J2ME & Gaming

Jason Lam 著
Deaboway Chou 译

http://www.jasonlam604.com/
http://sourceforge.net/projects/j2megamingbook/

译者	日期	内容	版本
deaboway	2004-11-20	创建文档	0.0.1
deaboway	2004-12-9	完成英文版本 0.5.6 的翻译工作	0.1.0

作者	日期	内容	版本
Jason Lam	2003-9-1	创建文档	0.5.0
Jason Lam	2003-9-18	在文档中并入所有已完成章节	0.5.1
Jason Lam	2003-10-4	在 JEMBlazer 中加入一个简短的介绍 在有限的章节中的第 10 章（提高可用性）中添加暂停、删除、保存组件	0.5.2
Jason Lam	2003-10-13	完成第 12 章	0.5.3
Jason Lam	2003-11-29	完成第 18 章	0.5.4
Jason Lam	2004-5-8	完成第 17 章	0.5.5
Jason.Lam	2004-6-30	完成第 13 章，创建第 14 章，增加了第 19 章的部分内容 将一些NPC行动代码分配给Jonathan Knudsen完成 由Leeman Cheng 和 Carlo Casimiro两人完成艺术设计的工作	0.5.6

译者　序

I have a dream! It's "**Play Games Everywhere…**".

我有一个梦想，就是"**让游戏无处不在**"!

时光匆匆滑过了 2004 年，也带来了很多改变。从 MIDP 2.0 发布开始到现在已经有两年多了，它的出现让移动设备上的游戏开发更加方便、也更为简单快捷。当我一年前发现 MIDP 2.0 技术的时候，我同时感觉到我们离随时随地玩游戏的那一天不再遥远。

每一个人都会玩游戏，都喜欢玩游戏；但是并不是每一个人都会开发制作游戏。要让游戏无处不在，还需要更多的人学会开发游戏。相信好学的你在玩过几个好玩的游戏后总会问：这个游戏是怎么做出来的？我是否也可以做出同样出色的游戏？我该如何学习游戏的制作？……　游戏有很多种——这在本书中会讲到——在当今的 2004 年如果想找一种游戏，它甚至只需要一个人业余很少的时间和精力就可以制作出，而且有可能会非常受欢迎的游戏，我想非移动游戏莫属！

本书就是 J2ME 移动游戏（主要是手机游戏）制作的入门读物，只要你具备 JAVA 编程的基础知识并且了解一些 J2ME 的背景知识，通过本书的学习，你就能创造出自己的游戏。当然，最最重要的还是读者你的创意！毕竟**知识有崖，创意无限**！

译者本身也是一个业余的游戏制作者，本着自由软件的精神翻译这本书，读者可以在确保完整性的前提下自由的分发本电子书，出版本书请联系译者，出版原英文版请联系原作者。由于本书原文还没有完成，译者将在原作者写作的同时进行翻译。由于翻译都是在业余时间进行，在工作了一整天后，精力难免难以集中，本书中肯定会有很多错误，期待读者的批评指正，译者绝对知错就改。☺☺☺☺☺

接下来，译者准备制作几个基于 MIDP 2.0 的手机游戏，并且打算翻译 J2ME API，感兴趣的朋友请跟我联系。此外，也希望同对 RUP 感兴趣的读者交个朋友。

译者的联系方式如下：

- E-mail/MSN: deaboway@hotmail.com
- E-mail: deaboway@163.com
- OICQ: 77337595
- Blog: http://deaboway.blogchina.com/

译者

2004 年末

内容简介

本书涉及在无线设备上使用J2ME技术的程序设计，主要介绍游戏的开发。需要读者具有一些J2ME和J2SE的相关知识及程序设计的经验。书中不会过多地描写类似如何创建高级的GUI菜单的方法，但是会演示游戏的主菜单究竟是什么样子。当然也不会解释如何使用RMS（Record Management System），但是会使用RMS技术实现高分记录和游戏设置功能。同时，读者具有线程的相关知识和经验对进行游戏开发比较有帮助。总而言之，本书不会过多地关注J2ME的具体知识，而是着眼于实际游戏相关功能的实现。不过，书中将具体介绍一些在MIDP2.0中新加入的与Game相关的类。

本书也可作为对手机游戏开发感兴趣的Java程序设计人员的快速参考。同时，也可作为具有其他语言或者平台的游戏开发经验并且想使用J2ME开发游戏的游戏开发者的优秀入门读物。

目　录

第一篇　基 础 知 识

- 绪论
- 移动游戏的约束
- 编码之前
- MIDP2 新加的游戏类
- 数学约束

第一章 绪论

1.1 本书的目标

欢迎来到移动游戏的世界！第一篇（第一章到第五章）主要包含如下内容：

- 移动游戏产业的方向
- 开发自己移动游戏可以使用的工具
- 开发移动游戏可能会面临的困难
- 游戏设计和开发过程中要注意的事项

然后，剩下的章节将介绍一个基础游戏的建立过程，虽然游戏本身并没有太大价值，但是它的确提供了设计一个游戏、特别是基于 J2ME 技术的移动设备的游戏的基本流程。除了介绍游戏本身，它还涉及了移动设备程序开发的其它领域，例如游戏主菜单的界面设计。

1.2 移动游戏

1.2.1 时机成熟

移动游戏就在眼前，时候到了！随着游戏开发技术的进步，现在你可以随时随地的玩所有你喜欢的游戏，从Pac-man, Bust-A-Move到具有更加交互性的规则的游戏。然而，移动工业领域的游戏和应用程序却仍然处在它的幼年期，开发人员、制造商、运营商都一直在寻找一个震撼人心的游戏或者应用程序的到来，从而促进移动设备工业的革新并带来数以百万计的税收。诚然，图像越来越华美、游戏性越来越完善，但是每一个手机游戏的用户却只有寥寥几千名而不是上百万。这对我们游戏开发者来说是好事，它留给我们很大的成长空间，有数不清的金钱让我们去兑现。

1.2.2 与传统游戏的差别

现在是独行侠的大好时机，因为 J2ME 技术需要相对较少的花费和工作任务。这就好像回到了过去一个人就可以在地下室搞定程序并且制作出完整的游戏的时代。比较一下当前的移动设备时代和以前的任天堂（Nintendo）时代，现在可以称为 Atari 时代的开元。大约在一年之前大量的经典游戏在移动设备上被重新制作，同时，游戏的图案的进化却是比较缓慢，当前的游戏非常类似于 NES 或者 Super NES 上的游戏。

移动设备还处在黑暗时代，只能提供很少的资源给图案、人工智能（AI）和多用户同时游戏功能。它不需要60个美工、20个程序员和先进的图形图像设备所组成的团队来开发一个为期一年的游戏，相反，移动设备的游戏开发可以由1到5人组成的小组容易地开发，一般预算只有几千美元到上万美元，而不像PC游戏世界的动辄上百

万美元的预算。最近，移动设备上的游戏开始实现网络通讯，从而支持多用户同时游戏。手机制造商、运营商或者其它组织并不关心你做什么样的游戏，不像PC和控制台上的游戏往往基于几个要素或者几个人的决定，这里的几个人却通常不是游戏开发团队，而是对利润感兴趣的组织，比如市场营销部门。大多数移动游戏开发工具和标准都是被原始制作者、制造商和运营商开放的。此外，传统游戏世界需要寻找针对Xbox、PlayStation或者其它控制台的各种信息和资料，而且同一个游戏很难运行于所有这些控制台。移动游戏在一定程度上克服了这个缺点。

总而言之，移动游戏和传统游戏的区别如下：

- 低预算、低成本
- 开发团队小
- 技术易于管理，不需要像图形图像设备这类特殊的设备
- 基于开放的标准
- 移动电话本身的特点为支持多人同时游戏铺平了道路

1.3 开发工具

移动游戏开发的一些可用平台如下：

- SUN公司的J2ME(Java 2 Micro Edition)——http://wireless.java.sun.com/
- MoPhun——http://www.mophum.com/
- Brew——http://www.brew.com/

这里没有将移动设备的操作系统（例如 PalmOS 和 SymbianOS）加入，是因为这些操作系统本书可以支持 J2ME 环境。

本书将主要介绍J2ME（Java 2 Micro Edition），这里有一些软件开发包可以使用。为了使你开发的游戏可以在不同的制造商和不同型号的移动设备上正常运行，你需要使用尝试很多不同SDK。最主要也是最常用的SDK是由Sun公司开发的J2ME SDK，可以由http://wireless.java.sun.com/下载，其它开发包的下载地址如下：

开发商	网址
Nokia	http://forum.nokia.com/
SonyEricsson	http://www.sonyericsson.com/
Sony Clie	http://www.cliedeveloper.com/top.html
Motorola	http://www.motocoder.com/
Nextel	http://developer.nextel.com/
RIM	http://www.blackberry.net/developers/na/index.shtml
Siemens	http://www.siemens-mobile.com/
SprintPCS	http://developer.sprintpcs.com/adp/index.do
Palm	http://www.palmsource.com/developers/
Symbian	http://www.symbian.com/

BellMobility	http://www.developer.bellmobility.ca/

有一些用于J2ME开发的IDE。一些IDE能完全整合了其它的开发包，例如Sun One Studio Mobile Edition 可以很容易地整合SprintPC J2ME SDK、Sun公司的SDK、Nokia公司的SDK以及Siemens公司的SDK。

厂商	网址
Sun One Mobile Edition	http://wwws.sun.com/software/sundev/jde/
Jbuilder Mobile Edition	http://www.borland.com/
Code Warrior Wireless Studio	http://www.metrowerks.com/MW/Develop/Wireless/Wireless_Studio/default.htm
WhiteBoard SDK	http://www.zucotto.com/home/index.html
IBM Studio Device Developer formly know as IBMVisualAge MicroEdition	http://www-3.ibm.com/software/wireless/wsdd/

1.4 其他

除了各种软件组织、制造商和运营商提供的工具和 IDE 外，想必你会对下面介绍的工具和软件包感兴趣。

1.4.1 3D

虽然由于移动设备的限制现在的很多移动游戏都是 2D 的，但是还是在一些高端的移动设备上实现了 3D。当然，技术的发展也使游戏工作站和 PC 机上的 3D 游戏更加普遍。

J2ME中正开发的支持3D图形的技术有JSR，在MIDP 2.0 3D JSR 184发布后，JSR有可能会发布新的版本——*Mobile 3D Graphics API for J2ME*，想了解更多的信息请参考http://jcp.org/en/jsr/detail?id=184。

被广泛用于3D图形显示的OpenGL也开发了针对移动设备的版本——OpenES，更多信息请参考http://www.khronos.org/embeddedapi/index.html。FatHammer公司（http://www.fathammer.com/）已经将OpenES OMAP平台整合到他们的X-Forge的3D游戏引擎的SDK中。

还有一个3D引擎是MoPhun，请参考http://www.mophun.com/。

从现在的移动设备情况看，虽说谈3D还有些为时过早，但是这方面的技术的确在稳步地发展中。

1.4.2 多人游戏

除了单机游戏，有大量的其他技术来开发出多人游戏。

J2ME 技术默认支持通过 HTTP/HTTPS 的通讯，另一个选择是使用被称为 Jini 的代理 J2ME 体系结构。如果运营商支持 socket 通讯，我们还可以在游戏中通过扩展 socket 实现联网。然而，这样就不仅仅需要通讯协议的支持，还需要一个可以生成和处理各种相关信息的游戏服务器，这些信息有可能是：

- 游戏大厅
- 游戏场所
- 跟踪用户的交互
- 身份验证
- 聊天室
- 即时信息
- 监控和统计工具

游戏场所本身就扮演仲裁者或者交警的角色，它可以中转信息给用户或者给当前运行的各个游戏。

实现这个功能的确需要做大量的工作，然而，这里有一些公司已经提前做好了游戏服务器：

- DemiVision－http://www.demivision.com/（已经整合到JamDat.com）
- Mforma－http://www.mforma.com/
- Xadra－http://www.xadra.com/
- Butterfly.net－http://www.butterfly.net/
- TerraPlay－http://www.terraplya.com/

不仅仅是开发基于广域网的游戏，还可以开发小区域网的游戏，这项技术叫做蓝牙。蓝牙实现的功能与基于 HTTP 协议实现互联的游戏类似，但是蓝牙有一定的地域限制，一些比较好的与蓝牙和 J2ME 技术相关的链接如下：

- Zucotto Wireless－http://www.zucotto.com/
- RococoSoft－http://www.rococosoft.com/

此外，还有其它的技术可以实现移动游戏的联网，其中有因为Napster的成功而最广为人知的P2P技术。Java技术的P2P技术可以访问http://www.jxta.org/和http://jxme.jxta.org/。另一个由苹果公司开发的P2P技术是Rendezvous，它支持自动广播及客户端和服务器端之间的发现服务。

1.4.3 运行移动游戏的设备

不光是蜂窝电话，移动游戏还可以运行在 PDA、SumbianOS、微软的 SmartPhone 上，虽然它们中的一些并没有安装运行 J2ME 程序和游戏的 JVM 环境，也就是说你

也许要在开发过程中自己安装 JVM 环境。

也存在像 JEMBlazer 这种小生境设备，有关 JEMBlazer 的介绍如下：

“是不是厌倦了在只有小小的屏幕、困难的控制、差劲的系统性能的手机上玩移动游戏？aJile Systems JEMBlazer Jflame扩展卡将使你的GBA或GBA SP支持Java游戏，从而使你可以随意地下载因特网上的各种Java游戏进行游戏。JEMBlazer基于Sun公司的MIDP，它可以运行基于MIDP1.0/2.0及其它多媒体MIDlet开发的游戏。同时，JEMBlazer扩展卡也具有超炫的游戏音乐，使你尽情享受究极的MIDP游戏”

更多有关JEMBlazer的信息请访问他们的网站http://www.jemblazer.com/。

图 1：JEMBlazer 的庐山真面目

1.5 开天辟地第一步——开始开发

1.5.1 开个小头

以上介绍了支持移动游戏开发的工具和各种技术，在阅读本书中的例子之前先要确定制作哪种类型的游戏。如果你尚没有任何游戏编程的经验，就需要从简单的游戏编程开始。编写一个单机版的游戏不需要太多的可视和交互的界面，但是仍然可以使你对游戏开发以及开发一个完整的游戏需要的信息有很好的理解。不要试图一蹴而就，第一个游戏就是白金销量，你会遇到很多障碍，也许就会因此而气馁，从此一蹶不振。先完成一个简单的游戏并体会成功的快感，然后在做一些困难的游戏吧！

一般来说，以图片较少的简单场景切换游戏开始比较好，因为它的图形和线程都比较容易控制。这里面的佼佼者是 TicTacToe，从这里你可以学到游戏的运行是如何实现的，也可以学习简单的人工智能。也许下一个选择就是学习开发类似 Tetris

的游戏，通过对这种伟大游戏的模仿，可以从中学到游戏的基本交互、用户控制/输入、图像处理、平铺显示的使用。然后你可能想要开发一个类似于 Space Invaders 的基础射击游戏，或者像 Mario Brothers 一样的卷轴游戏。在开发了所有这些游戏之后，可以说你应该具有了开发更为复杂的游戏的能力。当然，开发更为吸引人的更具有可玩性的游戏需要更加出色的图像和人工智能。

1.5.2 游戏的种类和流派

游戏有多少种呢？日常所见的任何游戏都可以区分到特定的流派下，可以说这个世界上没有全新的游戏！一些游戏可能看起来超炫的，但是如果你将它们拆开来看的话，它们仅仅是重新应用了以前游戏的一点或几点好的创意。像DukeNukem，Doom，Quake，Freelancer，Counter Strike，Return Castle Wolfestein。它们其实都是同一种类型的游戏，唯一的区别是有的是2D的而有的是3D的。

游戏可以分为以下几种类型：

种类	描述
动作游戏(Arcade/Action)	快节奏、图像丰富、高度交互性的。如 Quake 和 Freelancer
益智类游戏(Puzzle)	需要逻辑推理的游戏，如 Tetris
卡片游戏(Card)	类似 Poker、BlackJack 的游戏
战略游戏(Strategy)	需要大量思考和战术移动以及微操，如 Command and Conquer、Warcraft
角色扮演游戏 (Role Playing Game)	需要用户遵守游戏规则，而且通常包括长时间的人物塑造，如 Dungeons and Dragons
运动游戏(Sport)	和运动相关的游戏

表中是最常见的游戏类型，当然还有其它游戏类型像琐事类游戏(trivia)、模拟游戏(simulation)、机会类游戏(chance)如 Magic 8 Ball 等等…

1.5.3 学习游戏创意

一般情况下，你以爱好或者学习的目的重新制作一个游戏不会有什么问题，但是如果你想克隆一个商业游戏并把它推向市场就会有麻烦了。然而，重新制作和克隆游戏之间的界限并不是非常明显的。例如，你或许看到市面上 N 多类似于 Tetris 的商业游戏，那是因为它们都没有使用 Tetris 这个具有版权的名称。而所谓的界线也就是看你模仿的是内容还是游戏创意了，这的确比较难以区分。如果你确实计划制作一款商业游戏，你最好尝试获得一个自己原创的创意或者整合其它一些成功游戏的各种创意于一身。如果你的游戏类似于市面上已有的游戏，很多游戏发行商不会愿意发布你游戏的，在游戏的生命周期只有 3 到 6 个月的移动工业中更是如此。即使你只是由于个人爱好发布一个游戏，大多数人还是会选择全新的游戏而不是对 Tetris 的另一份克隆。

1.5.4 竞争

在任何一种产业中，都可以通过向其他人学习并评估究竟什么是没有被完成的，来了解这种产业当前的竞争状态。

下面是一些可以了解行业最新资讯的网站：

- Midlet Review – http://www.midlet-review.com/
- Midlet.org – http://www.midlet.org/
- Midlet Centrel – http://www.midletcentral.com/
- Micro Java – http://www.microjava.com/
- AllNetDevices – http://www.allnetdevices.com/
- WirelessDevNet –http://www.wirelessedevnet.com/
- Infosync – http://www.infosync.no/
- WGR (Wireless Gaming Review) - http://www.wirelessgamingreview.com/

下面是已存在的你将面对的竞争者的列表，当然他们也有可能成为你的同盟或伙伴。下一节我们将详细讲述如何将游戏推向市场。

AbstractWorlds
AirG
AnfyMobile
Astraware
Atatio
BlueSphereGames
CheekyGroup
Cocoasoft
Codetoys
CoffeebreakMedia
Comzexx
CovertOperations
DigitalBridges
DistinctiveDevelopment
Dreamquest
DSEffects
Eemo
Elkware
FutureDesignGroup
Gameloft
Getsnax
HandyGames
Hudson
Ifone
In-Fusio
Iomo
Jsmart
Jamdat
livingmobile.net
Macrospace
Mforma
MicroJocs
Mobilegames
MobileScope
Mobizexx
MoBro
Morpheme
Mr.Goodliving
Notthefly
Paletsoft
Perimind
Picofun
segawireless
SoftexIndia
Sorrent
Sumea
Toomtam
THQWireless
WES

注意：这个列表并不完整。

1.6 推向市场

现在你已经有了工具和游戏创意，在后面的章节你还将学习如何做一个游戏，问题是怎么样从自己做的游戏中赚到钱？

你有以下几种选择：

1.6.1 直接出售

你可以建立自己的网站，并且通过已有的网络贸易商如 PayPal 收取费用。如何收取呢？首先，如果用户将游戏下载到 PC 机，他如何将游戏上载到他的手机中？随后的问题是用户是否知道如何将游戏上载到手机中，他们是否有上载需要的数据线？所以，如果想更为成功的出售自己的游戏，直接提供 OTA 下载不失为一个好方法，这就意味着你需要提供一个自己的计费系统，而不是使用 PayPal 这种网络。注意，在使用 PayPal 这类系统的时候，要亲自处理用户的订单，并且为下载游戏创建可用的临行空间。

另一个可以直接出售游戏却不用建立自己的服务器的方式是将游戏放到网上商店，如Handago：http://www.handago.com/，Handago将会从出售的游戏中获得一定百分比的佣金。但是通过广告和与单个用户的直接联系来出售的游戏都是很有限的，用户需要直接登陆Handago去购买游戏。

1.6.2 加入集团(Aggregator)

另一个选择是加入一个集团。集团好像开发商和将产品发放到他们的各个用户手中运营商之间的中间人，这对不想运作自己的公司并且对各种商业文件深恶痛绝的独立开发者有很大吸引力。与集团合作还是非常简单的，你要做的只是签署一个NDA、签订一个协议、上交一个完成的游戏，集团来处理市场和收费。当然，不同的集团会有不同的条款和份额制度，你需要调查一下哪个集团适合你。最需要关心的当然是收费模式，SLA(Service Level Agreement)意味着你提供的支持的类型，如果是非独占的说明你可以通过其它集团出售你的产品，或者说你可以与其它集团合作。

1.6.3 游戏发行商

游戏发行商和确定的游戏组织可以帮助你发布你的游戏。你可以直接联系游戏公司看他们是否有正在进行的项目，如果有的话，你就可以争取在其中占有一席之地。

1.6.4 运营商

直接与运营商建立联系也是一个不错的选择，但是现实是运营商一般只与大的确定的游戏公司合作，这些公司往往已经在游戏领域有了一定的声誉，不过也是开始制作移动设备上的游戏。运营商也倾向跟这样一些集团合作，他们会处理开发者不愿意处理的事情，也会处理运营商不想处理的任务。

1.6.5 制造商

一些移动设备制造商如 Nokia 会有与开发组织有市场合作的机会，你可以与你有兴趣合作的制造商联系来获取详细的资料。

1.6.6 无线产品代理商

也可以与无线产品代理商合作，类似于电影工业的代理，他们着眼于营销你及你的能力和业务。

1.7 总结

如果想将游戏市场化，在移动游戏开发过程中要注意 N 多问题，如果是最初进入移动游戏开发领域的，这里有大量的工具和资源，想必有所裨益。当然，你也可以仅仅是以编写移动游戏为爱好！如果你对游戏编程没有兴趣，最好还是思考一下做其它一下感兴趣的事比较好一些。由于爱好或者追求利润而进行的游戏编码是在计算机编程中很有挑战性的工作之一，特别在技术瞬息万变和竞争日趋激烈的今天。

如果你想将你的作品市场化，下表列出的一些组织会有所帮助：

公司名称	公司类型	URL
AirG	游戏开发公司	http://www.airg.com/
AnfyMobile	游戏开发公司	http://www.anfymobile.com/
Nokia	制造商	http://forum.nokia.com/
WirelessDeveloper	代理商	http://www.wirelessdeveloper.com/
TiraWireless	集团(Aggregator)	http://developer.tirawireless.com/
Nextel	运营商	http://developer.nextel/
T-Mobile	运营商	http://www.tmobile.com/
CellMania	集团(Aggregator)	http://www.cellmania.com/
Cingular	运营商	http://alliance.cingularinteractive.com/
Telus	运营商	http://www.telus.net/
BellMobility	运营商	http://www.developer.bellmobility.ca/
4thPass	集团(Aggregator)	http://www.4thpass.com/

