框中设置工程的名字；“标题”文本框中设置游戏的名字；“位置”文本框中输入工程文件的存放路径，如图 7-28 所示。

图 7-27

图 7-28

工程文件夹内有一个文件类型为 rxproj 的文件，可以通过双击它启动 RPG Maker 并打开工程；也可以先打开 RPG Maker，再通过“打开工程”命令选择路径打开相应工程。

RPG Maker 没有自动保存功能，需要手动选择“保存工程”选项或单击工具栏上的“保存”按钮保存工程进度。

7.3 制作地图

地图不仅仅是背景，它也是 RPG Maker 制作游戏的基础，角色的移动、对话、互动、探索等都是以地图为基础的；事件的设置也要在布置完地图后才能进行。

下面通过实例逐步学习地图的制作过程。

1. 素材的制作

图块的格式和分割在前文已经详细说明过了，之前也提到过虽然游戏素材库提供的素材都是类似早期日式 RPG 中的“正视图”，但只要有相应素材实际也可以做成像《博得之门》、《暗黑破坏神》那种欧美 RPG 的假 3D 斜 45°视角——这个实例中要做的就是这种视角的素材，如图 7-29 所示。

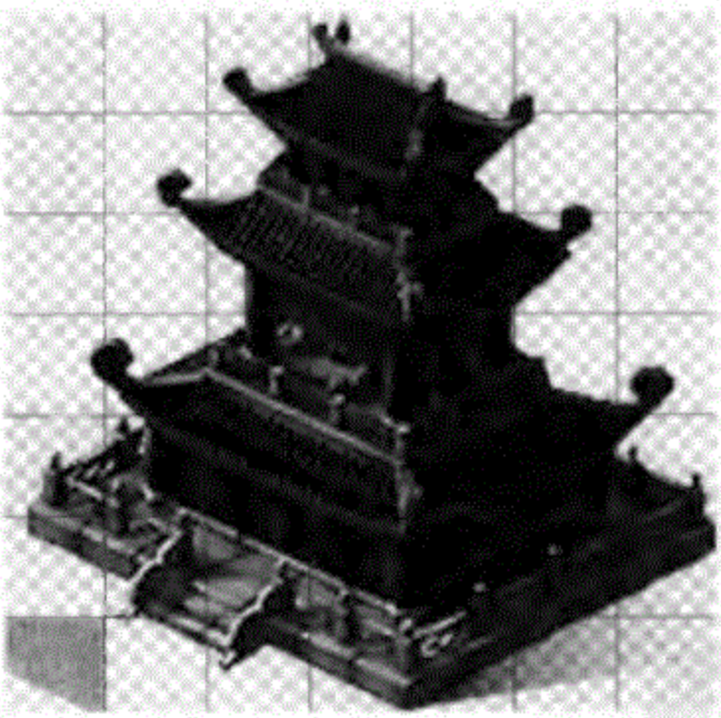


图 7-29

为了把握建筑的透视结构，先在 3ds max 等 3D 制作软件中进行建筑的建模。渲染出所需要的角度的位图后再用 Photoshop 等平面软件修改。

大型建筑分为 32×32 的小块处理，导入 RPG Maker 后再在拼合时还原成原貌。下面就来学习文件的导入。

2. 素材导入

单击工具栏上的媒体库按钮 、单击“工具”→“媒体库”命令或者直接按快

捷键 F10 都可以打开“媒体库”，如图 7-30 所示。这里主要通过媒体库来导入和管理素材。

媒体库下左侧为工程文件夹下子文件夹的目录，不同类型的素材存放在不同文件夹中；右侧是相应文件夹里的素材；最右侧的按钮可以对素材库进行操作，例如导入。

地图使用的是图块素材，对应的是 Graphics/Tilesets 文件夹。选择这个文件夹后如图 7-31，单击“导入”按钮，弹出“导入”对话框。选择素材所在的路径进行导入，导入后的素材会出现在工程文件夹下相应的子目录内，如图 7-32 所示。

完成导入后可以单击“预览”按钮进行检查，如图 7-33、图 7-34 所示。

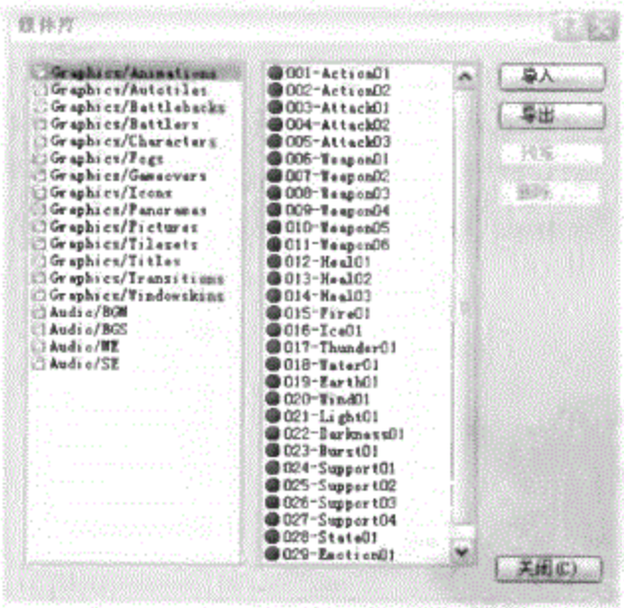


图 7-30

图 7-31

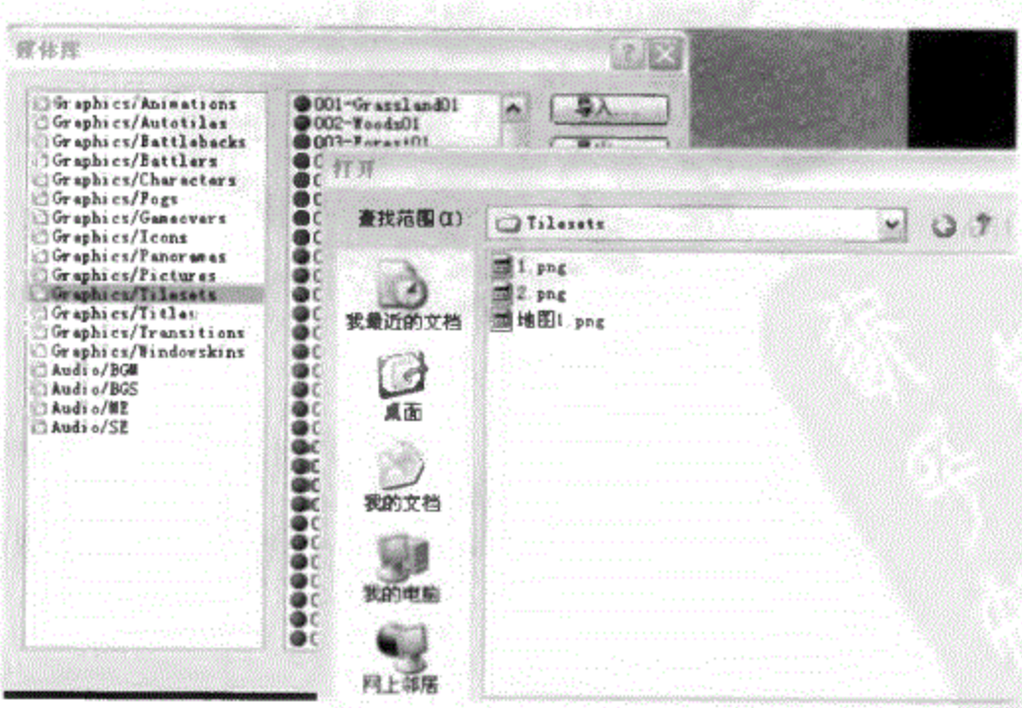


图 7-32

图 7-33

图 7-34

然后导入战斗背景素材，选择媒体库中的 Graphics/Battlebacks 文件夹，单击“导入”按钮，导入两张绘制好的战斗背景素材，如图 7-35 所示。

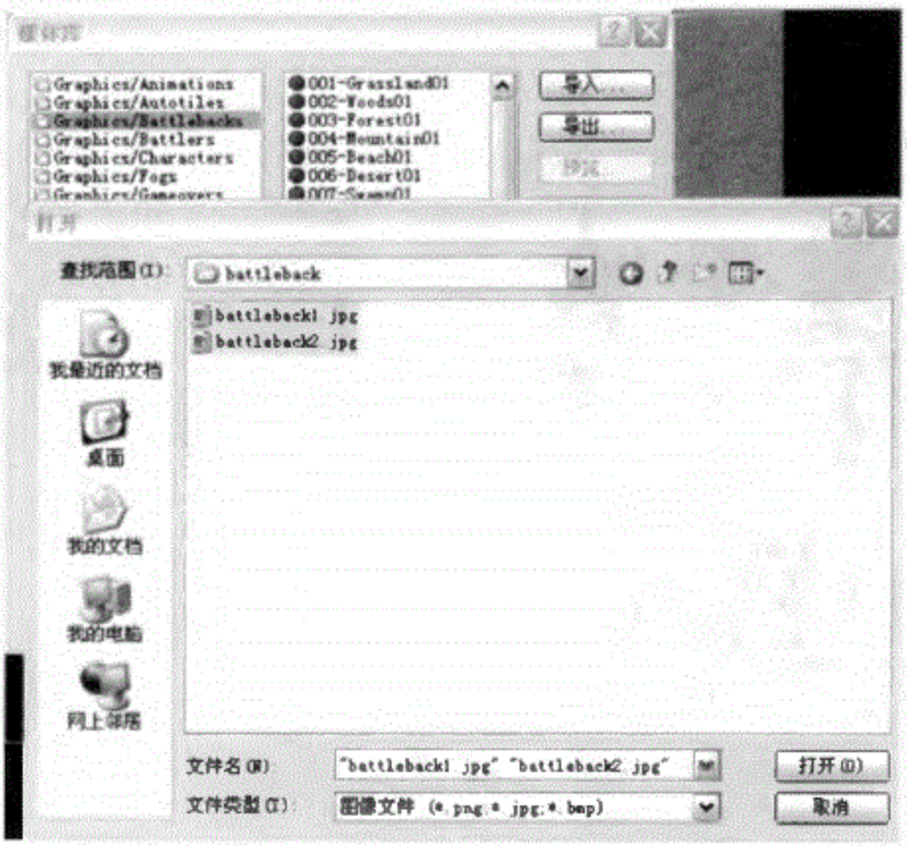


图 7-35

导入之后可单击“预览”按钮进行检查。预览效果如图 7-36、图 7-37 所示。

图 7-36

图 7-37

3. 图块的设置

把素材绘制完成并导入 RPG Maker 后，其实它还只能叫做地图元件图形，需要进行下一个操作才能使其成为真正的图块。

单击工具栏上“数据库”按钮、选择“工具”→“数据库”命令或直接按快捷键 F9 打开“数据库”对话框，并选择“图块”选项卡。

单击“数据库”按钮 打开“数据库”对话框，选择“图块”选项卡，如图 7-38 所示。

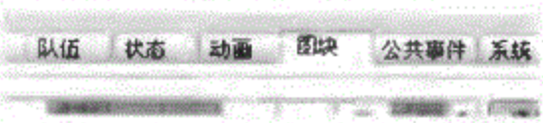


图 7-38

（1）加入数据库

对话框左边的部分为软件中自带的图块素材，共有 50 张，要是再需要增加地图就需要单击“更改最大值”按钮，如图 7-39 所示。由于要增加两张图块素材，所以这里把最大值改为 52，如图 7-40 所示，这样便可以将图块加入数据库了。

图 7-39

图 7-40

在“名称”文本框中设置图块名称，起名为“古代 1”；在“地图元件图形”下拉列表框中选择刚才导入的图片素材，如图 7-41 所示。

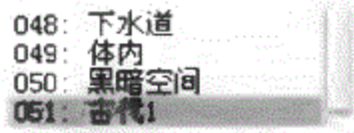


图 7-41

（2）通行：块设置

导入的素材出现在中间列表框中并被分割成小块，每个小块上有都“O”或“X”的标记，如图 7-42 所示，在它右侧有“通行：块”、“通行：4 方向”、“优先级”、“草木繁茂处”、“柜台属性”和“地形标志”等按钮。

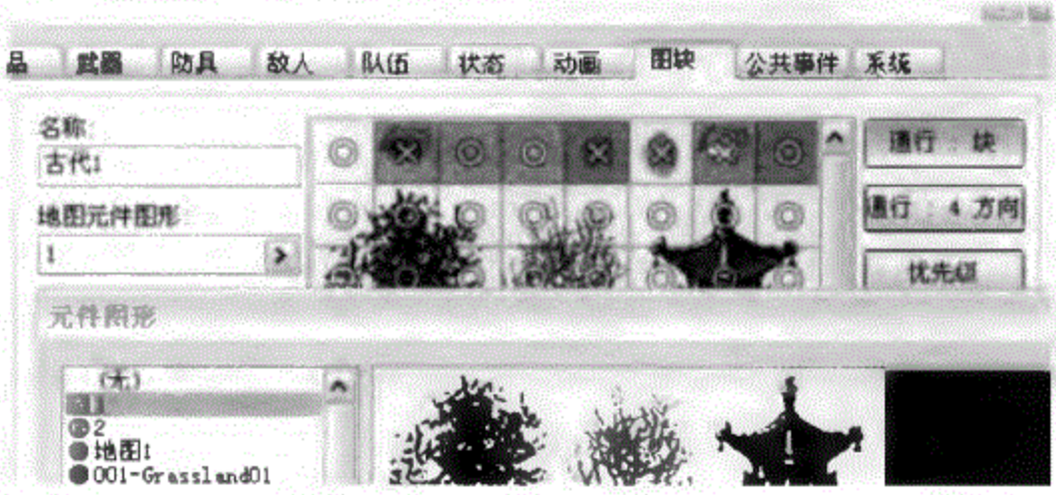


图 7-42

单击“通行：块”按钮之后，就会显示“O”或“X”标记，“O”代表在游戏中角色可以从此处通行，“X”代表游戏中角色无法从此处通过，但是并不是一个地图元件就全部是“X”，因为有些东西即使是障碍物，但依然有可以通过的地方，例如图 7-43 中所示的神坛。

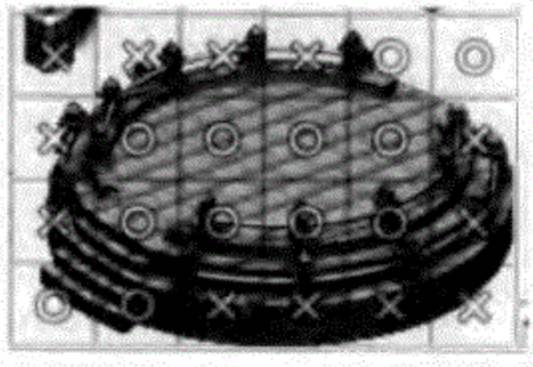


图 7-43

边缘围栏的部分为“X”图标，而梯子以及中间部分则是“O”，表示这块区域可以通行。

该图块的通行设置如图 7-44、图 7-45 所示。

（3）优先级设置

优先级的设置决定了物件在游戏进行中的遮挡效果，优先级越高，遮挡的层级也就越高，在 RPG Maker 中，优先级的最高级数为 5，图块的优先级设置如图 7-46、图 7-47 所示。

设置的时候要注意有些物件例如树木、建筑等由于在角色行走到其背后时能够穿透、又有遮挡效果，所以要配合优先级设置让游戏在进行中物件和角色能够更有层次感。

图 7-44

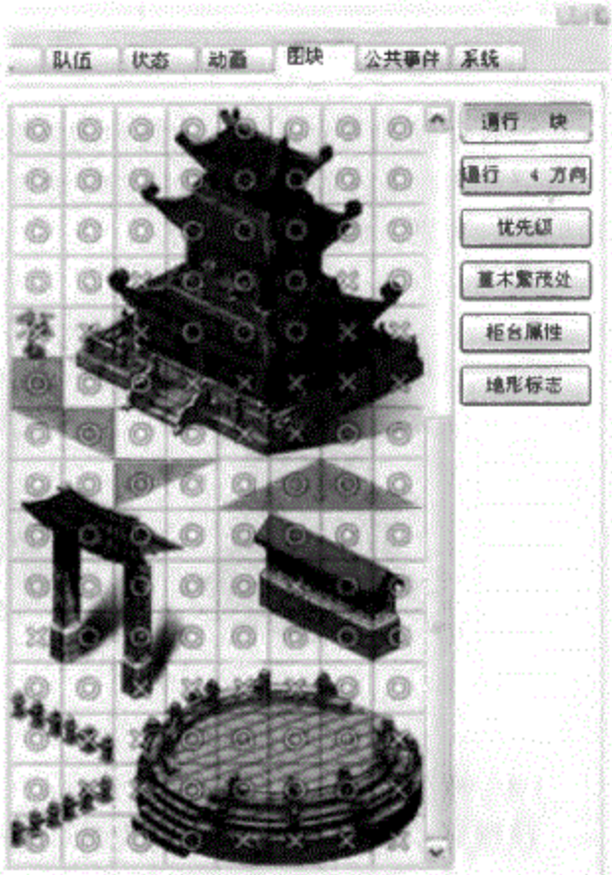


图 7-45

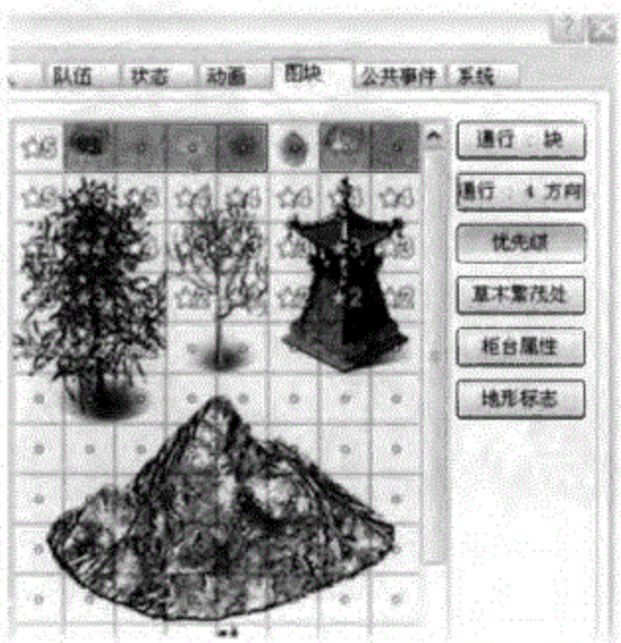


图 7-46

图 7-47

例如图 7-48 中所示的建筑物，在其底部区域使用了“X”图标，说明底部区域不可通过，但是却又在其上半部分使用“O”，这样角色在与该物体的上半部分发生碰撞之后会有穿透的现象产生，这时候就需要配合优先级设置，如图 7-49 所示，物体的上半部分在角色通过的时候能够遮挡住角色，这样便增强了画面的立体感，可以看到游戏进行时的效果，如图 7-50 所示。

图 7-48

图 7-49

图 7-50

（4）战斗背景图片设置

设置完“通行：块”和“优先级”之后，在这张图块上我们还需要设置战斗时的背景图片，在“图块”选项卡中选择“战斗背景图形”，选择之前绘制完成并导入的“battleback1”，如图 7-51 所示。

图 7-51

紧接着来设置第二张地图图块，还是单击“数据库”按钮，打开“数据库”对话框，在“图块”选项卡中用“更改最大值”按钮来增加一个图块，如图 7-52 所示。

将第二个图形元件导入，如图 7-53、图 7-54 所示。

图 7-52

图 7-53

图 7-54

单击“通行:块”按钮之后，设置“O”和“X”标记，来标明可以阻挡的地图物件，设置内容如图 7-55、图 7-56 所示。

图 7-55

图 7-56

该图块的优先级设置如图 7-57、图 7-58 所示。

图 7-57

图 7-58

在设置第一张地图的时候就已经提到过，设置的时候要注意有些物件例如树木，建筑等由于在角色行走到其背后时能够穿透，又有遮挡效果，所以还是要配合优先级设置让游戏在进行中物件和角色层次感更强，如图 7-59 所示。

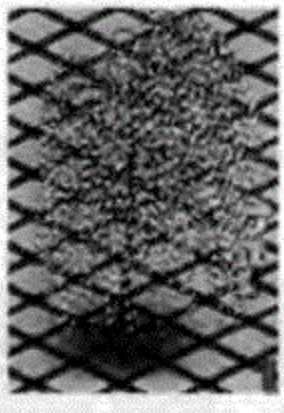


图 7-59

4. 新建并设置地图

图块设置完成之后，就可以使用设置好的图块建立并设置地图了。

RPG Maker 中，在窗口的左下脚，有一个工程文件夹的标志。下面有一个默认名称为 MAP001 的地图文件。

RPG Maker 中的地图是树状分布的，如果在工程文件夹上右击地图文件 RESCUE，在弹出的快捷菜单中单击“新建地图”命令，就会生成新地图并出现在工程文件夹下，如图 7-60 所示，与 MAP001 是并列关系。如果在 MAP001 上右击新建地图，新地图会出现在 MAP001 下，与 MAP001 是从属关系。这方便了地图的管理，例如可以在 MAP001 层级内放世界地图，在世界地图层级下放城镇地图，在城镇地图下放城镇内的室内地图。当然，如果选中工程文件夹并用鼠标右键单击创建出来的地

图便是与之前的地图成并列关系的地图。

在这里将第一章地图命名为“古代”，如图 7-61 所示，在“背景”下拉列表框中可以选择在这个地图中使用哪个图块，选择之前在数据库中设置好的 051 号图块，宽度为 20 个单元，高度为 35 个单元，需要说明的是，无论新建还是编辑地图，都会弹出名称不同的同一个对话框，地图大小单位是行和列，每个单元与图块中的一个元素大小相当，为 32 × 32。

图 7-60

在“遇敌率”中的“队伍”选项中，如果单击鼠标右键选择敌人队伍，将使角色在地图上移动的过程中，走一定的步数就会随机遇到一队敌人，但是这个游戏的实例想要敌人在地图上能够显示出来，并且在与主角接触的同时进行战斗，这就需要在之后的地图事件中进行设置，所以在这里的“遇敌率”选项组中不做任何处理，最后选择“自动切换 BGM”复选框，选择一首 RPG Maker XP 自带的背景音乐，如图 7-62 所示。

图 7-61

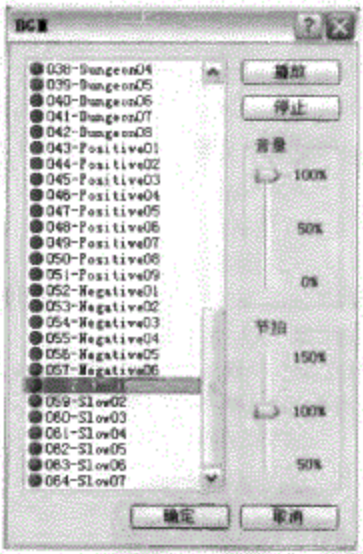


图 7-62

5. 绘制地图

地图设置完成后，就可以利用图块来绘制地图了。

（1）绘制第一张地图

1）绘制第一层

在第一个地图中使用的“古代”城外的地图地面以草地为主，所以可以使用一块四方连续的草地元件来铺设第一层，如图 7-63 所示。

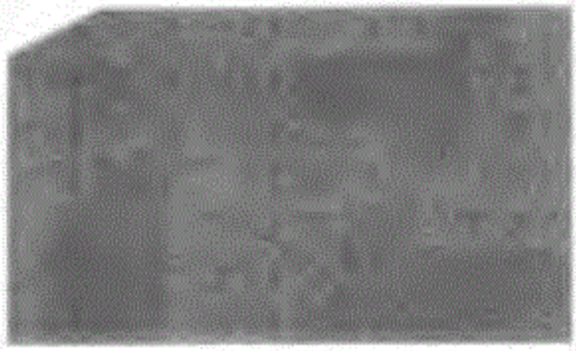


图 7-63

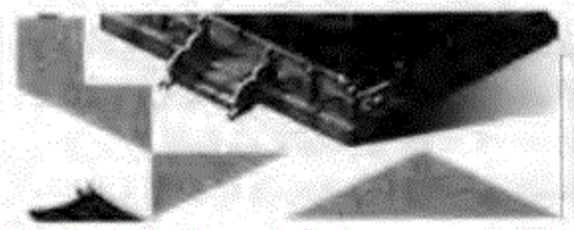


图 7-64

但是由于整个游戏处于45°斜角状态，所以在草地的素材中绘制了一些斜角度的草作为草地的边界，如图 7-64 所示。

正是利用这些斜角度的草地单元，才能够最终确定斜角度地图的边界。第一层铺设的草地效果如图 7-64 所示：为了避免画面过于单调，草地上可以配合一些其他颜色的草，让整个画面看起来丰富一些，如图 7-65、图 7-66 所示。然后加上一些主角出场时要走的小路，这样第一层地图基本上就绘制好了，如图 7-65、图 7-66 所示。

2）绘制第二层

接下来绘制第二层地图，在工具栏上选择第二层。主要的地图元件都应放在第二层，例如建筑物、植物、小道具等，所以说，在绘制第二层地图的时候，基本就是考虑加上整个地图所需要的元件了。

把草地周围的边界用围栏的单元将其围上，如图 7-67 所示，这样地图的边缘就不会显得那么生硬了，之后加上植物，注意植物不要遮挡移动的路线。由于在绘制图块的时候加上了半透明阴影，所以有些物件放置到地面的时候结合度会比较高，看起来也不觉得单调。

将塔楼置于画面中央，如图 7-68 所示，两侧使用城墙，注意城墙要做成可以两方连续的，并且两个方向都要有，如图 7-69 所示，中间留出两个通道，方便角色通行，正面绘制一些植物作为点缀。

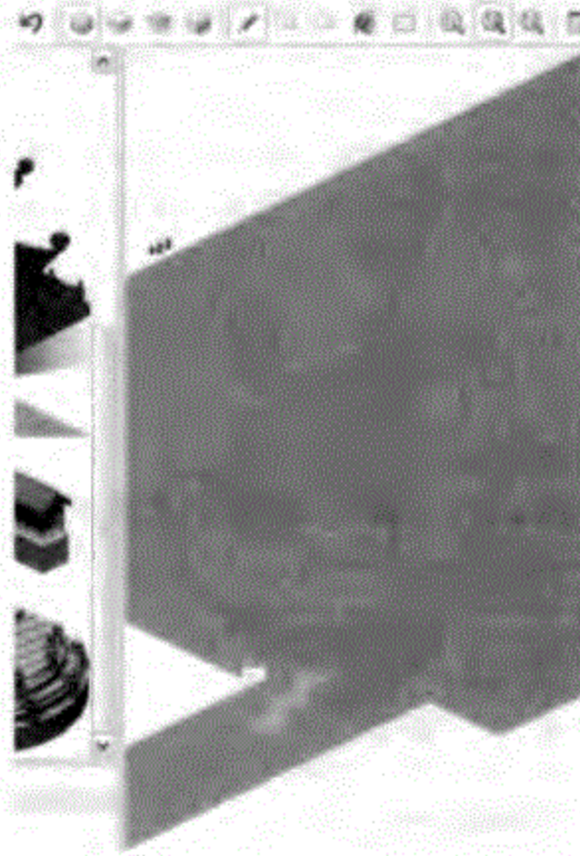


图 7-65

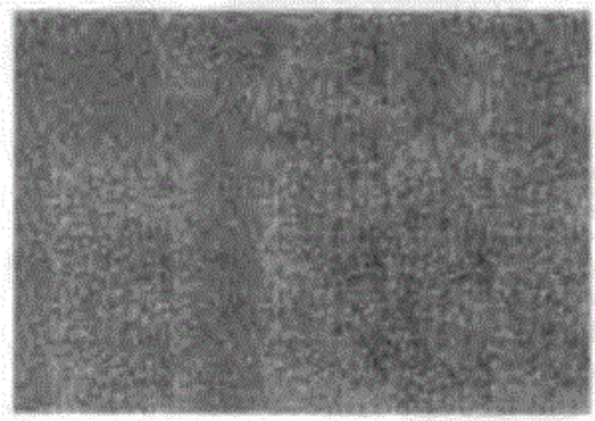


图 7-66

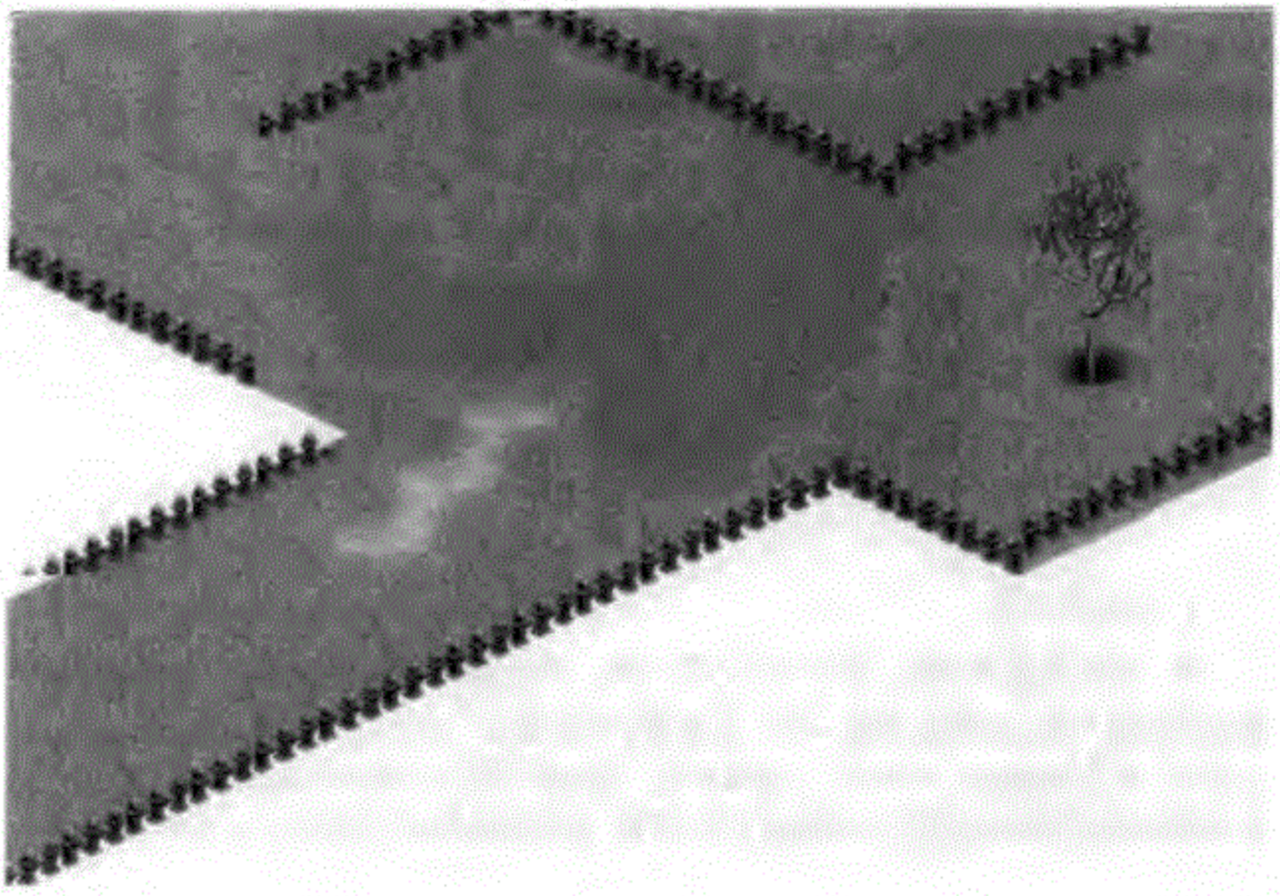


图 7-67

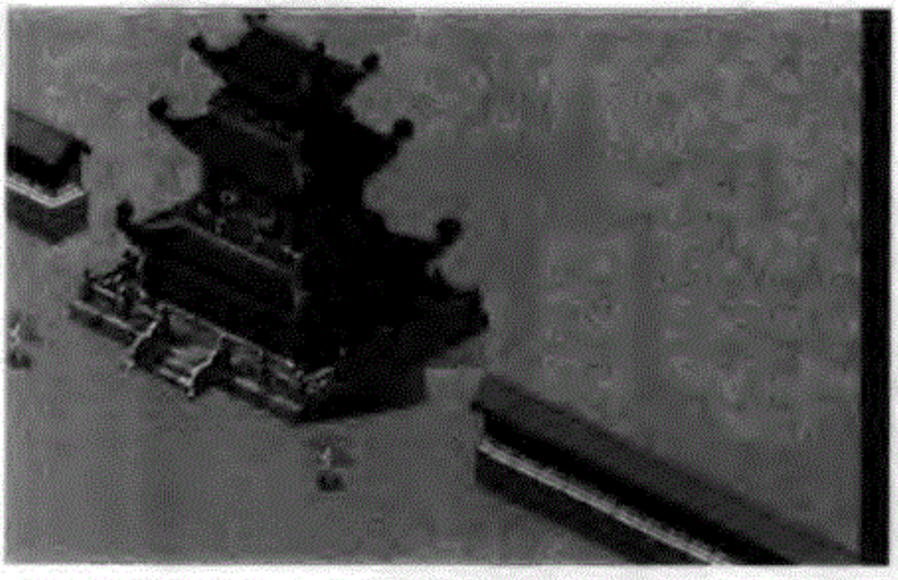


图 7-68

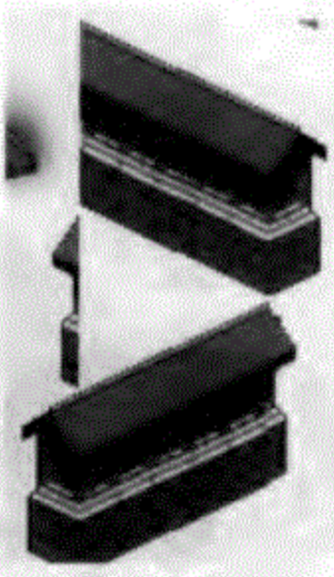


图 7-69

如果一个物件要考虑到前后叠放次序，那么这些物件可以分两层绘制，例如山丘，如图 7-70 所示，要重复利用几个并且层叠起来，造成连绵起伏的感觉，可以在第二层上先放远处的几块，然后就可以到第三层去绘制了。

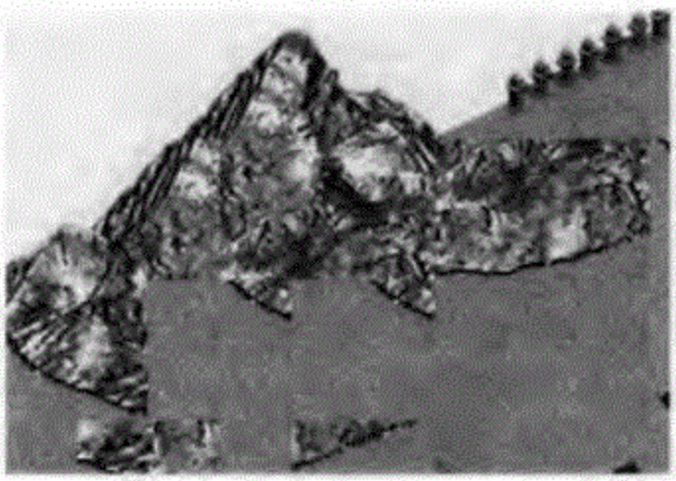


图 7-70

3）绘制第三层

第二层绘制基本完成，接下来进入第三层，单击层按钮，然后给地图做最后的绘制，将山和树木丰富一下，有些地方可以增加一些遮挡，这样效果就比较逼真，这里用门作为地图的入口和出口，如果在某一层的话只能显示某层的地图单元，若完整的地图效果要完全显示的话可以按键盘上的 F4 键。最终的地图效果如图 7-71 和图 7-72 所示。

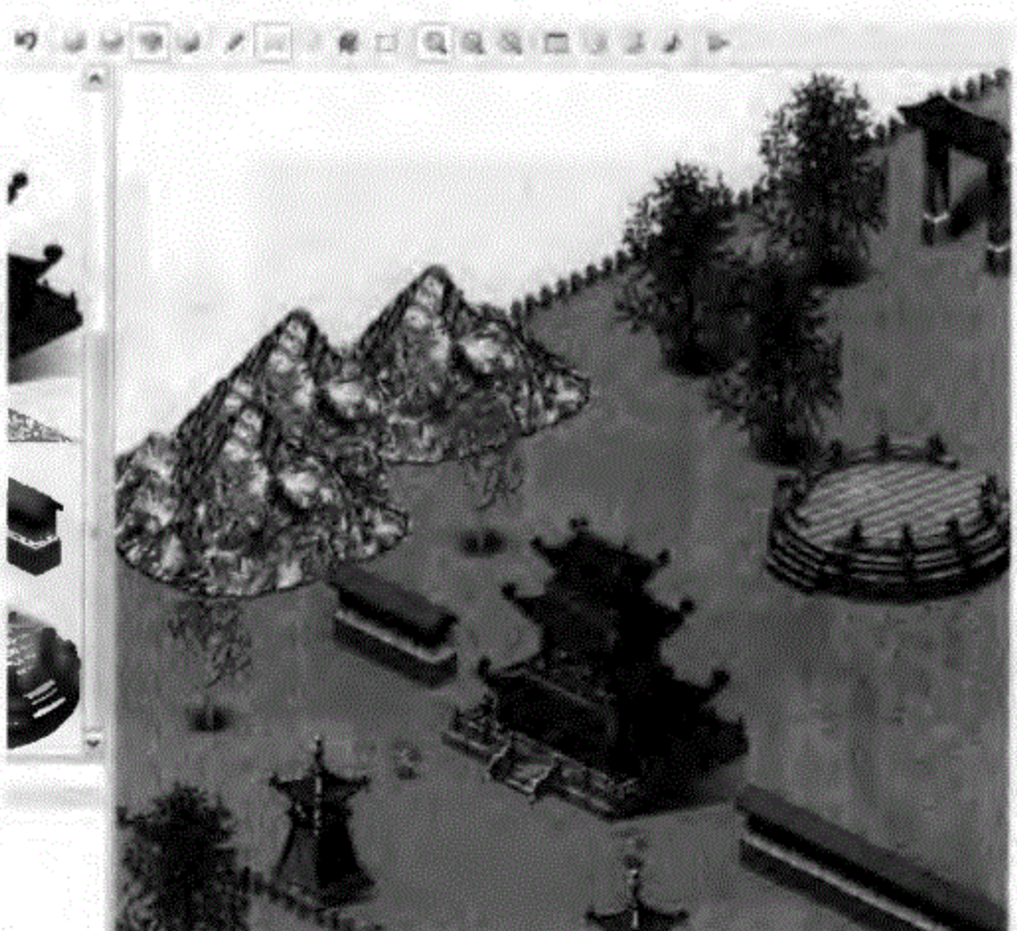


图 7-71

图 7-74

图 7-75

2）绘制第一层

进入到第一层，选择“铅笔工具”进行绘制，首先将地图的最底层——地面铺满，选择四方连续的地面图块，配合各个角度的地面边缘图块将地图整体定型，如图 7-76 所示。

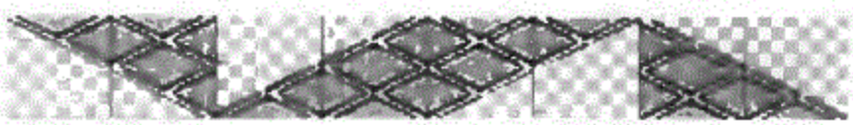


图 7-76

简单铺设城内的入口通道，这样角色进入到城内之后就要显得开阔一些了，整体的最终效果如图 7-77 所示。在这里可以看到地面的边缘部分显然比较生硬，但后面可以利用城墙将这些地面的边缘包裹起来。

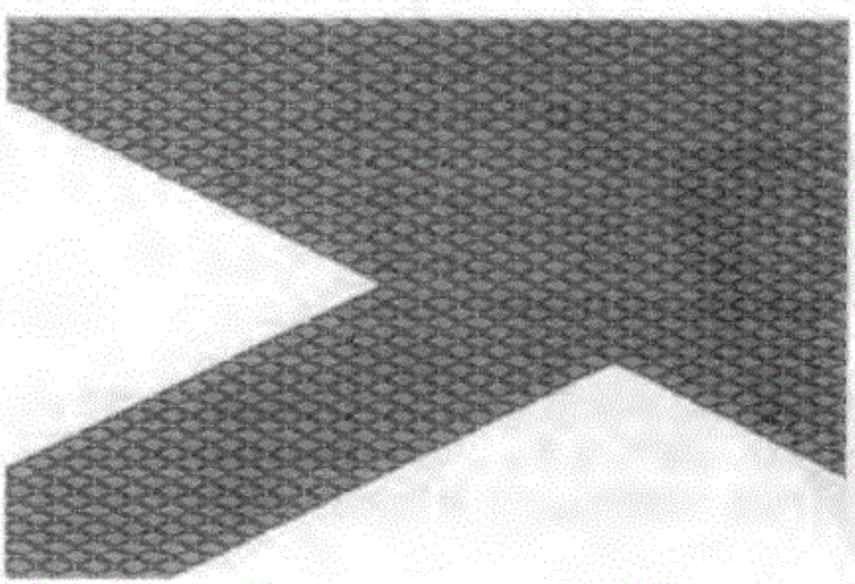


图 7-77

3）绘制第二层

地面绘制完成，切换到第二层，在图块中有之前绘制好的四方连续的城墙，如图 7-78 所示，这里可以将它应用到地面的边缘处。

在这里使用的时候也是如同地面一样，要灵活运用两个方向的城墙以达到拼接的目的，如图 7-79 所示。

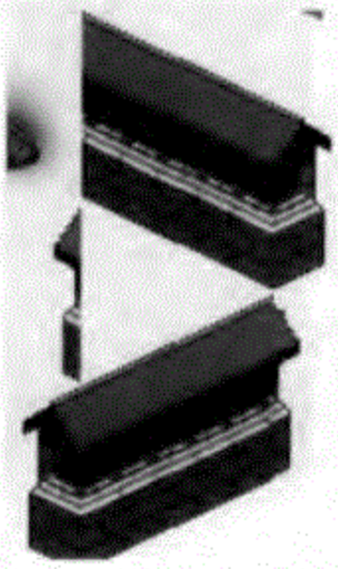


图 7-78

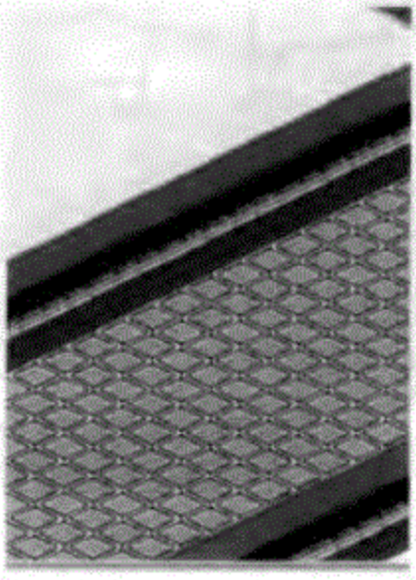


图 7-79

转角处的效果如图 7-80 所示。然后就是一些其他物件的摆放，例如建筑物、花草树木、水桶和推车等。在摆放这些物件时，有些东西是可以重复使用的，例如客栈边上的木桶、木头桌椅等，如图 7-81、图 7-82 所示。这样画面中的元素丰富一些，整个地图的感觉就不会太单调。

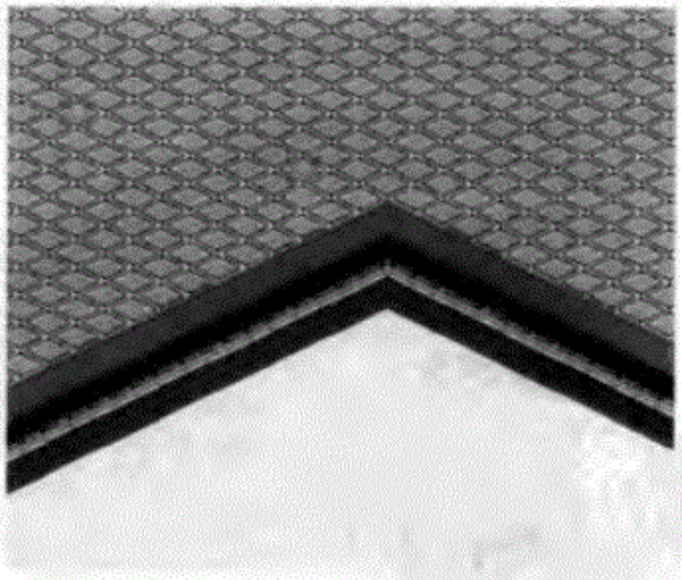


图 7-80

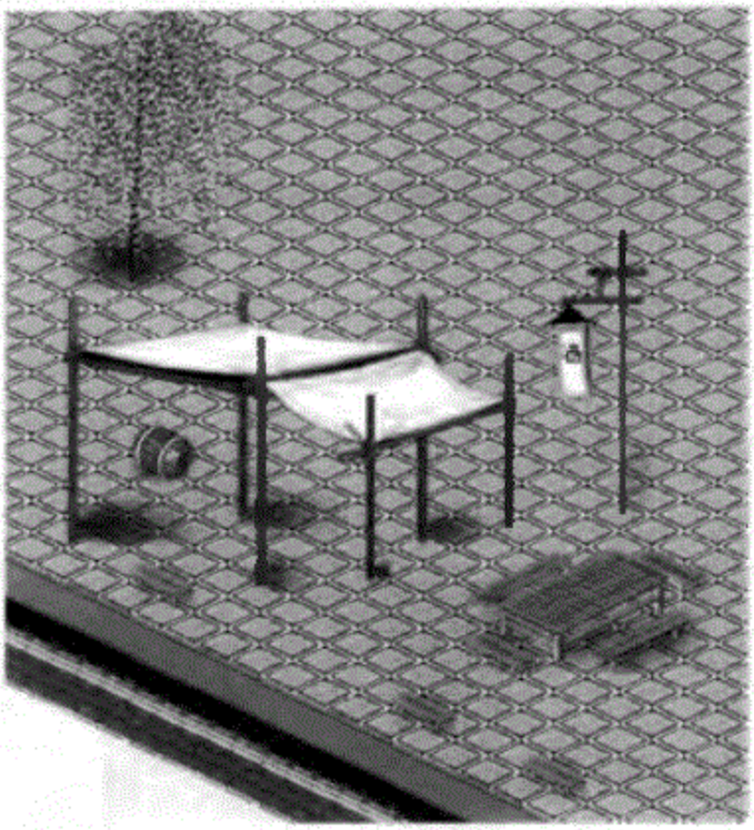


图 7-81

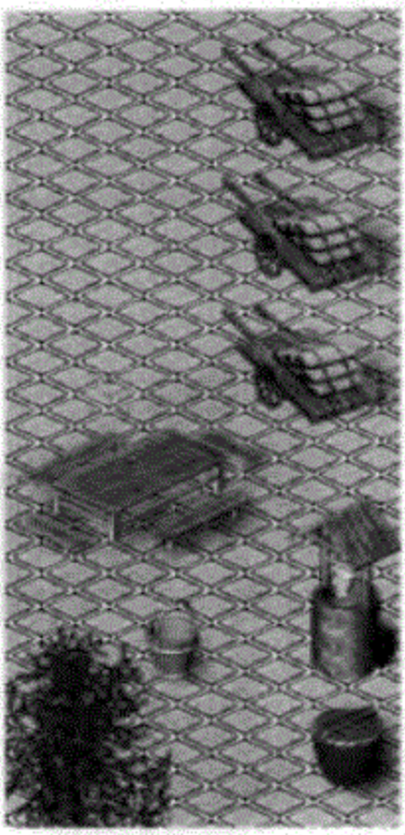


图 7-82

4）绘制第三层

切换到第三层，给地图加上最后的点缀。客栈和酒桶可以互相层叠以增加画面立体感，如图 7-83 所示。

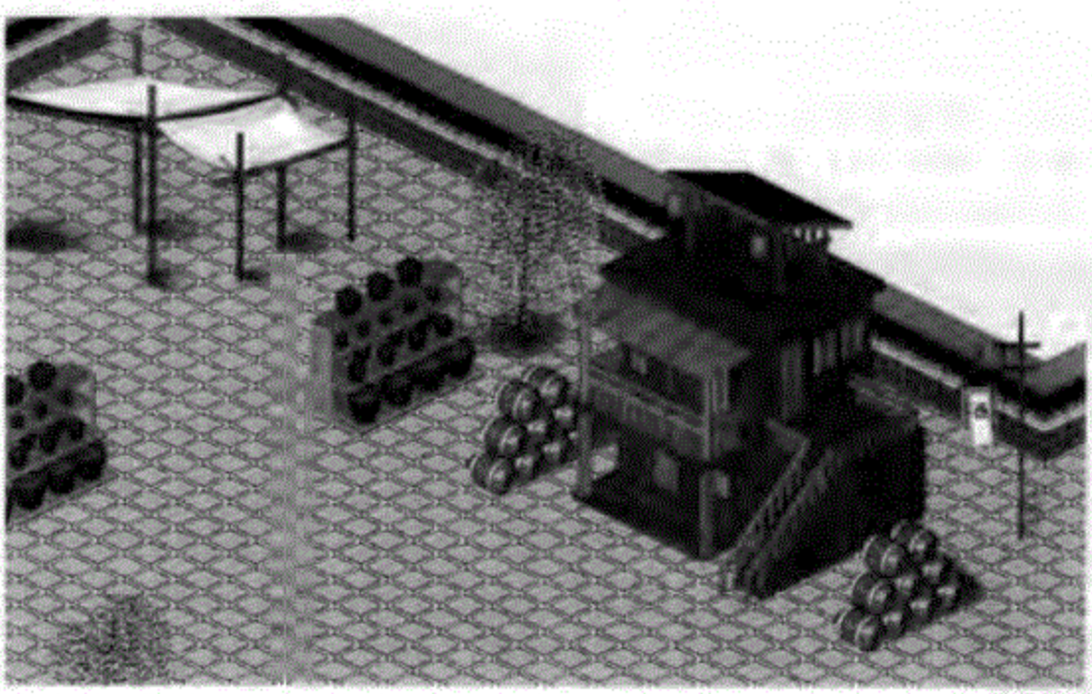


图 7-83

第二张城内地图的最终效果如图 7-84 所示。

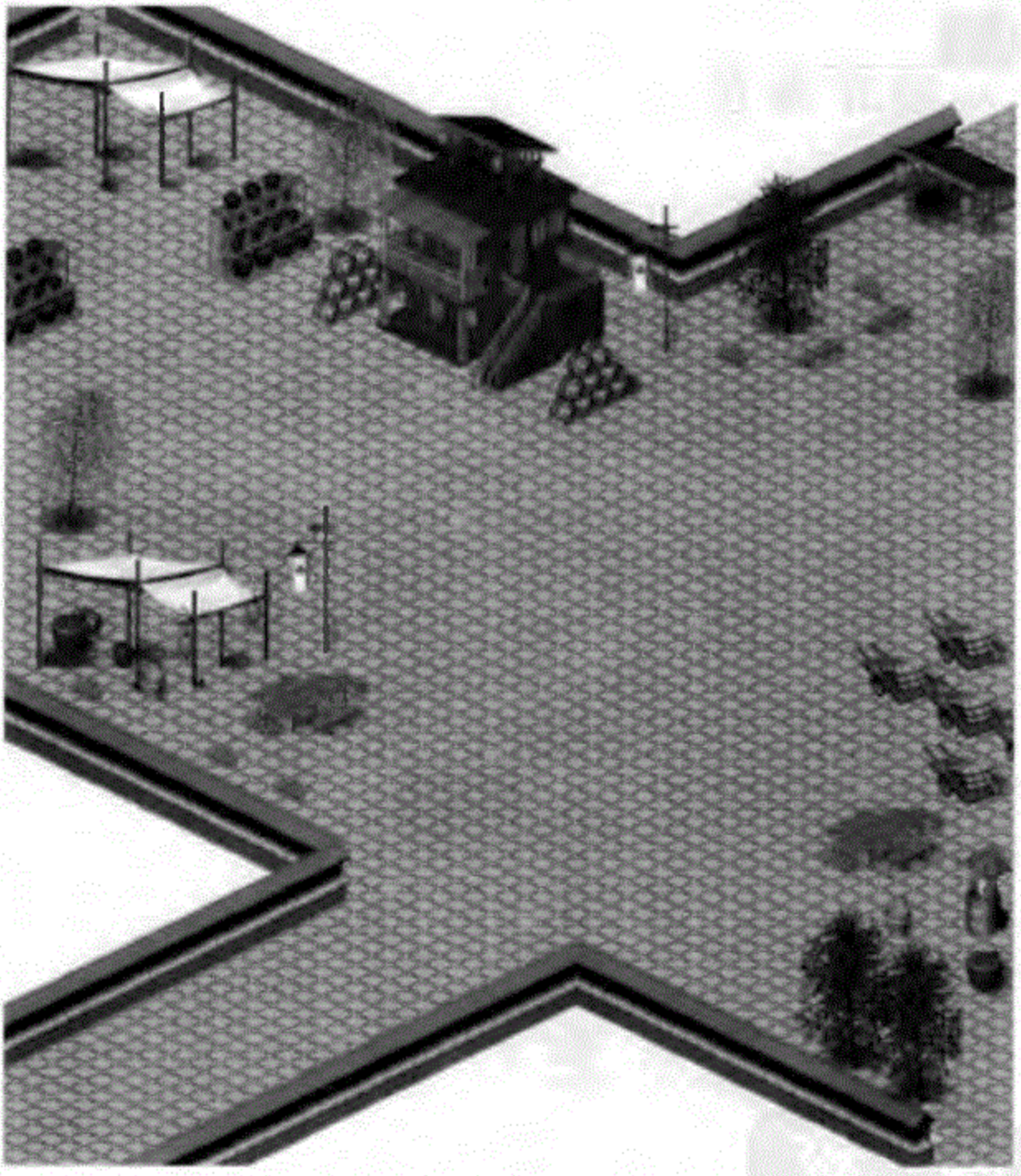


图 7-84

到这里，本游戏的地图设置就全部结束了，虽然说它这只是 RPG Maker 中的一个步骤，但无论怎样，地图在 RPG 游戏中起着至关重要的作用，可以说，整个游戏画面的组成元素中，地图占了 60% 以上的比重。在游戏的互动部分地图的碰撞效果也是至关重要的一部分。