第二章 移动游戏的限制

在开发 J2ME 移动游戏过程中，你会很快地意识到最大地困难和障碍是，本来在 PC 机和控制台上习以为常的应用在移动设备上却遇到各种限制，包括内存的、屏幕尺寸的、甚至色彩的限制。

由于游戏中的用户输入、图形图像、动画、声音和振动的高交互性，这种限制在移动游戏中表现得比移动应用程序更加明显。此外，你不仅要注意不同制造商之间产品的不同，也要了解同一个制造商的不同移动设备模型的差别。不同手机的模式在内存、色彩、屏幕尺寸和用户界面等各个方面都有很多不同。

2.1 内存

2.1.1 内存的分类

一般情况下，内存被认为是内存区域中的堆栈，在游戏执行的时候储存信息，在游戏终止后被释放。读者需要查阅制造商手册来获得确切的规格说明。如果游戏需要的内存空间超过了移动设备可以分配的大小，游戏一般就不能够运行了。

可以算作内存的还有被称作 RMS(Record Management System)的存储空间。读者需要确定你打算开发的游戏所针对的机型允许使用的空间的总大小，可能的话，建立一个选择逻辑来处理内存耗尽情况的发生。你也许会问 RMS 的作用是什么？在游戏开发过程中，你也许想存储最高分或用户首选项，或者需要提供给用户一个选项来保存上一次游戏的进度来使用户可以继续游戏，这些都是 RMS 可以做到的。简单来说，RMS 就是一个相当于 ROM 的存储空间。

2.1.2 存储空间的碎片

PC 机上的硬盘在使用一段时间后就会出现碎片，在手机的存储空间上也会出现这种情况。例如需要写一个大的对象到内存中，但是却找不到足够的一整块连续的空间，于是系统就将它分成几块分别存储在内存的各个空闲空间中。这不但会导致存储空间访问时间的增加，当内存中大的对象被清除后会留下很多内存空洞，从而导致新的对象被存储时也会遍布内存各处。

2.2 显示大小和分辨率

除了内存，你还要考虑的每个移动设备的显示大小。拿 Sony Ericsson P800 来说，它的屏幕分辨率是 208x320，而 Nokia 3650 的分辨率就是 176x208，所以你需要考虑发布对应大小的特殊版本的游戏。而且 Nokia 也许支持大小是 16x16 的精灵，P800 支持的精灵的大小也许就是 32x32。如果你对精灵(sprite)不太了解，本书的后面会有相应讲解，现在可以理解成图形图像。

图 2：不同手机显示大小不尽相同

虽然越来越多新近开发的手机都是彩屏的，但是还是要考虑到已经存在的数以百万计的黑白屏幕的手机用户。

2.3 手机界面

最近几年，各个手机制造商开始放弃传统的手机界面，进而追求更为个性化的按键和控制，这归结于制造商对更加友好的用户界面的孜孜不倦的追求。虽然一些制造商已经意识到没有绝对最好的用户界面，只有对某些有特殊需求的特定的群体比较好的用户界面，这主要取决于用户除了使用手机打电话外还做什么，比如听音乐、玩游戏、或这记录商业事件。当然，新界面的设计也扩展了手机市场的边界和吸引力。这对游戏开发者来说有什么意义呢？这又增加了开发者需要考虑的一个因素，比如，你把数字键“5”设定为开火，但是也许在某些手机上这个键不是很方便就可以按到的。下面是现在市场上存在的一些不同类型不同外观的手机：

图 3：手机界面各不相同

2.4 缺少类文件

由于 J2ME 是 J2SE 的子集，所以它并没有包含所有的包和类。也就是说你要么不使用这些类，要么自己扩展所需要的类。例如，J2ME 中不支持浮点数(float)，故不能进行小数运算，也不能使用 sin、cos 和 tan 运算，但是在游戏开发中，小数是经常要用到的。然而，可以用定点数学运算的方式来代替小数运算。比方说，如果想精确到小数点后三位，你可以用 1000 来表示 1.000。在变换的过程中要不断的乘以或者除以 10，以得到正确的结果。至于角度可以用 0、30、60、90、120、150、180、210、240、270、300、330、360 这种对应角度值的整数值来代替，事实上由于移动设备的屏幕往往都很小，所以太精确的计算也是没有意义的。第五章将详细讨论数学的约束。

还有一些其它的类有没有包含在J2ME中，在没有用的它们之前你也许不会感觉到。像float类已经有一些人预先写出了可以使用的对应类，如果你要使用的类在J2ME中找不到，切记先在网上搜索一下，如果没有才需要考虑自己去写。这里有一个可用的浮点数类，参考http://henson.newmail.ru/j2me/Float.htm。

2.5 开发者和使用者眼中的性能

在测量性能之前需要理解一些概念。性能究竟是指什么？通常意味着响应时间和吞吐量。响应时间又是指什么呢？是不是指返回一个特定的函数，抑或是用户得到了游戏的一些反馈信息？不同点在什么地方？例如，当游戏在加载的时候，用户看到了指示加载状态的信息，但是事实上游戏的加载并没有完成。吞吐量通常是指在给定的时间内完成的工作量。另一个重要的性能指标是网络游戏的网络延时，通常是指处理任务的最短时间。所以在你开始测量性能之前，你需要澄清性能的涵义以及如何测量性能。

用户对性能的理解主要基于感觉的性能和反应的性能，即用户看到的或者从游戏得到的反馈。游戏中永远都不要有没有任何反馈的暂停，例如初始化加载游戏资源的时候或者有网络通讯的时候。在这种耗时间的任务执行的时候，必须给用户一个提示信息，如游戏正在加载中，或者有一些反馈的信息，否则，用户会怀疑是不是游戏有了什么问题。

2.6 提高性能

2. 6. 1 先编码，然后考虑性能的提高

首先，在设计和开发游戏的时候，首要得规则是不要考虑任何性能方面的因素只专心的编码。最初编码的时候，需要将注意力集中在保持代码整洁、可运行且易懂上。然后才谈得上性能评价和修正该修正的地方。否则，你所能得到的也许仅仅是时间的浪费，甚至由于不必要的性能扩展阻碍了游戏的开发。通过剖析程序代码

你也能发现最经常被执行到的部分，增加仅仅会有 10%的几率执行到的那部分代码的执行效率是没有什么意思的，这常常被称为二八原理、或一九原理，所以要尽量增加最经常被执行到的部分的执行效率。提高效率的好的方法是，测试被更改代码确定的分支的，通过比较于更改前效率的差别来确定更改后是否对效率的提高有帮助，还是仅仅提高了一点点效率却给代码带来了很大的混乱，这样就得不偿失了。当对代码的更改对效率的提高不明显时，建议还是维持代码的抽象性、复用性和易维护性比较好。

2.6.2 减少面向对象的代码

通常，面向对象的语言往往趋向于更多的类、更多的使用帧、设计模式被用于几乎每一个设计中。这种情况通常是好的，但是在移动领域，由于内存的限制，面向过程的程序设计方法有时候是更受欢迎的。虽然不太容易让人接受，你需要避免使用 MVC(Model-View-Controller)模式，因为这种设计模式会让你的代码量呈指数性增长。话虽这样说，也不要完全按照面向过程的方式编码，只要有这样的想法就好了，当面对性能或内存的问题时，你也许需要用面向过程的方法代替面向对象的编程方式来解决。

2.6.3 减少第三方库的使用

上面已经讲过了尽量减少类的使用，当然也要避免使用第三方的 API 来存储空间和运行空间的使用。例如，在网络通讯时使用自己定义的数据解析可能会提高游戏的运行效率，要避免使用一个更加抽象的工业化定义的通讯方法如 SOAP 网络协议。因为这个协议不仅会增加程序的大小，也会带来解析和建立 XML 数据传递的额外过程。

2.6.4 最少的通讯

为了提高网络通讯的性能，需要避免不断地从服务器取回数据。例如，Web 开发中，大的图像被分成很多小的部分然后分别取回，但是在移动设备的程序中，为了避免延迟出现几率的增加，一次取回所有要用到的资源是明智的。

2.6.5 组合图像

每个游戏都会有的资源是图片，通过将游戏用到的各个图像组合到一个大的图像中，不但可以减少网络游戏中的延迟，也可以减少游戏的大小。因为每个图片的开头都包含预定义的信息，如果有 10 个图片那么就有 10 个相同的这种信息。当然需要从组合图像中析取单个的图像的算法，后面会详细讲述。

游戏中的图像要尽量最优化并且压缩。但是要注意，并不是所有的图形图像优化工具的默认选项都是完全符合要求的。

2.6.6 垃圾回收

作为一名 JAVA 程序员，虽然我们不能不使用 JVM 的垃圾自动回收机制，但是我们可以通过一些措施来促进垃圾回收，如将不再使用的对象赋值为 NULL，又如使用对象池重新利用对象从而避免建立新的对象。也可以通过调用 System.gc()或者 Runtie.getRuntime().gc()来强制执行垃圾回收，顺便提一下，有一些人相信这种方法是可行的，而另一些人则不这么认为。为了防止陷入另一个永无止境的争论，建议读者还是按照自己的喜好处理内存，并且自己决定采取何种方式。但是要注意在不同的移动设备上相同的代码可能会有完全不同的执行结果。

2.6.7 短小的名称 及 混淆机制

同减少类的总量相比，通过减少变量名、类名、方法名的长度来压缩应用程序的大小，不失为一个更好的办法。如果可能的话，只使用默认的包，而不要创建不需要的包结构。

如果将所有的变量、方法、类的名称都用一个或两个字母代替，那程序代码的可读性就会大大降低，这里介绍一下混淆器的概念。使用混淆器来处理代码后，它会将代码中长的名称都替换成简短的名称，从而使代码更短小。同时也带来了另一个好处，就是通过混淆器编译后的代码，就算再被反编译也很难读懂。如下是一些可用的混淆器：

- ProGuard – http://proguard.sourceforge.net/
- SourceGuard – http://www.javalobby.com/
- RetroGuard – http://www.retrologic.com/
- CodeShield – http://www.codingart.com/codeshield.html
- yGurad – http://www.yworks.de/en/products_yguard_about.htm
- IBM JAX – http://www.alphaworks.ibm.com/tech/JAX
- JShrink – http://www.e-t.com/jshrink.html

2.6.8 编码技巧

下面的编码技巧不一定对性能的提高有所帮助，请根据读者自己的判断决定是否使用。

1. 用 StringBuffer 代替 String，因为 String 对象不能被更改，任何对 String 对象的更改都是创建一个新的 String 对象。
2. 直接存取类中的变量要比通过 setter 和 getter 方法快。
3. 使用本地变量比使用类或实例变量更有效率。
4. 使用变量比使用数组更有效率。
5. 避免在循环中进行同步，因为每一次循环都会有 lock 和 unlock 的过程，会严重影响程序执行效率。
6. 循环中倒数（递减）比正数（递增）要快。

7. 使用类似于 x+=1 替代 x=x+1，因为这样生成的代码小。
8. 删除循环中的常量运算。
9. 重复利用对象。
10. 把不再使用的对象赋值为 null，特别是不再使用的 thread。
11. 尽量使用内置的方法。比如，想实现将数据从一个数组拷贝到另一个数组的功能，使用 System.arraycopy 比自己创建新的方法的效率更有效率。

2.6.9 好的编码习惯

虽然这对性能没有直接影响，但是无论在那种类型的开发中，都是比任何其它技巧都要重要的一个技巧。想必很多开发人员都用过类似 CVS 的版本控制软件，更多的相关内容请参考第三章，我们将讲述如何使开发的设计和过程更为优秀。读者也许还需要考虑使用 Ant 这类工具编译代码。在开发过程中所做的控制越多，在问题发生的时候就越容易跟踪和查找

附录 A 示范了 J2ME 和 Ant 的使用，及一个称为 ProGuard 的混淆器的使用。

2.7 总结

在有各种各样的限制的移动设备上开发程序特别是开发游戏是很有挑战性的一种工作，然而，一旦掌握了提高执行效率的技术就可以在移动设备上创造奇迹！

第三章 编码之前

3.1 概述

无论你编写游戏是因为兴趣还是追求利润，特别是为了赚钱而编写游戏的话，在开发游戏的过程中理解和应用游戏策划和开发过程是极其重要。很多人认为这是浪费时间，甚至会扼杀创造力。游戏开发是很有挑战性的，但是除了对创造性、图形图像和动画的重视，与软件开发世界的其它工业也没有太大的不同。这在游戏业向多人同时游戏和 MMOG(Massively Multiplayer Online Gaming)游戏发展的今天更为明显。移动游戏开发也是如此，虽然现在的很多手机游戏只是单机版，但是对网络游戏的需求的确是日趋激烈的。

游戏业在渐渐演变成服务性行业。服务意味着什么呢？游戏本身就是服务，它们一般都是一星期 7x24 小时不间断服务，特别当游戏包含适当的预售服务的时候。除了最最重要的使用户可以随时随地访问外，在任何一个网络游戏中，身份验证、隐私和安全都是最重要的要素之一。

下面比较一下在线游戏和支持 CRM、供货渠道管理和在线订单的大型实时销售系统的差别。

方面	网络游戏	销售系统
优秀的游戏性	具有	—
优秀的人工智能	具有－需要智能的敌人和智能的友军	具有－销售前景预期、特征、跟踪顾客偏好、提供特色服务
直观的用户界面	具有－容易使用和适应	具有－容易使用和适应
永久的数据和数据库	具有－控制游戏信息、保存游戏、游戏特征设定	具有－保存所有条目、特征、补贴、顾客信息…比游戏需要更多的存储空间。
实时性	具有－减少延迟，提高吞吐量和反应速度	具有－减少延迟，提高吞吐量和反应速度
监控	具有－监控性能，防治潜在的问题，如用户作弊	具有－监控产品的整个流程、发生的一切问题
身份认证、隐私和安全	具有－加密私有数据，特别是具有适当的签名模块和信用卡交易的时候	具有－加密私有数据，如顾客的个人信息（地址、电话、信用卡等）
优秀的图形图像	具有－3D 渲染的游戏中会更多	具有－特别是消费者应用程序如网络商店，成功的市场竞争带来成功的销售

从上表可以看出，二者的特征几乎没有差别。只是某个系统的一些方面会有些

许不同，如类似 Doom 3 的游戏就需要大量的图形图像；另一方面，放射性产品的监控系统的精确性就要远远高于普通的多人游戏的服务主机。

这里要讲的是游戏开发与其它软件开发工作没什么两样，它们都需要优秀的策划和严谨的开发过程。

虽然上述讨论都是基于网络游戏的，但是优秀的策划和严谨的开发过程对单机游戏一样有用，这都会直接决定是否可以成功地按时发布一个在预算内的优秀品质的游戏。

3.2 游戏策划

游戏策划是每个成功游戏的必要条件。今天的游戏开发已经告别了野蛮时代，那时候，一两个编码人员躲在地下室或阁楼里也许就可以编写出激动人心的游戏。现代，这种情况不太常见了，一谈到游戏一般都是好几百万的预算。当然，这样说还为时过早，移动游戏还没有达到那样的规模，但这并不意味着好的游戏策划不再重要了。移动游戏的开发费用一般比 PC 机或控制台上的游戏开发费用低，但是相关的其它花费却是一样的。例如，各种资源的花费、市场策略的花费、广告的花费，还有基础设施建设的花费，尤其当游戏是多人同时游戏类型的时候。综上所述，要开发一款成功的游戏，需要从开发的最初就有大量的投入，这就需要优秀的游戏策划。

撇开商业层面不谈，如果你开发出一个垃圾的游戏，没有任何人会玩它，你作为独立开发者的名誉将失去光泽。随后介绍游戏的游戏策划的一些要素，简要地讲述了开发一款游戏的游戏需要做哪些努力。市面上有专门讲游戏策划的书，如果你对游戏策划不是非常了解，强烈建议你花一定的时间和精力在这些书上，这是非常值得的。

3.2.1 复用性 和 组件开发

在任何软件开发活动中都要考虑组件的复用，如人工智能算法、3D 运算引擎、甚至小到定制的 sprite 类。

除了在自己开发的几个项目中复用代码，你还可以购买或者使用开源代码。例如，当开发自己的数据库的时候，可以参考市场上可用来记录数据的几十种数据库。另一个要考虑的问题是，如果原本负责数据库系统开发的队友离开了团队，谁将接替他？一般情况是，在浪费了大量时间和资源后，在剩下的人中又出现一个数据库方面的专家之前，数据库这一块将是空白的。当然，如果你使用 mySQL 或 Oracle 这类已经存在的数据库，就会有成千上万个这方面的专家，也有数不清的厂商支持这种著名的数据库。

3.2.2 用户调查和目标用户

制作一款游戏首先要了解它的用户。例如，在北美足球游戏就没有曲棍球和棒球游戏受欢迎，在欧洲足球却是最流行的游戏之一；在欧洲的一部分和亚洲的大部分，成人游戏和赌博游戏是很容易被接受的，但是在北美包含色情和赌博内容的游戏常常被大众所排斥。用户调查中也要考虑用户的文化和传统，为有牛图腾的民族开发有宰牛场景的游戏是不明智的。

即使只在一个国家发行的游戏，也要考虑目标用户的年龄段、背景等等。高尔夫球游戏只能吸引高尔夫运动爱好者，而快节奏的完全是血腥的屠杀的游戏对青少年有很大的吸引力。也可以考虑将目标用户定位到少女市场、儿童市场等。

游戏用户可以被分成三类：

- **核心游戏用户**－这类人花费大量的时间和金钱在游戏上，他们甚至会为了游戏放弃其它。核心游戏用户会尽量的玩遍所有最新的和最有名的游戏，而且不会计成本。
- **适度游戏用户**－适度游戏用户同核心用户一样会花费大量的时间和金钱在游戏上，然而，他们不会沉迷于游戏中，在购买游戏的时候也会考虑到价格的因素。
- **普通游戏用户**－这种类型的用户只会玩那种非常经典的游戏，如 TicTacToe、Monopoly、Blackjack……

通过上述分类和当前移动技术的发展情况，将目标用户定位到普通类型和适度类型的玩家是比较明智的。

决定游戏针对的目标用户无疑是非常重要的，这将对整个游戏策划都有指导性的意义。

3.2.3 游戏种类

第一章中提到了游戏的类型划分，每一种类型的游戏都是下面一种游戏之一：

- 没有存档的单机游戏
- 有存档的单机游戏
- 没有存档的网络游戏
- 有存档的网络游戏

单机的意思是显而易见的，没有存档的单机游戏是指不能从上次游戏退出的地方继续游戏的那种游戏，而有存档的单元游戏可以。可以很容易地通过普通文件、XML 文件或者数据库实现。多人游戏一般都是可存档的，最常见的是 MMOG（Multiplayer Massively Online Gaming），用户可以从上一次退出时地状态继续游

戏，但是游戏本身和其它在线的用户却是不间断地改变着。这使 MMOG 变得很特殊，使 MMOG 不只是游戏，而是现实生活的人们感兴趣的一直社会活动。

3.2.4 长期的计划

虽然大多数游戏都是独立的，但是仍需要考虑是否做一些有意义的升级和补丁，如附加的地图、人物或者级别。这当然需要游戏被设计成组件模式从而易于扩展和维护。

3.2.5 技术

这样说也许会激起激烈的争论，很多人认为技术只是设计游戏时的工具。但是，另一方面技术本身也会约束什么事情可以做到而什么事情不能做到，在移动游戏开发领域这种约束就更为明显了。建议读者在游戏策划的时候保持开放的思想，不要被技术所桎梏，在策划的过程中也要考虑到当前使用技术的约束。

3.2.6 游戏性

好的游戏性要求有办法使用户有兴趣不断地一遍一遍玩这个游戏。游戏本身必须对用户有一定的挑战性，或者以技能、存储、思想、问题解决、获得高分乃至纯萃的寻找和发现的形式来满足用户，使用户通过一定的努力可以获得成就感。一定要注意，将目标定成不能轻松达到，但是也不能非常难以达到，不然就走上了逼迫用户放弃继续玩这个游戏的极端，可以说游戏难度的把握非常重要。还有两个要注意的要素，一个是角色的开发，要使用户在游戏进行的过程中可以完善游戏中的角色；另一个使游戏的平衡性。在 Pan-Man 游戏中，平衡性表现在鬼魂有能力抓住用户取不能抓住用户这种平衡。

3.2.7 真实性

游戏的真实性会增加游戏的吸引力，如果用户发现游戏太不可思议了，他就会对游戏比较失望。当然，太真实的游戏一方面会增加游戏制作的难道，也会导致用户的厌倦。

3.2.8 创意

虽然强调了策划的重要性和优秀的开发过程的作用，创意还是不容忽视的。诀窍在于找到创意和结构之间的平衡。说起来简单，真正要建立一种以前从来没有过的游戏类型真的很难。除非你的商业目标就是做一个大众都爱玩的类似 Connect Four 或 Tetris 的游戏，就算这样，一个全新的益智游戏也比新瓶装旧酒的游戏要受欢迎得多。原创的游戏将成就你的成功。

3.2.9 模仿

创意是可遇不可求的！现在更是如此！但是这并不影响从其它游戏甚至其它任何事物上学习创意。可以是你日常生活中看到的任何事，如电影中的片断，开车时看到的场景等等。灵感总是层出不穷的，关键看你能否抓住她。

3.2.10 设计模式

当使用面向对象的语言开发软件的时候，设计模式的应用时非常普遍的，但是必须要考虑额外的内存占有。这并不是说设计模式不好，前面也提到了，应该先尽最大可能设计好游戏，才能顺利编码，最后才是对特殊部分的优化。

3.2.11 小结

游戏策划是一门大学问，需要理论和游戏开发经验的结合，当要进入比当前游戏设计更为复杂的大型游戏如 MMOG 时，游戏设计就更为重要了。希望读者已经对游戏策划有了基本的了解，当你创建、设计你的第一个游戏的时候，这些理论有助于你游戏的成功。

3.3 开发过程控制

开发游戏时使用开发过程可以提高成功的几率，虽然按部就班地设计被当成浪费时间和对创造力的抹煞，但是它的确有助于开发的正常进行。从设计、执行（开发时）、调度、后期调度到资源控制都需要一个系统的过程。开发过程的优点如下：

- 防止作用域漂移，在游戏世界中阻止用户增加“neat/cool to have features”
- 处理所面临的高风险的情况
- 针对每一阶段（设计、开发、调度、投递产品）的详细计划
- 源代码控制管理
- 迭代开发
- 编译管理

上面列出的只是开发中应用好的过程控制的一部分优点，即使你是一个人开发一个游戏，使用过程控制也不止是一个好的练习，但你超越了简单的开发环境后，它还可以帮你解决很多问题。当你决定开发不止支持一种型号的手机的游戏的时候，过程控制可以很容易地做到，从而，你可以为不同运营商和制造商的三个、四个、甚至一打不同型号的手机开发游戏。很有可能在第一款游戏成功发布给运营商的刺激下，你会很快地推出第二个、第三个…… 无论怎么强调都不过分，控制和遵守开发过程控制可以增加成功的几率。

3.3.1 开发模型

在讨论了过程控制的重要性之后，随后大概介绍一下几种著名的过程方法，在介绍的时候再比较出最适合游戏开发的过程方法。

混沌模型

被称为混乱，这的确不是一种模型；不幸的是，很多项目都在使用这种模型进行开发。这种模型通常出现在小公司里，一个或几个人有了一个好的主意，然后就决定立即编码，他们的想法是尽快地把想法变成实在的产品。最终，也许会有完成的游戏，也发行到了几千部甚至百万部手机上，莫名其妙的 bug 就开始接踵而至了。所以，即使刚开始可以成功的卖出，还是避免不了因为对第一个不完善版本的额外支持和游戏退订造成的利润损失。

混沌模型显然不是游戏开发的首选，也不是任何实际的软件开发需要考虑的模型。但是不要跟概念证明或 RAD（Rapid Application Development 快速应用开发）混为一谈，在这些情况下混沌模型是可以接受的。问题是，在很多情况下概念证明应该被抛开而重新使用更加产品的新增特征的过程模型。

瀑布模型

瀑布模型是古老和传统的开发过程控制之一，开发过程的每一个阶段都有对应的文档，一个阶段结束后才开始下一个。后一个阶段的文档都是基于前一阶段的文档，当一个阶段完成后，它对应的文档将被冻结，除非有什么重大的改变，所以图 4 中返回上一阶段的箭头是虚线。另一个主要的特征是在项目完成的最后，不是所有的测试都会被完成。

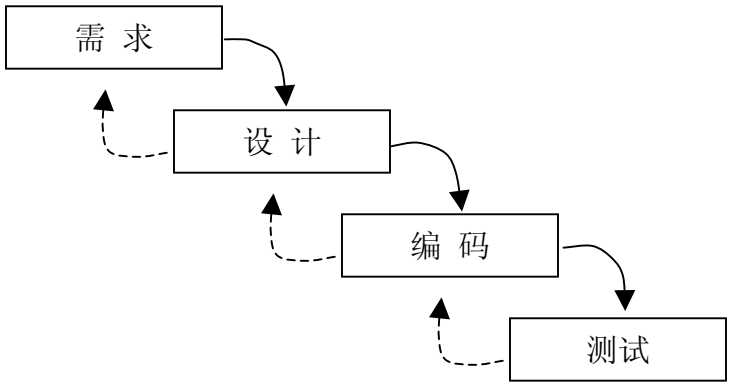

图 4：瀑布模型图示

开发游戏时，瀑布模型也许不太适合，在每个阶段完成后对文档的冻结，妨碍了需要加入的任何更改，在游戏开发者看来就是妨碍了创意的发挥。最终的测试就更加糟糕了，因为在游戏都是基于创意和未知的事物，测试一般在开发过程中进行，

特别是在组件开发还没有完成时。例如，新开发出的遵循新的规则的多人游戏引擎在具体应用前就需要被证明，这个引擎可以高效正常地工作，而且可以做专用的加载工作。

通常，瀑布模型在现代的开发世界中是不被赞成的。当然，如果你有定义得非常好的需求，有一套完整的系统规范，而且知道最后的结果将是如何的，瀑布模型将是比较好的选择。有一个收费系统的例子，涵盖了瀑布模型的所有相关规则和结果。在游戏开发中，总是有很多的改变、创意和促进现有技术的开发，瀑布模型的确不是一个适合的过程控制模型。由于瀑布模型是从七十年代就有的传统的开发模型，并且在游戏开发中应用不成功，使游戏开发者有了过程控制和策划浪费时间的错误认识。

螺旋模型

螺旋模型在瀑布模型的基础上更进一步，这里只有一下大概的比较，如果你对螺旋模型感兴趣的话，你需要自己去做进一步的研究。可以将螺旋模型想象成很多小的瀑布模型链接在一起，螺旋模型的优点是可以容易地加入软件开发中地改变，用户反馈贯穿开发和测试全程。螺旋的特征本身就容易导致松散的项目管理、项目难以跟踪、难以确定项目结束日期等一系列问题。所以它也不适合于对项目结束日期和要求严谨的跟踪策略的游戏开发项目。

RUP（Rational Unified Process－Rational 统一过程）

RUP 是一个以文档为中心的过程模型，这些文档被称为人工因素。用例作为关键的人工因素来指导整个开发过程。RUP 统一过程会涵盖软件开发的从头至尾，从而成为确保项目可跟踪、处理高风险、提供项目各部门间交流的一门学科。软件开发是一个不断变化的反覆的过程。

RUP 的一个缺点是开发过程的每个阶段都需要所有的人工因素。很多项目的失败不是因为 RUP 本身，而是团队的成员不能按照文档规范做事。特别是在游戏开发世界里，文档被认为是无聊的事情，并且会扼杀创造力。如果游戏项目非常大而且有很多人工因素需要面对，RUP 不失为一个好的选择。例如 MMOG 项目，它需要需求分析/特征搜集的好的过程控制，优秀的游戏策划和设计、很多专家（AI 开发者、美工、网络开发专家、网络基础设施专家、QA 团队等等）组成一个团队，这种项目使用 RUP 就比较合适。

有关RUP的更多内容请参考：http://www.rational.com/

XP 极限编程

相对于传统的文档驱动的软件开发模型，专注于小团队，短周期，小版本开发的极限编程近几年变得比较流行。如下是　XP的特征：

- 小团队
- 开发者成对开发（成对编程）
- 编码前测试
- 多次迭代
- 与最终用户高度交互

（译者：这里顺便向大家推荐一个中文技术网址，里面有各种软件工程技术文章：http://www.sawin.com.cn ）

顾名思义，极限编程就是尽可能快速地获得需求，并且软件开发中两个人一起编写一个项目的一种技术。每个伙伴工作在同一台机器上，当一个程序员在写代码时，另一个伙伴在一旁观看，同时认真所写的代码。写代码者从战术上考虑具体实现，其伙伴则从战略上考虑整个程序。他们之间频繁的交换角色，这样将使得可以更快写完代码，并且减少错误，更重要的是：代码将至少有两个人非常清楚的掌握。

只有当项目和项目团队完全符合 XP 极限编程的要求，游戏开发项目才可以使用它，否则 XP 很可能演变成瀑布模型。XP 只适合 2 到 12 人的小团队，而且成对编程的两个人必须习惯于彼此配合并且水平和经验要差不多。

有关XP极限编程的更多内容请参考：http://www.extremeprogramming.org/。

Agile 敏捷方法

上面讲的 XP 极限编程方法是 Agile 敏捷开发中最著名的一个。敏捷开发过程的方法还有：

- SCRUM
- Crystal
- SDM
- 特征驱动软件开发（Feature Driven Development，简称 FDD）
- 自适应软件开发(Adaptive Software Development，简称 ASD)
- 领导开发

2001 年，为了解决许多公司的软件团队陷入不断增长的过程泥潭，一批业界专家一起概括出了一些可以让软件开发团队具有快速工作、响应变化能力的价值观和原则，他们称自己为敏捷联盟。敏捷方法有如下特征：

- 通过可运行的软件衡量开发过程
- 用户的需求被放在第一位
- 小的团队规模
- 不间断的测试
- 文档的编写和提交贯穿项目始终

有关Agile敏捷方法的更多内容请参考：http://www.agilealliance.org/。

小结

我们已经简要的介绍了开发领域中的一些过程控制方法。问题是这些过程控制方法对游戏开发有效吗？当然有效！XP 极限编程和 Agile 敏捷方法对小团队的需要重复开发的项目有很好的实用性；由不同资历的人员组成的打的开发团队负责的项目，RUP 则是个不错的选择。

以防万一，可以讲 RUP 作为主要的过程控制方法，在 RUP 的特定的部分应用 XP 极限编程的特征。

对于独立开发者来说，遵循过程控制方法需要做大量的文档，而且难以满足特定过程控制方法的要求（如因为只有一个人而没有办法进行成对编程），从而因为气馁而回到混沌开发的老路上。作为个体，至少要做到以下几点：有完整的游戏策划和设计文档，编译过程和源代码控制。虽然还不是非常足够，但是由于只有一个人，这样做还是情有可原的。同时也要做好单元测试和迭代测试。

上面提到的过程控制方法都是简短的介绍，其中的每一种都涵盖了大量的信息，这一部分的主要内容是简单介绍一下常用的过程控制方法，并且看它们是否适合用来处理游戏开发项目。最终还是要由你或者项目的雇主来决定哪种过程控制方法最适合于你所在的软件开发环境。

最后，绝对不要忘记开发游戏这个终极目标，如果不能在预算内按时交付一个高质量的游戏，即使做再多的策划和过程控制也没有用。

3.3.2 编译过程

好的编译过程对软件开发有很多好处，编译过程没有什么特殊的地方，只是一些有关如何讲代码编译成确定格式的文件的规则和方针。例如，简单如一个目录的建立都有助于代码和资源的组织。在一个工程中，资源都放在一个目录中，源代码放在一个目录中，jad和manifest放在一个目录中，最终编译的结果又要放在另一个目录中，每个工程都会有类似的划分。这一般可以设定为编译工程文件后的默认输出，而在Sun公司的WTK（Wireless Tool Kit）中这种划分是必须的。JUNIT工具（http://www.junit.org/）可以自动执行编译，而且包含了自动的单元测试，它推荐使用像ANT（http://ant.apache.org/）这类Java编译工具。切记你可能不是开发同一个游戏的唯一一个，通过定义统一的编译过程有助于开发团队中每个人之间的同步。

3.3.3 源代码控制

源代码的有效管理对项目的成功意义重大，如果不是一个开发者开发一个项目的话，源代码管理是必须的。源代码控制确保了代码被代码在读入读出的过程中不会丢失或者被覆盖，在开发、测试、质量评估和最终生产过程中，源代码控制提供了很多常规管理。

作为独立开发者，源代码控制有助于同时开始多个项目或者同时开发针对不同运营商或不同机型的同一个项目的不同版本。

通过对每个项目相关事物的保存，源代码控制可以减少源代码冲突、生产力的降低及代码损失的可能。

一些常用的源代码控制软件有 CVS、Visual Source Safe，BitKeeper 和 eChangeMan。

第四章 MIDP 2.0 的 Game 类

J2ME 的流行促使几个运营商和制造商开发了一些支持游戏的类，却造成了游戏缺乏可移植性的问题。例如，很难将使用 Siemens 的 Sprite（精灵）类的游戏移植到 Nokia 手机上，因为你需要重新实现 Sprite 类。

在 MIDP 2.0 版本发布后，这些游戏移植性的问题得到了初步的解决，因为 MIDP 2.0 新加入了五个与游戏开发相关的类：

- GameCanvas
- Sprite
- Layer
- LayerManager
- TiledLayer

Layer 类一般不会直接用到，它是一个抽象类，被 Sprite 类、LayerManager 类和 TiledLayer 类使用，这些类都在包 java.microedition.lcdui.game 中。

Game 类的出现不仅降低了错误出现的几率，也使游戏代码变得更小，因为开发者不需要自己编写像 Sprite 这种类了。Game 类现在是每个支持 MIDP 2.0 的手机上的基础 Java 环境。

4.1 GameCanvas 类

与 MIDP 1.0 中相比较，MIDP 2.0 中的 GameCanvas 带来了两点好处。一个是创建的每一个 GameCanvas 类都有一个独立的 buffer，从而不但减少了堆的使用率，也可以使用一个单独的堆控制游戏。以下是 MIDP 1.0 实现的循环：

```
public void theGameCanvas extends Canvas implements Runnable {
   public void run() {
      while (true) {
         repaint(); // update game display
      }
   }
   public void paint(Graphics g) {
   // painting
   // redraw occurs here
   }
   protected void keyPressed(int keyCode) {
   // obtain user inputs
   }
}
```

上述代码有三个显而易见的功能块：屏幕绘制、run()和消息处理；所有这些都是在不同的线程中运行的。正是由于多个线程的缘故，最终的显示有可能偶尔会出现不平稳的情况，特别是需要很多图形图像和交互内容的动作游戏。在三个不同的部分跟踪三个不同的功能也是很麻烦的。

在 MIDP 2.0 中，对 GameCanvas 的扩展，使一切都变得更清洁、更易使用也更有效率。GameCanvas 运行在一个线程中，同时运用 polling 的技术避免了对消息处理（keyPressed）的等待，这意味着可以随时通过 GameCanvas 类提供的 getKeysState()方法来获取按键的情况。通过 buffer 的使用，一种称为双缓冲（double buffering）的技术被自动应用，开发者需要做到只是调用 flushGraphics()来显示图像。双缓冲技术是用来避免屏幕显示的抖动问题，一般先在后台绘制一个临时的图像，绘制完成后直接切换到显示区域，从而避免了直接在显示区域绘制时因延时发生的闪烁。

4.2 GameCanvas 基础实例

下面是 GameCanvas 使用的实例，在这个例子中用户可以通过按键控制“x”的移动：

GameCanvas

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true

    private long delay; // To give thread consistency

    private int currentX, currentY; // To hold current position of the 'X'

    private int width; // To hold screen width

    private int height; // To hold screen height
    // Constructor and initialization

    public ExampleGameCanvas() {
        super(true);
        width = getWidth();
        height = getHeight();
        currentX = width / 2;
        currentY = height / 2;
        delay = 20;
    }
```

```
// Automatically start thread for game loop
public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
}

public void stop() {
    isPlay = false;
}

// Main Game Loop
public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
        input();
        drawScreen(g);
        try {
            Thread.sleep(delay);
        } catch (InterruptedException ie) {
        }
    }
}

// Method to Handle User Inputs
private void input() {
    int keyStates = getKeyStates();
    // Left
    if ((keyStates & LEFT_PRESSED) != 0)
        currentX = Math.max(0, currentX - 1);
    // Right
    if ((keyStates & RIGHT_PRESSED) != 0)
        if (currentX + 5 < width)
            currentX = Math.min(width, currentX + 1);
    // Up
    if ((keyStates & UP_PRESSED) != 0)
        currentY = Math.max(0, currentY - 1);
    // Down
    if ((keyStates & DOWN_PRESSED) != 0)
        if (currentY + 10 < height)
            currentY = Math.min(height, currentY + 1);
}

// Method to Display Graphics
private void drawScreen(Graphics g) {
```

```
            g.setColor(0xffffff);
            g.fillRect(0, 0, getWidth(), getHeight());
            g.setColor(0x0000ff);
            g.drawString("X",currentX,currentY,Graphics.TOP| raphics.LEFT);
            flushGraphics();
        }
}
```

Main Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleGameCanvasMidlet extends MIDlet {
    private Display display;

    public void startApp() {
        display = Display.getDisplay(this);
        ExampleGameCanvas gameCanvas = new ExampleGameCanvas();
        gameCanvas.start();
        display.setCurrent(gameCanvas);
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

图 5：GameCanvas 实例源代码

图 6：GameCanvas 实例的模拟器截图

4.3 Sprite 类

4.3.1 Sprite 的定义

成功的游戏一般都会有优秀的图像。游戏中的大多数对象都被组织成了一种特殊的图像，称为精灵。精灵可以是任何图像，如子弹、怪物、游戏主角、敌人、特殊物品、钥匙和门等等。

图 7：精灵实例

大多数情况下，精灵是运动的图像。这些运动的图像是通过很多平滑的差别的相同精灵组合而成的，这种精灵的集合通常被称为帧集合，这些帧会顺序地或者非顺序地渲染到屏幕上，通过简易的代码可以顺序的现实特定的精灵。

图 8：精灵集合

图八所示内容表示一个等待帧和另外四个基础帧的精灵集合。

4.3.2 Sprite 类构造函数

Sprite 类有三个内置的构造函数：

- Sprite(Image image)－创建不动的单帧精灵
- Sprite(Sprite sprite)－用已有精灵创建一个新的精灵
- Sprite(Image image,int frameWidth,int frameHeight)－创建一个超过两帧的动态的精灵，frameWidth 是精灵的宽度，frameHeight 是精灵的高度。

图 9：精灵集合

在图 8 中，将整个精灵集合分成五个独立的帧，它的总宽度是 160 象素，则每一帧的宽度是 32 象素。每个帧的高度同样是 32 象素。每一帧的长和宽不一定要相同，但是在同一个精灵集合中所有帧的长都要相同，宽也都要相同。在 Sprite(Image image,int frameWidth,int frameHeight)构造函数中，不需要确定帧的个数，sprite 类会自动计算。

4.3.3 8Bit,16Bit 还是 32Bit?

包含精灵的图像一般都有相同的长度和高度的约束，这是由于象素的数量与色素的数量相关。这被称为位深和色深，每一个象素的位数越大，它能显示的颜色就越多。关系式是：

$$2^{\text{bit 位数}} = \text{颜色总数}$$

当 bit 位数是 8、16、24 和 32 时，可以显示的颜色如下：

- 2^8=256 色
- 2^{16}=65536 色
- 2^{24}=16777216 色
- 2^{32}=16777216 色和一个 8 位深的 α 通道。α 通道实际上是一个掩码，它定义了合并两个象素的方式。象素的合并发生在一个象素在另一个象素上时。

当然，可以仅仅使用 8 位象素（253 色）显示超过 8 位的精灵，这可以有效地节省移动设备资源。颜色越多，渲染图像地过程就会越长。

4.3.4 精灵碰撞

精灵碰撞是所有交互式的动作游戏必须考虑地功能，比如，精灵碰到墙壁、精灵的射击或者两个精灵相互碰撞。碰撞探测并不意味着刮掉或者游戏结束，它可以代表升级、力量增加、赛车游戏中的赛车失控、开门、指示玩家爬梯子等等一切可能的事情。

那么，如何探测精灵碰撞呢？一般我们可以之间判断一个精灵的象素是否跟另一个的有所交叠或接触。

图 10：精灵碰撞

幸运的是，精灵类包含了如下方法：

- collidesWith(Image image,int x,int y,Boolean pixelLevel)
- collidesWith(Sprite sprite,Boolean pixedLevel)

通过上面函数，你可以检测精灵与其它精灵或者图像碰撞的情况。第一个使用 image 作为输入变量的函数需要指定 image 的位置，通过 x、y 表示图像的到左上角的距离。PixelLevel 是一个 boolean 变量，true 表示象素级别的检测，false 指矩形交

叉（rectangle intersect）检测。可以通过定义矩形交叉的值来决定检测的精细程度，一般将矩形交叉的值设为比图像大小略微小一些。不可视的矩形碰撞可以消除不成对的精灵碰撞，但是不包括不透明的象素的交叠。

象素级别的检测是指一个精灵的不透明象素与另一个精灵的不透明象素相交叠的时候，最常用的方法是两个精灵的矩形交叉的冲突，而且矩形的大小通常就是图像边框的范围。

这是第三个方法：

- collidedWidth(TiledLayer tiledLayer,Boolean pixelLevel)

这个方法与上两个类似，唯一的不同是碰撞通过图像平铺层检测。TiledLayers 将在下一部分详细讲述。

4.3.5 精灵显示

一般调用绘图方法（paint）来显示或渲染精灵，这需要通过 Graphics 对象来实现，因为 Graphics 是其中一个必须的输入参数。而且需要先使用 setVisible(boolean) 方法调用绘图方法，boolean 要为 true。

4.3.6 显示精灵序列

可以处理精灵帧序列的一些方法：

- getFrameSequenceLength()－返回帧队列的元素个数。
- getFrame()－取回帧队列中当前登录的索引号，注意不是当前显示的帧。
- nextFrame()－移动到帧队列的下个帧，如果当前是最后一个帧，则移动到第一个帧。
- prevFrame()－移动到帧队列的下个帧，如果当前是第一个帧，则移动到最后一个帧。
- setFrame(int sequenceIndex)－在帧队列中手动设置队列索引号。
- setFrameSequence(int[] sequence)－预设已经定义好的帧队列。

更多细节请参考 J2ME API。

4.3.7 精灵透视

要根据具体情况来决定是使用透明的还是不透明的图像，如果是类似 TicTacToe 这种包含很少的动画的简单游戏，透明图像并不是必须的。在包含大量的精灵碰撞和不断改变的背景图像的高交互性的游戏中，则需要考虑透明图像。当然，在你决定使用透明图像的时候，你需要联系移动设备制造商，以确定你要在其上开发的游

戏的移动设备是否支持透明图像。

图 11：透明背景的精灵 VS. 非透明背景的精灵

4.3.8 精灵变换

有一些操作精灵的方法可以将精灵做 90 度旋转、镜像变换。变换通过调用 setTransform(transform)方法实现。参数实际上是整型变量，有如下默认静态常量供用户使用：

静态常量	整数值
TRANS_NONE	0
TRANS_MIRROR_ROT180	1
TRANS_MIRROR	2
TRANS_ROT180	3
TRANS_MIRROR_ROT270	4
TRANS_ROT90	5
TRANS_ROT270	6
TRANS_MIRROR_ROT90	7

变换被应用后，精灵被自动定位，仍然处在中间。因此，对精灵的引用指针没有改变，但是，getX()和 getY()的返回值会变成当前精灵相对屏幕左上角的位置。

4.3.9 精灵优化

为了使针对的计算机的色彩和分辨率支持你所开发的游戏，需要最优化游戏中所有的图形图像。图像的优化通常是移除人的眼睛不能分辨的颜色，优化的好处是降低图像的尺寸大小，即减少 jar 文件的大小。

4.3.10 Sprite 类实例

下面是在 GameCanvas 部分的实例基础上建立的 sprite 类实例；你可以上下左右移动中间的精灵。左上角的精灵展示没有应用透明时精灵的显示情况。

Main Game Canvas with Sprites:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true

    private long delay; // To give thread consistency

    private int currentX, currentY; // To hold current position of the 'X'
```

```
    private int width; // To hold screen width

    private int height; // To hold screen height
    // Sprites to be used

    private Sprite sprite;

    private Sprite nonTransparentSprite;

    // Constructor and initialization
    public ExampleGameCanvas() throws Exception {
        super(true);
        width = getWidth();
        height = getHeight();
        currentX = width / 2;
        currentY = height / 2;
        delay = 20;
        // Load Images to Sprites
        Image image = Image.createImage("/transparent.png");
        sprite = new Sprite(image, 32, 32);
        Image imageTemp = Image.createImage("/nontransparent.png");
        nonTransparentSprite = new Sprite(imageTemp, 32, 32);
    }

    // Automatically start thread for game loop
    public void start() {
        isPlay = true;
        Thread t = new Thread(this);
        t.start();
    }

    public void stop() {
        isPlay = false;
    }

    // Main Game Loop
    public void run() {
        Graphics g = getGraphics();
        while (isPlay == true) {
            input();
            drawScreen(g);
            try {
                Thread.sleep(delay);
            } catch (InterruptedException ie) {
            }
        }
```

```
    }

    // Method to Handle User Inputs
    private void input() {
        int keyStates = getKeyStates();
        sprite.setFrame(0);
        // Left
        if ((keyStates & LEFT_PRESSED) != 0) {
            currentX = Math.max(0, currentX - 1);
            sprite.setFrame(1);
        }
        // Right
        if ((keyStates & RIGHT_PRESSED) != 0)
            if (currentX + 5 < width) {
                currentX = Math.min(width, currentX + 1);
                sprite.setFrame(3);
            }
        // Up
        if ((keyStates & UP_PRESSED) != 0) {
            currentY = Math.max(0, currentY - 1);
            sprite.setFrame(2);
        }
        // Down
        if ((keyStates & DOWN_PRESSED) != 0)
            if (currentY + 10 < height) {
                currentY = Math.min(height, currentY + 1);
                sprite.setFrame(4);
            }
    }

    // Method to Display Graphics
    private void drawScreen(Graphics g) {
        g.setColor(0xFF0000);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x0000ff);
        // display sprites
        sprite.setPosition(currentX, currentY);
        sprite.paint(g);
        nonTransparentSprite.paint(g);
        flushGraphics();
    }
}
```