7.4 制作角色

1. 角色素材导入

（1）角色行走图

接下来就是制作游戏角色的工作了，角色素材分为地图上移动的素材，和战斗时的素材，首先导入地图上移动的角色素材。

单击工具栏上的媒体库按钮、单击“工具”→“媒体库”命令或者直接按快捷键 F10 都可以打开“媒体库”对话框，在弹出的对话框中选择 Graphics/Characters，导入绘制好的 3 个角色，如图 7-85 所示。

图 7-85

单击“预览”按钮预览一下素材是否显示正确，如图 7-86 所示。

（2）角色战斗图

地图行走图像导入完成后，接着导入战斗图像，选择“媒体库”对话框中的 Graphic/Battlers，将 3 个角色所对应的 3 张战斗图像导入到 RPG Maker 中，如图 7-87 所示。

单击“预览”按钮，检查导入的图像素材是否正确，如图 7-88 所示。

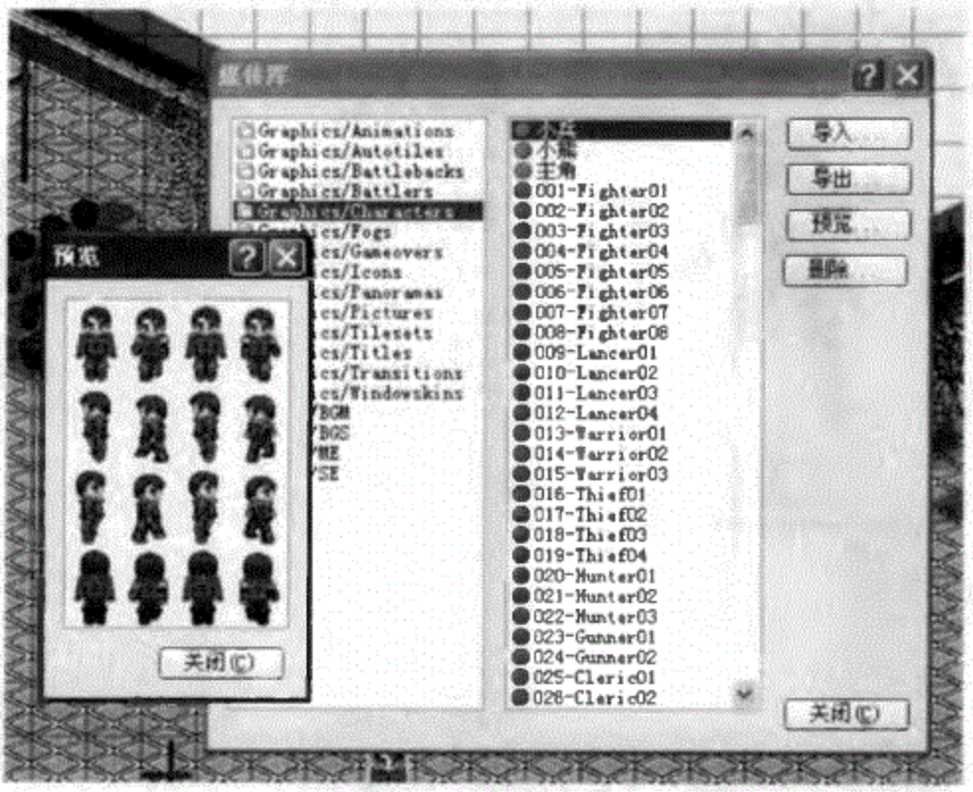

媒体库
Graphics/Animations
Graphics/Autotiles
Graphics/Battlebacks
Graphics/Battlers
Graphics/Characters
Graphics/Fogs
Graphics/Gameovers
Graphics/Icons
Graphics/Panoramas
Graphics/Pictures
Graphics/Tilesets
Graphics/Titles
Graphics/Transitions
Graphics/Windowskins
Audio/BGM
Audio/BGS
Audio/ME
Audio/SE
小兵
小熊
主角
001-Fighter01
002-Fighter02
003-Fighter03
004-Fighter04
005-Fighter05
006-Fighter06
007-Fighter07
008-Fighter08
009-Lancer01
010-Lancer02
011-Lancer03
012-Lancer04
013-Warrior01
014-Warrior02
015-Warrior03
016-Thief01
017-Thief02
018-Thief03
019-Thief04
020-Hunter01
021-Hunter02
022-Hunter03
023-Gunner01
024-Gunner02
025-Cleric01
026-Cleric02
导入...
导出...
预览...
删除
关闭(C)
预览
关闭(C)

图 7-86


媒体库
Graphics/Animations
Graphics/Autotiles
Graphics/Battlebacks
Graphics/Battlers
Graphics/Characters
Graphics/Fogs
Graphics/Gameovers
Graphics/Icons
Graphics/Panoramas
Graphics/Pictures
Graphics/Tilesets
Graphics/Titles
Graphics/Transitions
Graphics/Windowskins
Audio/BGM
Audio/BGS
Audio/ME
Audio/SE
小兵(大图)
主角(大图)
001-Fighter01
002-Fighter02
003-Fighter03
004-Fighter04
005-Fighter05
006-Fighter06
007-Fighter07
008-Fighter08
009-Lancer01
010-Lancer02
011-Lancer03
012-Lancer04
013-Warrior01
014-Warrior02
015-Warrior03
016-Thief01
017-Thief02
018-Thief03
019-Thief04
020-Hunter01
021-Hunter02
022-Hunter03
023-Gunner01
024-Gunner02
025-Cleric01
026-Cleric02
导入...
导出...
预览...
删除
关闭(C)

图 7-87

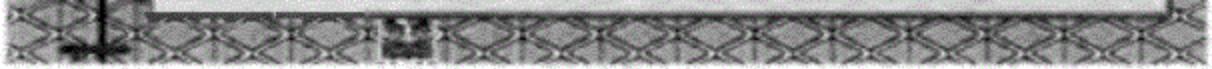


图 7-88

2. 角色数据设置

（1）设置角色

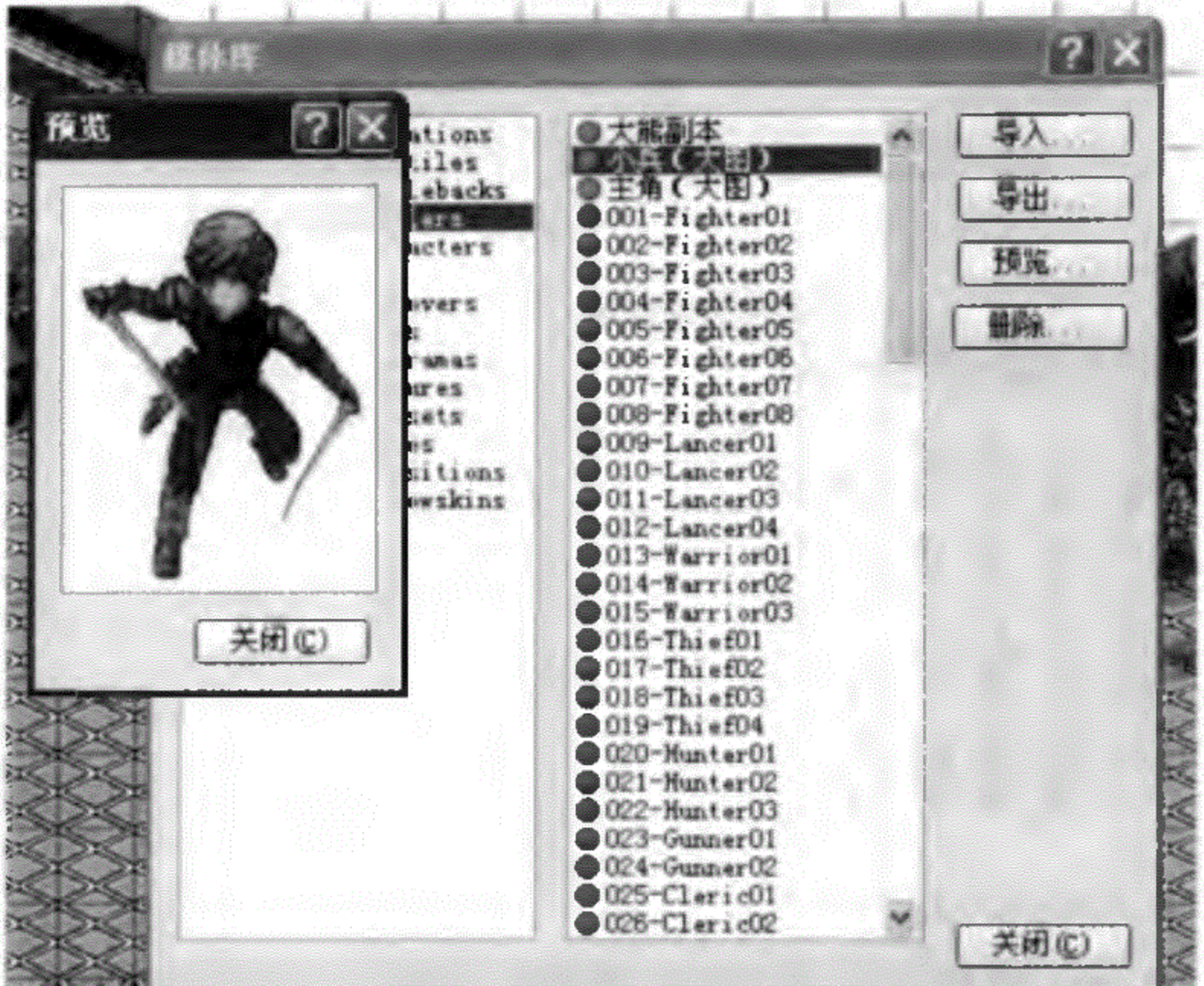
1）角色若零设置

完成媒体库的素材导入设置之后就可以使用这些素材了，要使图像素材转变为游戏中可以使用的物件或单位，就需要在数据库中进行设置，先设置主角（玩家控制的角色），单击“数据库”按钮 打开“数据库”对话框，选择“角色”选项卡，如图 7-89 所示。

为了添加自己制作的角色，在这里需要将 RPG Maker 中默认的 8 名角色删除，选中系统中默认的角色，单击鼠标右键选择“清除”命令，如图 7-90 所示，将 8 个角色清空，如图 7-91 所示。

在这里有两个角色可供玩家使用，一个是女性角色若零，还有一个是在经过第一轮战斗之后会加入我方阵营的敌军士兵，所以要重新定义一下角色的最大值，单击“角色”选项卡中的“更改最大值”按钮，如图 7-92 所示，在弹出的对话框中将“最大值”改为 2，如图 7-93 所示。

选择角色 001，角色名称命名为“若零”，职业选择战士，等级设置 1~10，如图 7-94 所示。


模仿库
预览
关闭(C)
大能副本
小兵(多人副)
主角(大图)
001-Fighter01
002-Fighter02
003-Fighter03
004-Fighter04
005-Fighter05
006-Fighter06
007-Fighter07
008-Fighter08
009-Lancer01
010-Lancer02
011-Lancer03
012-Lancer04
013-Warrior01
014-Warrior02
015-Warrior03
016-Thief01
017-Thief02
018-Thief03
019-Thief04
020-Hunter01
021-Hunter02
022-Hunter03
023-Gunner01
024-Gunner02
025-Cleric01
026-Cleric02
导入...
导出...
预览...
删除...
关闭(C)

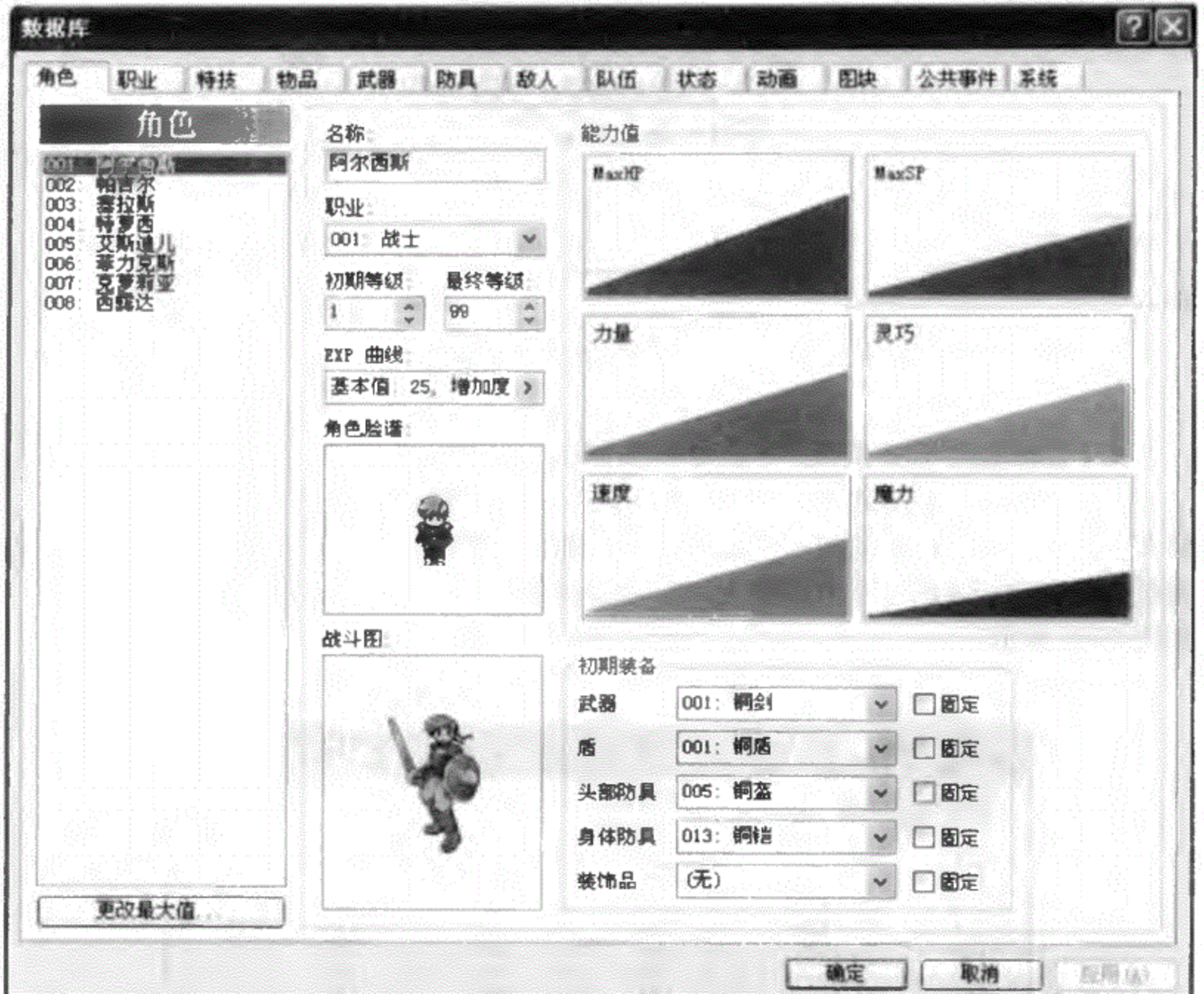

数据库
角色
职业
特技
物品
武器
防具
敌人
队伍
状态
动画
图块
公共事件
系统
角色
001
002 帕吉尔
003 塞拉斯
004 特罗西
005 艾斯迪儿
006 菲力克斯
007 克罗莉亚
008 西露达
更改最大值
名称
阿尔西斯
职业
001 战士
初期等级
1
最终等级
99
EXP 曲线
基本值 25, 增加度
角色脸谱
战斗图
能力值
MaxHP
MaxSP
力量
灵巧
速度
魔力
初期装备
武器 001: 铜剑 固定
盾 001: 铜盾 固定
头部防具 005: 铜盔 固定
身体防具 013: 铜铠 固定
装饰品 (无) 固定
确定
取消

图 7-89

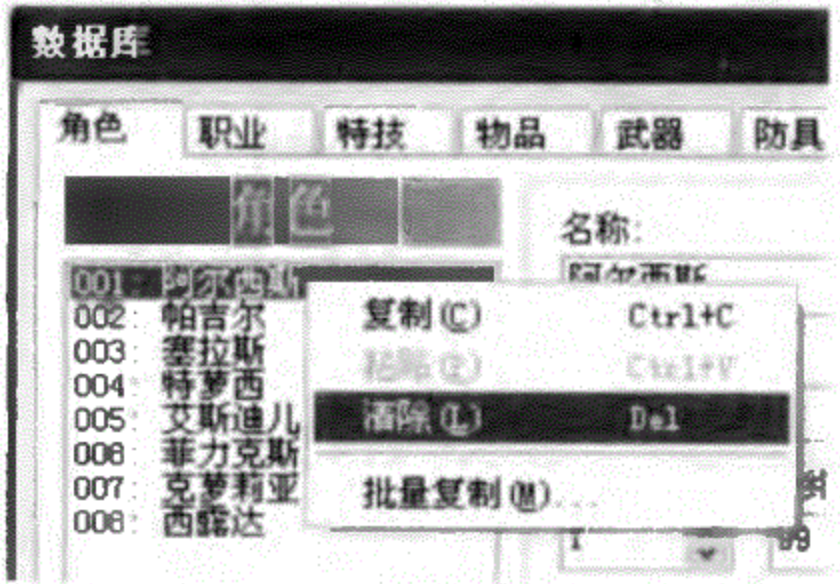


图 7-90

图 7-91

图 7-92

图 7-93

图 7-94

往下能够看到“EXP 曲线:”下拉列表框，如图 7-95 所示。

EXP 曲线即经验值曲线，它记录了距离下一个等级还需要多少经验值，EXP 曲线可以通过基本值和增加度进行调整，将主角若零的基本值调整为 14，增加度调整为 38，如图 7-96 所示。

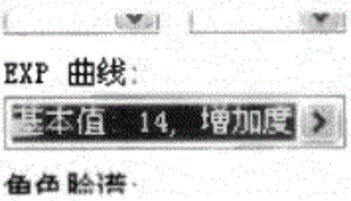


图 7-95

图 7-96

在“能力值”选项组中记录了角色的成长曲线，共有 6 项，分别为 MaxHP、MaxSP、力量、灵巧、速度和魔力，如图 7-97 所示。

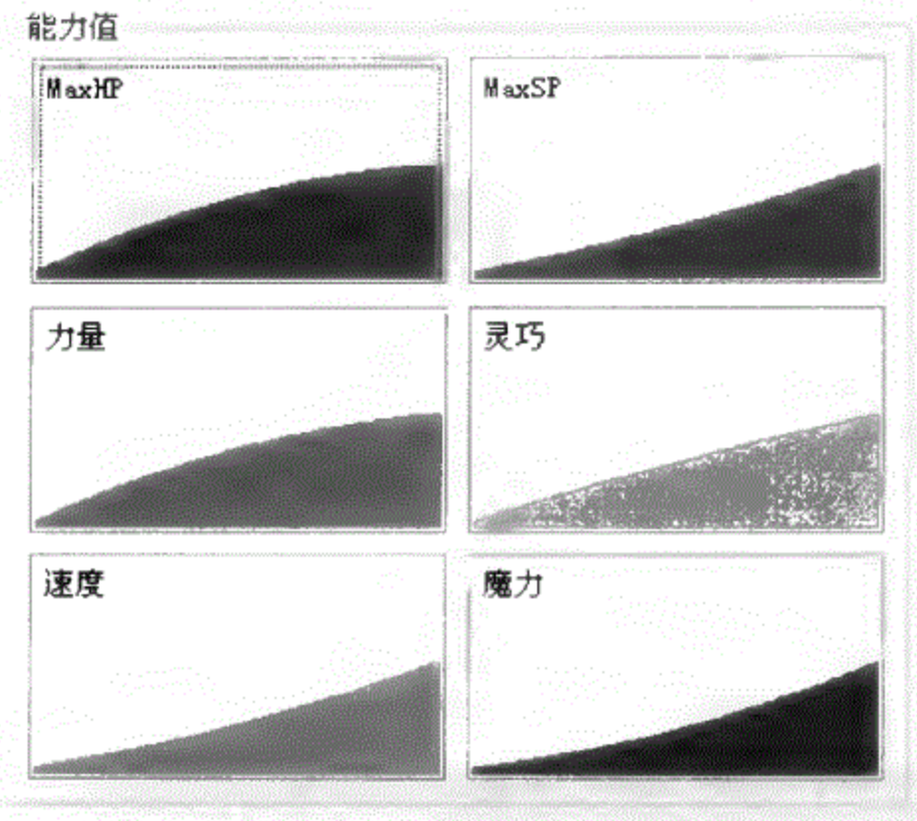


图 7-97

双击其中任何一个选项都能够打开“能力值”对话框，如图 7-98 所示，在该对话框中可以直接用鼠标绘制能力值的成长曲线，系统还提供了 A、B、C、D、E 这 5 种简易的曲线设置，通过选择顶部的选项卡可以选择到其他的能力值的设定页面。

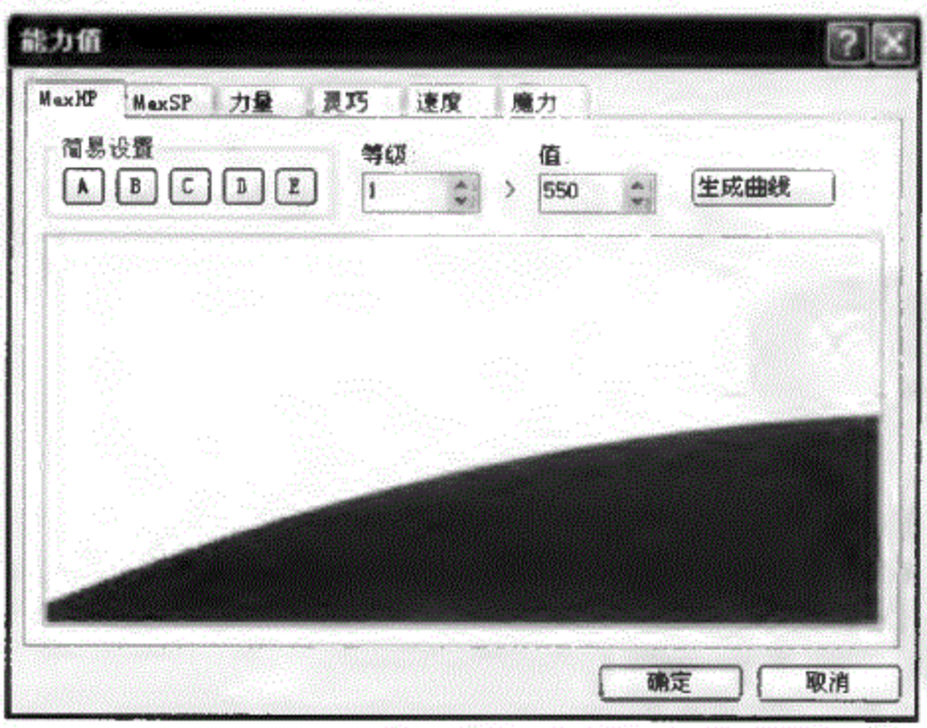


图 7-98

单击“生成曲线”按钮，打开“生成曲线”对话框，如图 7-99 所示，在该对话框中拖动滑动块到“早熟”，将若零的 MaxHP（最大体力）成长曲线设为早熟型，如图 7-100 所示，单击“确定”按钮。

图 7-99

图 7-100

切换到“力量”选项卡，将若零的力量成长曲线同样设为早熟，如图 7-101 所示。这样，在成长初期的时候，主角就已经拥有较高的 HP 和较强的攻击能力了。由于主角是一个强力型角色，为了平衡游戏性，所以可以把其他能力的成长曲线设定为晚熟。

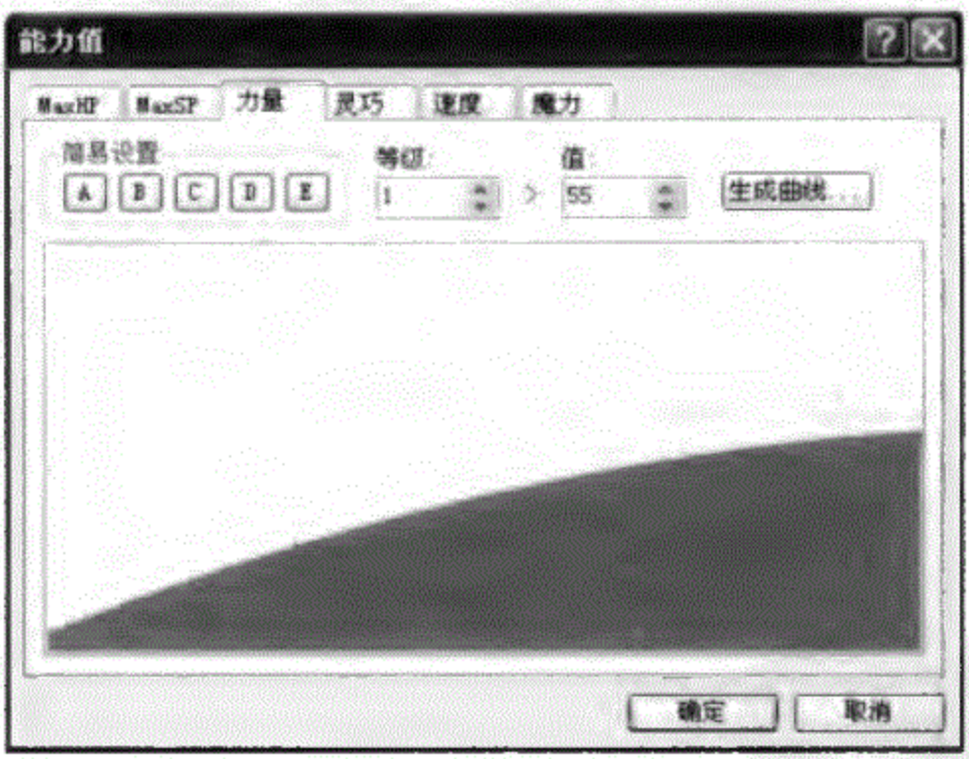


图 7-101

曲线设置完毕后，回到原来窗口中右下角的“初期装备”选项组，给角色增加一些登场时就拥有的武器装备，按照角色设定，选择给她使用铜槌作为武器，头部和身体的防具依次使用铜盔和铜铠，如图 7-102 所示，当然在游戏的运行过程中可以设置一些中途得到的装备，例如通过宝箱获得、商店购买等方式。