Main Midlet

```
import javax.microedition.midlet.*;
```

```
import javax.microedition.lcdui.*;

public class ExampleGameSpriteMidlet extends MIDlet {
    private Display display;

    public void startApp() {
        try {
            display = Display.getDisplay(this);
            ExampleGameCanvas gameCanvas = new
ExampleGameCanvas();
            gameCanvas.start();
            display.setCurrent(gameCanvas);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

图 12：Sprite 实例源代码

图 13：Sprite 类实例的模拟器截图

4.3.11 扩充 Sprite 类

需要考虑继承 Sprite 类的几个原因：

1. 高效添加附加的 Sprite
2. 对所有游戏中的精灵使用 Sprite 类作为基类
3. 高效封装

就第一点来说，你会发现 sun 公司提供的当前 Sprite 类没有提供所有开发者可能会用到的特征。如在赛车游戏中，速度是一个很关键的因素，故需要继承 Sprite 类并扩展要用到的变量和方法。

另一种情况是，有几个不同类型的精灵类，但是每一个类型都有它们自己的特征集。例如，玩家控制的精灵（player sprite）有一些特征如生命、体力、速度、智力，而敌人精灵（enemy sprite）则有另一些特征如区分它是哪种敌人的值、生命条。但是两种精灵仍共享相同类型的特征如位置坐标和名称。通常可以创建包含对所有精灵都通用的特征的 Sprite 基类，然后，通过继承基类来建立同时包含自己独特的特征和公共特征的子类。

注意到在 Sprite 类实例中，精灵只能上、下、左、右的移动。如果你希望通过按 3 键可以使精灵同时向右上角移动，就需要定义其它的基础方向。要实现 SW、SE、NW、NE 这四个方向，需要正确地定义精灵和移动数值。虽然这仅仅涉及到很基础的数学处理，但是为了保持代码的整洁，建立一个封装好的自己的 Sprite 类是一个比较好的选择。

4.3.12 建立自己的 Sprite 类

如果 Sprite 类由于无论何种原因不能满足你的需求，你可以选择继承 Layer 类，或者重新扩展自己的 Sprite 类，当然需要以其它名称命名。

下面是一个重新扩展 Sprite 类的不完整的示例，你可以根据需要将它完成。

```
import java.microedition.lcdui.*;

public class MySprite {
    private Image image;
    private int positionX;
    private int positionY;
    private int frameWidth;
    private int frameHeight;
    private int numFrames;
    private int currentFrame;
    private boolean visible;

    public MySprite(Image image, int frameWidth, int frameHeight, int numFrames) throws Exception {
        this.image = image;
        this.frameWidth = frameWidth;
        this.frameHeight = frameHeight;
        this.numFrames = numFrames;
    }
```

```
    public int getX() {
        return this.positionX;
    }

    public void setX(int positionX) {
        this.positionX = positionX;
    }

    public int getY() {
        return this.positionY;
    }

    public void setY(int positionY) {
        this.positionY = positionY;
    }   // Continue Your Code here
}
```

4.3.13 Sprite 百草园

自己编码来处理精灵是一回事，制作和寻找精灵则是另一回事了。很多开发者一般都没有制作精美的精灵的能力。

有关精灵的著名站点：

- http://www.arifeldman.com
- http://www.idevgames.com
- http://www.spriteworks.com

要注意，在免费使用精灵之前，需要先阅读并遵守使用条款。

另一个选择是雇用专业的精灵设计（sprite artist），或称为象素作家（pixel pusher）。你可能会想要联系自由图像设计者并确定他们的收费标准，有时候他们会根据项目的不同而免费制作精灵图像，特别当项目是开源的时候。在寻找自由图像设计者之前需要做一些调查，就好像开发者一样，艺术家也有他们自己的社群。

你也可以学习自己制作精灵。如果你计划做一个 2D 的游戏而且你也不介意花费很多时间在学习精灵绘制的更多相关知识，这是一个不错的主意。但是，如果你自认为没有什么艺术天分，最好还是接受上面两条建议比较好。注意：更好的图像是重要的，更重要的是游戏可以正常运行；所以，建议刚开始先使用一般的不是那么好看的图像，直到开发出完全功能的游戏后，再考虑提高游戏中图像的质量。

4.4 LayerManager 类

4.4.1 什么是 LayerManager

在本章的上一部分我们讨论了 sprite 的处理，当然，我们可以在同一屏中加入不止一个 sprite，但是，场景呢？事实上，所有的游戏，无论是动画的、静态的还是卷轴的，都需要背景来使游戏更加栩栩如生。这就是 LayerManager 存在的原因。就像 LayerManager 类的名称，这个类用来管理图像的各个层，它将按正确的位置和次序渲染所有在它控制下的图层。换句话来说，它会使用一种易于管理的方式来处理图像之间的重叠。

可以将图层的次序理解为第三维称为 z 轴：

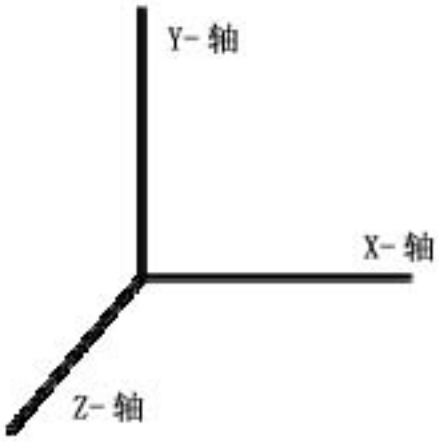

图 14：Z 轴图示

z 轴上的图层用数字从 0 开始编号，0 号图层最靠近用户，编号最大的图层最远离用户。如果图层的个数有变化，LayerManager 会重新对所有图层编码，从而保持图层次序的连续性。这通常发生在游戏中动态的添加和删除图层的时候。

4.4.2 如何使用 LayerManager

append(Layer layer)用来整加一个图层：

```
LayerManager = new LayerManager();
LayerManager.append(layer);
```

通过索引号重新获得图层可以使用方法 getLayerAt(int index)；获得当前 LayerManager 中的图层总数使用 getSize()；移除一个图层使用 remove(Layer layer)；在确定的索引号处加入特定的图层可以使用 insert(Layer layer,int index)方法；最后，可以通过调用 paint(Graphics g, int x, int y)方法来显示图层。

4.4.3 更进一步

上节提到的方法都是比较直接的，而且就算不使用 LayerManager，也可以容易地自己扩展。然而，这里有一个提供更为高级功能的方法：setViewWindow(int x,int y,int width,int height)。它可以定义屏幕的可视区域大小，也可以定义显示哪一个图层。初听起来的确让人困惑，下图具体描述了可视区域的概念：

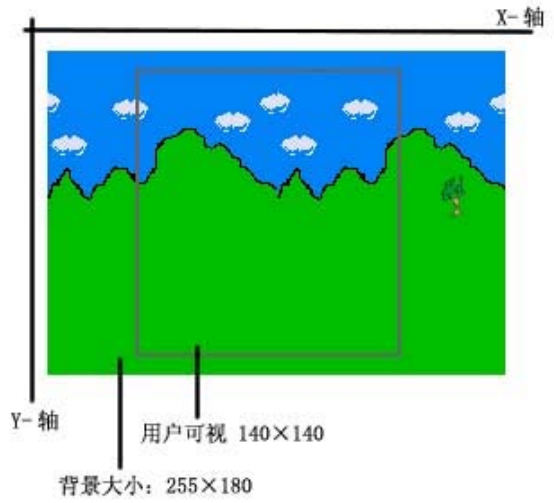

图 15：setViewWindow()方法实现的功能

实现上图情况的方法应该是——setViewWindow(55,20,140,140)，前两个数字(55,20)代表用户可视范围的左上角相对整个背景左上角的坐标。后面两个数字(140,140)则表示用户视图的实际宽度和高度。具体的代码和模拟器截图请见下一节。

4.4.4 LayerManager 类实例

Game Canvas using the LayerManager:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true
    private long delay; // To give thread consistency
    private int currentX, currentY; // To hold current position of the 'X'
    private int width; // To hold screen width
    private int height; // To hold screen height
    // Sprites to be used
    private Sprite playerSprite;
    private Sprite backgroundSprite;
    // Layer Manager
```

```
    private LayerManager layerManager;

    // Constructor and initialization
    public ExampleGameCanvas() throws Exception {
        super(true);
        width = getWidth();
        height = getHeight();
        currentX = width / 2;
        currentY = height / 2;
        delay = 20;
        // Load Images to Sprites
        Image playerImage = Image.createImage("/transparent.png");
        playerSprite = new Sprite(playerImage, 32, 32);
        Image backgroundImage = Image.createImage("/background.png");
        backgroundSprite = new Sprite(backgroundImage);
        layerManager = new LayerManager();
        layerManager.append(playerSprite);
        layerManager.append(backgroundSprite);
    }

    // Automatically start thread for game loop
    public void start() {
        isPlay = true;
        Thread t = new Thread(this);
        t.start();
    }

    public void stop() {
        isPlay = false;
    }

    // Main Game Loop
    public void run() {
        Graphics g = getGraphics();
        while (isPlay == true) {
            input();
            drawScreen(g);
            try {
                Thread.sleep(delay);
            } catch (InterruptedException ie) {
            }
        }
    }

    // Method to Handle User Inputs
    private void input() {
```

```
            int keyStates = getKeyStates();
            playerSprite.setFrame(0);
            // Left
            if ((keyStates & LEFT_PRESSED) != 0) {
                currentX = Math.max(0, currentX - 1);
                playerSprite.setFrame(1);
            }
            // Right
            if ((keyStates & RIGHT_PRESSED) != 0)
                if (currentX + 5 < width) {
                    currentX = Math.min(width, currentX + 1);
                    playerSprite.setFrame(3);
                }
            // Up
            if ((keyStates & UP_PRESSED) != 0) {
                currentY = Math.max(0, currentY - 1);
                playerSprite.setFrame(2);
            }
            // Down
            if ((keyStates & DOWN_PRESSED) != 0)
                if (currentY + 10 < height) {
                    currentY = Math.min(height, currentY + 1);
                    playerSprite.setFrame(4);
                }
        }

        // Method to Display Graphics
        private void drawScreen(Graphics g) {
            //g.setColor(0x00C000);
            g.fillRect(0, 0, getWidth(), getHeight());
            g.setColor(0x0000ff);
            // updating player sprite position
            playerSprite.setPosition(currentX, currentY);
            // display all layers
            layerManager.paint(g, 0, 0);
            flushGraphics();
        }
}
```

Main Midlet:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleLayerManagerMidlet extends MIDlet {
    private Display display;
```

```
    public void startApp() {
        try {
            display = Display.getDisplay(this);
            ExampleGameCanvas gameCanvas = new
ExampleGameCanvas();
            gameCanvas.start();
            display.setCurrent(gameCanvas);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

图 16：LayerManager 类实例的模拟器截屏

图 17：使用 setViewWindow 函数的 LayerManager 类实例的模拟器截屏

上图显示的是使用 setViewWindow 函数实现的效果，下面是对应的代码（替换掉原代码的 drawScreen 方法）：

```
    //Method to Display Graphics
    private void drawScreen(Graphics g) {
        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x0000ff);
        // updating player sprite position
        playerSprite.setPosition(currentX,currentY);
        // display all layers
        layerManager.setViewWindow(55,20,140,140);
        layerManager.paint(g,20,20);
        flushGraphics();
    }
```

仅使用 setViewWindow 方法，用户可视区域不会出现在屏幕中间，如果想调节可视区域在屏幕的位置，需要在 paint 方法中设定适当的值。在本例中，如果想使可视区域在屏幕中间，传入的值为 X-轴：20 象素、Y-轴：20 象素，这一点是可视区域对实际屏幕的偏移量。请不要与 setViewWindow 的前两个参数混淆了，setViewWindow 的参数是相对整个背景图左上角的偏移量。

现在你在屏幕的顶端和底端有空间加入更多对用户的反馈，例如，你可以显示最高分，当前用户得分、级别和剩余生命数。

图 18：有更多用户反馈的游戏模拟器截图

4.4.5 LayerManager 类和滚动背景

背景图像的使用要比屏幕本身要好的多。

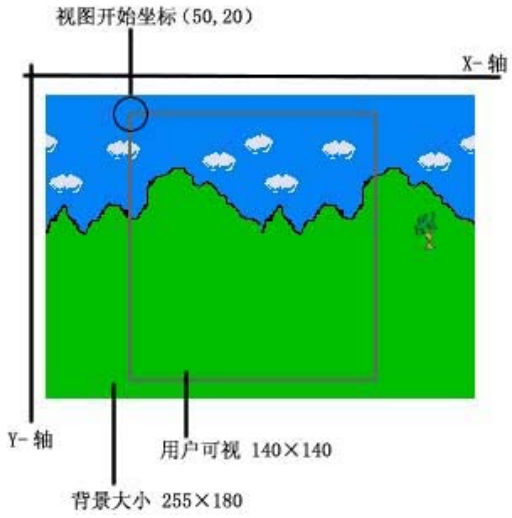

图 19：滚动视图

在这个例子中，开始坐标是（50,20），为了使背景卷动需要使这里的“50”可变。当需要背景向左卷动时，第一个参数需要减小；反之，当需要背景向右卷动时，第一个参数需要增大。

下面的代码显示了实现此功能的方法，简单起见精灵已经被移除了，现在按游戏对应的左键则背景向左卷动，按游戏对应的右键则背景向右卷动。而且，背景只有水平方向上可以卷动，在竖直方向不能卷动。当然，你可以很容易地将其改为四个方向都能卷动的背景。

Scrolling Game Canvas

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true
    private long delay; // To give thread consistency
    private int width; // To hold screen width
    private int height; // To hold screen height
    private int scnX, scnY; // To hold screen starting viewpoint
    // Sprites to be used
    Image backgroundImage;
    private Sprite backgroundSprite;
```

```
// Layer Manager
private LayerManager layerManager;

// Constructor and initialization
public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    scnX = 55;
    scnY = 20;
    delay = 20;
    // Load Images to Sprites
    backgroundImage = Image.createImage("/background.png");
    backgroundSprite = new Sprite(backgroundImage);
    layerManager = new LayerManager();
    layerManager.append(backgroundSprite);
}

// Automatically start thread for game loop
public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
}

public void stop() {
    isPlay = false;
}

// Main Game Loop
public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
        input();
        drawScreen(g);
        try {
            Thread.sleep(delay);
        } catch (InterruptedException ie) {
        }
    }
}

// Method to Handle User Inputs
private void input() {
    int keyStates = getKeyStates();
    if ((keyStates & LEFT_PRESSED) != 0) {
```

```
                if (scnX - 1 > 0)
                    scnX--;
            }
            if ((keyStates & RIGHT_PRESSED) != 0) {
                if (scnX + 1 + 140 < backgroundImage.getWidth())
                    scnX++;
            }
        }
        // Method to Display Graphics
        private void drawScreen(Graphics g) {
            //g.setColor(0x00C000);
            g.setColor(0xffffff);
            g.fillRect(0, 0, getWidth(), getHeight());
            g.setColor(0x0000ff);
            // display all layers
            layerManager.setViewWindow(scnX, scnY, 140, 140);
            layerManager.paint(g, 20, 20);
            flushGraphics();
        }
}
```

Main Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class SimpleScrollingLayerManger extends MIDlet {
    private Display display;

    public void startApp() {
        try {
            display = Display.getDisplay(this);
            ExampleGameCanvas gameCanvas = new
ExampleGameCanvas();
            gameCanvas.start();
            display.setCurrent(gameCanvas);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
```

```
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

图 20：使用 LayerManager 类实现的卷动背景模拟器截图

4.5 TiledLayer 类

4.5.1 什么是 TiledLayer 类？

TiledLayer 是 game 包中的最后一个类，当然不是最没有意义的类，TiledLayer 是用来处理拥有巨大的背景或虚拟区域的完美选择。通过使用这个函数可以定义游戏背景的特定区域，并且可以重复利用它们生成一个完整的图像。例如下图：

图 21：TiledLayer 类实例

在图 21 中，仔细观察图像你会发现很多区域比较类似，事实上，你可以将整个图像分解成 6 种截然不同的区域，一般称为贴砖（Tile）。

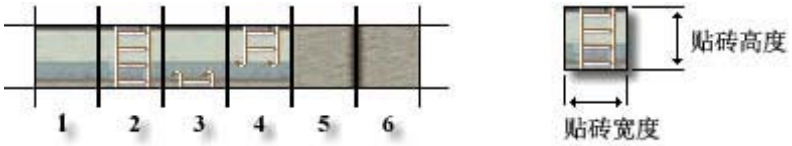

图 22：贴砖实例

所有的贴砖都是 32×32 象素的，这是 TiledLayer 类要求的。每个贴砖都有一个编号，从 1 开始，自左向右、自上而下顺次变大，即贴砖被一行一行地赋值的。所以，只要保证贴砖的索引值不变，就可以按照想要的形状用几块贴砖铺设出巨大的图像。

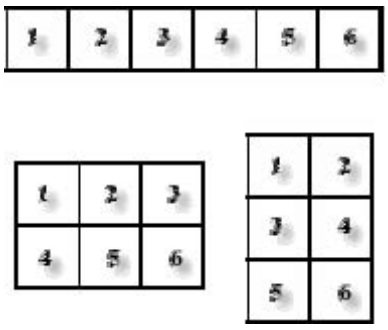

图 23：指定索引号的 TiledLayer

上例中的贴砖和索引号是一一对应的，从而通过索引号确定图像中使用哪个贴砖。如果索引值为 0 则表示图像中的这个单元格是空的，就是说如果图像中的某个单元格对应的值为 0，在游戏运行时将不会在这个单元中贴入任何贴砖。

4.5.2 TiledLayer 类的构造函数

建立一个 TiledLayer 实例的构造函数：

TiledLayer(int columns, int rows, Image image, int tileWidth, int tileHeight)

Columns 和 rows 分别是生成最终图像所包含贴砖的列数和行数；image 是用来映射对应索引号的所有贴砖所在的图像；tileWidth 和 tileHeight 是每个确定贴砖的宽度和长度。

4.5.3 TiledLayer 处理

映射一个特定贴砖到图像中确定的单元格使用 *setCell(int col, int row, int tileIndex)*方法，其中 col 和 row 定位单元格的坐标，tileIndex 映射使用哪个贴砖。

在调用构造函数后想替换整个贴砖集或贴砖映射表，一般使用 *setStaticTileSet(Image image, int tileWidth,tileHeight)*函数。如果新的贴砖集与原砖图集相同或者包含了更多的贴砖，所生成图像的单元格内容不会改变。然而如果新的贴砖集比原来的集合包含的贴砖少，则映射将被重置；原来生成的图像中的所有的索引都被置为 0，而且动态贴砖将被删除。

想使用特定的贴砖填充特定区域中的单元格，可以使用 *fileCells(int col, int row, int numCols, int numRows, int tileIndex)* 函数。前两个参数指定了特定区域中左上角的单元格所在的行和列数；接着的两个参数定义此区域包含的行数和列数；最后一个参数是想放置到设定区域中的贴砖的索引。

4.5.4 TiledLayer 显示

同其它游戏对象一样，可以通过调用 paint 方法，或者使用 LayerManager 类来显示 TiledLayer 类的内容。

4.5.5 检索当前 TiledLayer 类的设置

下面一些知名达意的获得当前使用的 TiledLayer 类的信息的函数：

- getCell(int col,int row)
- getCellHeight()
- getCellWidth()
- getColumns()
- getRows()

4.5.6 动态的单元格

TiledLayer 类可以提供生成动态的贴砖的功能，动态贴砖用负索引值表示。每个动态单元格都是通过动态地关联静态地贴砖得到的，这就使我们可以很容易地通过变换关联的静态贴砖来得到一组动态单元格。这种技术可以用来生成背景动画，入欢呼的人群、飘动的云、还要碧波荡漾的水……除去上面提到的方法，下面还有三种方法来生成动态单元格：

- *createAnimatedTile(int staticTileIndex)* – 通过静态贴砖索引创建一个新的动态贴砖并返回动态贴砖的连续的负索引。默认包含一个静态贴砖（位置码）或者 0。
- *getAnimatedTile(int animatedTileIndex)* – 通过索引检索动态贴砖
- *setAnimatedtile(int animatedTileIndex, int staticTileIndex)* – 将一个动态贴砖链接到静态贴砖上

4.5.7 没有动态贴砖的 TiledLayer 类实例

GameCanvas Source Code:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true
    private long delay; // To give thread consistency
    private int width; // To hold screen width
    private int height; // To hold screen height
    // Layer Manager
```

```
private LayerManager layerManager;

// TiledLayer
private TiledLayer tiledBackground;

// Constructor and initialization
public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    delay = 20;
    tiledBackground = initBackground();
    layerManager = new LayerManager();
    layerManager.append(tiledBackground);
}

// Automatically start thread for game loop
public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
}

public void stop() {
    isPlay = false;
}

// Main Game Loop
public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
        input();
        drawScreen(g);
        try {
            Thread.sleep(delay);
        } catch (InterruptedException ie) {
        }
    }
}

// Method to Handle User Inputs
private void input() {
    // no inputs
}
```

```
    // Method to Display Graphics
    private void drawScreen(Graphics g) {
        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x0000ff);
        layerManager.paint(g, 0, 0);
        flushGraphics();
    }

    private TiledLayer initBackground() throws Exception {
        Image tileImages = Image.createImage("/tiles.png");
        TiledLayer tiledLayer = new TiledLayer(10, 10, tileImages, 32, 32);
        int[] map = { 5, 1, 1, 4, 1, 1, 1, 1, 1, 6, 5, 1, 3, 1, 1, 3, 1, 1, 1,
                6, 5, 1, 2, 1, 1, 2, 1, 1, 1, 6, 5, 1, 2, 3, 1, 2, 1, 1, 1, 6,
                5, 1, 4, 2, 1, 2, 1, 1, 1, 6, 5, 1, 1, 4, 1, 2, 1, 1, 1, 6, 5,
                1, 1, 1, 1, 4, 1, 1, 1, 6, 5, 1, 1, 1, 1, 1, 1, 1, 1, 6, 5, 1,
                1, 1, 1, 1, 1, 1, 1, 6, 5, 1, 1, 1, 1, 1, 1, 1, 1, 6 };
        for (int i = 0; i < map.length; i++) {
            int column = i % 10;
            int row = (i - column) / 10;
            tiledLayer.setCell(column, row, map[i]);
        }
        return tiledLayer;
    }
}
```