最后给角色加入脸谱和战斗图，选择素材为之前在媒体库中导入的 Character 和 Battler 素材，如图 7-103 所示。

图 7-102

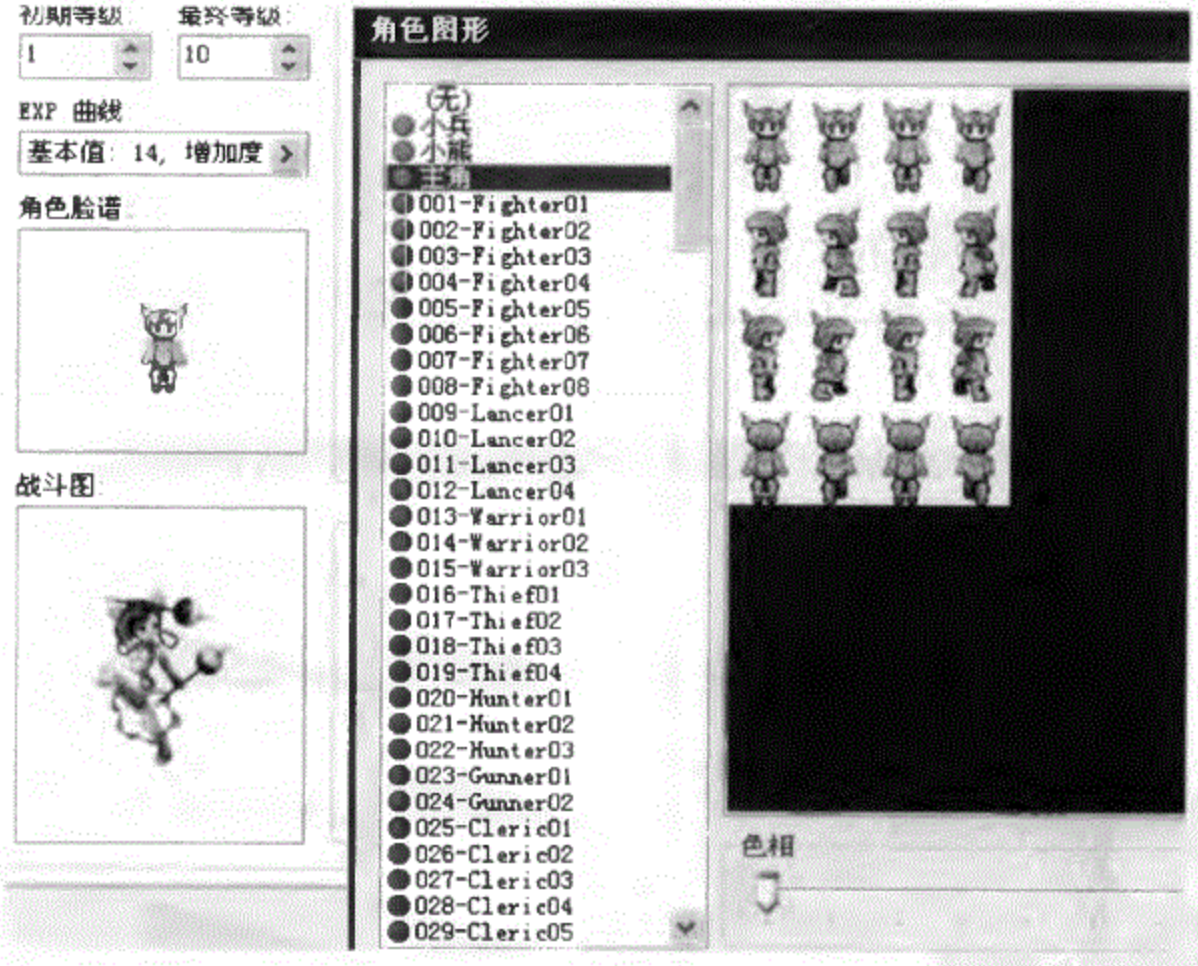


图 7-103

这样，主角若零的设置就已经全部完成了，具体参数最后效果如图 7-104 所示。

2）设置小兵

接下来设置小兵，该小兵为第一次战斗之后加入的角色，职业设定为战士，EXP 曲线和能力值没有做任何修改，按照默认的参数，初期没有给任何装备，设置结果如图 7-105 所示。

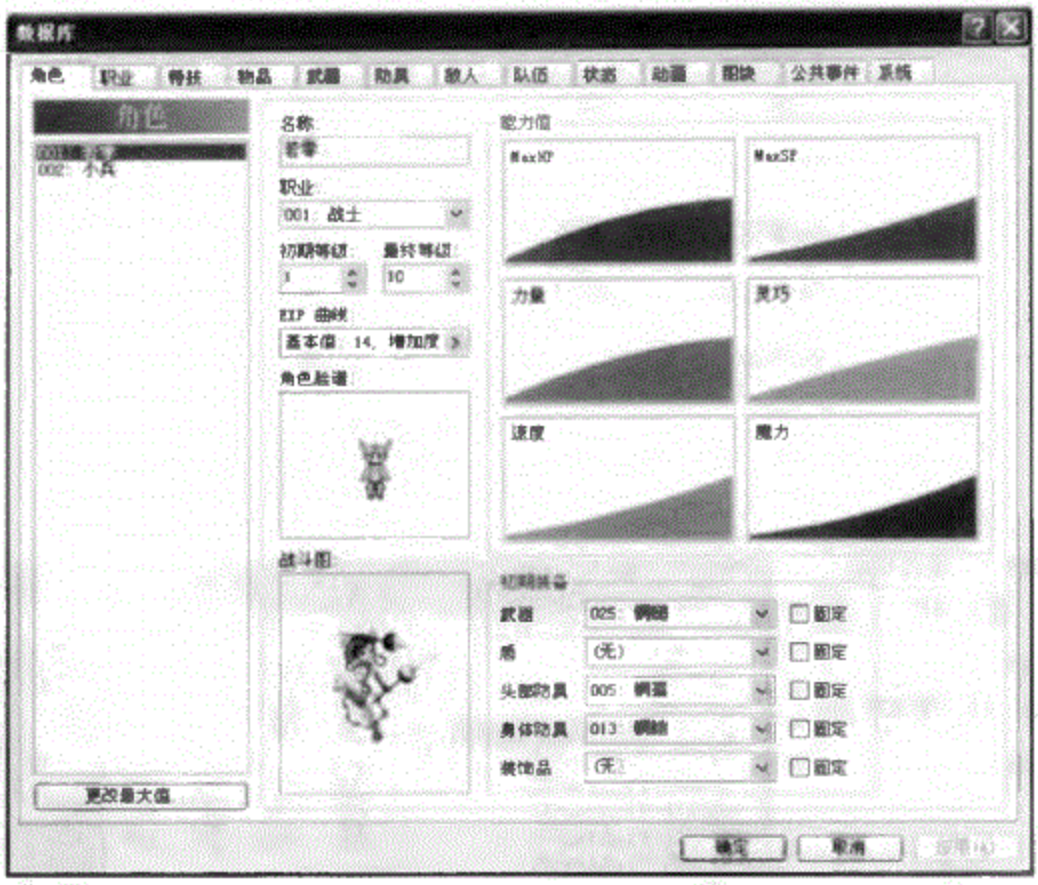

数据库
角色
职业
特技
物品
武器
防具
敌人
队伍
状态
动画
图块
公共事件
系统
角色
001: 若零
002: 小兵
名称
若零
职业
001: 战士
初期等级
1
最终等级
10
EXP 曲线
基本值 14, 增加度
角色脸谱
战斗图
能力值
MaxHP
MaxSP
力量
灵巧
速度
魔力
初期装备
武器
025: 钢剑
盾
(无)
头部防具
005: 钢盔
身体防具
013: 钢铠
装饰品
(无)
固定
更改最大值
确定
取消

图 7-104

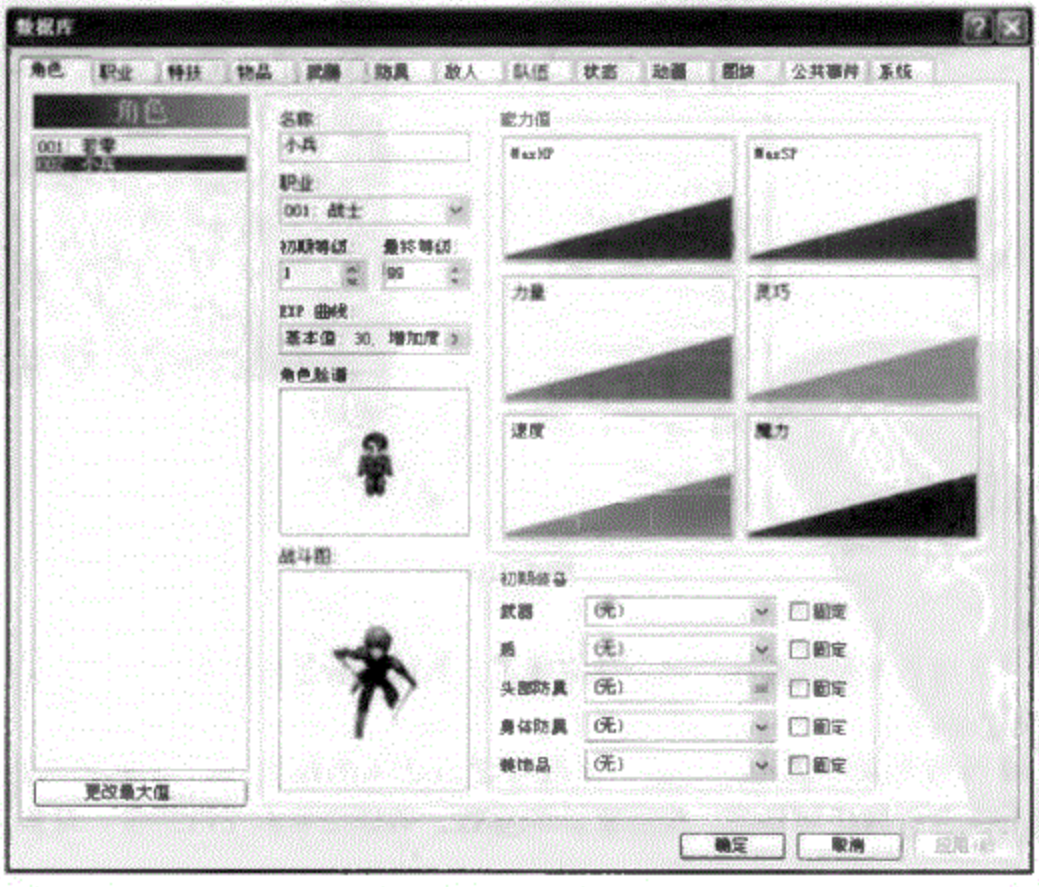

数据库
角色
职业
特技
物品
武器
防具
敌人
队伍
状态
动画
图块
公共事件
系统
角色
001: 若零
002: 小兵
名称
小兵
职业
001: 战士
初期等级
1
最终等级
99
EXP 曲线
基本值 30, 增加度
角色脸谱
战斗图
能力值
MaxHP
MaxSP
力量
灵巧
速度
魔力
初期装备
武器
(无)
盾
(无)
头部防具
(无)
身体防具
(无)
装饰品
(无)
固定
更改最大值
确定
取消

图 7-105

（2）设置敌人

1）设置敌兵

角色设置全部完成后，接下来要设置敌人了。切换到“数据库”对话框中的“敌人”选项卡，如图 7-107 所示，在“敌人”选项卡中不需要对地图中的角色脸谱进行设置，只要导入角色的战斗图便可以了。

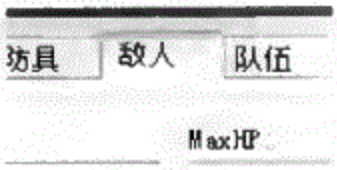


图 7-106

和之前设置角色的时候一样，在设置敌人的时候同样要先把系统中默认的一些敌人角色删除，然后通过“更改最大值”按钮来添加制作的角色。首先创建敌兵角色，命名为“士兵”，在这里 EXP 和金钱是打倒他之后主角获得的经验值和金钱，敌人的 HP、力量以及技能等选项如图 7-107 所示。

图 7-107

2）设置 BOSS

敌兵设置完成，添加第二个敌人角色，这次添加的是游戏最终的 BOSS，将名称改为“魔熊”。由于是 BOSS，所以要给他增加一些战斗时的特效，在“攻击方的动画：”下拉列表框中选择咏唱魔法，“对像方的动画：”下拉列表框中选择打击。EXP 和金钱比较高，分别设置为 100 和 2 000。HP、力量以及技能等具体设置如图 7-108 所示。

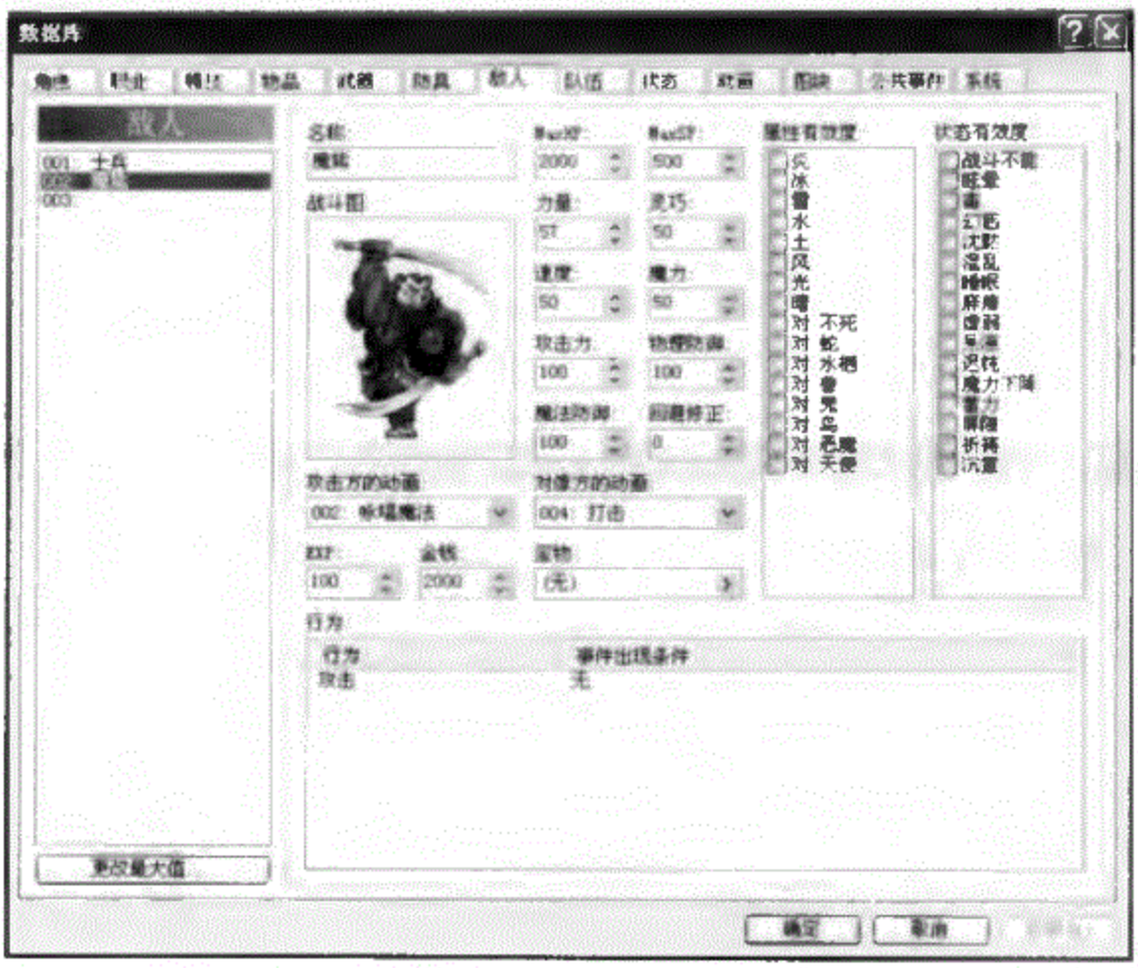


图 7-108

（3）队伍设置

敌人设置完成后，战斗时遇到的敌人都需要被编入队伍，切换到“队伍”选项卡。如同之前设置角色与队伍一样，删除系统默认的敌人队伍，并且使用“更改最大值”按钮将队伍最大值改为 4，分别添加 4 个队伍。

首先添加“士兵 *2”，从事先设置好的“敌人”选项中使用“添加”按钮将两个士兵添加到队伍中，如图 7-109 所示，添加完之后可以使用“战斗测试”按钮进行与该小队的游戏战斗测试。

为丰富游戏，可再添加一组“士兵 *3”，如图 7-110 所示。

轮到 BOSS 了，添加队伍“魔熊”，将敌人“魔熊”添加到新队伍中，如图 7-111 所示。

最后添加一支 BOSS 与士兵混合的队伍，这样可以为提高游戏难度做准备，如图 7-112 所示。


数据库
角色
职业
特技
物品
武器
防具
敌人
队伍
状态
动画
图块
公共事件
系统
队伍
001: 士兵*2
002: 士兵*3
003: 魔能
004: 士兵*2, 魔能
名称:
士兵*2
自动生成名称
[ED] 战斗背景
战斗测试
全部删除
整列
< 添加
001: 士兵*2
002: 魔能
003:
战斗事件:
1
新建事件页
复制事件页
清除事件页
条件:
不执行
距离:
战斗
更改最大值
确定
取消
应用(A)

图 7-109


数据库
角色
职业
特技
物品
武器
防具
敌人
队伍
状态
动画
图块
公共事件
系统
队伍
001: 士兵*2
003: 魔能
004: 士兵*2, 魔能
名称:
士兵*3
自动生成名称
[ED] 战斗背景
战斗测试
全部删除
整列
< 添加
001: 士兵*3
002: 魔能
003:
战斗事件:
1
新建事件页
复制事件页
清除事件页
条件:
不执行
距离:
战斗
更改最大值
确定
取消
应用(A)

图 7-110


数据库
角色 职业 特技 物品 武器 防具 敌人 队伍 状态 动画 图块 公共事件 系统
队伍
001: 士兵*2
002: 士兵*3
003:
004: 士兵*2, 魔能
名称
魔能
自动生成名称
[ID] 战斗背景
战斗测试
全部删除
整列
< 添加
001: 士兵
003:
战斗事件
1
新建事件页
复制事件页
粘贴事件页
清除事件页
事件 不执行
距离 战斗
更改最大值
确定
取消
应用(A)

图 7-111


数据库
角色 职业 特技 物品 武器 防具 敌人 队伍 状态 动画 图块 公共事件 系统
队伍
001: 士兵*2
002: 士兵*3
003: 魔能
名称
士兵*2, 魔能
自动生成名称
[ID] 战斗背景
战斗测试
全部删除
整列
< 添加
001: 士兵
003:
战斗事件
1
新建事件页
复制事件页
粘贴事件页
清除事件页
事件 不执行
距离 战斗
更改最大值
确定
取消
应用(A)

图 7-112

7.5 设置游戏事件

本节开始制作游戏事件，事件也就是游戏的程序模块。如果把游戏比做一个人，那么游戏中的美术部分就是这个人的躯体，而程序部分就是他的灵魂，游戏与玩家之间的交互都是由它来完成的。RPG Maker 中的事件设置并不复杂，所以游戏美术制作人员完全可以通过它来制作简单的 RPG 游戏。

1. 设置公共事件

战斗事件和公共事件都是在数据库中进行设置的，在这里主要使用公共事件，设置好的公共事件可以被之后的地图事件调用。操作步骤为，首先在“数据库”对话框中切换到“公共事件”选项卡，如图 7-113 所示。

图 7-113

通过单击“更改最大值”按钮新建一个名为“普通对话”的事件，在“执行内容”文本框中单击鼠标右键，在弹出的快捷菜单中选择“插入”命令，来插入一个事件，如图 7-114 所示。

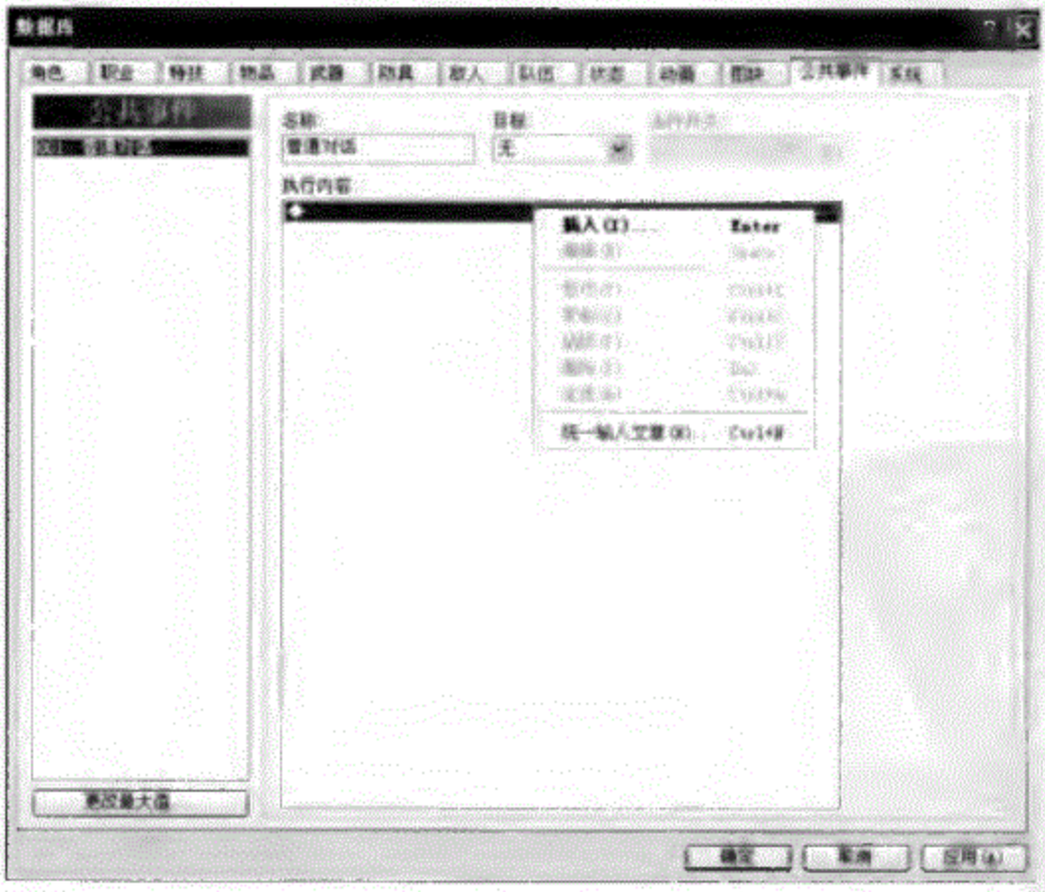


图 7-114

图 7-115

在“事件指令”对话框的“1”选项卡中单击“显示文章”按钮，如图 7-115 所示。

在“执行内容”文本框中输入“文章：去城里看看……”，如图 7-116 所示，设置好之后这条公共事件就能够被其他事件多次调用了，例如地图上的 NPC 角色都可以使用。

2. 设置地图事件

（1）场景—事件设置

1）设置 NPC 事件

公共事件设置完成后，要进入到地图添加地图事件了，把“数据库”对话框关闭，切换到层按钮边的“事件”按钮 ，在地图中选择一处位置放置地图上走动的 NPC，选择好位置之后双击，如图 7-117 所示，弹出“制作事件 -ID:005”对话框，如图 7-118 所示，

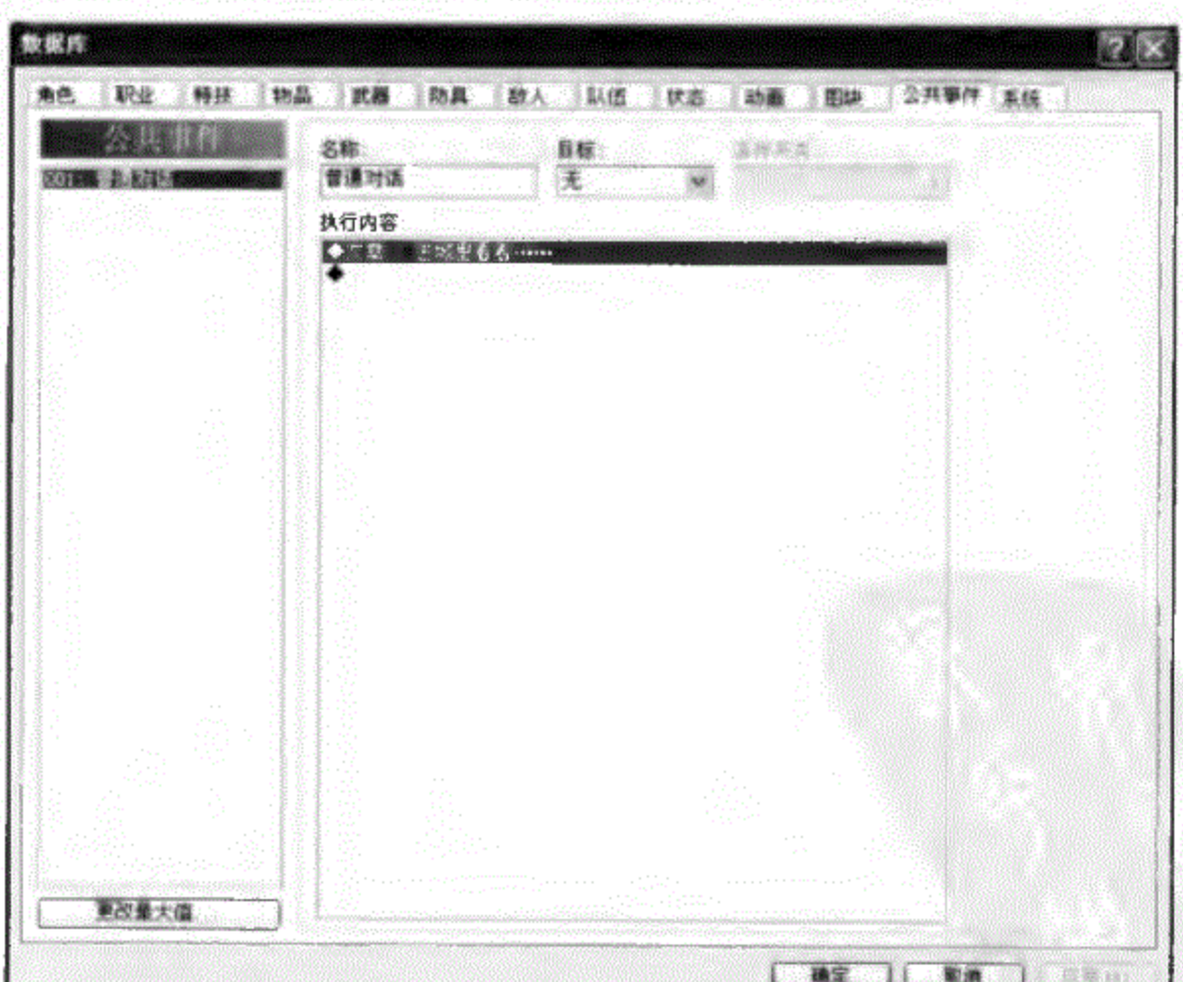


图 7-116

图 7-117

图 7-118

在此添加事件。

既然事件是角色，就该给它一张角色的图片，在“角色图片”下的空白处双击，从图片素材中选择一个角色作为 NPC 的地图形象，如图 7-119 所示。

在“事件开始条件”选项组中选择“与主角接触”单选按钮，如图 7-120 所示，这样玩家在移动角色到该 NPC 面前时便会触发事件了。

图 7-119

图 7-120

图 7-121

与添加公共事件时的步骤相同，选中“执行内容”文本框中的空白处，如图 7-121 所示，单击鼠标右键插入事件。

在“事件指令”对话框中选择“更改文章选项”按钮，如图 7-122 所示，在输入文章之前使用这个命令，可以调整游戏中文章框所在的位置。

在弹出的“更改文章选项”对话框中将文章的“显示位置”改为“中”，在“窗口显示”选项组中选中“显示”单选按钮，如图 7-123 所示，最后单击“确定”按钮。

再插入一个事件，这里可以用到之前在数据库中设置过的公共事件，之前的公共事件使用了显示文章，它可以被多个事件调用，这里 NPC 的角色如果要调用的话就要在“事件指令”对话框中单击“公共事件”按钮，如图 7-124 所示。

图 7-122

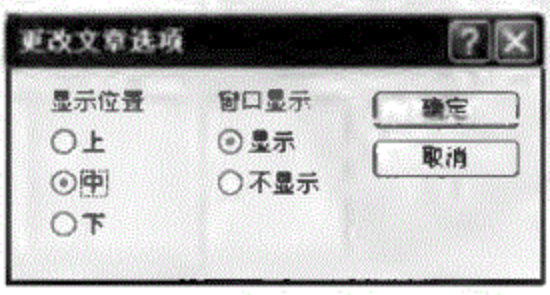


图 7-123

在弹出的“公共事件”对话框中选择之前便设置好的“001:普通对话”，如图 7-125 所示，然后单击“确定”按钮。

这样，在执行内容中“公共事件:普通对话”便被调用进来了，并且它显示为蓝色，如图 7-126 所示，这说明该事件为公共事件。

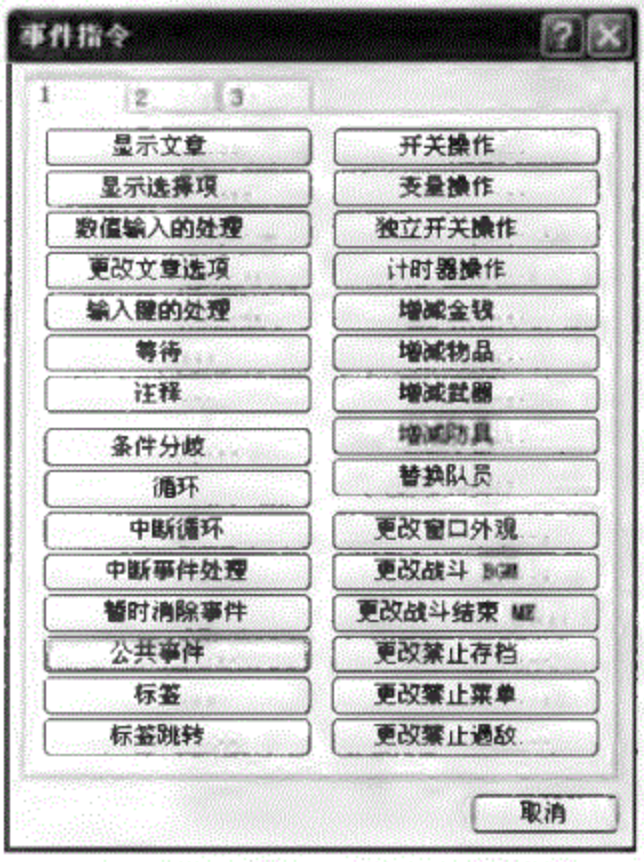


图 7-124

图 7-125

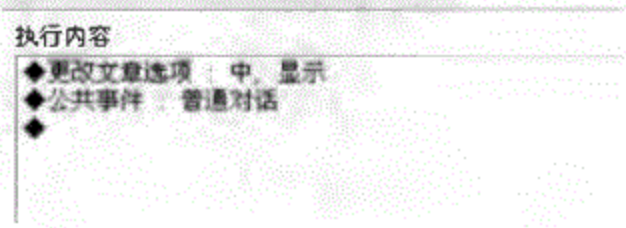


图 7-126

最后在地图上显示的结果，如图 7-127 所示，只是事件的图标，在游戏运行的时候，在地图上便能显示出完整的角色了。

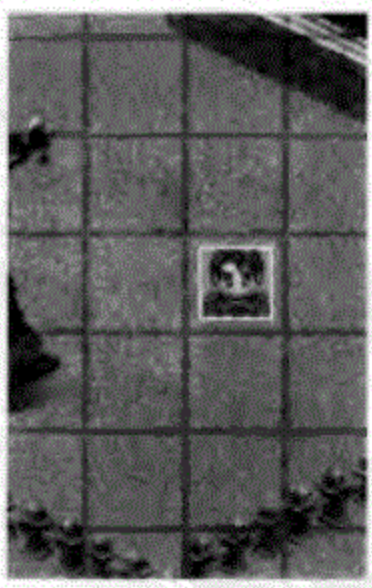


图 7-127

2）主导故事主线的 NPC

要在地图的城楼前创建第二个 NPC 角色，如图 7-128 所示，该角色是委托给主角任务的角色，所以这个事件的设置比起之前的事件会更复杂些，接下来逐步讲解。

基本的设置和之前大体相同，首先选择角色图片，在“事件开始条件”选项组中选择“与主角接触”单选按钮，但是在“移动规则”选项组中的移动“类型”下拉列表框中选择“接近”，这样当玩家控制主角进入游戏地图之后该事件便会主动靠近玩家与其进行对话，而“速度”和“频度”可以调节该事件的移动速度，如图 7-129 所示。

在“执行内容”文本框中加入事件指令，如图 7-130 所示，单击鼠标右键，在弹出

图 7-128

图 7-129

图 7-130

的快捷菜单中选择“插入”命令，在弹出的对话框中单击“更改文章选项”按钮。

和之前的事件相同，将文章的“显示位置”调整为“中”，“窗口显示”设置为“显示”，如图 7-131 所示，最后单击“确定”按钮。

在“执行内容”文本框的“更改文章选项:中,显示”的下方再加入一条,如图 7-132 所示。

图 7-131

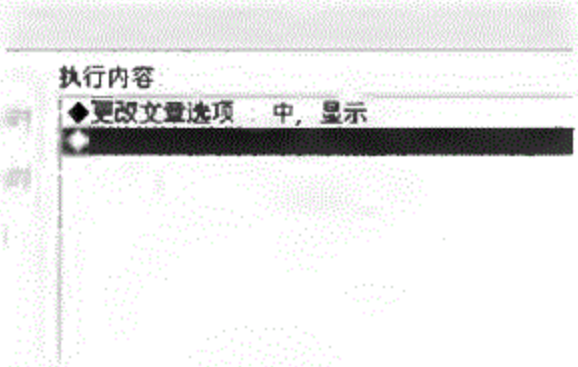


图 7-132

这里需要说明的是：显示文章之前如果需要加入说话的人的名称，可以在最开头处输入角色名称，然后再输入该角色所说的话；也可以用另外一种方法，由于在之前的数据库中已经指定了游戏的角色，所以可以用符号来代替角色的名称，例如定义的角色 1 为若零,角色 2 为小兵,那么在 RPG Maker 中,就可以用“\N［1］”代表若零,用“\N［2］”代表小兵，那么该显示文章的最终结果如图 7-133 所示。

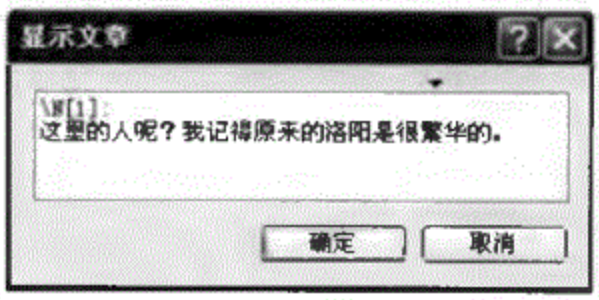


图 7-133

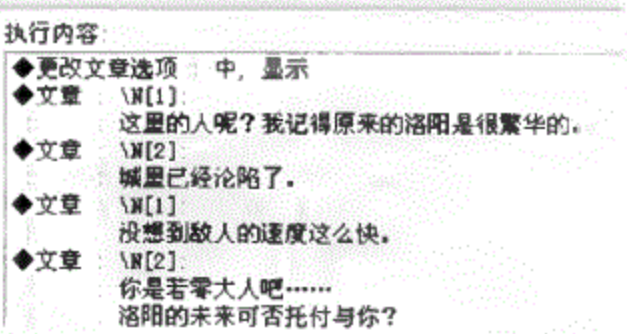


图 7-134

输入几段若零与小兵之间对话的显示文章事件，如图 7-134 所示。

以上对话中我方士兵已经向主角提出了拯救洛阳城的邀请，所以接下来就需要加入选项事件从而能够选择接受或不接受任务委托，在文章事件的下方插入事件指令内容，如图 7-135 所示。

在弹出的“事件指令”对话框中单击“显示选择项”按钮,如图 7-136 所示。

在弹出的“显示选择项”对话框中

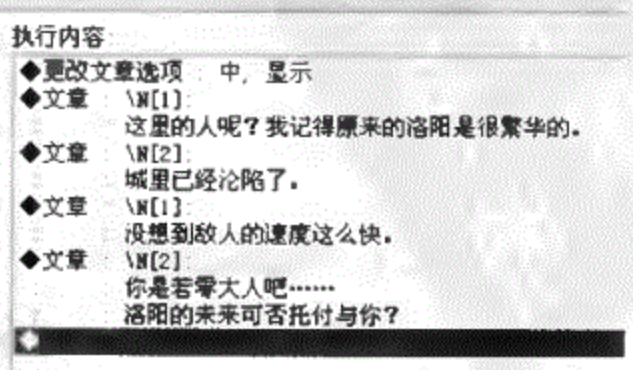


图 7-135

可以输入 4 个选择项，在这里只要输入前两个就可以了，“选择项 1”文本框中输入“接受”，“选择项 2”文本框中输入“不接受”，然后设置一个取消选项的命令，在“取消的场合”选项组中单击“选择项 2”单选按钮，如图 7-137 所示。这样在游戏进行的过程中，若选择了“不接受”选项，便会退出选择对话框了。

图 7-136

图 7-137

已添加的“显示选择项”事件指令，如图 7-138 所示，拥有两个可以添加事件指令的选择项，分别是“[接受] 的场合”和“[不接受] 的场合”。选择“接受的场合”便可以在接受 NPC 委托之后添加事件指令，如图 7-139 所示。

图 7-138

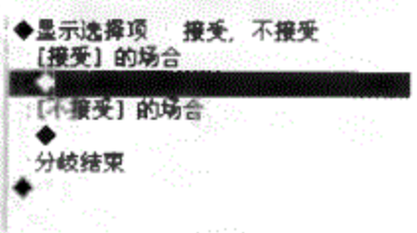


图 7-139

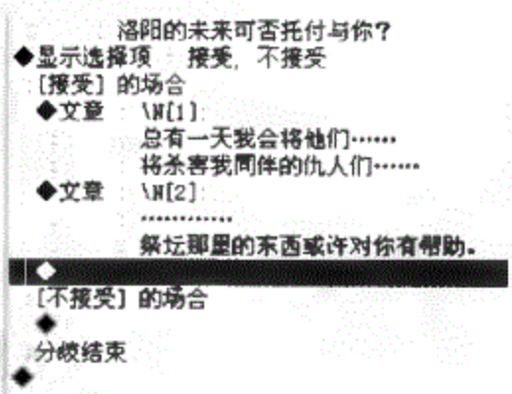


图 7-140

在接受的场合中先插入两段对话，交代一下剧情，如图 7-140 所示。

为了更好地交代剧情，可以将画面缓慢移动到剧情将要发生的地方并且移动到角色的身上，添加“画面卷动”指令如图 7-141 所示。在弹出的“画面的卷动”对话框中进行设置，将“方向”和“距离”分别设置成“上”和“7”，“速度”设置为“3: Slow”，如图 7-142 所示。

在向上的卷动画面设置完之后再设置一个向下的画面卷动。

图 7-141

图 7-142

仍然在“接受的场合”中添加新的事件指令，如图 7-143 所示。

添加一个“开关操作”，如图 7-144 所示。

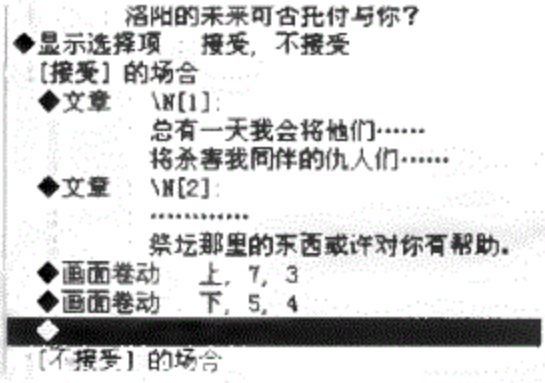


图 7-143

图 7-144

在这里简单介绍一下开关的用法，开关类似于程序中的逻辑变量，有 ON/OFF（打开 / 关闭）两种状态，一般 RPG Maker 中默认的开关状态为 OFF，在游戏中它可以起到前因后果的作用，例如，进入游戏下一关卡大门时需要得到钥匙才能够进入，这时候设置一个名为“钥匙”的开关，这个开关的默认值是 OFF，在角色拿到钥匙之后把开关打开则“钥匙”开关改为 ON，这样开门的时候做一个判断，如果“钥匙”开关为 ON，那么便可以切换场景。对于整个游戏剧情来讲，开关起着至关重要的作用。

继续之前的步骤，“开关操作”对话框中选中“单独”单选按钮，再在相应的下拉列表框中选择 0001 号开关，将该开关命名为“接受委托”，并且把操作内容改成“ON”（打开），最后单击“确定”按钮，如图 7-145 所示。

如果选择不接受任务，那么就退出选择事件，这在之前设置“显示选择项”时已经设置过了，现在退出时再设置一下文章的显示，在“[不接受]的场合”中插入“显示文章”事件指令，如图 7-146 所示。

第二个 NPC 事件设置完成后，最终效果如图 7-147 所示，可以看到这次设置的事件比起之前更加复杂了，但是只有复杂的事件才能让游戏变得互动性更强。

图 7-145

图 7-146

图 7-147

3）设置水晶和敌兵

敌兵事件的设置。在地图上找到神坛，这里是下一处剧情发生的地方，在接受委托并将“接受委托”开关改成 ON 之后在这个位置便会出现一颗水晶和一个敌兵，首先添加敌兵，在入口台阶处添加事件如图 7-148 所示。

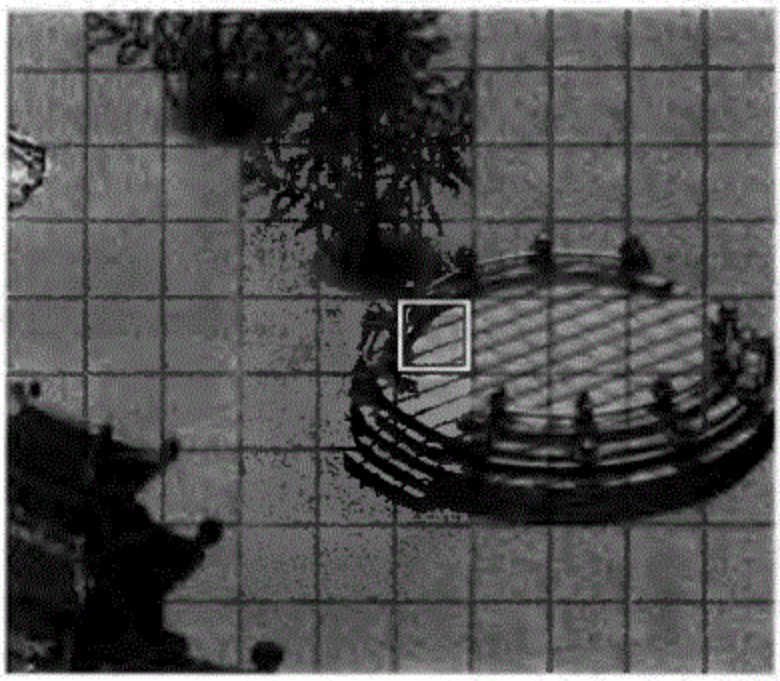


图 7-148

之前的事件没有设置事件出现条件，因为游戏开始之后那些事件都是无条件执行的，但是这里的敌兵由于要在接受任务委托之后才会出现，所以必须要设置“事件出现条件”，事件出现条件通常有开关的判断和变量的判断，在主角若零接受委托之后“接受委托”开关被改成 ON，所以敌兵在这里的出现条件就是“接受委托”为 ON，如图 7-149 所示，如果没有接受委托，那么这个敌兵的事件便不会在地图上出现。

“角色图片”、“移动规则”和“事件开始条件”，如图 7-150 所示。

图 7-149

图 7-150

还是和之前设置的事件时一样，首先“更改文章选项”，让文章显示框显示在画面中央，然后输入两段敌兵与主角的对话，如图 7-151 所示。

简单的对话之后便要开始战斗了，在文章指令的后方插入事件指令，如图 7-152 所示。

图 7-151

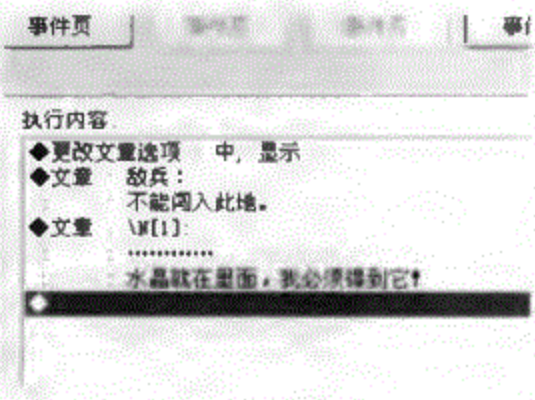


图 7-152

在“事件指令”对话框中选择“战斗处理”按钮，如图 7-153 所示。

选择之后会弹出“战斗处理”对话框，选择对应的敌人队伍（这在之前的数据库的队伍中已经设置过），选择完之后单击“确定”按钮，如图 7-154 所示。

图 7-153

图 7-154

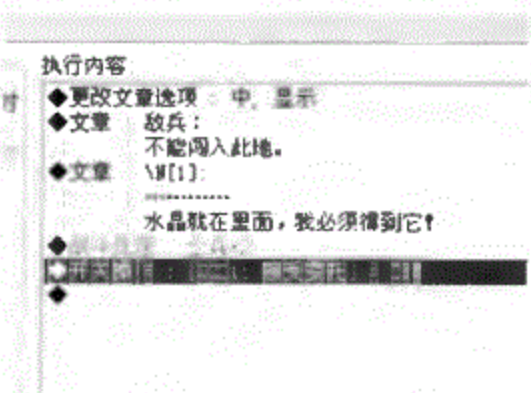


图 7-155

战斗之后可以再插入一个“开关操作”，将“接受委托”开关关闭，如图 7-155 所示，这样战斗结束之后敌兵就消失了。

水晶事件的设置。在敌兵事件的边上添加事件，如图 7-156 所示，该事件作为即将被敌兵抢走的水晶。

图片选择 RPG Maker 里的水晶图片，事件移动类型设为“固定”，在“事件开始条件”选项组中选择“与主角接触”单选按钮，如图 7-157 所示。

图 7-156

图 7-157

执行内容先显示两段文章，同时通过文章提示之前被击败的敌兵可以要求他加入。再插入一个开关指令，定义开关“得到水晶”，并将其改成 ON，如图 7-158 所示。

添加敌兵事件页。由于在之前水晶设置中提到了可以要求看守水晶的敌兵加入队伍，所以玩家需要再次回到敌兵的面前和其对话，由于在击败敌兵之后他已经消失了，要再次让他出现，就得继续对敌兵进行事件设置，选择敌兵事件，如图 7-159 所示。

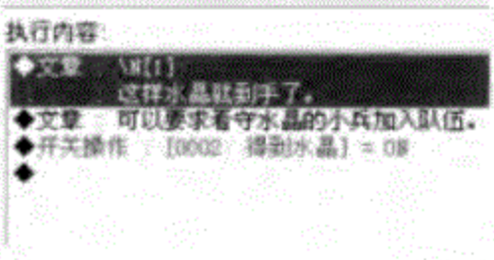


图 7-158

图 7-159

原本设置好的事件无需进行改动，所要做的就是新建一个事件页，选择“编辑事件”对话框上方的 按钮，创建第二个事件页。在新的事件页中设置事件的出现条件，选中得到水晶“开关”复选框，如图 7-160 所示。在上面中设置水晶的时候，在得到水晶之后将水晶的开关改成了 ON，所以即使敌兵在被击败之后消失了，但在“得到水晶”开关开启之后他又能够出现。

执行内容加入“文章”和“显示选择项”用来选择是否接受加入，选项内容分别为“可以”和“不用了”，如图 7-161 所示。在“可以”的场合中插入事件指令。

图 7-160

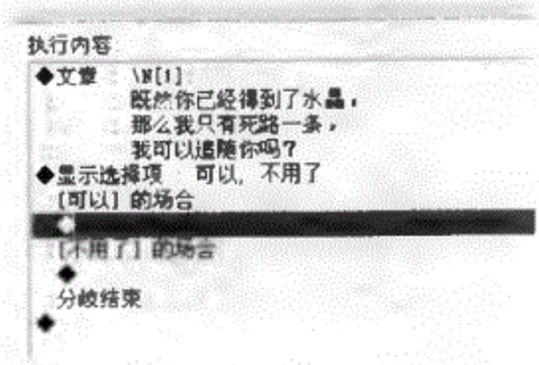


图 7-161

单击“事件指令”对话框中的“替换队员”按钮，如图 7-162 所示。

在“队员的替换”对话框中选择之前就在数据库中设置好的小兵作为要替换的角色，在“操作”选项组中指出游戏中加入角色或是从队伍中已有的角色中移除角色，这里选中“加入”单选按钮，如图 7-163 所示。

图 7-162

图 7-163

图 7-164

加入之后的士兵已经能够在战斗中受玩家的控制了，但是在地图上依然无法看到，如果要让他在地图上也跟随玩家一起行动，就必须再插入事件指令，如图 7-164 所示。

在弹出的“事件指令”对话框中选择“2”选项卡，然后选择“设置移动路线”按钮，如图 7-165 所示。

在弹出的“移动路线”对话框中首先选择“本事件”，如图 7-166 所示，如果选择了“角色”或是其他事件，那么事件移动就不会在该事件上发生了。

图 7-165

图 7-166

移动路线中有各个方向的选择，单击 靠近主角 按钮，使该事件靠近主角移动，选中“重复动作”复选框，如图 7-167 所示。

士兵加入完成，由于接下去就要进入第二张地图中进行战斗了，所以最好在这个时候保存一下游戏，所以添加事件指令“呼叫存档画面”，如图 7-168 所示。

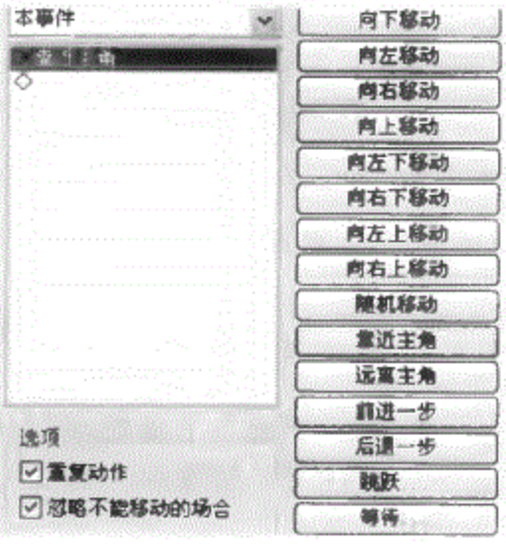


图 7-167

图 7-168

存档可以加在“显示选择项”中，也可以加在“分歧结束”之后，如图 7-169 所示。

这样，在游戏进行中便会弹出保存提示，如图 7-170 所示，保存完之后，玩家就能够放心地挑战下一个场景了。

保存之后需要交代一下剧情，添加文章指令，告诉玩家要向上走才能进入到下一个场景，并且通过“画面卷动”