Main Midlet Source Code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleTiledLayer extends MIDlet {
    private Display display;

    public void startApp() {
        try {
            display = Display.getDisplay(this);
            ExampleGameCanvas gameCanvas = new
ExampleGameCanvas();
            gameCanvas.start();
            display.setCurrent(gameCanvas);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
```

```
    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

模拟器输出：

见图 21。

4.5.8 包含动态贴砖的 TiledLayer 类实例

这个实例演示了动态贴砖的建立，虽然生成的动态贴砖并没有什么实际意思，但这并不是重点，重点是搞清楚动态贴砖的生成过程。

GameCanvas Source Code:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ExampleGameCanvas extends GameCanvas implements Runnable {
    private boolean isPlay; // Game Loop runs when isPlay is true
    private long delay; // To give thread consistency
    private int currentX, currentY; // To hold current position of the 'X'
    private int width; // To hold screen width
    private int height; // To hold screen height
    // Layer Manager
    private LayerManager layerManager;
    // TiledLayer
    private TiledLayer tiledBackground;
    // Flag to indicate tile switch
    private boolean switchTile;
```

```
// To hold the animated tile index
private int animatedIdx;

// Constructor and initialization
public ExampleGameCanvas() throws Exception {
    super(true);
    width = getWidth();
    height = getHeight();
    currentX = width / 2;
    currentY = height / 2;
    delay = 20;
    tiledBackground = initBackground();
    layerManager = new LayerManager();
    layerManager.append(tiledBackground);
}

// Automatically start thread for game loop
public void start() {
    isPlay = true;
    Thread t = new Thread(this);
    t.start();
}

public void stop() {
    isPlay = false;
}

// Main Game Loop
public void run() {
    Graphics g = getGraphics();
    while (isPlay == true) {
        input();
        drawScreen(g);
        try {
            Thread.sleep(delay);
        } catch (InterruptedException ie) {
        }
    }
}

// Method to Handle User Inputs
private void input() {
    // no inputs
}

// Method to Display Graphics
```

```
    private void drawScreen(Graphics g) {
        g.setColor(0xffffff);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x0000ff);
        // Determine which tile to show
        if (switchTile) {
            tiledBackground.setAnimatedTile(animatedIdx, 3);
        } else {
            tiledBackground.setAnimatedTile(animatedIdx, 4);
        }
        // Set tile file to opposite boolean value
        switchTile = !switchTile;
        layerManager.paint(g, 0, 0);
        flushGraphics();
    }

    private TiledLayer initBackground() throws Exception {
        Image tileImages = Image.createImage("/tiles.png");
        TiledLayer tiledLayer = new TiledLayer(10, 10, tileImages, 32, 32);
        int[] map = { 5, 1, 1, 4, 1, 1, 1, 1, 1, 6, 5, 1, 3, 1, 1, 3, 1, 1, 1,
                6, 5, 1, 2, 1, 1, 2, 1, 1, 1, 6, 5, 1, 2, 3, 1, 2, 1, 1, 1, 6,
                5, 1, 4, 2, 1, 2, 1, 1, 1, 6, 5, 1, 1, 4, 1, 2, 1, 1, 1, 6, 5,
                1, 1, 1, 1, 4, 1, 1, 1, 6, 5, 1, 1, 1, 1, 1, 1, 1, 1, 6, 5, 1,
                1, 1, 1, 1, 1, 1, 1, 6, 5, 1, 1, 1, 1, 1, 1, 1, 1, 6 };
        for (int i = 0; i < map.length; i++) {
            int column = i % 10;
            int row = (i - column) / 10;
            tiledLayer.setCell(column, row, map[i]);
        }
        // Created animate tile and hold animated tile index
        animatedIdx = tiledLayer.createAnimatedTile(5);
        // Set Cell with animated tile index
        tiledLayer.setCell(1, 1, animatedIdx);
        return tiledLayer;
    }
}
```

Main Midlet Source Code:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExampleTiledLayerAnimated extends MIDlet {
    private Display display;

    public void startApp() {
```

```
        try {
            display = Display.getDisplay(this);
            ExampleGameCanvas gameCanvas = new
ExampleGameCanvas();
            gameCanvas.start();
            display.setCurrent(gameCanvas);
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        exit();
    }

    public void exit() {
        System.gc();
        destroyApp(false);
        notifyDestroyed();
    }
}
```

模拟器输出：

“梯子顶部”和“梯子底部”两个图片组成的动画

第五章 数学约束

5.1 概述

如果打算开发高交互性的特定游戏类型，如动作类游戏，则需要更接近现实的游戏算法。一个方法是在游戏中增加更多的与游戏相关的基础数学内容。本章不会详细介绍游戏设计中的所有数学内容，那也许整整一本书都讲不完，而是介绍在 J2ME 编程中如何规避数学的约束。数学约束中最关键的是缺少小数的支持，从而三角函数等在 J2ME 中都是不支持的。

5.2 定点数

什么是定点数？定点数是 float 或 double 这种类型的扩展。为什么需要定点数？一个原因是 MIDP1.0 或者 MIDP2.0 中都没有这种数学形式，所以，你不能得出下面算式的结果：

2.245×5.423

当然 J2ME 也不是完全不支持数学运算，它包括有限的数学支持，包括对整型和长整型的支持，你甚至不能用它做任何运算。相关函数如下：

- abs(int a)－返回绝对值
- abs(long a)－返回绝对值
- max(int a,int b)－返回较大值
- max(long a, long b)－返回较大值
- min(int a,int b)－返回较小值
- min(long a,long b)－返回较小值

使用定点数还有两个优点：一个是所有定点数之间都有一个相同间隔，而不像分数那样随着指数的增加，各个分数之间的间隔会增大；另一优点是在同一个平台上同一种语言扩展的函数中，定点数运算要比分数运算快得多。

在介绍了定点数的基础后，接下来的问题是如何使用它以及如何获得定点数，具体来说就是定点小数的处理。举个例子来说，如果要精确到小数点后面 1000 位，（在 J2ME 开发中这已经足够了）不论它是不是小数都会在小数点后包含固定的有效数字，在本例中是三位有效数字。例如数值 1 将表示成 1.000，当然由于 J2ME 不支持小数值故表示为 1000，所以需要跟踪小数点的位置。从而，实际的计算还是整型或者长整型的数值计算。前面的例子的计算过程如下：

2245 乘以 5423，同时记录小数点的位置。

现在可以自己扩展一个定点数类，或者使用其它人扩展好的类，当然，使用他人的代码时切记遵守使用许可。下面是一些相关类的链接：

- http://bearlib.sourceforge.net
- http://home.rochester.rr.com/ohommes/MathFP
- http://henson.newmail.ru/j2me/Float.htm

处理数字的时候，你也许希望通过移位操作来提高运算效率。当然，不管是你自主开发的还是别人的代码，你都会想要修改其原始版本以更便于游戏开发中使用。也就是说，代码不单要考虑到功能的实现，还要考虑到大小，否则移动设备也许根本没有足够的空间。

5.3 基础三角函数和精灵移动

最终，你可以扩展 sin、cos、tan，但是先调用其它扩展类（如定点数类）来实现三角函数计算功能，运算效率会比较低。在不知道角度范围的情况下，这是没有选择余地的，但是，在移动游戏领域，由于屏幕大小的限制，定义 16 个方向的移动就可以使精灵在各个方向平滑的移动。这些方向可以很容易的计算，并存储在一个查找表中，下表列出了各个角度对应的三角函数值。在表中，我们使用定点数表示三角函数值，如 $\sin(22.5^{o})=0.383$，定点数表示为 383。下面的章节将介绍 2D 变换或称 2D 精灵变换，涉及对图像进行翻转和镜像的实现。

角度	Sin(小数值×1000)	Cos(小数值×1000)
0/360	0	1000
22.5	383	924
45	707	707
67.5	924	383
90	1000	0
112.5	924	-383
135	707	-707
157.5	383	-924
180	0	-1000
202.5	-383	-924
225	-707	-707
247.5	-924	-383
270	-1000	0
292.5	-924	383
315	-707	707
360	0	1000

5.4 精灵移动实例

5.4.1 Graphics Credit

在讲解实例之前，不得不满怀敬意地提到Ari Feldman，他已经在GPL（General Public License）下发布了很多精灵，更多有关Ari Feldman和他的精灵的信息请参考他的个人网址http://www.arifeldman.com。

5.4.2 不包含定点数

在例子中，一个简单的精灵——坦克，它的移动基于整型数的，可以通过对应的按键使坦克前进、后退、转向。

5.4.3 包含定点数

5.4.4 2D 转换

第二篇　Eliminator 实 例

- 介绍
- 启动画面
- 游戏主菜单
- 异常处理
- 设置和高分记录
- 地形（卷动背景）
- 玩家和子弹
- 场景切换
- 游戏内容
- 老怪（Boss）
- 游戏其它

第六章 Eliminator：介绍

6.1 概述

在随后的一些章节中我们将学习一个垂直滚动射击游戏的制作，主要目的是理解一个典型的游戏的编码规范，同时也学习将前面所学知识包括 MIDP2.0 的类结合到实际应用中，以及处理移动设备限制的挑战。虽然这个游戏不一定是下一个流行游戏，但是它将揭示开发完整游戏从开始到结束的基本过程。

说到源代码，这里会使用 J2ME 仿真器，更多有关开发游戏要用到的工具的信息参考本书的相关部分。在实际将游戏上载到移动设备上时需要对游戏进行适当的调整。

代码通常尽量写得用户友好些，并且预留区域用来进行以后的设计、内存使用和性能方面的扩展。

6.2 工具使用

这里使用Sun公司的WTK2.0（J2ME Wireless Toolkit version 2.0），它在渲染滚动的TiledLayer时有一些问题，Nokia的J2ME开发包对滚动TiledLayer处理得更有效率，最好的选择是使用Nokia的开发包。在本书创作过程中，据说Sun公司新发布的WTK已经解决了这些问题。

6.3 简短的游戏描述

游戏名称为：Eliminator，它是类似于Raiden或1942的竖直滚动的射击游戏，游戏的主要特征如下：

- 基于使用TiledLayer类预定义好的地图的三级关卡
- 没级关卡都有一个老怪（BOSS）
- 两个供选择船只（玩家控制）
- 自动存档
- 最高得分（本地或网络同步）
- 各种不同的武器(Wing Man、炸弹Bombs、Spread、激光Laser、防护罩shield)

<u>完成的产品</u>

///这里放置截屏

第七章 Eliminator：游戏启动画面

7.1 概述

启动画面通常在主菜单显示之前，游戏加载时立即出现。虽然启动画面不是必须的，但是它增加了游戏的吸引力。它一般显示一些图片、公司信息、公司注册商标等。

启动画面可以用Alert类简单的生成，也可以结合timer类和canvas类定制启动画面。启动画面只需显示很短的时间，一般为三秒。在用户化的启动画面中最好有选项可以使用户可以跳过启动画面立即进入游戏主菜单。

7.2 源代码

这里直接使用了Timer类和Canves类，而时间倒计时控制使用了另外一个名为CountDown的类。CountDown类可以重复应用到未来的开发中，也许需要更为复杂的计数。这段代码本书的后面还将用到，但将不再列出。

当然，启动画面本身没有任何意义，它需要主类Midlet调用才有用。详细代码参考游戏主菜单那一章。

SplashScreen.java

```
import java.util.Timer;
import javax.microedition.lcdui.*;

public final class SplashScreen extends Canvas {
    private Display display;
    private Displayable next;
    private Timer timer;
    private Image image;
    private int dismissTime;

    public SplashScreen(Display display, Displayable next, Image image,
            int dismissTime) {
        timer = new Timer();
        this.display = display;
        this.next = next;
        this.image = image;
        this.dismissTime = dismissTime;
        display.setCurrent(this);
    }
```

```
    static void access(SplashScreen splashScreen) {
        splashScreen.dismiss();
    }

    private void dismiss() {
        timer.cancel();
        display.setCurrent(next);
    }

    protected void keyPressed(int keyCode) {
        dismiss();
    }

    protected void paint(Graphics g) {
        g.setColor(0x00FFFFFF);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0x00000000);
        g.drawImage(image, getWidth() / 2, getHeight() / 2 - 5, 3);
    }

    protected void pointerPressed(int x, int y) {
        dismiss();
    }

    protected void showNotify() {
        if (dismissTime > 0)
            timer.schedule(new CountDown(this), dismissTime);
    }
}
```

CountDown.java

```
import java.util.TimerTask;

class CountDown extends TimerTask {
    private final SplashScreen splashScreen;

    CountDown(SplashScreen splashScreen) {
        this.splashScreen = splashScreen;
    }

    public void run() {
        SplashScreen.access(this.splashScreen);
    }
}
```

启动画面截屏

图24：“启动画面”截图

第八章 Eliminator：游戏主菜单

8.1 概述

主菜单无论对游戏还是应用程序都很重要，在移动设备上更是如此。造就一款成功的游戏，抛开其它各种因素不谈，主菜单的设计非常重要，其关键是可用性。

主菜单的可用性是指高效、易用、易学、易记。如果一个邮件客户端难以使用或者用户不友好，而无论定价多少，用户都不会使用。在游戏中，如果用户不友好就更糟，因为即使玩家对游戏感兴趣，由于菜单的原因使游戏难以控制，玩家也不会再玩这个游戏。一个可用的主菜单是非常关键的，所有主菜单系统必须在游戏设计时就进行。而且如果第一次就做得比较好，可以在以后得游戏开发中继续沿用相同的主菜单帧。

8.2 主菜单外观

设计主菜单有两个大的方向，一个是使用MIDP扩展得高级GUI组件，一个是自己扩展图像主菜单系统。下面两个表是它们的优缺点比较：

高级GUI组件

优点	缺点
易扩展	视觉吸引力不强
符合移动设备的外观	有些功能不能实现，如滚转图像转换
代码量少，减少最终jar大小	经常需要用户向下滚动来查看选单；如果能避免用户滚动菜单将会非常好
高级GUI更易于不同机型、不同厂商之间的移植	
易于维护和更改	
易于在其它游戏中复用	

扩展图像主菜单

优点	缺点
视觉吸引力强	更多的设计和扩展
加强公司标志	更多资源，如公司图像等
加强客户程序标志	更多代码，增加最终jar文件大小
更多控制和定制。如可以用小字体和间距使菜单控制在一个屏幕中；避免用户滚动	降低了菜单可复用性
	对菜单的改动需要更多资源，如新的主菜单需要新的图片

8.3 基本主菜单

下图说明了一个基本的主菜单流程：

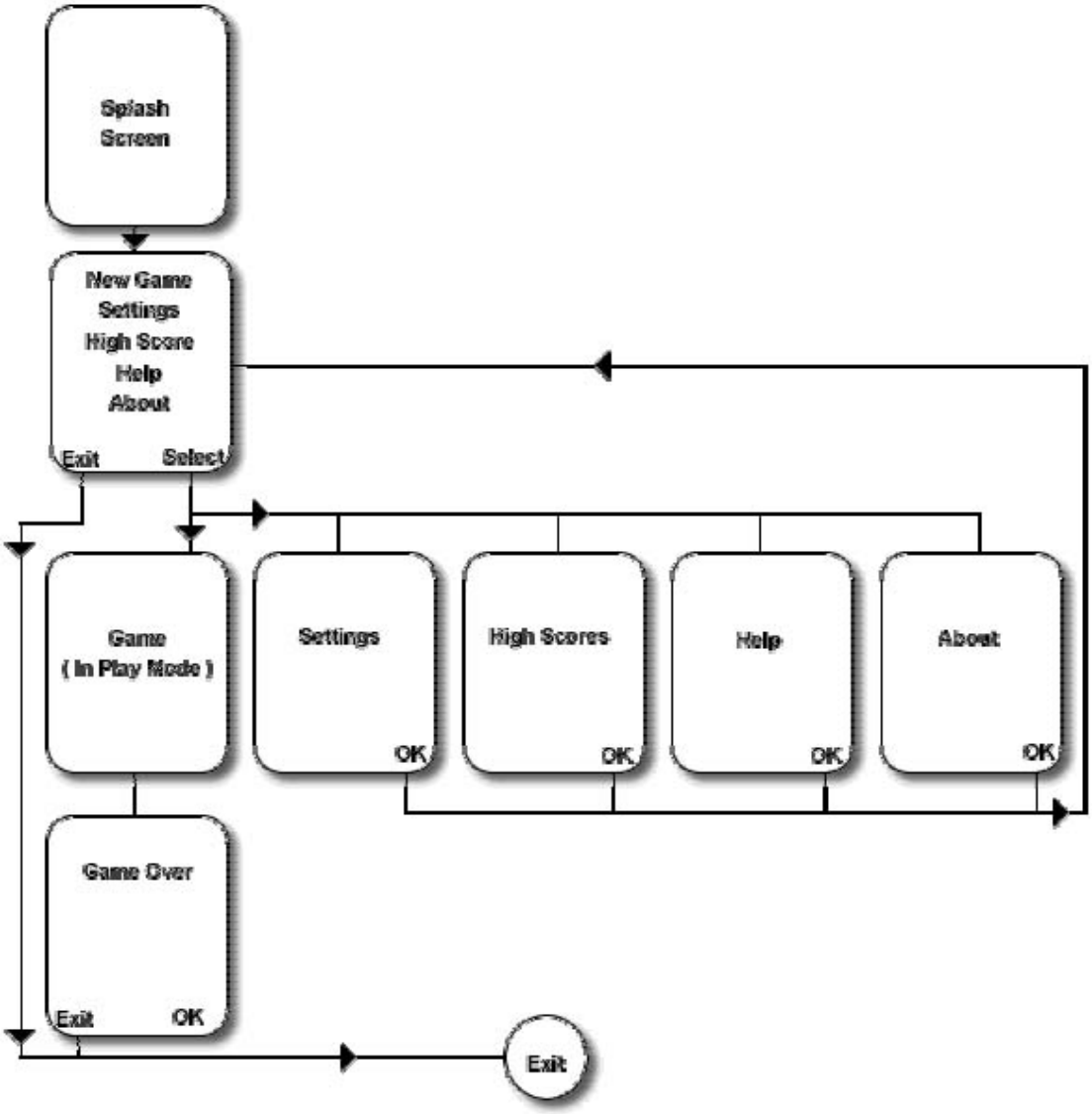

图25：基本主菜单流程图

例子中的*设置Settings*、*最高分High Scores*、*帮助Help*、*关于About*都是警告对话框（Alert box）。*帮助Help*、*关于About*使用警告对话框（Alert box）比较合理，但是*设置Settings*、*最高分High Scores*也使用警告对话框（Alert box）还是不太符合逻辑。这里这样做只是为了简单起见，而不建议你所有主菜单都这样制作。下一节将对*设置Settings*、*最高分High Scores*使用子菜单。甚至像Tic-Tac-Toe这样的游戏都没有最高分和设置这样的选项，可见本例提供的主菜单帧还是比较完善的。

基本主菜单源代码：

```
import javax.microedition.lcdui.*;

public class MainMenuScreen extends List implements CommandListener {
    private Eliminator midlet;
    private Command selectCommand = new Command("Select",
Command.ITEM, 1);
    private Command exitCommand = new Command("Exit",
Command.EXIT, 1);
    private Alert alert;

    public MainMenuScreen(Eliminator midlet) {
        super("Eliminator", Choice.IMPLICIT);
        this.midlet = midlet;
        append("New Game", null);
        append("Settings", null);
        append("High Scores", null);
        append("Help", null);
        append("About", null);
        addCommand(exitCommand);
        addCommand(selectCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            midlet.mainMenuScreenQuit();
            return;
        } else if (c == selectCommand) {
            processMenu();
            return;
        } else {
            processMenu();
            return;
        }
    }

    private void processMenu() {
        try {
            List down = (List) midlet.display.getCurrent();
            switch (down.getSelectedIndex()) {
            case 0:
                scnNewGame();
                break;
            case 1:
```

```
                scnSettings();
                break;
            case 2:
                scnHighScores();
                break;
            case 3:
                scnHelp();
                break;
            case 4:
                scnAbout();
                break;
            }
            ;
        } catch (Exception ex) {
            // Proper Error Handling should be done here
            System.out.println("processMenu::" + ex);
        }
    }

    private void scnNewGame() {
        midlet.mainMenuScreenShow(null);
    }

    private void scnSettings() {
        alert = new Alert("Settings", "Settings.....", null, null);
        alert.setTimeout(Alert.FOREVER);
        alert.setType(AlertType.INFO);
        midlet.mainMenuScreenShow(alert);
    }

    private void scnHighScores() {
        alert = new Alert("High Scores", "High Scores.....", null, null);
        alert.setTimeout(Alert.FOREVER);
        alert.setType(AlertType.INFO);
        midlet.mainMenuScreenShow(alert);
    }

    private void scnHelp() {
        alert = new Alert("Help", "Help.....", null, null);
        alert.setTimeout(Alert.FOREVER);
        alert.setType(AlertType.INFO);
        midlet.mainMenuScreenShow(alert);
    }

    private void scnAbout() {
        alert = new Alert("About", "Eliminator\nVersion 1.0.0\nby Jason
```

```
Lam",
                    null, null);
            alert.setTimeout(Alert.FOREVER);
            alert.setType(AlertType.INFO);
            midlet.mainMenuScreenShow(alert);
        }
}
```

Main Midlet 源代码：

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Eliminator extends MIDlet {
    protected Display display;
    private Image splashLogo;
    private boolean isSplash = true;
    MainMenuScreen mainMenuScreen;

    public Eliminator() {
    }

    public void startApp() {
        display = Display.getDisplay(this);
        mainMenuScreen = new MainMenuScreen(this);
        if (isSplash) {
            isSplash = false;
            try {
                splashLogo = Image.createImage("/splash.png");
                new SplashScreen(display, mainMenuScreen, splashLogo,
3000);
            } catch (Exception ex) {
                mainMenuScreenShow(null);
            }
        } else {
            mainMenuScreenShow(null);
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }
```