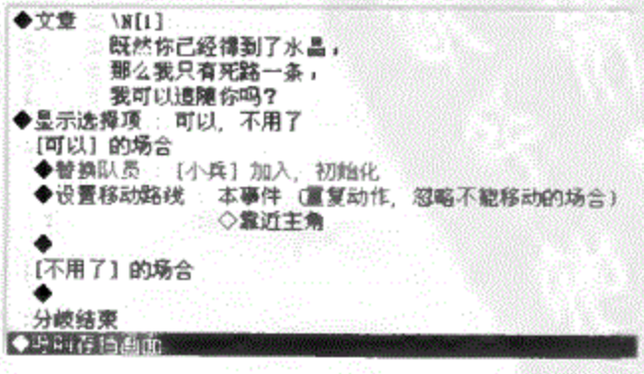


图 7-169

指令将画面移动到本场景的出口，使用“等待”指令使画面在出口处停留 20 帧时间再移动回来，如图 7-171 所示。

图 7-170

图 7-171

设置地图切换事件。

设置出口，在门外的出口处添加事件，如图 7-172 所示。

给该事件设置一个事件出现条件，由于该事件必须在击败敌兵获得水晶之后才能被触发，所以选择得到水晶开关为 ON 时，如图 7-173 所示。

图 7-172

图 7-173

该事件为场景切换事件，在“事件指令”对话框的“2”选项卡中单击“场所移动”按钮，如图 7-174 所示。

单击之后弹出“场所移动”对话框，在“朝向”和“淡入淡出”下拉列表框中选择默认的设置，选中“直接指定”单选按钮，然后选择指定的场景，如图 7-175 所示。

图 7-174

图 7-175

在弹出的“场所”对话框中选择第二张“城内”地图，并在地图中选择一个入口位置，以定位角色进入场景之后的位置，这里将它定在地图的左下角，如图 7-176 所示。

这样，场所移动的执行内容就添加完成了，如图 7-177 所示。

这样，第一个场景的全部事件便添加完成了。

（2）场景二事件设置

回到地图目录中，切换到第二张名为“城内”的地图场景，如图 7-178 所示。

打开“城内”场景，如图 7-179 所示，在设置地图事件之前还是先交代一下游戏流程：

图 7-176

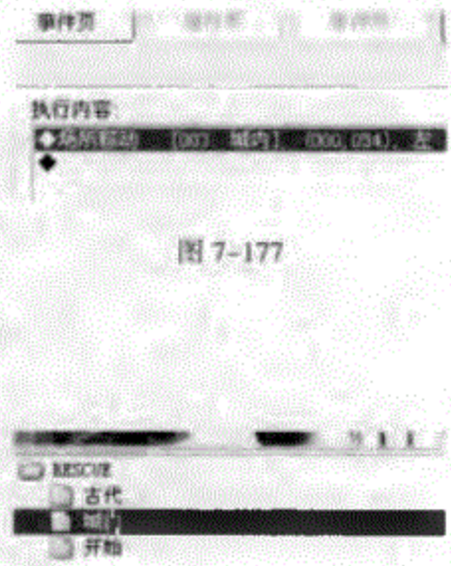


图 7-177

图 7-178

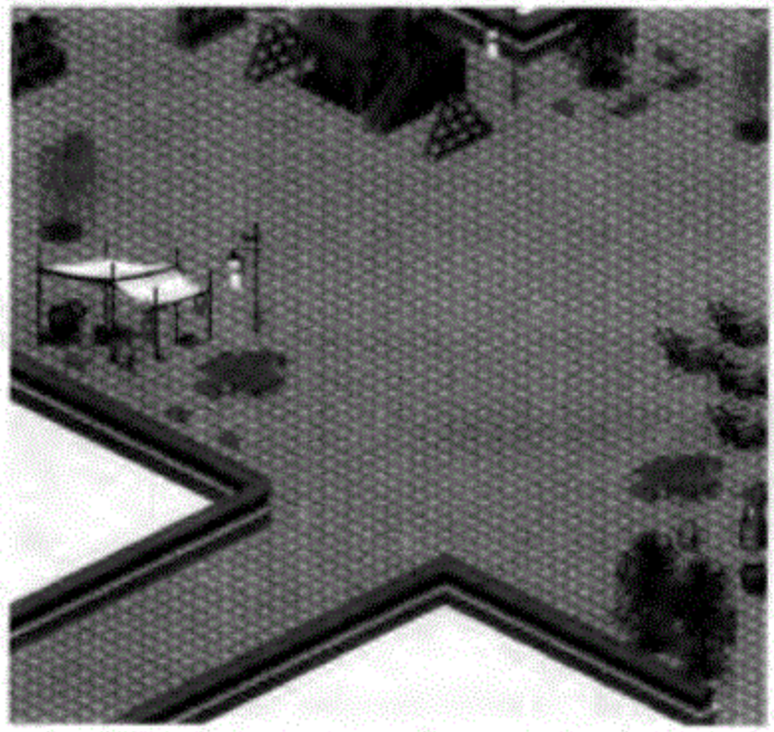


图 7-179

在这个地图中有个客栈，门口需要放置一个商人，游戏中玩家可以使用击败敌兵所得到的金钱到商人这边购买药品和装备，另外在地图上安排了3队敌兵，将他们击败之后就可以和BOSS战斗了。

接下来就以这个游戏流程为思路来建立事件。

1）设置商人事件

首先在地图中找到客栈，在附近的空地添加一个事件，如图7-180所示。

给角色选择一张图片（由于之前绘制的素材中没有商人的图片，这里暂且使用RPG Maker中自带的角色代替一下），移动“类型”设为“固定”，在“事件开始条件”选项组中选中“与主角接触”单选按钮，如图7-181所示。

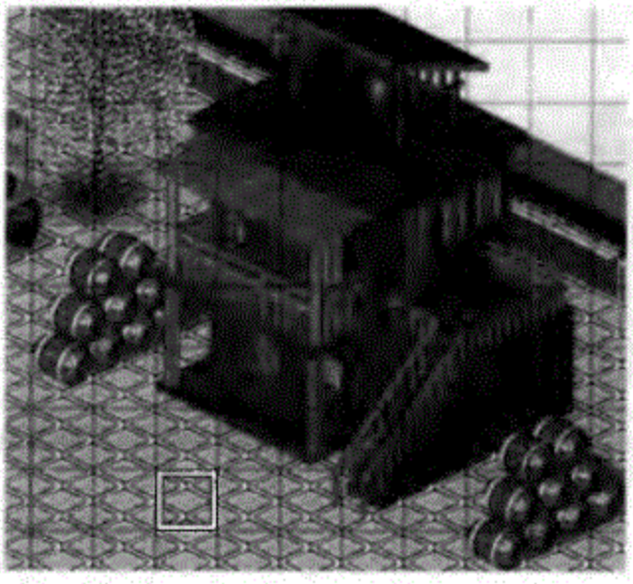


图7-180

图7-181

在“执行内容”文本框中先添加更改文章选项，把主角和商人对话的对话框调整到画面的上方显示，添加“文章”事件指令，简单交代一下剧情，如图7-182所示。

要在商人这里购买商品，就必须要添加商店事件，在“事件指令”对话框的“3”选项卡中单击“商店处理”按钮，如图7-183所示。

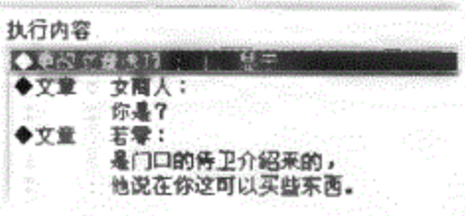


图7-182

图7-183

可以在弹出的“商店处理”对话框中添加商品，商品有“物品”、“武器”和“防具”3种选择，如图7-184所示，玩家可以根据需要将物品添加到商店事件中。

目前先选择 8 种商品，分别为“回复剂”、“完全回复剂”、“香水”、“高级香水”、“力量之石”、“生命之种”、“钢槌”和“钢盾”，如图 7-185 所示，其中回复剂和香水用于游戏中。

图 7-184

图 7-185

在“商店处理”之后再添加两段显示文章的事件指令给玩家以简单的提示，如图 7-186 所示。

2）多个敌兵事件的设置

商人事件处理完毕所，在场景中再添加一些可以看见的敌人士兵，在需要添加的位置添加事件，在设置时首先要给他们一个出现状态，在“事件出现条件”处打开开关，将“士兵 2 战斗状态”打开（这里暂且指定这名士兵为士兵 2，如果把该士兵制作完成并复制成另一个事件的话，就必须要将下一个士兵的开关设为“士兵 3 战斗状态”）。

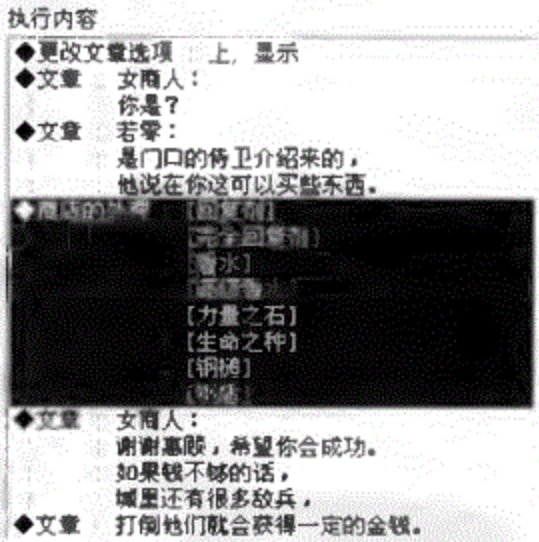


图 7-186

移动规则“类型”为“随机”，“事件开始条件”选择为“与主角接触”如图 7-187 所示。事件执行内容依次为“更改文章选项”→“显示文章”→“战斗处理”→“开关操作”，需要说明的是，这里的战斗处理选择“士兵 *3”的队伍，最后一步开关操作，将“士兵 2 战斗状态”关闭，这样之前设置的“事件出现条件”便不成立了，士兵在被击败后便会消失。

由于游戏在消灭 3 队敌兵之后便能够与 BOSS 进行战斗，所以这里要对击败的敌兵数量进行统计。一般在游戏中统计数据需要用到变量，变量的功能和开关类似，都可以通过改变状态来做为判定条件，区别在于开关只有打开和关闭两种状态，而变量则是可变的数值，变量可以有加、减、乘、除等操作。

在执行内容中插入一条事件指令，单击“变量操作”按钮，如图 7-188 所示。

选择一个变量，起名为“击破敌兵数”，在“操作变量”对话框中，将“操作”改成“加法”，如图 7-189 所示，这样，在消灭一队敌兵之后，“击破敌兵数”这个变量就会加一。

图 7-187

图 7-188

图 7-189

添加完变量操作之后，最终效果如图 7-190 所示，变量操作之后，该事件便设置完成了。

由于敌兵事件基本相同，所以既然设置完一个敌兵事件就可以将他复制若干个，为了简单，这里暂且复制 3 个，在地图上选中事件，单击鼠标右键选择“复制”命令，如图 7-191 所示。

图 7-190

图 7-191

在地图上选择适合的位置进行粘贴，如图 7-192 所示，粘贴完成后要修改每个事件的“事件出现条件”。

3）设置最终 BOSS 事件

敌兵设置全部完成，接着设置 BOSS。到画面的最末端，给 BOSS 找一个合适的位置，添加一个事件，如图 7-193 所示。

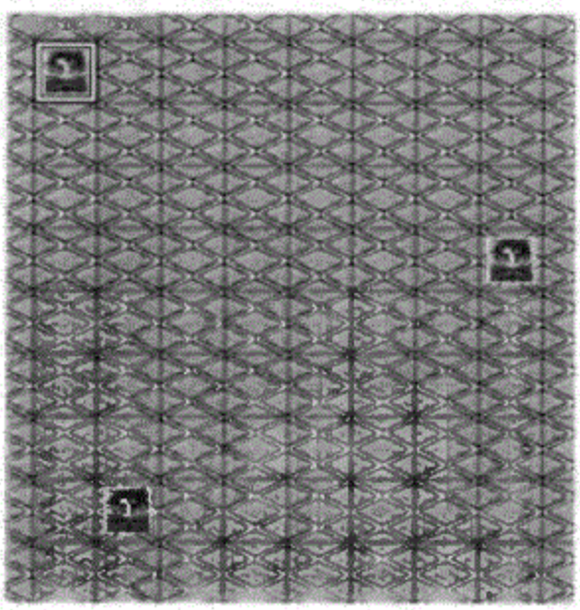


图 7-192

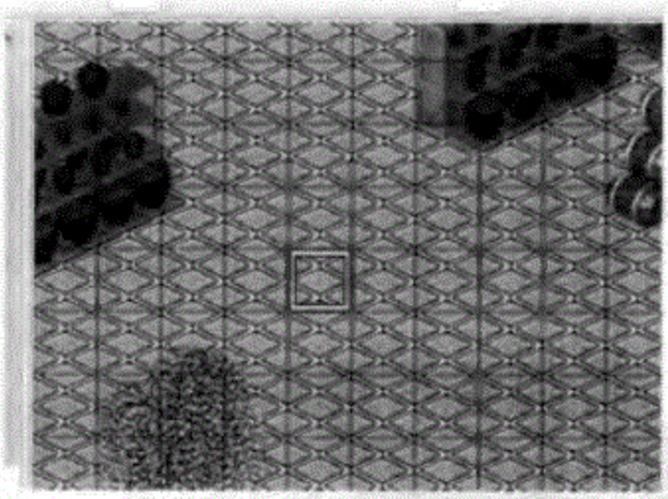


图 7-193

首先就要设置事件的出现条件，之前在设置敌兵的时候最后都会设置一个“击破敌兵数”变量加一的事件指令，那么在击败 3 个敌兵后，“击破敌兵数”的值就会等于 3，所以这里 BOSS 的出现条件就是“击破敌兵数”值大于 3，如图 7-194 所示。

为 BOSS 选择一个地图上显示的图形，选择之前绘制好并且导入 RPG Maker 中的魔熊图片，如图 7-195 所示。

图 7-194

图 7-195

事件执行内容首先插入“更改文章选项”以及 3 段显示“文章”事件，文章具体内容如图 7-196 所示，再添加一个“战斗处理”指令。

战斗处理对象为 BOSS 魔熊，战斗时不能设置逃跑选项。如果战斗失败，还要继续，选中“失败的话继续”复选框，如图 7-197 所示。

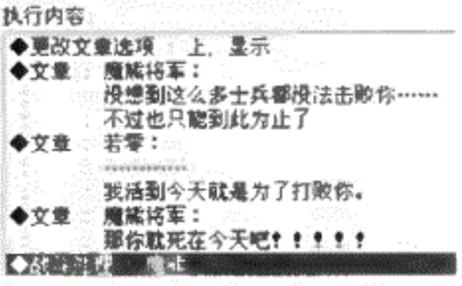


图 7-196

图 7-197

与 BOSS 的战斗战与其他战斗不同，因为它难度很大，所以给人紧张、刺激的感觉，为了烘托战斗时的气氛，战斗的背景音乐最好也能够有所变化，RPG Maker 事件指令也支持音乐的变换。

插入一条事件指令，单击“事件指令”中“2”选项卡中的“演奏 BGM”按钮，如图 7-198 所示。

图 7-198

选择之后会弹出“BGM”对话框，选中音乐后单击对话框中的“播放”按钮预览音乐，为 BOSS 选择一首紧张刺激的背景音乐“011-LastBoss03”，如图 7-199 所示。

单击“确定”按钮之后可以稍做些修改，将刚刚添加的“演奏 BGM”事件指令拖动到“战斗处理”事件的上方，如图 7-200 所示。

之前插入“战斗处理”的时候，由于选择了“失败的话继续”复选框，所以出现了“胜利的场合”和“失败的场合”，如果是在战斗

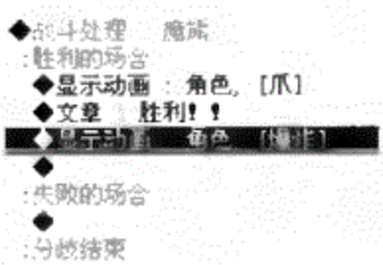


图 7-204

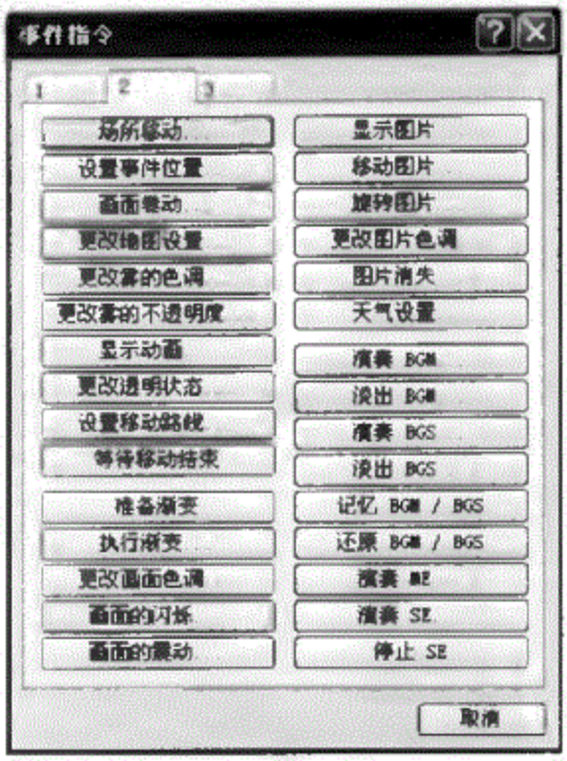


图 7-205

调整画面闪烁的 RGB(红绿蓝)值和强度值，把“时间间隔”设置为 10 帧，如图 7-206 所示。这样游戏进行过程中就能够使画面保持 10 帧左右时间的闪烁。

为了让画面再稍做停留，可以在游戏即将结束之前再增加一个画面等待的事件指令，单击“等待”按钮，如图 7-207 所示。

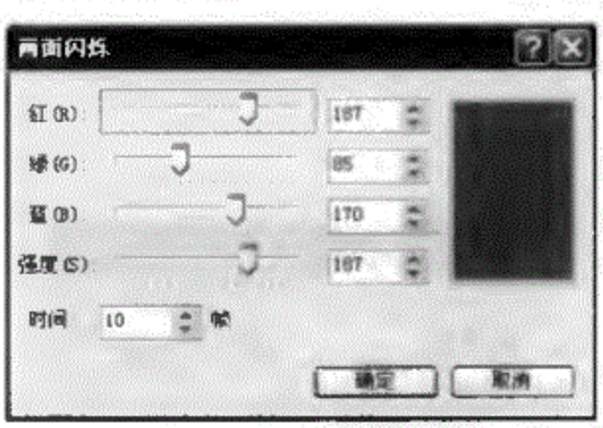


图 7-206

图 7-207

图 7-208

将等待时间改为 20 帧，如图 7-208 所示。

战胜了 BOSS 之后，游戏将跳到结束画面，在“事件指令”对话框中单击“游戏结束”按钮，如图 7-209 所示。

BOSS 事件的最终完整设置如图 7-210 所示。


事件指令
1
2
3
战斗处理
商店处理
名称输入处理
增减 HP
增减 SP
更改状态
完全回复
增减 EXP
增减等级
增减能力值
增减特技
变更装备
更改角色姓名
更改角色职业
更改角色图形
增减敌人 HP
增减敌人 SP
更改敌人状态
敌人完全回复
敌人出现
敌人变身
显示动画
伤害处理
强制行动
战斗中断
呼叫菜单画面
呼叫存档画面
游戏结束
返回标题画面
脚本
取消

图 7-209

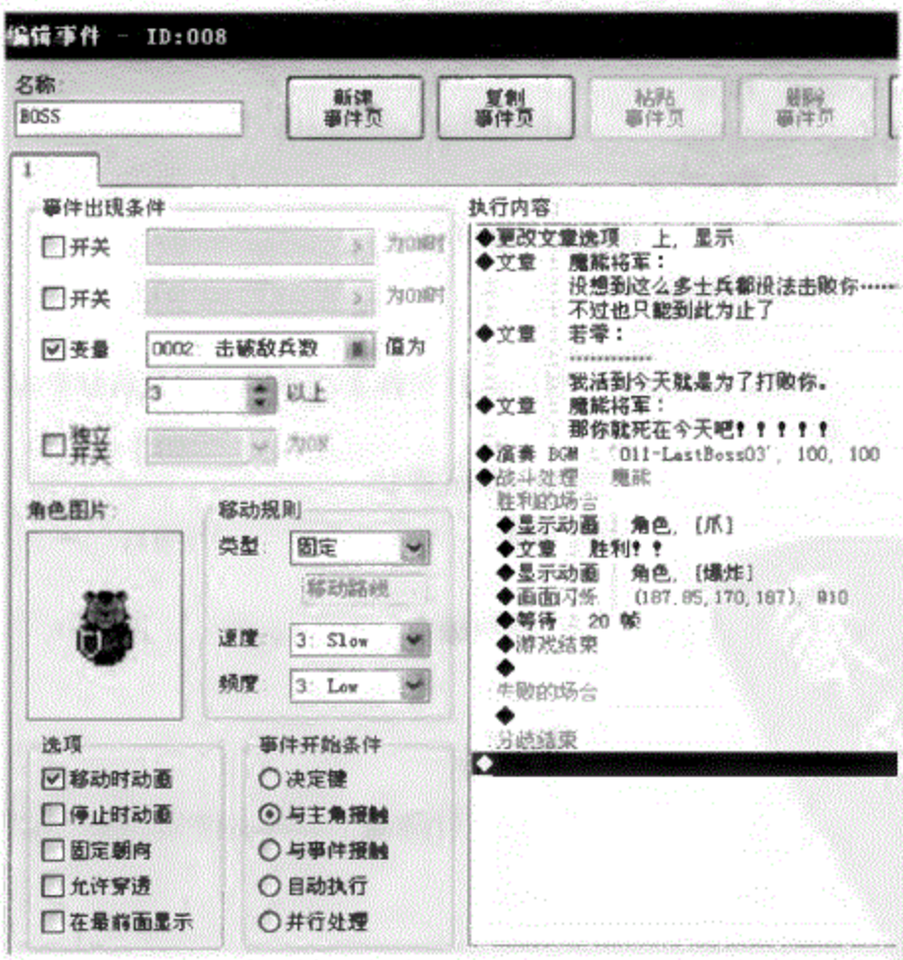

编辑事件 - ID:008
名称
BOSS
新建事件页
复制事件页
粘贴事件页
删除事件页
1
事件出现条件
开关
开关
变量
0002: 击破敌兵数
值为
3
以上
独立开关
角色图片
移动规则
类型
固定
移动路线
速度
3: Slow
频度
3: Low
选项
移动时动画
停止时动画
固定朝向
允许穿透
在最前面显示
事件开始条件
决定键
与主角接触
与事件接触
自动执行
并行处理
执行内容
◆更改文章选项：上，显示
◆文章：魔能将军：
没想到这么多士兵都没法击败你……
不过也只能到此为止了
◆文章：若零：
……
我活到今天就是为了打败你。
◆文章：魔能将军：
那你就死在今天吧！！！！！
◆演奏 BGM：'011-LastBoss03', 100, 100
◆战斗处理：魔能
胜利的场合
◆显示动画：角色，〔爪〕
◆文章：胜利！！
◆显示动画：角色，〔爆炸〕
◆画面闪烁：(187,85,170,187), 910
◆等待：20 帧
◆游戏结束
◆
失败的场合
◆
分歧结束

图 7-210

至此，整个游戏的制作部分已经完成。回顾之前所做的工作，细心的人可以发现整个游戏的制作离不开美术部分和事件设置部分。事件的设置主导了游戏的整个流程，所以说，RPG Maker 中的事件是游戏的主心骨。

7.6 游戏测试

1. 测试要点

测试游戏时需要注意几个要点：

- 场景与角色碰撞是否有效，是否会出现穿透。
- 事件是否有效。
- 游戏流程是否正确。
- 游戏的平衡性。

2. 游戏按键

在运行游戏之后，RPG Maker 的按键控制方式如下：

- Esc/Num 0/X：取消 / 菜单。
- Space/Enter/C：决定。
- Q/PageUp：回到前页。
- W/PageDown：移向后页。
- Alt+Enter：窗口模式和全屏模式的切换。
- Alt+F4：强制结束游戏。
- F12：强制回到标题画面。
- F1：显示游戏属性对话框（可以自行分配游戏手柄和键盘的按钮设定）。
- F2：在标题栏上显示 FPS。
- F9：在移动中按下该键可以打开调试窗口。
- Ctrl：按下该键不放同时移动鼠标，可以使地图图块的通行设置无效，而能在不允许通行的元件上自由移动。

如果使用游戏手柄的话，其方向键和键盘上的方向键（↑ ↓ ← →）功能一致，用来控制角色的移动。

3. 开始测试

现在就按照基本流程，测试游戏。运行游戏，在弹出的提示框中选择“新游戏”，如图 7-211 所示。

游戏流程大致如下：

① 玩家操纵主角开始了游戏的旅程，在城外地图中可以清楚地看到两个 NPC 角色在地图中走动，如图 7-212 所示。

图 7-211

图 7-212

② 与其中一个 NPC 进行对话之后他会要求主角接受拯救洛阳城的任务，如图 7-213 所示，这时主角必须选择“接受”游戏才能继续。

图 7-213

③ 主角接受任务之后地图上方的神坛中便会出现敌兵和水晶，主角上去与敌兵说话，同时会发生战斗，如图 7-214 所示。

图 7-214

④ 主角击败敌兵之后可以得到水晶，这时敌兵会要求加入玩家的队伍，如图 7-215 所示，这时玩家选择“接受”。

图 7-215

⑤ 敌兵加入了战斗队伍，通过上面的门主角已经可以进入到第二个场景了。第二个场景是城内场景，如图 7-216 所示，其中有不少敌兵，主角会和他们进行战斗。

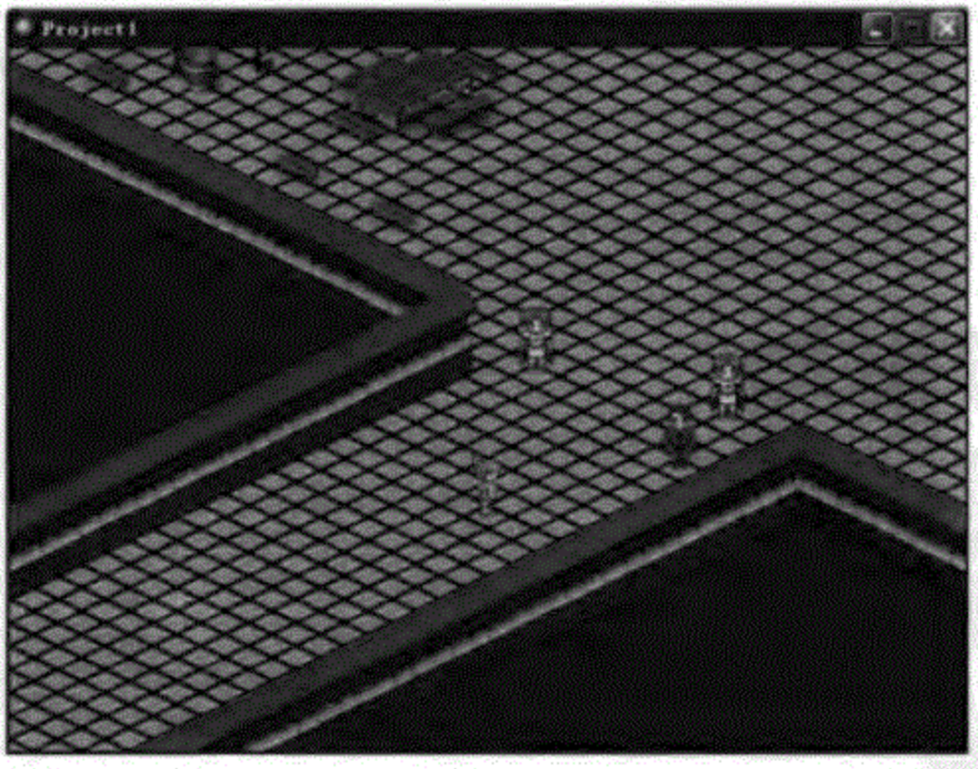


图 7-216

⑥ 战斗时，原来的敌兵也出现在了主角阵中，如图 7-217 所示，但是由于他能力较低，在战斗过程中很容易被击倒，所以玩家要及时使用回复魔法对他的 HP 进行补充。