```
    public void destroyApp(boolean unconditional) {
        System.gc();
        notifyDestroyed();
    }

    private Image createImage(String filename) {
        Image image = null;
        try {
            image = Image.createImage(filename);
        } catch (Exception e) {
        }
        return image;
    }

    public void mainMenuScreenShow(Alert alert) {
        if (alert == null)
            display.setCurrent(mainMenuScreen);
        else
            display.setCurrent(alert, mainMenuScreen);
    }

    public void mainMenuScreenQuit() {
        destroyApp(true);
    }
}
```

Main Midlet调用启动画面，控制着所有屏幕显示，这在下一节*包含子菜单的主菜单*中更加明显。其一可能看起来比较奇怪：

```
public void mainMenuScreenShow(Alert alert) {
    if (alert==null)
    display.setCurrent(mainMenuScreen);
    else
    display.setCurrent(alert,mainMenuScreen);
}
```

为什么不直接写成：

```
public void mainMenuScreenShow() {
    display.setCurrent(alert,mainMenuScreen);
}
```

这里的alert用来满足在转向下一个屏幕前想先显示一条信息的功能;也可以更好的进行错误处理。例如，当用户选择最高分，但是一个在加载最高分的时候错误发生了，就可以显示一个错误信息同时返回主菜单。

基本主菜单截屏：

图26：基本主菜单截屏

8.4 包含子菜单的主菜单

在这一节，我们用form组件替换掉了*最高分High Scores*、*帮助Help*、*关于About*原来的Alert消息；*设置Settings*使用了新的List组件来显示两个子菜单；*游戏结束Game Over*则是包含分配了软键的菜单项的GameCanvas。

屏幕内容是虚设的，将在后面的章节中进一步完善，本节的目标是基本理解Eliminator游戏的菜单流程。而且这种菜单流程也可以用在以后的游戏制作中。下图是包含子菜单的主菜单的图示：

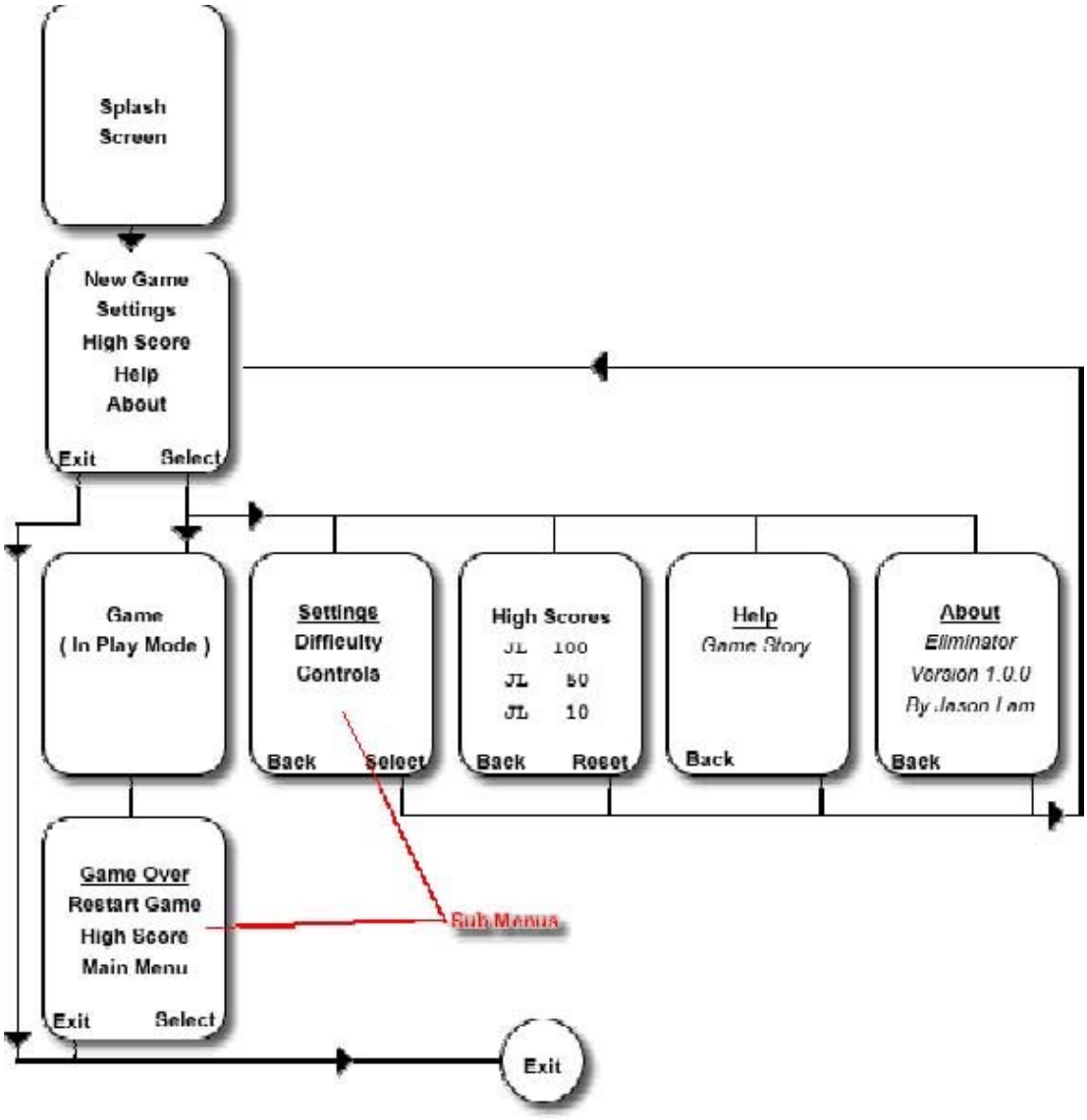

图27：包含子菜单的主菜单的图示

下面是每个相关屏幕和主菜单、主类的对应代码段：

主 Midlet 源代码：

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Eliminator extends MIDlet {
    protected Display display;
    private Image splashLogo;
    private boolean isSplash = true;
    private MainMenuScreen mainMenuScreen;
    private SettingsScreen settingsScreen;
    private HighScoreScreen highScoreScreen;
    private HelpScreen helpScreen;
    private AboutScreen aboutScreen;

    public Eliminator() {
    }

    public void startApp() {
        display = Display.getDisplay(this);
        mainMenuScreen = new MainMenuScreen(this);
        settingsScreen = new SettingsScreen(this);
        highScoreScreen = new HighScoreScreen(this);
        helpScreen = new HelpScreen(this);
        aboutScreen = new AboutScreen(this);
        if (isSplash) {
            isSplash = false;
            try {
                splashLogo = Image.createImage("/splash.png");
                new SplashScreen(display, mainMenuScreen, splashLogo,
3000);
            } catch (Exception ex) {
                mainMenuScreenShow();
            }
        } else {
            mainMenuScreenShow();
        }
    }

    public Display getDisplay() {
        return display;
    }

    public void pauseApp() {
    }
```

```
    public void destroyApp(boolean unconditional) {
        System.gc();
        notifyDestroyed();
    }

    private Image createImage(String filename) {
        Image image = null;
        try {
            image = Image.createImage(filename);
        } catch (Exception e) {
        }
        return image;
    }

    public void mainMenuScreenShow() {
        display.setCurrent(mainMenuScreen);
    }

    public void settingsScreenShow() {
        display.setCurrent(settingsScreen);
    }

    public void highScoreScreenShow() {
        display.setCurrent(highScoreScreen);
    }

    public void helpScreenShow() {
        display.setCurrent(helpScreen);
    }

    public void aboutScreenShow() {
        display.setCurrent(aboutScreen);
    }

    public void mainMenuScreenQuit() {
        destroyApp(true);
    }
}
```

包含子菜单的主菜单源代码

```
import javax.microedition.lcdui.*;

public class MainMenuScreen extends List implements CommandListener {
    private Eliminator midlet;
    private Command selectCommand = new Command("Select",
```

```
Command.ITEM, 1);
    private Command exitCommand = new Command("Exit", Command.EXIT, 1);

    public MainMenuScreen(Eliminator midlet) {
        super("Eliminator", Choice.IMPLICIT);
        this.midlet = midlet;
        append("New Game", null);
        append("Settings", null);
        append("High Scores", null);
        append("Help", null);
        append("About", null);
        addCommand(exitCommand);
        addCommand(selectCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            midlet.mainMenuScreenQuit();
        } else if (c == selectCommand) {
            processMenu();
        } else {
            processMenu();
        }
    }

    private void processMenu() {
        try {
            List down = (List) midlet.display.getCurrent();
            switch (down.getSelectedIndex()) {
            case 0:
                scnNewGame();
                break;
            case 1:
                scnSettings();
                break;
            case 2:
                scnHighScore();
                break;
            case 3:
                scnHelp();
                break;
            case 4:
                scnAbout();
                break;
```

```
                }
                ;
            } catch (Exception ex) {
                // Proper Error Handling should be done here
                System.out.println("processMenu::" + ex);
            }
        }

        private void scnNewGame() {
            midlet.mainMenuScreenShow();
        }

        private void scnSettings() {
            midlet.settingsScreenShow();
        }

        private void scnHighScore() {
            midlet.highScoreScreenShow();
        }

        private void scnHelp() {
            midlet.helpScreenShow();
        }

        private void scnAbout() {
            midlet.aboutScreenShow();
        }
}
```

“设置”界面的源代码

```
import javax.microedition.lcdui.*;

public class SettingsScreen extends List implements CommandListener {
    private Eliminator midlet;
    private Command backCommand = new Command("Back",
Command.BACK, 1);
    private Command selectCommand = new Command("Select",
Command.SCREEN, 1);

    public SettingsScreen(Eliminator midlet) {
        super("Settings", Choice.IMPLICIT);
        append("Difficulty", null);
        append("Controls", null);
        this.midlet = midlet;
        addCommand(backCommand);
```

```
        addCommand(selectCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow();
        } else if (c == selectCommand) {
            processMenu();
        } else {
            processMenu();
        }
    }

    private void processMenu() {
        try {
            List down = (List) midlet.display.getCurrent();
            switch (down.getSelectedIndex()) {
            case 0:
                setDifficulty();
                break;
            case 1:
                setControls();
                break;
            }
            ;
        } catch (Exception ex) {
            // Proper Error Handling should be done here
            System.out.println("processMenu::" + ex);
        }
    }

    private void setDifficulty() {
        System.out.println("Difficutly Settings Not Implemented Yet");
    }

    private void setControls() {
        System.out.println("Controls Settings Not Implemented Yet");
    }
}
```

“最高分”界面的源代码

```
import javax.microedition.lcdui.*;

public class HighScoreScreen extends Form implements CommandListener {
```

```
    private Eliminator midlet;
    private Command backCommand = new Command("Back", Command.BACK, 1);
    private Command resetCommand = new Command("Rest", Command.SCREEN, 1);

    public HighScoreScreen(Eliminator midlet) {
        super("High Score");
        this.midlet = midlet;
        StringItem stringItem = new StringItem(null, "JL 100\nJL 50\nJL 10");
        append(stringItem);
        addCommand(backCommand);
        addCommand(resetCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow();
            return;
        }
        if (c == resetCommand) {
            // not implemented yet
            System.out.println("Reset High Scores Not Implemented Yet");
        }
    }
}
```

“帮助”界面的源代码

```
import javax.microedition.lcdui.*;

public class HelpScreen extends Form implements CommandListener {
    private Eliminator midlet;
    private Command backCommand = new Command("Back", Command.BACK, 1);

    public HelpScreen(Eliminator midlet) {
        super("Help");
        this.midlet = midlet;
        StringItem stringItem = new StringItem(
                null,
            "It is the year 3023, many things have changed over the years "
        + "including the Great Migration to Earth II. "
        + "But the one thing that hasn't changed is the struggle for "
```

```
            + "Power and Wealth. You are one the last remaning hopes "
            + "to defend Earth II from total dominance by Evil Ruler named "
            + "Zikron. You mission is to eliminate Zikron and his "
            + "minions from the face of the Earth II");
            append(stringItem);
            addCommand(backCommand);
            setCommandListener(this);
        }

        public void commandAction(Command c, Displayable d) {
            if (c == backCommand) {
                midlet.mainMenuScreenShow();
                return;
            }
        }
}
```

“关于”界面的源代码

```
import javax.microedition.lcdui.*;

public class AboutScreen extends Form implements CommandListener {
    private Eliminator midlet;
    private Command backCommand = new Command("Back",
Command.BACK, 1);

    public AboutScreen(Eliminator midlet) {
        super("About");
        this.midlet = midlet;
        StringItem stringItem = new StringItem(null,
                "Eliminator\nVersion 1.0.0\nBy Jason Lam");
        append(stringItem);
        addCommand(backCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow();
            return;
        }
    }
}
```

包含子菜单的菜单模拟器截屏

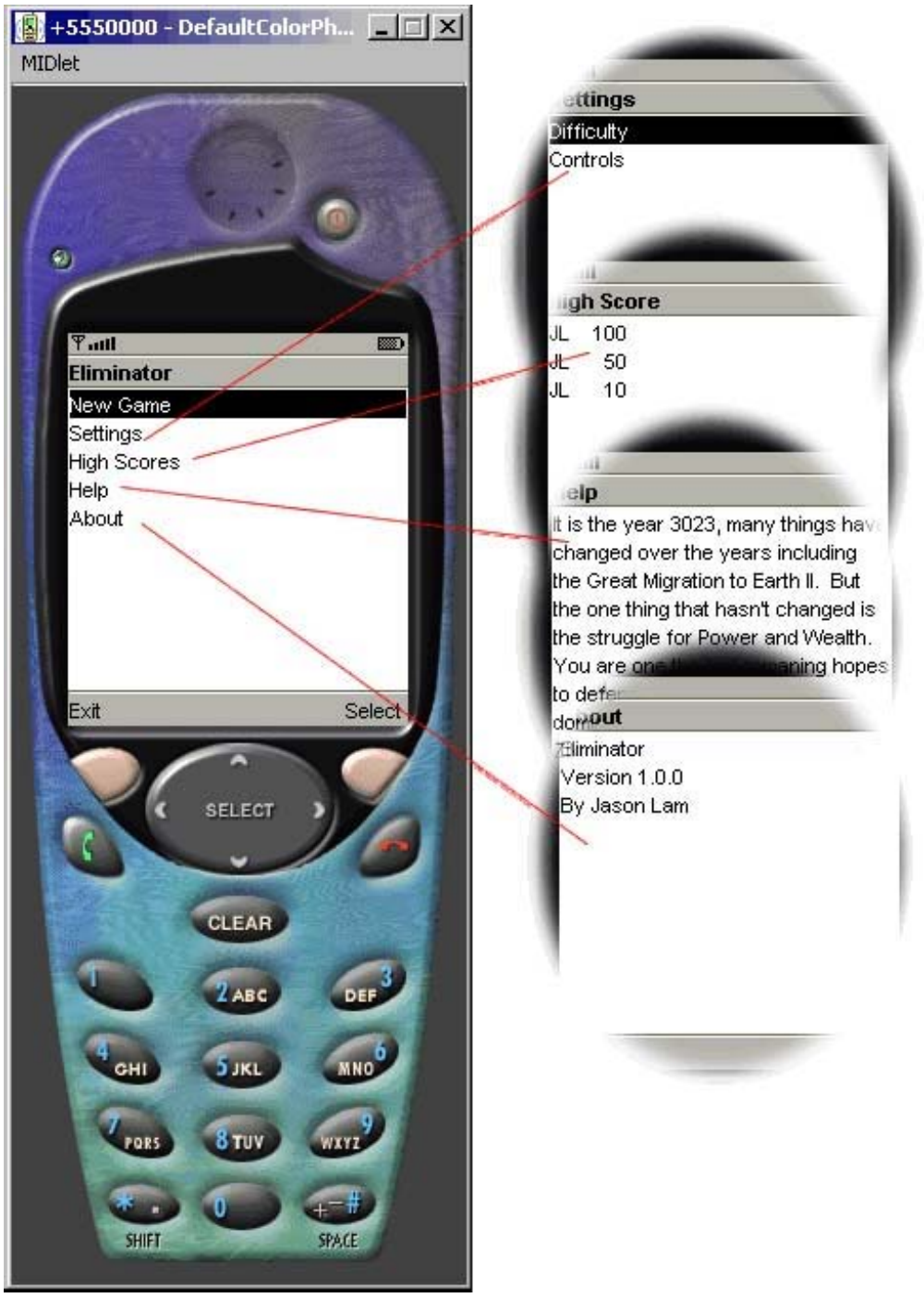

图28：包含子菜单的菜单模拟器截屏

8.5 菜单设计的建议和提高

前面几节介绍了显示菜单的相关知识，确保可用性的前提下，有很多方法实现菜单的显示。这里有一些建议和提供，或者称改变可以用在前面的代码中：

1. 在启动画面中加入包含跳过选择的按键以使用户可以跳过启动画面。同时，加入一个*离开Exit*按键使用户可以根据需要立即离开游戏。

2. 上面例子中的软键不一定适合所有的移动设备，需要同运行游戏的移动设备的运营商、制造商、集团协商按键的标志、菜单及按键的位置等。例如，上面代码中的*Back*和*Exit*是左软件，但在Nokia Series 40则是右软键。又如，一些集团比较喜欢Options而不是Setting。

3. 在了解了菜单系统的设计过程之后，你会发现即使主菜单没有上面列出的所有选项依然可以满足用户需求。事实上，并不需要如此多的游戏选项，因为其所导致的滚动会使用户不爽。然而，为了使游戏健壮、易维护，你也许会使用设计模式：MVC模式、Command模式、针对菜单过程的Reflection。但是，不要忘记，在添加更为健壮的代码带来更多内存使用的同时，我们甚至还没有开始编写游戏呢！当前，如果需要添加或移除菜单项，你只要在Elminator.java和MainMenuScreen.java文件中编辑一下就好了，同时添加或删除[name]Screen.java文件。

4. 如果你决定不使用高级GUI组件，而选择自己扩展图形界面，则你需要做很多更改，并且，请在一次确定你的界面是易学易用的。

8.6 小结

现在你掌握了菜单选项的流程和菜单系统的源代码编写方法，切记好的菜单是造就优秀游戏和应用程序的重要因素，菜单要高效、易用、易学、易记。游戏不但要好玩，还要容易进入。

第九章 Eliminator：异常处理

随着加入越来越多的代码以得到Eliminator的最终版本，我们需要添加一些处理错误和异常的功能。有很多种方法实现异常处理，直观起见，这里直接在主midlet函数中抛出所有异常；也就是说所有其它的类需要传播异常。

这里仅覆盖了游戏执行过程中的一些常见的异常抛出，在将游戏发布之前，你需要仔细研究你的游戏代码，并对所有可能的异常进行处理。异常处理超出了本书的范围。

下面列出的是需要更改的源代码，主要的变化是主midlet在需要的适合抛出一个alert对话框，从而使用户可以在主菜单显示之前看到错误信息，当然，前提是主菜单代码没有问题。无论如何，当用户在玩游戏时加载*高分记录High Scores*，或者更新*设置Setting*信息时，如果出现错误，游戏就会显示异常信息，并且返回主菜单。

Alert 方法源代码:

```
protected void showErrorMsg(String alertMsg) {
    if (alertMsg == null || CONST_DEBUG == false) {
        alertMsg = "Error Starting Elminator, may or may not function
        correctly. Please contact support.";
    }
    alert =new Alert("Eliminator ERROR",alertMsg,null,null);
    alert.setTimeout(Alert.FOREVER);
    alert.setType(AlertType.ERROR);
    this.mainMenuScreenShow(alert);
}
```

主菜单显示方法源代码：

```
protected void mainMenuScreenShow(Alert alert) {
    if (alert==null)
        display.setCurrent(mainMenuScreen);
    else
        display.setCurrent(alert,mainMenuScreen);
}
```

ShowErrorMsg方法中的CONST_DEBUG==false相当于一个开关，它允许开发者在开发过程中输出特定的开发信息，当发布产品时，可以通过更改布尔值来仅仅输出一般用户信息。当然，这取决于是否愿意向用户输出有意义的信息，如错误代码；这一般由游戏开发者决定，或者遵从制造商、运营商或者集团的意愿。

异常实例的模拟器截屏

本例中，将splash.png从工程文件中移除，从而显示错误信息：

图29：Eliminator出现错误的模拟器截屏

第十章 Eliminator：设置和高分记录

10.1 概述

在编写游戏本身代码之前还有一些内容需要设置，包括游戏困难度设置、控制设置和高分记录。这些都需要持久性存储，即需要记录管理系统（Record Management System，RMS）的支持。如果你不熟悉RMS，请在继续阅读本章的剩余部分之前先阅读rms相关的资料。

设置和高分记录将被存储在两个独立的记录空间。简单起见，控制设置只有一个选项：开启或关闭自动射击。从而，*设置Setting*子菜单将用*Difficulty*和*Auto Fire*代替*Difficulty*和*Controls*。

作为示范的目的，我们包含了设置难度的选项；然而通常情况下，如果存在设置游戏模式的选项，则必须在游戏中添加一个有效的值来控制游戏的模式。但是，如前面所提到的那样，这些章节的主要目的是介绍J2ME游戏编程，而不是希望开发出一款成功的游戏，所以，仅仅给出了不完全的功能实例。

10.2 RMS 数据结构

*设置Setting*的存储模式是由两个整型字段组成一个记录（行）。

名称	数据类型	值	内容
难度	Integer	1、2、3	1＝容易，2＝一般，3＝困难
自动射击	Integer	1、0	1＝开启，0＝关闭

高分记录的存储格式是三行记录，每行记录都由两个字段组成；记录1对应第一行，记录2对应第二行，记录3对应第三行。

名称	数据类型	值	内容
高分记录（初始值或名称）	String	3 Char	
分值	Integer		

10.3 设置 Settings 和高分记录 High Score 设计

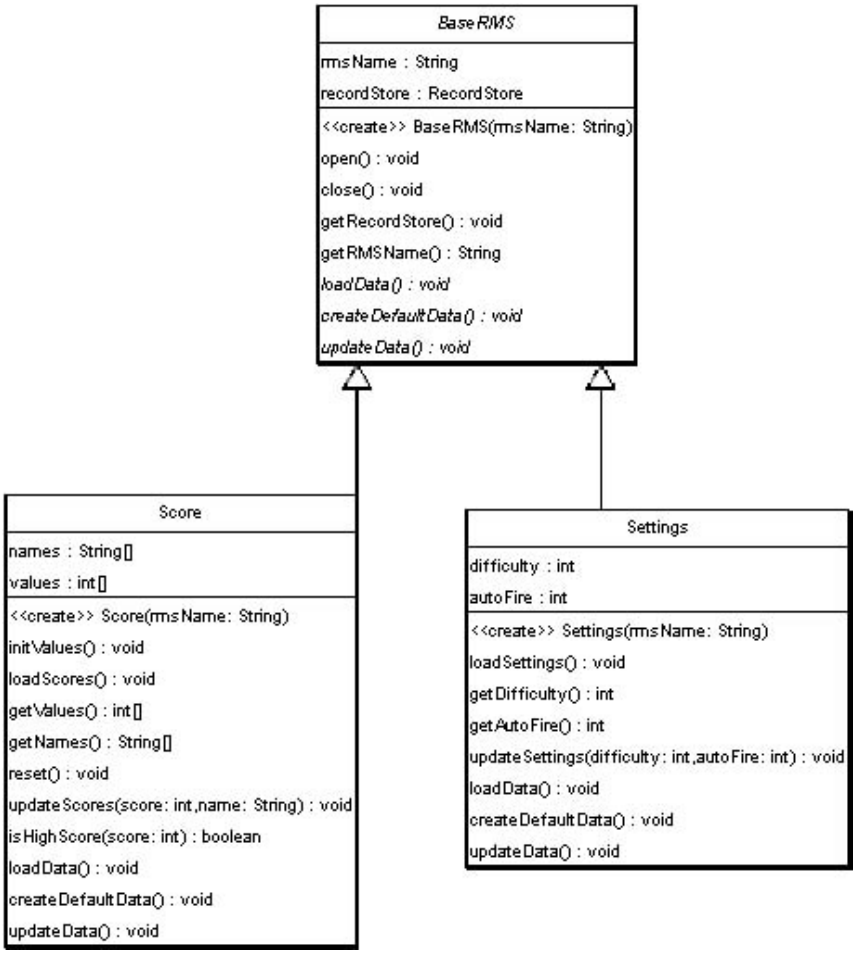

图30：Settings和High Score的UML类图

因为设置和高分记录存在公共的功能，所以使用继承是合理的设计方法。不仅可以容易地增加用到RMS地子类；也可以在其它项目中重用选中的组件。例如，在另一个游戏中也要用到Settings，你需要做的只是拷贝BaseRMS类和Score类，而不是仅仅使用Settings类。这种结构也利于在不影响其它代码块的前提下，编辑或完全移除确定地组件。