图 7-217

⑦ 这时，地图的上方出现一个商人，如图 7-218 所示，主角可以利用前面击败敌人所得到的金钱来购买商人的东西。

图 7-218

⑧ 进入购买菜单，如图 7-219 所示，在这里主角可以买入物品，也可以将身上多余的物品卖出，玩家这时要注意检查一下自己之前设置的商店物品是否正确。

图 7-219

⑨ 主角做完买卖之后就可以去挑战 BOSS 了。主角注意要在挑战 BOSS 之前先将自己战斗时失去的 HP 补充回来，BOSS 就在地图的上方，如图 7-220 所示。

图 7-220

⑩ 主角战胜 BOSS 之后，整个游戏便结束了，如图 7-221 所示。

图 7-221

总结：

至此，RPG Maker 实例演示全部结束，虽然用了不少的篇幅，但是提供给读者的只是一个游戏的基础，真正复杂的游戏是需要制作人精雕细琢的。读者也完全可以按照自己的构思去创造和制作属于自己的游戏。

第 8 章　RPG Maker XP 程序编写

在第 6 章中曾经提到过 RPG Maker 中的脚本，本章将对 RPG Maker 中使用的脚本语言 Ruby 进行较为详细的介绍。

熟练地掌握 Ruby 的语法，并了解 RPG Maker 的对象结构，是熟练使用 RGSS 为 RPG Maker 开发脚本的前提。本章内容是针对有一定程序开发基础的读者编写的，如果有疑问，读者还可以参考 RPG Maker XP 中的帮助文档。

8.1 Ruby 和 RGSS 的概念

Ruby 是一种起源于日本的新语言，它在二十世纪九十年代中期由松本幸宏发明，松本至今还维护着 Ruby 的官方发布版本。

最初的 Ruby 设计受启发于 Perl 与 Smalltalk 的部分特性。正如它的名字“红宝石”所暗示的，Ruby 是一门优雅的语言，它简单而精巧。

减少编程时候不必要的琐碎时间，使程序员高兴，是松本的首先考虑的；其次是进行良好的接口设计。他强调系统设计必须人性化，而不是一味从机器的角度设想。

遵循这样的理念，Ruby 语言非常简单易用。Ruby 是一种面向对象的语言，所有 Ruby 中的数据都是对象，这样也使得 Ruby 的程序更加容易理解。在后面我们将对面向对象的分析和设计进行更多的论述。

RGSS 是 Ruby 语言的游戏定制，其语法与 Ruby 完全相同。

8.2 Ruby 环境

虽然 RPG XP 内嵌了 RGSS，但是 RGSS 带有复杂的对象结构和游戏逻辑，对于单独学习 Ruby 来说过于复杂，不便于了解和测试基本的语法。因此我们将利用 Ruby 的单独发行版本讲解此节。Ruby 可以运行在各种类型的计算机和操作系统上。Ruby 的安装程序可以在网络上搜索下载。

1. Ruby 的工具集

安装完成后，“所有程序”菜单中将会增加如下的命令：

```
Ruby-1.8.6-27（安装的版本号）
→ Documentation                          （目录，Ruby 的相关文档）
→ RubyGems                               （目录，Ruby 的包安装程序 RubyGems）
→ fxri-interactive Ruby Help & Console   （交互式 Ruby 帮助文档和解释器）
→ IRB-interactive Ruby Console           （交互式 Ruby 解释器）
→ SciTE                                  （具有用户界面的 Ruby 编辑器，IDE）
→ Uninstall Ruby                         （卸载 Ruby）
```

IRB 是一个基于命令行的交互式 Ruby 解析器，它按行接受用户输入的 Ruby 脚本并解析执行。

SciTE 是一个简单的 IDE，包含了 Ruby 语言的高级功能，开发脚本文件的时候可以提供帮助。

Ruby 同时提供了一些命令行工具帮助实现常用的功能。

```
Ruby source.rb  #Ruby，此命令执行指定的 Ruby 源文件，本例中为 source.rb
Ri Time         #Ri 即 Ruby info。查看指定的 Ruby 类或模块的信息，本例中为 Time
```

在后面的例子中，我们将主要使用 IRB（交互式 console）和 Ruby 命令行。

2. 运行一个 Ruby 程序

用 Ruby 命令运行 Ruby 程序。按照惯例，Ruby 的源文件后缀为 .rb，用 SciTE 或者任意文本编辑器输入如下文本：

```
D:\Ruby\code\hello.rb
print "Hello world"
```

打开命令行工具，执行如下命令：

```
cd D:\Ruby\code
Ruby hello.rb
```

执行的结果在命令行中显示：

```
Hello world
```

用 IRB 运行一个 Ruby 程序。从“开始”菜单启动 IRB，输入如下命令：

```
IRB(main):001:0> print "hello world"
```

执行结果如下：

```
hello world=> nil
```

IRB 所有的表达式都会有返回值，print 语句的返回值为 nil。

8.3 基本语法

本节中我们将介绍 Ruby 的基本类型、标识符、循环控制结构等基本语言模块。熟练地使用这些功能是开发 Ruby 及 RGSS 程序的基础。

1. Ruby 基础语法元素

（1）数据和变量定义

变量是一个值不确定的数据引用，其引用的数据值和类型都可以变化。常量是一个值确定的数据引用。在程序中，变量被用来进行数据操作，例如玩家的 HP 等；而常量则用于保存那些程序运行期间不会变化的值，例如经验值上限、圆周率等。

标识符用于命名变量和函数。

Ruby 的标识符以字母或“_”开始，后面是字母、“_”或数字，没有长度的限制。

Ruby 变量和常量的种类包括全局变量、实变量、局部变量和常量，它们靠其名称的第一个字符来区分。变量的名称由一个可选的前缀符号和一个标识符组成。

1）全局变量

以“$”开头的变量就是全局变量，程序的任何地方都能引用（因此使用时要特别注意，避免冲突）。全局变量没有必要事先声明。引用尚未初始化的全局变量其值为 nil。

例如：$foobar，$_var32，$_1st

2）实变量

以“@”开头的变量就是实变量，属于特定的对象。实变量可以在其类或子类的方法中引用。引用尚未初始化的实变量其值为 nil。

例如：@foobar

3）局部变量

以小写字母或“_”开头的标识符就是局部变量或方法调用。

在局部变量作用域（类、模块、方法定义的部分）中小写字母开头的标识符要初始赋值。引用未声明的标识符会当做无参数的方法调用。

例如：foobar

4）常量

以大写字母开头的标识符就是常量。常量的定义（和初始化）就是进行代入赋值，但是在方法里是不能定义的。访问未定义的常量会发生 NameError 异常。

常量可在类/模块中定义，除在该类/模块中可以引用外，该类的子类或嵌套该模块的类/模块中也能引用该常量。从外部引用常量要用到“::”运算符。

例如：SYSNAME

当常量不是在当前模块内调用的时候，需要同时使用类和模块的名称。

5）伪变量

在普通变量以外，还有一种被称为伪变量的特殊变量。这是为了方便程序设计，由语言提供的对象。

```
self        当前方法的执行对象本身。
nil         NilClass 类唯一的实例，表示伪。
true        TrueClass 类唯一的实例，表示真。
false       FalseClass 类唯一的实例，表示伪。
```

用户不能更改伪变量的值。为伪变数代入赋值会出现语法错误。除 self 只能在类中使用外，伪变量可以在程序中任意位置使用。

注意：两个 nil 是相等的（这一点与其他语言中的 null 不同）。变量名称中不可以使用中文。

用户可以在程序中添加注释来方便理解，用 # 开始的内容代表注释。注释放在相关代码块的上方，单独一行输入。

```
# This is a comment lines
```

注意：在字符串中间的 # 并非注释。

（2）保留字

编译器对保留字有特殊的解释，保留字不能作为类名称、变量名称等使用。但是，以接头符 $、@ 等开头就不再是保留字。在 def 和调用方法的对象后面时，如明确为方法名称就可以作为方法使用。

保留字包含下面一些：

```
BEGIN     class     ensure    nil       self
when      END       def       false     not
super     while     alias     defined?  for
or        then      yield     and       do
if        redo      true      begin     else
in        rescue    undef     break     elsif
module    retry     unless    case      end
next      return    until
```

（3）表达式

表达式包括变量和常量、字面值、运算式、函数调用和控制结构等。表达式组成了 Ruby 的程序。表达式和表达式之间用分号（；）或换行分开。但是，紧跟反斜线符号（\）

的换行并不是分隔符，而是表示到下一行继续。

用户可以使用括号将表达式括起来进行群组化。

提示：在 SciTE 中，用（）包围的代码块可以折叠起来。

表达式例子：

```
(1+2)*3
foo()
if test then print "符合条件" end
```

1）赋值表达式

使用赋值表达式向变量等对象进行赋值。赋值也可以用做局部变量和常数的声明。

语法：

变量 '=' 表达式

常量 '=' 表达式

表达式 '[' 表达式 ..']' '=' 表达式

表达式 '.' 标识符 '=' 表达式

若左边是变量的话，就将表达式的计算值代入其中。

```
A=[]            # 空数组
A[1]=32         # 为数组的元素赋值
```

2）变量与类型

Ruby 的类型系统是一种宽松的、基于推导的系统。当定义一个变量时，对于一个基本类型，不需要指定它的类型，编译器会基于变量的行为选择合适的类型。

Ruby 的类型系统被称为 Duck Type，设计者的意图是：如果一个对象看起来像鸭子，也会嘎嘎叫，它就是一只鸭子。在上面的例子中，对于编译器“1”符合数字 Fixnum 的格式，而“3”符合字符串的格式，编译器就会选择最匹配的类型来实例化需要的对象。

3）定义和使用数字

用赋值符（=）可以为变量赋值。Ruby 会自动识别以下格式的数字类型：

```
numbers.rb
a=5                     # 十进制，默认
b=0x5                   # 十六进制
c=012                   # 八进制
d=0b1010                # 二进制
d=-124;                 # 有符号十进制整数
e=1_000_000_000         # 使用 _ 作为分隔符
f= 0xffff_ffff          # 对十六进制使用分隔符
g=0.123                 # 浮点数，默认
h=1.2e-3                # 浮点数，科学计数法
```

整数类型为 FixNumber，可以使用有、无符号数，可以用二进制、八进制、十进制

和十六进制表示。

浮点数类型为 Float，可以选择使用科学计数法表示。

在数字值中可以插入“_”作为分隔。Ruby 编译器会忽略“_”，不会对其做特别的处理。这样在那些数值很大的数字中加入“_”，会很方便地快速看清楚其位数，了解其值的大小。

注意：浮点数不能像其他语言写成像“.1”那样直接以“.”开头。“_”加在数字的开头和结尾，或连续写入多个“_”都是不允许的。

4）简单算术运算

数字可以直接运行常用的数学运算：+，-，*，/，%（求余）。

整数的运算结果为整数，浮点数的运算结果为浮点数，混合使用时，结果为浮点数。

可以使用赋值语句将计算的结过存储到一个变量中，赋值语句由一个变量名后跟一个赋值运算符与算数表达式组成。下面是一个简单的算术运算赋值：

```
damage=baseDamage * raceFactor                    # 计算乘积并赋值
```

其中，等号为赋值运算符，等号右侧的值被计算出来赋给等号左侧的变量。baseDamage 和 raceFactor 两个变量在使用之前必须定义并赋值。如果上例中 raceFactor 为浮点数，baseDamage 为整数，则结果 damage 将为浮点数。

5）使用布尔值

布尔（bool）值可以取值为 true 或者 false，常常用于条件判断。

```
isNumber=true
isReady=false
```

布尔运算可以被使用在布尔值上：

```
isReady=haveMoney && haveTime                     # 是否同时满足两个条件
isWinner=isHighestScore || isNoEnemy              # 是否满足一条条件
isOK=! isOK                                       # 对条件取反
```

（a）表达式“&&”表达式

首先计算左边，若结果为真就接着计算右边。and 运算符与 && 的作用相同但优先级更低。将包含 and 的表达式作为某方法的参数时，必须使用双层括号。

（b）表达式“||”表达式

首先计算左边，若结果为假就接着计算右边。or 运算符与 || 的作用相同但优先级更低。将包含 or 的表达式作为某方法的参数时，必须使用双层括号。

（c）“!”表达式

此表达式为一元操作符，若表达式值为真就返回假，若表达式值为假则返回真。! 与 not 作用相同。

（d）自运算操作符

自运算操作符作用运算于变量本身，它由一个运算符与等号组合而成，其语法为：

```
表达式1 op= 表达式2
```

op 可以是下列运算符中的某一个，运算符与“=”之间不留间隔。

+, -, *, /, %, **, &, |, ^, <<, >>, &&, ||

这种赋值形式和“表达式1 = 表达式1 op 表达式2”等同。例如：

```
subTotal +=100          # 增加 100
amount*= 1.05           # amount 增加 1.05 倍，等同于 amount=amount*1.05
```

2. 定义和使用字符串

（1）定义字符串

在 Ruby 中，可以用单引号和双引号定义字符串，字符串是 String 类的实例。

```
"This is a string expression\n"     # 双引号字符串
'This is a string expression'       # 单引号字符串
%q|this is another string\t|        # %q 指定'|'为字符串起始字符
%Q{Seconds/day: #{24*60*60}}        # %Q 指定'{'位字符串的起始字符
```

以双引号包围的字符串控制码和内嵌表达式都是有效的，而以单引号包围的字符串，除了\\（反斜线符号本身）、\'（单引号）和行尾的\（忽略换行）外，其他的在字符串中将不做处理。中间夹有空字符的多个字符串值会被当做是一个字符串值来处理。"foo" "bar" 等效于 "foobar"。

```
控制码：反斜线符号 \ 是字符串中的特殊文字，当做控制码使用。
\t          制表符（0x09）
\n          换行符（0x0a）
\r          回车符（0x0d）
\f          换页符（0x0c）
\s          空格符（0x20）
\nnn        八进制代码（n 是 0~7）
\xnn        十六进制代码（n 是 0~9，a~f）
\"          可以用来在双引号字符串中包含双引号'"'，'\\'可用于输入'\'
```

（2）字符串中内嵌表达式

在双引号（"）包围的字符串和正则表达式中，可以使用“#{ 表达式 }”的形式把表达式的内容（以字符串的形式）嵌入其中。表达式为以变量符号（$、@）开头的变量时能够省略 {}。# 后如果不是跟着 {、$、@，就会被仅当做字符“#”解释。如果确定不使用内嵌表达式，在 # 前放置一个反斜线符号就可以了。

```
$name="world"
print"my name is #{$name}"      # 输出 "my name is world"
print 'my name is #{$name}'     # 输出 "my name is #{$name}"
```

%q 和 %Q 可以指定字符串的开始和结束符号，%q 与单引号相同，不会进行内嵌表达式评估，而 %Q 与双引号字符串相同。

3. 使用数组

（1）数组的语法

数组的语法是：

```
'[' 表达式 ',' ...']'
```

解释器分别计算每个表达式的值，然后返回由这些值构成的数组。数组是 Array 类的实例。

每次对数组表达式进行计算时都会生成新的数组对象。

```
[1,2,3]
[4+1, 6/3, 8*2]
[1]*10                  # 快速初始化数组，[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

数组中的值可以用 [] 来索引访问，例如

```
a=[1,2,3]
b=a[2]                  # 返回 3，索引从 0 开始，所以 a[2] 返回第三个值
```

（2）多重赋值

语法：

```
表达式 [','[ 表达式 ',' ...] ['*'[ 表达式 ]]]= 表达式 [, 表达式 ...][',' 表达式 ]
'*'[ 表达式 ]= 表达式 [, 表达式 ...][',' 表达式 ]
```

多重赋值是指在多个表达式以及数组中同时进行的赋值。左边的各个表达式必须是可以被赋值的。若右边只有一个表达式时，则将该表达式的计算值转为数组后，再把数组中的各个单元依次赋值给左边。若右边数组单元的数量超过左边的话，将忽略多余的数组单元。若右边数组单元个数不足的话，将向左边多余的单元中代入 nil。

若左边最后一个表达式前带 *，则将右边多余的单元以数组的形式代入这个带 * 的表达式中。若右边没有多余单元，就把空数组代入其中。

例如：

```
foo, bar   = [1, 2]      # foo = 1; bar = 2
foo, bar   = 1, 2        # foo = 1; bar = 2
foo, bar   = 1           # foo = 1; bar = nil
foo, bar   = 1, 2, 3     # foo = 1; bar = 2
foo        = 1, 2, 3     # foo = [1, 2, 3]
*foo       = 1, 2, 3     # foo = [1, 2, 3]
foo, *bar  = 1, 2, 3     # foo = 1; bar = [2, 3]
```

（3）使用哈希表

哈希表（也叫做关联数组）就是把任意种类的对象和另一任意种类的对象进行关联。它定义了一组键与值的对应关系，语法：

```
'{' 表达式 '=>' 表达式 ',' ...'}'
```

分别计算每个表达式的值，返回由主键和数值组成的哈希表对象。哈希表是 Hash 类的实例。

每次对哈希表表达式进行计算时都会生成新的哈希表对象。

```
a={1=>2, 3=>4, 5=>6}
b=a[1]    # 返回 2
```

4. 范围表达式

语法：

```
表达式 1'..' 表达式 2
表达式 1'...' 表达式 2
```

从表达式 1 到表达式 2 的范围对象，范围对象是 Range 类的实例。

".." 运算符生成的范围对象包含后面数值，"…" 运算符生成的范围对象不包含后面数值。

如果范围表达式两端是数值字面值，每次计算都会返回同一个对象，否则每次计算都会生成一个新的范围对象。

```
1..5                    # 从 1~5 的范围，包含五个数字
"str1000"…"str1005"     # 包含 str1000~str1004 共 5 个字符串
1.1 .. 2                # 包含 1.1, 1.2, ... 2, 共 10 个数字
```

5. 运算符表达式

除了前面提到的 "+"、"-"、"*"、"/"，Ruby 还提供了许多其他的运算符进行编程。Ruby 语言中有下列运算符。这些运算符拥有不同的优先级，具体如下所示：

```
操作符优先级列表
优先级高        []
                []
                **
                -（单项） +（单项） ! ~
                *  /  %
                +  -
                << >>
```

```
                &
                |  ^
                > >=  < <=
                <=> ==  === !=  =~  !~
                &&
                ||
                ..  ...
                ?:(条件运算符)
                =(+=, -= ... ,自运算符)
                not
优先级低        and  or
```

优先级相同的时候，按照从左向右的顺序进行，用括号包含的表达式具有最高的优先级。

```
a && b || c   等效于(a && b) || c
a || b && c   等效于a || (b && c)
```


（1）比较操作

进行比较的时候可以使用如表 8-1 所示的操作符：

表 8-1

运算符	描　述
==	两侧操作符相等，则为 true，否则为 false
>	左侧操作数大于右侧，则为 true，否则为 false
>=	左侧操作数不小于右侧，则为 true，否则为 false
!=	左侧操作数不等于右侧，则为 true，否则为 false
<	左侧操作数小于右侧，则为 true，否则为 false
<=	左侧操作数不大于右侧，则为 true，否则为 false

如表 8-1 所示，逻辑判断运算符的结果非 true 即 false，因此它们常被用于进行是非判断。例如：

```
isBiggerThan5 = var > 5                    # 变量是否大于 5
```

（2）条件运算符

语法：

```
表达式 1 ? 表达式 2 : 表达式 3
```

评估表达式 1 的结果，如果为 true 则选择返回表达式 2，反之则返回表达式 3。它与 if 表达式 1 then 表达式 2 else 表达式 3 end 完全相同。

```
Logon ? print("true"): print("error")
# 等同于
If logon == true then
   print "true"
else
   print "error"
end
```

小结：

数据和表达式是组成 Ruby 程序的基础。Ruby 提供了丰富的语言特征帮助我们实现各种计算表达。在开始实现更加复杂的程序之前，必须要熟练的掌握基础的数据和表达式操作。下一节我们将开始学习控制结构和函数，从而实现更加复杂的程序。

8.4 循环、控制结构与函数

1. 判断

在所有的程序中，逻辑判断是最基本的一项操作，游戏更加是逻辑判断的集合。你经常要进行类似下面的逻辑判断：

如果敌人是火系的，则火系魔法的伤害减少一半。

在程序设计中，这意味着用变量、常量和表达式的值来进行比较，利用获得的结果决定接下来执行的程序逻辑。判断经常使用上一节中提到的比较逻辑。

（1）if 语句

第一种可以利用判断结果的语句是 if 语句，它经常被用来处理“如果 ***，则 ***”这样的操作，它的基本格式为：

```
if 表达式 [then]
  表达式 ...
[elsif 表达式 [then]
  表达式 ... ]
...
[else
  表达式 ... ]
end
```

若条件表达式的计算结果为真时，将计算 then 以下的表达式。若 if 的条件表达式为假时，将计算 elsif 的条件部分。可以存在若干个 elsif 部分，若所有的 if 以及 elsif 的条件表达式都为假的话，如果有 else 部分，就计算它的表达式。

if 表达式的结果取决于条件成立部分（或 else 部分）中最后被计算的表达式的结果。

若没有 else 部分，且所有条件均不成立的话，就返回 nil。

Ruby 中只有 false 和 nil 表示假，其他的都是真，甚至 0 或空字符串也是如此。

注意：在 Ruby 中，和 if 对应的是 elsif，而并非 else if（C 语句）或者 elif。

if 还可以作为修饰符，用在语句的结尾，作为是否执行的判断条件。

语法：

```
表达式 if 表达式
```

当右边的条件成立时，计算左边的表达式，并返回其结果，若条件不成立则返回 nil。

例子：

```
print "debug\n" if $DEBUG          # 如果 $DEBUG 全局变量为 true，则输出字符串
```

需要注意的是，if 表达式具有一个返回值表达式，其返回值的规则与函数相同，最后一个被选择的条件表达式模块中语句的返回值被作为 if 的返回值。

（2）unless 语句

语法：

```
unless 表达式 [then]
  表达式 ...
[else
  表达式 ... ]
end
```

unless 与 if 正好相反，当条件表达式结果为假时，才计算 then 后面的表达式。unless 表达式中不能插入 elsif 语句。

unless 也可以作为修饰符

例子：

```
print "stop\n" unless valid(passwd)    # 如果 valid 返回 false，则输出字符串
```

语法：

```
表达式 unless 表达式
```

当右边的条件不成立时，计算左边的表达式，并返回其结果，否则返回 nil。

（3）case

语法：

```
case 表达式
[when 表达式 [, 表达式] ... [then]
  表达式 ..]..
[else
  表达式 ..]
end
```

case 语句可以对应多个 when 和一个可选的 else。case 先依次对 when 中的表达式进行匹配判断，然后根据匹配结果进行分支选择。它使用 === 运算符比较 when 的指定值和最初那个表达式的计算值，若相同就计算 when 部分的内容。

一个 when 中可以对应多个匹配或者 Range，在这种情况下，每一个值会被分别的匹配，只要有一个匹配就会返回该分支对应的代码。

case 将返回条件成立的 when 部分（或 else 部分）中最后被计算的表达式的结果。若所有条件都不成立，则返回 nil。

case 常常被用来处理“如果 A 则…，如果 B 则…”这样的场景。

```
case inputLine
  when "debug"

  when "quit", "exit"              # 多个匹配条件
    exit
  else
    print "Illegal command:    #{inputLine}"
end
```

2. 循环

（1）while 语句

语法：

```
while 表达式 [do]
  ...
end
```

只要表达式的值为真，就循环执行 while 语句中的内容。

while 返回 nil。另外，可以使用带参数的 break，将 while 表达式的返回值设为那个参数的值。

例子：

```
ary = [0, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
i = 0
while i < ary.length
  print ary[i]
  i += 1
end
```

语法：

```
表达式 1 while 表达式 2
```

当表达式 2 为真时，重复执行表达式 1。注意，要保证表达式 2 在一定的循环后返回 false，否则可能导致死循环。

例如：

```
index  *=2 while index < 10000        # 循环直至 index 超过指定的值
```

（2）until 语句

语法：

```
until 表达式 [do]
   ...
end
```

与 while 相反，until 在表达式的值变为真之前，一直循环执行 until 中的内容。

until 返回 nil。另外，可以使用带参数的 break，将 until 表达式的返回值设定为那个参数的值。until 用来处理“直到…，进行…处理”。

语法：

```
表达式 until 表达式
```

重复执行表达式 1 直到表达式 2 为 true 时结束循环。注意，要保证表达式 2 在一定的循环后返回 true，否则可能导致死循环。

例子：

```
print(f.gets) until f.eof?          # 输出每一行直至文件结尾
```

（3）for 语句

语法：

```
for Lhs ...  in 表达式 [do]
  表达式 ..
end
```

先计算表达式得到一个列表，然后分别针对该对象中的每个元素，循环执行 for 的内容。Lhs 表示当前循环中选中的元素值。

例子：

```
sum = 0                  # 计算从 1 到 100 的和
for i in 1..100
  sum +=i                #i 为当前循环的值
end
```

（4）break 语句

语法：

```
break [表达式]
```

break 将退出最内层的循环。

例子：

```
i = 0
while i < 3
  print i, "\n"
  break                          # 在第一次循环时即退出循环
end
```

与 C 语言不同，break 只能从循环中退出，而不能从 case 中退出。

若使用 break 退出 for 或迭代循环后，该循环将返回 nil。但如果使用了参数的话，循环将返回那个参数的值。

（5）next 语句

语法：

```
next [表达式]
```

next 将跳转到最内侧循环的头部。在迭代器中，它将跳离 yield 调用。

用 next 跳离 yield 后，yield 表达式将返回 nil。但如果使用了参数，则 yield 表达式的返回值就是该参数的值。

例子：

```
str.each_line do |line|
  next if line.empty?      # 如果 line 为空，调到 str.each_line，执行下一循环
  print line
end
```

（6）redo 语句

语法：

```
redo[表达式]
```

redo 将重新执行最内侧的本次循环，但是循环变量不会变化。

（7）retry

语法：

```
retry[表达式]
```

retry 将重新执行最内侧的整个循环。循环将会被重置并重新执行。

（8）迭代器

语法：

```
method(arg1, arg2, ...) do ['|' 表达式 ...'|'] 表达式 ... end
method(arg1, arg2, ...) '{'['|' 表达式 ...'|'] 表达式 ...'}'
```

迭代器是指为了对控制结构（多见于循环）进行抽象化而设计的方法。将 do...end 或 {...} 中的代码片段（也就是块）添加在方法后面，然后再调用该方法时，就能从该方法内部对块进行计算。在迭代器内进行块调用时使用 yield 表达式。传给 yield 的值会被赋值给夹在“||”中的变量。

下面是几个简单的迭代器例子：

```
3.times do
      print "Hello!"                                  # 执行三次打印
end
0.upto(9) do |x|                                      # 从 0 循环至 9，循环变量赋给 x
  sum +=x                                             # 使用循环变量进行运算
end
0.step(100, 2) {|x| sum+=x}                           # 计算从 0~100 所有偶数的和
[1, 1, 2, 3, 5].each {|val| print val, " " }  # 打印列表
```

（9）yield 语句

语法：