相比较同时使用三个类，虽然在一个类中可以更容易地实现所有功能。但是浪费少量的内存空间来换取程序的更易于维护是值得的。

10.4 源代码

这里添加了三个新的类：BaseRMS.java、Score.java 和 Settings.java，同时为了使用Settings和Score功能对Eliminator.java、 SettingsScreen.java 和 HighScore.java

三个类作了相应的改变。

10.4.1 BaseRMS

BaseRMS是实现RMS功能的基础类，它扩展了open和close两个方法，同时定义了三个抽象方法。Close方法用来关闭连接；Open方法用于打开记录空间，如果它不能找到现存记录空间就会自动建立一个缺省的记录空间，如果找到现存记录，就会加载它。

```
import javax.microedition.rms.*;
import java.util.*;
import java.io.*;

abstract public class BaseRMS {
    private String rmsName;

    private RecordStore recordStore;

    BaseRMS(String rmsName) {
        this.rmsName = rmsName;
    }

    public void open() throws Exception {
        try {
            recordStore = RecordStore.openRecordStore(this.rmsName,
true);
            if (recordStore.getNumRecords() > 0) {
                loadData();
            } else {
                createDefaultData();
            }
        } catch (Exception e) {
            throw new Exception(this.rmsName + "::open::" + e);
        }
    }

    public void close() throws Exception {
        if (recordStore != null) {
            try {
                recordStore.closeRecordStore();
            } catch (Exception e) {
                throw new Exception(this.rmsName + "::close::" + e);
            }
        }
    }
```

```
    public RecordStore getRecordStore() {
        return this.recordStore;
    }

    public String getRMSName() {
        return this.rmsName;
    }

    abstract void loadData() throws Exception;
    abstract void createDefaultData() throws Exception;
    abstract void updateData() throws Exception;
}
```

10.4.2 Score

Score类继承了BaseRMS类，并且扩展了抽象方法，源代码如下：

```
import javax.microedition.rms.*;
import java.util.*;
import java.io.*;

public class Score extends BaseRMS {
    private String[] names = new String[3];

    private int[] values = new int[3];

    public Score(String rmsName) {
        super(rmsName);
        initValues();
    }

    private void initValues() {
        names[0] = "JCL";
        names[1] = "YYK";
        names[2] = "RKL";
        values[0] = 100;
        values[1] = 50;
        values[2] = 10;
    }

    public void loadScores() throws Exception {
        try {
            // Will call either loadData() or createDefaultData()
            this.open();
```

```
            // Add any other neccessary processing, in this case there is none
            // Close
            if (this.getRecordStore() != null)
                this.close();
        } catch (Exception e) {
            throw new Exception("Error loading Scores" + e);
        }
    }

    public int[] getValues() {
        return this.values;
    }

    public String[] getNames() {
        return this.names;
    }

    public void reset() throws Exception {
        this.open();
        initValues();
        updateData();
        if (this.getRecordStore() != null)
            this.close();
    }

    public void updateScores(int score, String name) throws Exception {
        try {
            for (int i = 0; i < names.length; i++) {
                if (score >= values[i]) {
                    // load current scores
                    this.open();
                    // Shift the score table.
                    for (int j = names.length - 1; j > i; j--) {
                        values[j] = values[j - 1];
                        names[j] = names[j - 1];
                    }
                    // Insert the new score.
                    this.values[i] = score;
                    this.names[i] = name;
                    // Update High Scores
                    updateData();
                    // close
                    if (this.getRecordStore() != null)
                        this.close();
                    break;
                }
```

```
                }
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::updateScores::"
+ e);
            }
        }

        public boolean isHighScore(int score) throws Exception {
            boolean isHighScore = false;
            try {
                for (int i = 0; i < names.length; i++) {
                    if (score >= values[i])
                        isHighScore = true;
                }
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::isHighScore::" +
e);
            }
            return isHighScore;
        }

        protected void loadData() throws Exception {
            try {
                for (int i = 0; i < names.length; i++) {
                    byte[] record = this.getRecordStore().getRecord(i + 1);
                    DataInputStream istream = new DataInputStream(
                            new ByteArrayInputStream(record, 0,
record.length));
                    values[i] = istream.readInt();
                    names[i] = istream.readUTF();
                }
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::loadData::" + e);
            }
        }

        protected void createDefaultData() throws Exception {
            try {
                for (int i = 0; i < names.length; i++) {
                    ByteArrayOutputStream bstream = new
ByteArrayOutputStream(12);
                    DataOutputStream ostream = new
DataOutputStream(bstream);
                    ostream.writeInt(values[i]);
                    ostream.writeUTF(names[i]);
                    ostream.flush();
```

```
                    ostream.close();
                    byte[] record = bstream.toByteArray();
                    this.getRecordStore().addRecord(record, 0, record.length);
                }
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::createDefaultData::" + e);
            }
        }

        protected void updateData() throws Exception {
            try {
                for (int i = 0; i < names.length; i++) {
                    ByteArrayOutputStream bstream = new ByteArrayOutputStream(12);
                    DataOutputStream ostream = new DataOutputStream(bstream);
                    ostream.writeInt(values[i]);
                    ostream.writeUTF(names[i]);
                    ostream.flush();
                    ostream.close();
                    byte[] record = bstream.toByteArray();
                    this.getRecordStore()
                            .setRecord(i + 1, record, 0, record.length);
                }
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::updateData::" + e);
            }
        }
    }
```

10.4.3 Settings

Settings类继承了BaseRMS类，并且扩展了抽象方法，源代码如下：

```
import javax.microedition.rms.*;
import java.util.*;
import java.io.*;

public class Settings extends BaseRMS {
    private int difficulty = 1;

    private int autoFire = 1;
```

```
    public Settings(String rmsName) {
        super(rmsName);
    }

    public void loadSettings() throws Exception {
        try {
            // Will call either loadData() or createDefaultData()
            this.open();
            // Add any other neccessary processing, in this case there is none
            // Close
            if (this.getRecordStore() != null)
                this.close();
        } catch (Exception e) {
            throw new Exception("Error loading Settings" + e);
        }
    }

    public int getDifficulty() {
        return this.difficulty;
    }

    public int getAutoFire() {
        return this.autoFire;
    }

    public void updateSettings(int difficulty, int autoFire) throws Exception {
        try {
            // load current scores
            this.open();
            // update data
            this.difficulty = difficulty;
            this.autoFire = autoFire;
            // Update High Scores
            updateData();
            // close
            if (this.getRecordStore() != null)
                this.close();
        } catch (Exception e) {
            throw new Exception(this.getRMSName() + "::updateSettings::"
+ e);
        }
    }

    protected void loadData() throws Exception {
        try {
            byte[] record = this.getRecordStore().getRecord(1);
```

```
                DataInputStream istream = new DataInputStream(
                        new ByteArrayInputStream(record, 0, record.length));
                difficulty = istream.readInt();
                autoFire = istream.readInt();
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::loadData::" + e);
            }
        }

        protected void createDefaultData() throws Exception {
            try {
                ByteArrayOutputStream bstream = new
ByteArrayOutputStream(12);
                DataOutputStream ostream = new DataOutputStream(bstream);
                ostream.writeInt(difficulty);
                ostream.writeInt(autoFire);
                ostream.flush();
                ostream.close();
                byte[] record = bstream.toByteArray();
                this.getRecordStore().addRecord(record, 0, record.length);
            } catch (Exception e) {
                throw new Exception(this.getRMSName() +
"::createDefaultData::" + e);
            }
        }

        protected void updateData() throws Exception {
            try {
                ByteArrayOutputStream bstream = new
ByteArrayOutputStream(12);
                DataOutputStream ostream = new DataOutputStream(bstream);
                ostream.writeInt(difficulty);
                ostream.writeInt(autoFire);
                ostream.flush();
                ostream.close();
                byte[] record = bstream.toByteArray();
                this.getRecordStore().setRecord(1, record, 0, record.length);
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::updateData::" +
e);
            }
        }
}
```

10.4.4 主 Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class Eliminator extends MIDlet {
    protected Display display;
    private Image splashLogo;
    private boolean isSplash = true;
    private MainMenuScreen mainMenuScreen;
    private SettingsScreen settingsScreen;
    private HighScoreScreen highScoreScreen;
    private HelpScreen helpScreen;
    private AboutScreen aboutScreen;

    private Alert alert;

    private static Score score;
    private static final String scoreRMSName = "EliminatorSettings";
    private static Settings settings;
    private static final String settingsRMSName = "EliminatorScore";
    private static final boolean CONST_DEBUG = true;

    public Eliminator() {
    }

    public void startApp() {
        display = Display.getDisplay(this);
        if (isSplash) {
            isSplash = false;
            try {

                settings = new Settings(settingsRMSName);
                settings.loadSettings();

                score = new Score(scoreRMSName);
                score.loadScores();

                mainMenuScreen = new MainMenuScreen(this);
                settingsScreen = new SettingsScreen(this, settings);
                highScoreScreen = new HighScoreScreen(this, score);
                helpScreen = new HelpScreen(this);
                aboutScreen = new AboutScreen(this);

                splashLogo = Image.createImage("/splash.png");
                new SplashScreen(display, mainMenuScreen, splashLogo,
```

```
3000);
                } catch (Exception ex) {
                    showErrorMsg(null);
                }
            } else {
                mainMenuScreenShow(null);
            }
        }

        public Display getDisplay() {
            return display;
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
            System.gc();
            notifyDestroyed();
        }

        private Image createImage(String filename) {
            Image image = null;
            try {
                image = Image.createImage(filename);
            } catch (Exception e) {
            }
            return image;
        }

        protected void mainMenuScreenShow(Alert alert) {
            if (alert == null)
                display.setCurrent(mainMenuScreen);
            else
                display.setCurrent(alert, mainMenuScreen);
        }

        protected void settingsScreenShow() {
            try {
                settingsScreen.init();
                display.setCurrent(settingsScreen);
            } catch (Exception e) {
                showErrorMsg(null);
            }
        }
```

```
    protected void highScoreScreenShow() {
        try {
            highScoreScreen.init();
            display.setCurrent(highScoreScreen);
        } catch (Exception e) {
            showErrorMsg(null);
        }
    }

    protected void helpScreenShow() {
        display.setCurrent(helpScreen);
    }

    protected void aboutScreenShow() {
        display.setCurrent(aboutScreen);
    }

    protected void mainMenuScreenQuit() {
        destroyApp(true);
    }

    protected void showErrorMsg(String alertMsg) {
        if (alertMsg == null || CONST_DEBUG == false) {
            alertMsg = "Error Starting Elminator, may or may not function
correctly. Please contact support.";
        }
        alert = new Alert("Eliminator ERROR", alertMsg, null, null);
        alert.setTimeout(Alert.FOREVER);
        alert.setType(AlertType.ERROR);
        this.mainMenuScreenShow(alert);
    }
}
```

10.4.5 Settings 界面

前面的SettingsScreen类继承了List组件，但是因为这里的Settings界面有两个不同的选项，所以继承Form组件。这个Form组件包含了实现难道选择和自动开火功能的适当的按钮和选择框。

```
import javax.microedition.lcdui.*;

public class SettingsScreen extends Form implements CommandListener {
    private Eliminator midlet;
    private Command backCommand = new Command("Back",
Command.BACK, 1);
```

```
    private Command okCommand = new Command("OK",
Command.SCREEN, 1);
    private Settings settings;
    private static final String[] difficultyOptions = { "Easy", "Medium",
            "Hard" };
    private static final String[] autoFireOptions = { "Off", "On" };
    private ChoiceGroup difficulty;
    private ChoiceGroup autoFire;

    public SettingsScreen(Eliminator midlet, Settings settings)
            throws Exception {
        super("Settings");
        this.midlet = midlet;
        this.settings = settings;
        difficulty = new ChoiceGroup("Difficulty", ChoiceGroup.POPUP,
                difficultyOptions, null);
        autoFire = new ChoiceGroup("Auto Fire", ChoiceGroup.POPUP,
                autoFireOptions, null);
        append(difficulty);
        append(autoFire);
        addCommand(backCommand);
        addCommand(okCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow(null);
        } else if (c == okCommand) {
            processMenu();
        }
    }

    public void init() throws Exception {
        settings.loadSettings();
        difficulty.setSelectedIndex(settings.getDifficulty() - 1, true);
        autoFire.setSelectedIndex(settings.getAutoFire(), true);
    }

    private void processMenu() {
        try {
            settings.updateSettings(difficulty.getSelectedIndex()      +      1,
autoFire
                    .getSelectedIndex());
            midlet.mainMenuScreenShow(null);
        } catch (Exception ex) {
```

```
                midlet.showErrorMsg("null");
            }
        }
}
```

10.4.6 高分记录显示

```
import javax.microedition.lcdui.*;

public class HighScoreScreen extends Form implements CommandListener {
    private Eliminator midlet;
    private Command backCommand = new Command("Back",
Command.BACK, 1);
    private Command resetCommand = new Command("Reset",
Command.SCREEN, 1);
    private Score score;

    StringItem stringItem;

    public HighScoreScreen(Eliminator midlet, Score score) throws
Exception {
        super("High Score");
        this.midlet = midlet;
        this.score = score;
        stringItem = new StringItem(null, "");
        append(stringItem);
        addCommand(backCommand);
        addCommand(resetCommand);
        setCommandListener(this);
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow(null);
            return;
        }
        if (c == resetCommand) {
            processMenu();
        }
    }

    public void init() throws Exception {
        score.loadScores();
        stringItem.setText(buildHighScore());
    }
```

```
    private void processMenu() {
        try {
            score.reset();
            midlet.mainMenuScreenShow(null);
        } catch (Exception ex) {
            midlet.showErrorMsg("null");
        }
    }

    private String buildHighScore() {
        String result = "";
        String[] names = score.getNames();
        int[] values = score.getValues();
        for (int i = 0; i < names.length; i++) {
            result = result + names[i] + " " + values[i] + "\n";
        }
        return result;
    }
}
```

10.5 模拟器截屏

10.5.1 Settings 显示

图31：Settings显示模拟器截屏

10.5.2 High Score 显示

图32：High Score显示模拟器截屏

第十一章 Eliminator：地形（卷动背景）

11.1 概述

终于到了开发游戏本身的时候了，让我们从游戏场景开发开始吧。

如前所述，Eliminator是一个垂直射击游戏，为了使游戏更加真实，游戏的背景要做成动态的，即滚动的。实现滚动的一种方法是随机获得背景，如星星落下的随机集合；另一种方法是游戏本身对应着整个背景。对背景的更进一步要求是玩家每一次玩游戏的时候所看到的背景应该是一样的，从而使玩家能通过背景来确定游戏进行到了什么阶段，也可以使开发者根据背景上的坐标设定特定的物品和敌人，你可以称它为地图。

在为内存限制非常严重的移动设备开发的游戏中，直接插入巨大的背景图片不是一个明智的选择。这就是通过重复利用图片Tiles生成大的背景图片的TiledLayer类出现的原因。

注意，从现在开始开发游戏本身，我们将用到第四章介绍的相关知识。

11.2 游戏场景

由于这是第一个与游戏场景处理有关的章节，我们先简要介绍一下调用游戏场景需要更改的源代码，这跟通过主函数调用其它界面没有什么特别的不同。本章最后会有完整的源代码。

11.2.1 Eliminator.java

在主midlet和Eliminator.java中，游戏场景对象被添加：

```
private GameScreen gameScreen;
```

上述方法被用来调用一个新的游戏，一个新的游戏场景可以通过midlet本身、*设置settings*对象和*高分记录score*对象等调用。场景被调用的同时就开始执行，通过start()函数。

```
protected void gameScreenShow() {
    try {
        gameScreen = null;
        gameScreen = new GameScreen(this,settings,score);
        gameScreen.start();
        display.setCurrent(gameScreen);
```

```
    }
    catch (Exception e) {
        System.out.println(e);
        showErrorMsg(null);
    }
}
```

11.2.2 MainMenuScreen.java

可以在主midlet中调用gameScreenShow方法：

```
private void scnNewGame() {
    midlet.gameScreenShow();
}
```

11.2.3 GameScreen.java

GameScreen是由基本的游戏循环和游戏组件所组成，它继承自GameCanvas类，有关GameCanvas类的详细介绍请参考第四章。它扩展了Runnable接口来提供主游戏循环的引用，扩展了CommandListener来提供软键的反馈。

游戏主循环类似于：

```
//Main Game Loop
public void run() {
    Graphics g = getGraphics();
    Thread currentThread = Thread.currentThread();
    try {
            while (currentThread == gameThread) {
                long startTime = System.currentTimeMillis();
                if (isShown()) {
                    if (isPlay) {
                        tick();
                    }
                    render(g);
                }
                long timeTake = System.currentTimeMillis() - startTime;
                if (timeTake < MILLIS_PER_TICK) {
                    synchronized (this) {
                        wait(MILLIS_PER_TICK - timeTake);
                    }
                }
                else {
                    currentThread.yield();
```

```
                    }
                }
        }
        catch (InterruptedException ex) {
            // won't be thrown
        }
}
```

用来绘制图像的代码如下：

```
//Method to Display Graphics
private void render(Graphics g) {
        // Set Background color to beige
        g.setColor(0xF8DDBE);
        g.fillRect(0,0,width,height);
        g.setColor(0x0000ff);
        layerManager.paint(g,0,0);
        flushGraphics();
}
```

实现图像更新，如移动和滚动的代码区：

```
// Handle dynamic changes to game including user input
public void tick() {
// update code goes here
}
```

虽然上面的代码都是组成游戏的基本片断，但是在没有动画、绘制或者执行的情况下，的确没有什么内容好介绍的。下一节将介绍滚动背景的游戏场景。

11.3 一个简单 Terrain(地形)

11.3.1 概述

有关TiledLayer类的使用我们第四章都讲过了，滚动是通过调整TiledLayer图像的相对位置实现的；具有这个功能的函数是setPosition。

11.3.2 构造函数

在构造函数中调用initTerrain方法讲地形添加到LayerManager中：

```
initTerrain();
```

layerManager = new LayerManager();
layerManager.append(terrain);

11.3.3 初始化 Terrain(地形)

TiledLayer类被initTerrain方法加载,用来管理地形,并通过tile的高度、tile的数量和可视屏幕的高度来设置TiledLayer的初始位置。

```
//Init Terrain and set Terrain at the bottom of the map
private void initTerrain() throws Exception {
    try {
        terrain = loadTerrain();
        terrainScroll = 1 - (TILE_HEIGHT * terrain.getRows()) + scnViewHeight;
        terrain.setPosition(0,terrainScroll);
    } catch (Exception e) {
        throw new Exception("GameScreen::initTerrain::" + e);
    }
}
```

11.3.4 加载 Terrain(地形)

上一节中的方法没有真正实现图像的TiledLayer映射，而是在loadTerrain()方法中实现，将它调整到不同的任务中可以使方法更加清晰。在下一节的多样的地形处理部分将图解这种调整。

```
private TiledLayer loadTerrain() throws Exception {

    Image tileImages = Image.createImage("/terrain.png");
    TiledLayer tiledLayer = new TiledLayer( TILE_NUM_COL , TILE_NUM_ROW , tileImages , TILE_WIDTH , TILE_HEIGHT);

//    Define Terrain Map
    int[][] map = {
    {0,0,0,0,0,0},
    {3,0,0,0,0,0},
    {6,0,0,0,0,0},
    {6,0,0,0,1,2},
    {6,0,0,0,4,5},
    {6,0,0,0,7,8},
    {6,0,0,0,0,0},
    {9,0,1,2,3,0},
    {0,0,4,5,6,0},
    {0,0,7,8,9,0},
```

```
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {3,0,0,0,0,0},
        {6,0,0,0,0,0},
        {6,0,0,0,1,2},
        {6,0,0,0,4,5},
        {6,0,0,0,7,8},
        {6,0,0,0,0,0},
        {9,0,1,2,3,0},
        {0,0,4,5,6,0},
        {0,0,7,8,9,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {3,0,0,0,0,0},
        {6,0,0,0,0,0},
        {6,0,0,0,1,2},
        {6,0,0,0,4,5},
        {6,0,0,0,7,8},
        {6,0,0,0,0,0},
        {9,0,0,0,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {0,0,0,0,0,0},
        {3,0,0,0,0,1}
        };
// Map Terrain Map with actual graphic from terrain.png
        for (int row=0; row<TILE_NUM_ROW; row++) {
            for (int col=0; col<TILE_NUM_COL; col++) {
                tiledLayer.setCell(col,row,map[row][col]);
            }
        }
        return tiledLayer;
}
```

11.3.5 滚动 Terrain(地形)

就像前面提到的那样，对TiledLayer图像的滚动没有什么特殊的地方，无非就是在显示图像的时候通过选择不同的坐标显示图像不同的部分。这里，每一次游戏循环都在y方向增加两个象素；因为一次增加一个象素的话滚动得太慢了。由于scrollTerrain是对图像的修改，所有是通过tick()方法调用的，这不同于第四章滚动背景的技术。在第四章中，通过LayerManager类实现背景的滚动，这样做的缺点是与LayerManager有关的内容也会受到LayerManager类的影响。为了避免像Sprite类受到背景滚动的影响,这里推荐使用调节TiledLayer本身的位置实现背景的滚动。