```
yield '('[ 表达式 [',' 表达式 ...]]')'
yield { 表达式 [',' 表达式 ...]]
```

把参数传给块之后，对块进行计算。因为 yield 定义迭代器，所以是在方法定义内使用。

下面的例子中，将会依次输出每一个 yield 的结果。

```
def foo
  yield(1, 2)                  # 每一个 yield 对应依次迭代器的执行，本例中会被执行两次
  yield(5, 6)
end
foo {|a, b| print [a, b]}  # 输出：[1, 2]   [5, 6]
```

对块参数进行赋值时遵从多重赋值规律。若执行 yield 时，方法并没有带块（不是迭代器），就会引发 LocalJumpError 异常。

yield 将会返回块内最后计算的表达式的值。若因 next 引起块的运行中断，则返回 nil。若 next 带参数，该参数的值就是 yield 的返回值。

3. 异常处理

在 Ruby 程序中，用 begin/end 块处理异常，语法如下：

```
begin
    code...
[rescue  [parm]* [=> var] [then]
        # 异常处理代码
]*
[else
        # 没有发生异常时候的处理代码
]
[ensure
        # 始终会被调用的代码
]
end
```

一个代码块可以包含多个 rescue，每一个 rescue 可以带有若干个错误类型参数，如果没有指定错误类型，则默认参数为 StandardError。

当程序发生异常时，Ruby 会扫描当前的调用堆栈直到找到一个 begin/end 块，然后 Ruby 会比较此块中的所有 rescue 条件，如果捕获的异常与某一个 rescue 的参数匹配，则会处理该条件对应的代码块。

Ruby 定义了全局变量 $! 用于表示当前的异常，如果没有异常，则 $! 为 nil。

（1）raise 异常

语法：

```
raise                                  # 触发一个 RuntimeError
raise(aString)                         # 触发一个 RuntimeError，指定错误
                                         信息
raise(anException[, aString[anArray]]) # 触发一个指定的 Error
```

raise 并不是 Ruby 的保留字，而是内部函数（Kernel.raise）。如果没有指定参数，raise 抛出 $! 中的异常。如果只提供一个字符串参数，则抛出的 RuntimeError 信息被设置为指定的字符串。如果提供多个参数，第一个参数必须为异常的类型，或者一个返回异常的对象。

（2）rescue 语句

语法：

```
表达式 1 rescue 表达式 2
```

若表达式 1 中发生异常时就计算表达式 2。

不能指定想捕捉的异常类（也就是说，只能捕捉 StandardError 异常类的子类）。

在包含 rescue 修饰符的表达式中，若没发生异常则返回表达式 1 的值，若发生异常

则返回表达式 2 的值。

例子：

```
File.open("file") rescue print "can't open\n"
```

4. 函数

（1）什么是函数

函数是一组表达式操作的集合，通常用于完成一项给定的任务。在 Ruby 中，使用 def 来定义函数。函数中可以包含前面提到的所有的控制结构和语句。

（2）定义函数

函数名与标识符的命名规则类似。通常函数名应该用小写字母开头。在执行用大写字母开头的函数定义时，解释器会把函数解析成常量，可能导致执行错误。

如果函数的功能是做查询，函数的结尾都用“?”命名，例如 instance_of?

如果函数的功能可能是“危险”的，比如修改重要的值，函数结尾用 ! 命名，例如字符串类中 chop! 方法。

? 和 ! 符号仅仅被用于方法命名。

函数可以带有零到任意个参数，参数紧接在函数名后面的括号中，用逗号分隔。函数参数是仅在函数内部有效的局部变量。

下面是一个函数定义的例子：

```
def func1(arg1, arg2, arg3)      # 三个参数
  # 函数包含的表达式
end
def func2                        # 没有参数
  # 函数包含的表达式
end
```

Ruby 中还可以为函数参数指定默认值，如果调用时不指定参数值，则默认值将被使用。

```
def func3(arg1="var1", arg2="var2", arg3=12)
# 函数包含的表达式
end
```

类似数组的定义方式，Ruby 中可以定义变长的参数列表：

```
def varargs(arg1, *rest)                    # rest 将被作为一个数组传递给函数体
  # 函数包含的表达式
end
```

（3）函数的调用

函数的调用比较简单，其语法如下：

```
func2                          # 调用无参数函数
func3 "param1", "param2"       # 对 arg3 使用默认值
varargs(1, 2, 3)               # 使用变长参数，相当于 varargs(1, [2, 3])
```

在函数的调用中，括号可选，但是保留括号可以使代码更加易读，而且如果需要使用函数返回值时，使用括号可以避免遇到优先级冲突的问题。

每一个函数都具有一个返回值，返回值用 return 标记。如果没有标记，则函数返回最后一个表达式语句的返回值。

1）return

语法：

```
return [表达式 [','表达式 ... ]]
```

结束方法的运行，且把表达式的值设定为方法的返回值。若给出了 2 个以上的表达式，则将这些表达式组成一个数组，然后把该数组设定为方法的返回值。若省略表达式，则返回值为 nil。

例子：

```
return            # 返回 nil
return 12         # 返回 12
return 1, 2, 3    # 返回数组 [1, 2, 3]
```

2）alias

给方法或全局变量添加别名，其语法如下：

```
alias 新方法名 旧方法名
```

可以给方法名指定一个标识符或 Symbol（不能写 obj.method 这样的表达式）。alias 的参数不会被计算。

给方法添加别名时，别名方法将继承此刻的原始方法。此后，即使原始方法被重新定义，别名方法仍然保持着重定义前的老方法的特性。若用户改变了某方法的内容后，又想使用修改前的方法时，别名会非常有用。

```
def foo                        # 定义方法
  "foo"
end
alias: _orig_foo: foo          # 设定别名
def foo                        # 重写方法定义（利用以前的定义）
  _orig_foo * 2
end

p foo                          # 输出 "foofoo"。
```

alias 表达式返回 nil。

小结：

在 Ruby 中，任何一个功能都可以通过多种途径完成。熟练地掌握基本的语法和控制结构是构造复杂系统的基础。下一节我们将解释面向对象的设计方法，面向对象是 Ruby 的一个重要特性，任何数据在 Ruby 中都是对象，从基本数值到类定义本身。了解了这种设计方法，将会更加有利于构建复杂的程序。RGSS 更是大量地利用了复杂的对象结构。

8.5 面向对象的设计方法

在面向对象的分析设计中，类是对一组行为相同的对象的规格说明。规格包含两个方面：行为（Behavior）和数据（Data）。行为由类的方法声明，数据则有类的字段声明。

例如，对于字符串类 String，它在内部定义了字符串的内存分配算法，用于保存字符串内容，同时它也为了操作这些数据而定义了处理字符串的方法，例如 String.strip。

1. 定义类

在 Ruby 中，使用 class 关键字定义一个类。例如：

```
class Song
  def initialize(name, artist, duration)    # 构造对象初始化
    @name     = name                        # 实变量不需要预先声明
    @artist   = artist
    @duration = duration
  end
  def initialize()                          # 构造对象初始化
    @name    ="default"                     # 实变量不需要预先声明
    @duration = 100
  end
end
```

类名需要满足常量的定义规则，即必须以大写字母开头。

对于类的初始化函数 initialize，其可以拥有任意个参数。对应实例化的时候也需要给定同样数量的参数。用户可以定义多个具有不同参数的初始化函数。

（1）实例化类

在 Ruby 中，使用类的 new 方法来实例化一个对象。

```
instance1 = Song.new
instance2 = Song.new("ruby", "me", 1000)
```

New 函数调用类的 initialize 方法来初始化一个对象，因此可以匹配 initialize 函数对应的参数。

如果没有定义 initialize 方法，则 Ruby 会提供一个默认无参数的初始化方法。如果提供了任意一个 initialize 的有参数方法，则系统不会自动提供无参数的 initialize。此时如调用无参数的 new 会导致 ArgumentError。

（2）定义属性

在类的属性定义中，除了上文定义的实变量以外，还可以定义可供外界访问的属性。

```
class Song
  def name                    # 定义属性，只读
    @name
  end
  def artist
    @artist
  end
  def duration
    @duration
  end
end
```

与上述代码等效的一种简单的表达方法为：

```
class Song
  attr_reader :name, :artist, :duration          # 只读属性
end
```

如果需要定义可写的属性，Ruby 的语法与其他的常用语言稍有不同：

```
class Song
  def duration=(newDuration)
    @duration = newDuration
  end
end
```

一个简单的做法如下：

```
class Song
  attr_writer :duration                          # 只写的属性
end
```

定义读写的属性的方法如下：

```
class Song
  attr_accessor:duration                    # 可读写的属性
end
```

（3）定义类的静态数据

通常定义的属性是变量级别的属性，每一个类的实例拥有不同的值。类的静态数据在类的所有实例中共享，即全局只有一个实例。

静态数据由两个 @@ 引导，例如：@@classVar

（4）定义函数

在类中定义函数与普通的函数定义差别并不大，所有的类属性，包括静态属性都可以在函数中使用。

```
class Song
def Song.create          # 定义类静态函数
    end
    def sing             # 定义类成员函数
    end
end
```

如果在类函数中加上类名前缀，则将定义成为类的静态函数。静态函数必须与类名同时使用，它不能访问类的实例成员变量。

使用类的方法：

```
s=Song.new               # 定义类的实例对象
s.sing()                 # 调用实例方法
Song.create              # 调用静态方法
```

2. 类的继承

可以基于一个现有的类进行扩充定义一个新的类，这个过程被称为派生。这个派生类被称为类的直接子类。原始类被称做基类，又被称做超类。

通常为了扩充一个类的定义，可以附加新的属性或者函数，隐藏现有的属性或函数，改写现有的函数等。

语法：

```
class 标识符['<' superclass]
  ...
end
```

（1）继承的例子

继承的例子如下：

```
class KaraokeSong < Song
  def initialize(name, artist, duration, lyrics)
    super(name, artist, duration)    # 调用基类的构造函数
    @lyrics = lyrics                 # 增加新的数据
  end
end
```

类定义实际上就是把类赋值给由类名指定的常数（在 Ruby 中，类也是一个对象，它是 Class 类的实例）。

若某个类已经被定义过，此时又用相同的类名进行类定义，则意味着对原有的类的定义进行追加。

```
class KaraokeSong < Array
  def take
  end
end
class KaraokeSong
  def take2
  end
end
```

类定义中可以出现嵌套。下例中，嵌套外侧的 Outter 类和内侧的 Inner 类之间根本没有功能上的联系（除了常数 Inner 是 Outter 中的常数 Outter::Inner 之外）。类的嵌套就是指把与类有关的类 / 模块放在该类的外侧，使它们构成一个整体，借以表达某种包含关系。

```
class Outter
  class Inner
  end
end
```

（2）重载操作符

在类里可以重载操作符从而改变操作符的行为。

```
class MyArray
  def initialize
    @ary = [0, 1, 2, 3, 4, 5, 6, 7]     # 成员的实变量数组
  end
  def [](i)                              # 重载 [] 操作符
    @ary[i * 2]
  end
  def []=( i, v )                        # 重载 []= 操作符
    @ary[i * 2] = v
```

```
  end
end
c = MyArray.new
print c[3]                          # 变成 c.[](3),结果为6
print c[3] = 1                      # 变成 c.[]=(3,1),结果为1
```

3. 定义模块

模块可以用于组织方法、类和常量。它可以被用做命名空间从而防止命名冲突。

语法

```
module 标识符
    class 类名                      # 定义类
    end
    常量名 = 常量值                 # 定义常量
end
```

模块名是由大写字母开头的标识符。模块定义实际上就是把模块赋值给由模块名指定的常数（在 Ruby 中，模块也是一个对象，它是 Module 类的实例）。

若某个模块已经被定义过，此时又用相同的模块名来定义模块，则意味着对原有的模块定义进行追加。

模块定义表达式将返回最后计算的表达式的值。若该表达式不返回值，则返回 nil。

使用模块中定义的类和常量的时候，需要使用“::”操作符，而调用方法的时候，则需要使用“.”操作符。定义和使用模块的例子：

```
module Mytest                       # 定义模块
    def Mytest.test                 # 定义模块方法
      print "this is my test"
    end
    class MClass                    # 定义模块类
      def classfunc
          print "class test"
      end
    end
    CONST_VAR=32                    # 定义模块常量
end

print Mytest::CONST_VAR             # 调用模块常量
c = Mytest::MClass.new              # 实例化模块类
c.classfunc
Mytest.test                         # 调用模块方法
```

需要注意的是，在使用模块方法的时候，如果需要在外部直接使用，则需要定义为静态方法，即增加模块名前缀。

require 加载另外一个源文件或者模块。当我们实现复杂系统时候，一个文件无法描述所有的逻辑，这个时候就需要使用多个文件来实现。用 require 语句可以加载另外的文件和模块。例如：

```
require "anotherfile.rb"
```

注意：因为 Ruby 是一种脚本语言，在使用任何一个类型之前，该类型必须被加载，因此需要正确地指定文件的加载顺序。

小结：

了解了类和模块的使用规则后，我们就可以更好地抽象和表达数据，构造更加复杂的程序了。

下一节将讲述 RGSS 的应用，会应用到我们在前几节学到的知识。读者会发现如果了解了 RGSS 的对象结构，用脚本扩展 RPG Maker 将会变得非常简单。

8.6 RGSS 在 RPG Maker 中的简单应用

RPG Maker 中使用的是 Ruby Game Scripting System（RGSS）。RGSS 使用 Ruby 的语法，同时提供了大量的对象来帮助游戏制作者快速地实现游戏中的动态内容。

1. 使用声音

在 RPG XP 中，声音分为 4 种：

- 背景音乐：BackGroundMusic，BGM。
- 背景声音：BackGroundSound，BGS。
- 事件音乐：MusicEvent，ME。
- 事件声音：SoundEvent，SE。

注意：在 RPG Maker XP 中，使用的是 ME 和 SE。

在 Audio 包中定义了播放、停止、淡出的方法所对应的各种声音（SE 没有淡出）。

```
Audio.bgm_start        # 开始播放背景音乐
Audio.bgm_stop         # 停止播放背景音乐
Audio.bgm_fade         # 背景音乐淡出
```

但是在游戏中，我们并不直接使用 Audio 播放音乐，使用 RPG 包下的几个类可实现同样的功能：

```
RPG::BGM.start         # 开始播放背景音乐
RPG::BGM.stop          # 停止播放背景音乐
RPG::BGM.fade          # 背景音乐淡出
```

RPG::BGM.start 与 Audio.bgm_start 的区别在于，前者从 Audio/BGM 目录下启动指定的音乐文件，而后者需要提供相对应用程序的全路径。下面的两个语句的功能基本相同。

```
RPG::BGM.start("bg.mp3")
Audio.bgm_start("Audio/BGM/bg.mp3")
```

同理可以使用 RPG::BGS，RPG::SE，RPG::ME。

2. Graphic 模块

Graphic 模块管理显示相关的一些方法，比如淡入淡出、画面切换等。

Graphics.update 包含一些重要的方法（见表 8-2）：

表 8-2

函　数	作　用
update	更新显示，将在 Scene_Base 的循环中调用
fadeout（duration）	淡出
freeze	固定在当前画面上，在调用 transition 之前画面不允许改变
transition（duration）	转换到当前画面
snap_to_bitmap	RGSS2 新方法，截屏当前画面
width	屏幕宽度
height	屏幕高度
resize_screen	重置画面尺寸，不大于 640×480

3. input 模块

input 模块用于判断输入，包括键盘和手柄，这个模块包含以下一些重要方法（见表 8-3）：

表 8-3

函数	作　用
update	更新输入状态，在 Scene_Base 的循环中调用
press?（num）	Num 对应的键是否被按下，例如 input.press?（Input.C）
trigger?（num）	指定按键是否被重新按下，只有从未按下状态向按下状态变化的瞬间才被认定是“重新按下”
repeat?（num）	指定按键如果被新按下和 trigger? 不同的是，其考虑了连续按下按钮时的重复。如果按下返回 true，未按下则返回 false
dir4	对于四方向手柄，返回当前选中的方向，返回（2，4，6，8），如果没有被按下，则返回 0
dir8	返回八方向手柄返回的方向，（1，2，3，4，5，6，7，8，9），如果没有被按下，则返回 0
按键常量	重置画面尺寸，不大于 640×480

Input 模块还定义了如下常量（见表 8-4）：

表 8-4

常量	作用	常量	作用
Down Left Right Up	方向键上、下、左、右	Shift Ctrl Alt	键盘按键
A B C X Y Z L R	与键盘按键对应	F5 F6 F7 F8 F9	键盘按键

4. 查看 RGSS 的内置模块和类

单击“工具”→“RGSS 编辑”命令（对应 VX），或“工具”→“脚本编辑器”命令（对应 RPG Maker XP），弹出脚本编辑器窗口，窗口左侧包含了所有的加载到当前游戏的脚本。基本分为以下几类：

（1）模块

- Vocab：定义游戏中使用的常量、字符串等。
- Sound：定义声音播放相关的帮助函数。
- Cache：定义缓存。

（2）游戏对象

游戏对象定义了游戏中玩家、事件、敌人等类。重要的类包括 GameActor、GameCharacter、GameEnemy 等。

（3）活动块

活动块是画面精灵的类，用于处理动画等。

（4）窗口

游戏中使用的窗口。Window_Base 是所有窗口的基类。

（5）场景

游戏中的场景主要包括商店、战斗等。场景是一系列窗口的组合。SceneBase 是一切场景的基类。

（6）素材

用于加载用户自己的脚本。

（7）主处理

Main 模块是游戏程序的入口。RGSS 将依次加载所有的脚本，并从主处理开始执行。修改此处的脚本可以修改开始屏幕中的信息，默认的字体信息等。

5. RGSS 游戏的执行流程

任何一个 RGSS 游戏都将从 Main 模块开始执行。

```
begin
    Font.default_name = ["黑体"]          # 默认字体
    Graphics.freeze                       # 锁定当前屏幕，准备淡入到新场景
    $scene = Scene_Title.new              # 创建标题选项界面
    $scene.main while $scene != nil       # 主循环
```

```
    Graphics.transition(30)
rescue Errno::ENOENT                    # 异常处理
    filename = $!.message.sub("No such file or directory - ","")
    print("文件 #{filename} 无法找到 ")
end
```

默认的程序将首先初始化一个标题选项界面。

ScreenTitle 将加载所有数据库，生成游戏对象，初始化一个选项窗口并循环等待用户输入：

```
def start
    super
    load_database               # 读取数据库
    create_game_objects         # 生成游戏对象
    check_continue              # 继续游戏的有效判定
    create_title_graphic        # 生成标题图像
    create_command_window       # 生成指令窗口
    play_title_music            # 演奏标题音乐
end
```

Update 方法中将判断指令窗口的状态，并转到选择的处理逻辑上：

```
def update
    super                                   # 基类中调用了 Input.update
    @command_window.update
    if Input.trigger?(Input::C)             # 对应键盘空格键或回车键
      case @command_window.index
      when 0                                # 新的游戏
        command_new_game
      when 1                                # 继续游戏
        command_continue
      when 2     # 离开游戏
        command_shutdown
      end
    end
  end
```

选择新建游戏的处理：

```
def command_new_game
    confirm_player_location
    Sound.play_decision
    $game_party.setup_starting_members          # 初期队伍
```

```
    $game_map.setup($data_system.start_map_id)     # 初期位置的地图
    $game_player.moveto($data_system.start_x, $data_system.start_y)
    $game_player.refresh
    $scene = Scene_Map.new                         # $scene 的改变将会导致场景切换
                                                   # Main 中定义了 $scene.main while
$scene != nil，因此将触发新场景的 main 函数
    RPG::BGM.fade(1500)
    close_command_window
    Graphics.fadeout(60)
    Graphics.wait(40)
    Graphics.frame_count = 0
    RPG::BGM.stop
    $game_map.autoplay
end
```

SceneMap 和 SceneBattle 定义了游戏地图和战斗之间的切换，请参考其中代码以了解更多细节。

（1）实例 Demo1：增加一个自定义窗口

在本例中我们将增加一个窗口，在标题选项界面显示。

打开 RGSS 编辑器，在窗口分组最后的位置右击鼠标，选择插入、增加一个新的窗口类，如图 8-1 所示。

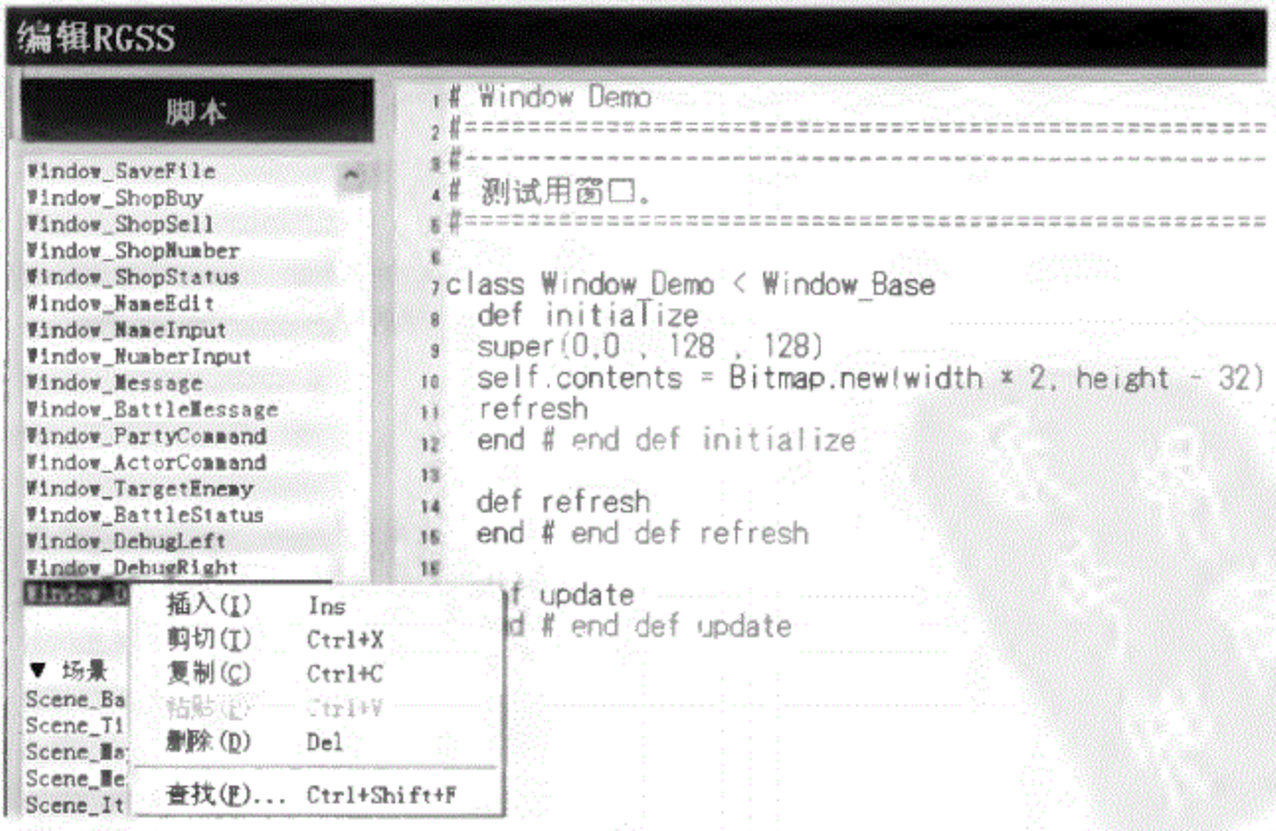


图 8-1

定义一个 Window_Base 的衍生类 Window_Demo。

```
class Window_Demo < Window_Base
  def initialize
    super(0, 0, 128 , 128)
    self.contents = Bitmap.new(width * 2, height - 32)      # 窗口内容
    refresh
  end
def refresh
    self.contents.draw_text(5, -5, 128, 32, "Test String")
  end
end
```

选择 Scene_Title，找到 start 函数，在结束前增加一个 Window_Demo 的实变量：

```
def start
    ...
@demo_window = Window_Demo.new
end
```

这样将会使 Scene_Title 初始化的时候构造一个 Window_Demo 的对象并显示。找到 terminate 函数，在结束前增加 @demo_window 的销毁操作

```
def terminate
    ...
    @demo_window.dispose
end
```

这样，在标题选单界面结束时，新增加的窗口也将被关闭。

保存并执行，执行结果如图 8-2 所示。

可以看到新增的窗口显示在左上方。重载窗口类的 update 方法，可以实现窗口的动态刷新。注意，需要在外层窗口中的 update 方法中添加对新增窗口 update 的调用。

同理，可以实现对场景类的继承和扩展。

（2）实例 Demo2：读取文件数据

RGSS 中定义以下内部函数用于读写数据。

```
load_data(filename)
save_data(obj, filename)
```

读取 filename 指定的数据文件，重建对象。

用户可以读取加密档案文件，向 filename 指定的数据文件里写入对象 obj。

```
$data_actors = load_data("Data/Actors.rxdata")
save_data($data_actors, "Data/Actors.rxdata")
```

图 8-2

但在实际的程序开发中，可能经常会遇到自定义文件需要进行读写操作。下面我们将提供一个读取文件内容的例子。

在前面的 Demo 工程中，新增加一个模块。

```
class FileReader
  FILE_NAME="test.txt"
  @@content=[]                                    # 读取到的内容

  def FileReader.read
    @@content=[]
    File.open(FILE_NAME) do |file|                # 读取文件
      while line=file.gets                        # 逐行读取文件内容
      @@content.insert @@content.length, line.chomp   # 加入内容数组
      end
    end
  rescue
  print $!                                        # 如果出错则显示错
                                                    误消息
  end
  def FileReader.content                          # 内容访问
    @@content
  end
def FileReader.save                               # 存储修改过的内容
```

```
    File.open(FILE_NAME , 'w') do |file|
      content.each do |line|
        file.puts line
      end
    end
  end
end
```

在 RGSS 中，内部函数 p 和 print 被重定义为输出到信息框。

除了 file.gets 方法依次读取文件内容，还可以用 IO.readlines 直接将文件内容读入到一个数组。

修改 Window_Demo 的 refresh 函数：

```
def refresh
    FileReader.read
    var = FileReader.content
    i = 1
    if var != nil then
     var.each do |str|
     self.contents.draw_text(5, -5, 128, 32 * i, str)  # 在窗口中显示所有的行
       i +=1
     end
    end
end
```


具体的执行结果如图 8-3 所示。

图 8-3

如果需要保存文件，请参考 RPG Maker 中的帮助文件。

小结：本书虽然介绍了两个脚本的实例，但是使用脚本可以开发出更多、更丰富多彩的内容，能够更多地对 RPG Maker 进行扩展。