```
// Scroll Terrain
private void scrollTerrain() {
    if (terrainScroll < 0) {
        terrainScroll += 2;
        terrain.setPosition(0,terrainScroll);
    }
}
```

11.3.6 Terrain（地形）图片

Terrain（地形）图片由9个tile（贴砖）组成，32象素×32象素大小：

图33：定义好的地形片断

图34：地形

11.3.7 源代码

GameScreen.java:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class GameScreen extends GameCanvas implements Runnable,
CommandListener {
    private static final int MILLIS_PER_TICK = 50;
    private static final int TILE_WIDTH = 32;
    private static final int TILE_HEIGHT = 32;
    private static final int TILE_NUM_COL = 6;
```

```
    private static final int TILE_NUM_ROW = 36;

    private Eliminator midlet; // Hold the Main Midlet
    private Settings settings; // Hold Game Settings
    private Score score; // Hold Game Score

    private Command backCommand = new Command("Back",
Command.BACK, 1);

    private boolean isPlay; // Game Loop runs when isPlay is true
    private int width; // To hold screen width
    private int height; // To hold screen height
    private int scnViewWidth; // Hold Width Screen View Port
    private int scnViewHeight; // Hold Height Screen View Port

    private Thread gameThread = null;

    // Layer Manager to manager background (terrain)
    private LayerManager layerManager;

    // TiledLayer - Terrain
    private TiledLayer terrain;

    private int terrainScroll; // Hold Y position for scrolling

    // Constructor and initialization

    public GameScreen(Eliminator midlet, Settings settings, Score score)
            throws Exception {
        super(true);
        this.midlet = midlet;
        addCommand(backCommand);
        setCommandListener(this);
        width = getWidth(); // get screen width
        height = getHeight(); // get screen height
        scnViewWidth = width; // Set View Port width to screen width
        scnViewHeight = height; // Set View Port height to screen height
        isPlay = true;
        initTerrain();
        layerManager = new LayerManager();
        layerManager.append(terrain);
    }

    // Start thread for game loop
    public void start() {
        gameThread = new Thread(this);
```

```
        gameThread.start();
    }

    // Stop thread for game loop
    public void stop() {
        gameThread = null;
    }

    // Main Game Loop
    public void run() {
        Graphics g = getGraphics();
        Thread currentThread = Thread.currentThread();
        try {
            while (currentThread == gameThread) {
                long startTime = System.currentTimeMillis();
                if (isShown()) {
                    if (isPlay) {
                        tick();
                    }
                    render(g);
                }
                long timeTake = System.currentTimeMillis() - startTime;
                if (timeTake < MILLIS_PER_TICK) {
                    synchronized (this) {
                        wait(MILLIS_PER_TICK - timeTake);
                    }
                } else {
                    currentThread.yield();
                }
            }
        } catch (InterruptedException ex) {
            // won't be thrown
        }
    }

    // Handle dynamic changes to game including user input
    public void tick() {
        scrollTerrain();
    }

    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.mainMenuScreenShow(null);
        }
    }
```

```
    // Method to Display Graphics
    private void render(Graphics g) {
        // Set Background color to beige
        g.setColor(0xF8DDBE);
        g.fillRect(0, 0, width, height);
        g.setColor(0x0000ff);
        // LayerManager Paint Graphics
        layerManager.paint(g, 0, 0);
        flushGraphics();
    }

    private TiledLayer loadTerrain() throws Exception {
        Image tileImages = Image.createImage("/terrain.png");
        TiledLayer tiledLayer = new TiledLayer(TILE_NUM_COL,
TILE_NUM_ROW,
                tileImages, TILE_WIDTH, TILE_HEIGHT);
        // Define Terrain Map
        int[][] map = { { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 0 },
                { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
                { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 },
                { 6, 0, 0, 0, 0, 0 }, { 9, 0, 1, 2, 3, 0 },
                { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 0 },
                { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
                { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 },
                { 6, 0, 0, 0, 0, 0 }, { 9, 0, 1, 2, 3, 0 },
                { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 0 },
                { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
                { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 },
                { 6, 0, 0, 0, 0, 0 }, { 9, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
                { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 1 } };
        // Map Terrain Map with actual graphic from terrain.png
        for (int row = 0; row < TILE_NUM_ROW; row++) {
            for (int col = 0; col < TILE_NUM_COL; col++) {
                tiledLayer.setCell(col, row, map[row][col]);
            }
        }
        return tiledLayer;
    } // Scroll Terrain

    private void scrollTerrain() {
        if (terrainScroll < 0) {
```

```
                terrainScroll += 2;
                terrain.setPosition(0, terrainScroll);
            }
        }

        // Init Terrain and set Terrain at the bottom of the map
        private void initTerrain() throws Exception {
            try {
                terrain = loadTerrain();
                terrainScroll = 1 - (TILE_HEIGHT * terrain.getRows())
                        + scnViewHeight;
                terrain.setPosition(0, terrainScroll);
            } catch (Exception e) {
                throw new Exception("GameScreen::initTerrain::" + e);
            }
        }
}
```

11.4 多重地形的地图类

11.4.1 概述

特别是对垂直射击游戏，只有一关是不切实际的。所以在这一节中，通过少量的修改，我们将有能力很容易地更改游戏中的地图。

简单起见，一共就有两种地形，第一种是“一个简单Terrain(地形)”那一节用到过的地形，第二种是背景是草的绿色的山冈。

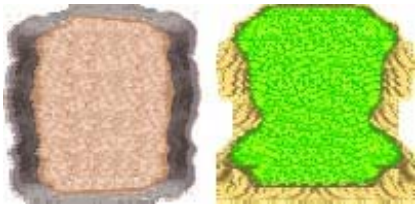

图35：多重地形

下面我们将与terrain设置相关的功能析取到一个独立的类中——GameMap，用来处理：

- Terrain选择

- Terrain滚动
- Terrain初始化
- Terrain背景/地板色

这个类可以作为一个组件重新用在其它垂直滚动游戏中，也可以经过很少的改动用在水平滚动游戏中。

11.4.2 构造函数、初始化和加载地形（Terrain）

通过调用GameMap的构造函数，地形被自动加载和初始化一个获得当前地形的方法调用。

```
gameMap = new GameMap(scnViewHeight);
terrain = gameMap.getTerrain();
```

11.4.3 滚动地形

GameMap类中的scrollTerrain方法用来实现滚动地形的功能：

```
gameMap.scrollTerrain();
```

可以通过调用GameScreen类的getter方法获得应用滚动之前最近的地形：

```
terrain = gameMap.getTerrain();
```

11.4.4 源代码

GameMap.java:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class GameMap {
    // Terrain 1
    private static final int[][] map1 = { { 0, 0, 0, 0, 0, 0 },
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 1, 2, 3, 0 }, { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 1, 2, 3, 0 }, { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
```

```
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 1 } };

    // Terrain 2
    private static final int[][] map2 = { { 0, 0, 0, 0, 0, 0 },
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 1, 2, 3, 0 }, { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 1, 2, 3, 0 }, { 0, 0, 4, 5, 6, 0 }, { 0, 0, 7, 8, 9, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
            { 3, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 0, 0 }, { 6, 0, 0, 0, 1, 2 },
            { 6, 0, 0, 0, 4, 5 }, { 6, 0, 0, 0, 7, 8 }, { 6, 0, 0, 0, 0, 0 },
            { 9, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 }, { 0, 0, 0, 0, 0, 0 },
            { 0, 0, 0, 0, 0, 0 }, { 3, 0, 0, 0, 0, 1 } };

    // Set Constant values, the values are direct
    // relation to the actually tile sizes
    // defined for the terrain graphics
    private static final int TILE_WIDTH = 32;

    private static final int TILE_HEIGHT = 32;

    private static final int TILE_NUM_COL = 6;

    private static final int TILE_NUM_ROW = 36;

    // To hold the current map
    private int[][] currentMap;

    // To hold the current terrain
    private TiledLayer terrain;

    // To hold the current background/floor color
    private int groundColor;

    // To hold the current screen, value neeeded for scrolling calculation
    private int screenHeight;

    // To hold Y position for scrolling
    private int terrainScroll;
```

```
    public GameMap(int screenHeight) throws Exception {
        this.screenHeight = screenHeight;
        setMap(1); // default to set to terrain 1
    }

    // Set Appropriate Terrain and Map
    public void setMap(int level) throws Exception {
        Image tileImages = null;
        switch (level) {
        case 1:
            tileImages = Image.createImage("/terrain1.png");
            currentMap = map1;
            groundColor = 0xF8DDBE;
            break;
        case 2:
            tileImages = Image.createImage("/terrain2.png");
            currentMap = map2;
            groundColor = 0xDECE6B;
            break;
        default:
            tileImages = Image.createImage("/terrain1.png");
            currentMap = map1;
            groundColor = 0xF8DDBE;
            break;
        }
        terrain = new TiledLayer(TILE_NUM_COL, TILE_NUM_ROW,
tileImages,
                TILE_WIDTH, TILE_HEIGHT);
        // Map Terrain Map with actual graphic from terrain.png
        for (int row = 0; row < TILE_NUM_ROW; row++) {
            for (int col = 0; col < TILE_NUM_COL; col++) {
                terrain.setCell(col, row, currentMap[row][col]);
            }
        }
        terrainScroll = 1 - (terrain.getCellHeight() * terrain.getRows())
                + screenHeight;
        terrain.setPosition(0, terrainScroll);
    }

    public void scrollTerrain() {
        if (terrainScroll < 0) {
            terrainScroll += 2;
            terrain.setPosition(0, terrainScroll);
        }
    }
```

```
    // Terrain Getter
    public TiledLayer getTerrain() {
        return terrain;
    }

    // Ground/Floor color Getter
    public int getGroundColor() {
        return groundColor;
    }
}
```

GameScreen.java:

GameScreen类因为GameMap类的使用而简化：

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class GameScreen extends GameCanvas implements Runnable,
CommandListener {
    private static final int MILLIS_PER_TICK = 50;
    private Eliminator midlet; // Hold the Main Midlet
    private Settings settings; // Hold Game Settings
    private Score score; // Hold Game Score

    private Command backCommand = new Command("Back",
Command.BACK, 1);
    private GameMap gameMap;

    private boolean isPlay; // Game Loop runs when isPlay is true
    private int width; // To hold screen width
    private int height; // To hold screen height
    private int scnViewWidth; // Hold Width Screen View Port
    private int scnViewHeight; // Hold Height Screen View Port

    private Thread gameThread = null;

    // Layer Manager to manager background (terrain)
    private LayerManager layerManager;

    // TiledLayer - Terrain
    private TiledLayer terrain;

    private int terrainScroll; // Hold Y position for scrolling
    // Constructor and initialization
```

```
public GameScreen(Eliminator midlet, Settings settings, Score score)
        throws Exception {
    super(true);
    this.midlet = midlet;
    addCommand(backCommand);
    setCommandListener(this);
    width = getWidth(); // get screen width
    height = getHeight(); // get screen height
    scnViewWidth = width; // Set View Port width to screen width
    scnViewHeight = height; // Set View Port height to screen height
    isPlay = true;
    gameMap = new GameMap(scnViewHeight);
    terrain = gameMap.getTerrain();
    layerManager = new LayerManager();
    layerManager.append(terrain);
}

// Start thread for game loop
public void start() {
    gameThread = new Thread(this);
    gameThread.start();
}

// Stop thread for game loop
public void stop() {
    gameThread = null;
}

// Main Game Loop
public void run() {
    Graphics g = getGraphics();
    Thread currentThread = Thread.currentThread();
    try {
        while (currentThread == gameThread) {
            long startTime = System.currentTimeMillis();
            if (isShown()) {
                if (isPlay) {
                    tick();
                }
                render(g);
            }
            long timeTake = System.currentTimeMillis() - startTime;
            if (timeTake < MILLIS_PER_TICK) {
                synchronized (this) {
                    wait(MILLIS_PER_TICK - timeTake);
                }
```

```
                    } else {
                        currentThread.yield();
                    }
                }
            } catch (InterruptedException ex) {
                // won't be thrown
            }
        }

        // Handle dynamic changes to game including user input
        public void tick() {
            // Scroll Terrain
            gameMap.scrollTerrain();
        }

        public void commandAction(Command c, Displayable d) {
            if (c == backCommand) {
                midlet.mainMenuScreenShow(null);
            }
        }

        // Method to Display Graphics
        private void render(Graphics g) {
            // Set Background color to beige
            //g.setColor(0xF8DDBE);
            g.setColor(gameMap.getGroundColor());
            g.fillRect(0, 0, width, height);
            g.setColor(0x0000ff);
            // Get Current Map
            terrain = gameMap.getTerrain();
            // LayerManager Paint Graphics
            layerManager.paint(g, 0, 0);
            flushGraphics();
        }
    }
```

第十二章 Eliminator：玩家和子弹

12.1 概述

12.1.1 玩家

现在来添加游戏中被用户控制的最主要的精灵——玩家（Player）。玩家会有几个特征，如剩余生命数、能量条、剩余炸弹数，还有各种不同的武器。玩家特征有可能会变得非常复杂，特别是在角色扮演类游戏中，会有许许多多的特征，包括智力、敏捷、力量、护甲的持久力等等。所以，通过继承MIDP 2中的Sprite类来创建一个新的精灵类非常有必要。

PlayerSprite类非常直接的包含了大多数Sprite类中存在的接口，只有一个新加的方法——控制发射子弹的fire方法。

```
public Sprite fire(Image bullets) {
    Sprite bullet = new Sprite(bullets,2,2);
    bullet.setFrame(0);
    getXY();
    bullet.setPosition(x - bullet.getWidth() / 2 + this.getWidth() / 2, y);
    return bullet;
}
```

这个方法用来获得玩家当前的位置,并且在相关的纵坐标上创建一个子弹精灵。在GameScreen.java中，这种应用会更加明显。

12.1.2 子弹

由于子弹不需要额外的功能，所以用默认的Sprite类就足够了，但是在大多数游戏中，屏幕中的子弹并不仅仅是一枚，需要使用引用（Vector）控制多个子弹。

为了降低资源的使用，每个射击的精灵包括玩家精灵都共享相同的引用，要解决的问题是如何很容易的决定子弹和发射子弹的精灵的对应关系。一般通过使用多个帧（Frame）来实现（参考下一节），第一个帧中用绿色的子弹来表示玩家发射的子弹，下一个帧中用红色的子弹表示敌人的子弹。

这种方法非常灵活，可以很容易的添加额外的子弹类型，如果需要的话甚至可以创建包含通过确定的子弹集合组成的动画的帧。

12.2 图像

12.2.1 玩家

下面是玩家图像文件：player.png

图36：玩家精灵的图表

玩家活动的时候只需要一个帧来表示；剩下的三个帧表示玩家被子弹击中后的情况。为了使图像更为平滑，也给用户更好的反馈，可以添加两个新的帧，分别表示向左转、向右转。

12.2.2 子弹

下面是子弹文件bullets.png，这个图表被放大了，它的原始尺寸是4×2象素，包含两种类型大小都是2×2的子弹。

图37：子弹精灵（放大）

12.3 源代码

12.3.1 玩家精灵

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

/**
 * Main Sprite / Player Inherits the MIDP 2.0 Sprite class
 */
public class PlayerSprite extends Sprite {
    private static final int MOVE = 3;

    private int x, y;

    private int scnWidth, scnHeight;
```

```
    private int frameWidth, frameHeight;

    private int frame;

    private int lives;

    public PlayerSprite(Image image, int frameWidth, int frameHeight,
            int scnWidth, int scnHeight) throws Exception {
        super(image, frameWidth, frameHeight);
        x = frameWidth / 2;
        y = frameHeight / 2;
        this.scnWidth = scnWidth;
        this.scnHeight = scnHeight;
        this.frameWidth = frameWidth;
        this.frameHeight = frameHeight;
        this.frame = 1;
        this.lives = 3;
    }

    public void startPosition() {
        setPosition(scnWidth / 2, scnHeight / 2);
    }

    public void moveLeft() {
        getXY();
        if (x - MOVE > 0)
            move(MOVE * -1, 0);
    }

    public void moveRight() {
        getXY();
        if (x + MOVE + frameWidth < scnWidth)
            move(MOVE, 0);
    }

    public void moveUp() {
        getXY();
        if (y - MOVE > 0)
            move(0, MOVE * -1);
    }

    public void moveDown() {
        getXY();
        if (y + MOVE + frameHeight < scnHeight)
            move(0, MOVE);
    }
```

```
    public Sprite fire(Image bullets) {
        Sprite bullet = new Sprite(bullets, 2, 2);
        bullet.setFrame(0);
        getXY();
        bullet.setPosition(x - bullet.getWidth() / 2 + this.getWidth() / 2, y);
        return bullet;
    }

    public void display(Graphics g) {
        this.setFrame(frame);
        this.paint(g);
    }

    public int getLives() {
        return lives;
    }

    public void setLives(int lives) {
        this.lives = lives;
    }

    private void getXY() {
        x = getX();
        y = getY();
    }
}
```

12.3.2 子弹精灵

子弹精灵像其它精灵一样初始化；但是却不是被加到LayerManager类的构造函数中，而是在玩家或其它精灵射击的时候动态添加的。

```
// Player Fires
if ((keyStates & FIRE_PRESSED) != 0) {
    Sprite bullet = player.fire(bulletImages);
    if (bullet != null) {
        bullets.addElement(bullet);
        layerManager.insert(bullet,1);
    }
}
```

我们可以通过引用逻辑的移除子弹，也可以通过LayerManager在子弹到达屏幕边缘时移除，还可以通过tick()方法来更新子弹的移动。处理在任意坐标的子弹移动需要额外的逻辑，当然当子弹击中有效的精灵后需要移除。

```
//Update Bullet(s) Movement
for (int i = 0; i < bullets.size(); ++i) {
    for (int j = 0; j < 2; ++j) {
        Sprite bullet = (Sprite)(bullets.elementAt(i));
        bullet.move(0, -1);
        if (bullet.getY() < 0) {
            bullets.removeElementAt(i);
            layerManager.remove(bullet);
            i--;
            break;
        }
    }
}
```

12.3.3 游戏屏幕

将上一节介绍的子弹的相关规则逻辑结合到一起，玩家精灵的相关特征需要是自解释的，成批的玩家代码请参考tick()方法。

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import java.util.*;

public class GameScreen extends GameCanvas implements Runnable,
CommandListener {
    private static final int MILLIS_PER_TICK = 50;

    private Eliminator midlet; // Hold the Main Midlet

    private Settings settings; // Hold Game Settings

    private Score score; // Hold Game Score

    private Command backCommand = new Command("Back",
Command.BACK, 1);

    private GameMap gameMap;

    private boolean isPlay; // Game Loop runs when isPlay is true

    private int width; // To hold screen width

    private int height; // To hold screen height
```

```
    private int scnViewWidth; // Hold Width Screen View Port

    private int scnViewHeight; // Hold Height Screen View Port

    private Thread gameThread = null;

    // Layer Manager to manager background (terrain)
    private LayerManager layerManager;

    // TiledLayer - Terrain
    private TiledLayer terrain;

    private int terrainScroll; // Hold Y position for scrolling
    // Sprites

    private PlayerSprite player;

    // Variables to hold bullet info
    private Vector bullets;

    private Image bulletImages;

    // Constructor and initialization
    public GameScreen(Eliminator midlet, Settings settings, Score score)
            throws Exception {
        super(true);
        this.midlet = midlet;
        addCommand(backCommand);
        setCommandListener(this);
        width = getWidth(); // get screen width
        height = getHeight(); // get screen height
        scnViewWidth = width; // Set View Port width to screen width
        scnViewHeight = height; // Set View Port height to screen height
        isPlay = true;
        // setup map
        gameMap = new GameMap(scnViewHeight);
        terrain = gameMap.getTerrain();
        // setup player sprite
        Image image = Image.createImage("/player.png");
        player = new PlayerSprite(image, 24, 18, width, height); // 24 =
width
                                                                 //    of
sprite in
                                                                 //
pixels, 18
                                                                 //    is
```

```
height of
                                                                    //
sprite in
                                                                    //
pixels
            player.startPosition();
            // init bullets
            bullets = new Vector();
            bulletImages = midlet.createImage("/bullets.png");
            layerManager = new LayerManager();
            layerManager.append(player);
            layerManager.append(terrain);
      }

      // Start thread for game loop
      public void start() {
            gameThread = new Thread(this);
            gameThread.start();
      }

      // Stop thread for game loop
      public void stop() {
            gameThread = null;
      }

      // Main Game Loop
      public void run() {
            Graphics g = getGraphics();
            Thread currentThread = Thread.currentThread();
            try {
                  while (currentThread == gameThread) {
                        long startTime = System.currentTimeMillis();
                        if (isShown()) {
                              if (isPlay) {
                                    tick();
                              }
                              render(g);
                        }
                        long timeTake = System.currentTimeMillis() - startTime;
                        if (timeTake < MILLIS_PER_TICK) {
                              synchronized (this) {
                                    wait(MILLIS_PER_TICK - timeTake);
                              }
                        } else {
                              currentThread.yield();
                        }
```

```
            }
        } catch (InterruptedException ex) {
            // won't be thrown
        }
    }

    // Handle dynamic changes to game including user input
    public void tick() {
        // Scroll Terrain
        gameMap.scrollTerrain();
        // Player Actions
        int keyStates = getKeyStates();
        // Player Moves
        if ((keyStates & LEFT_PRESSED) != 0) {
            player.moveLeft();
        } else if ((keyStates & RIGHT_PRESSED) != 0) {
            player.moveRight();
        } else if ((keyStates & UP_PRESSED) != 0) {
            player.moveUp();
        } else if ((keyStates & DOWN_PRESSED) != 0) {
            player.moveDown();
        }
        // Player Fires
        if ((keyStates & FIRE_PRESSED) != 0) {
            Sprite bullet = player.fire(bulletImages);
            if (bullet != null) {
                bullets.addElement(bullet);
                layerManager.insert(bullet, 1);
            }
        }
        // Update Bullet(s) Movement
        for (int i = 0; i < bullets.size(); ++i) {
            for (int j = 0; j < 2; ++j) {
                Sprite bullet = (Sprite) (bullets.elementAt(i));
                bullet.move(0, -1);
                if (bullet.getY() < 0) {
                    bullets.removeElementAt(i);
                    layerManager.remove(bullet);
                    i--;
                    break;
                }
            }
        }
    }

    public void commandAction(Command c, Displayable d) {
```

```
            if (c == backCommand) {
                midlet.mainMenuScreenShow(null);
            }
        }

        // Method to Display Graphics
        private void render(Graphics g) {
            // Set Background color to beige
            //g.setColor(0xF8DDBE);
            g.setColor(gameMap.getGroundColor());
            g.fillRect(0, 0, width, height);
            g.setColor(0x0000ff);
            // Get Current Map
            terrain = gameMap.getTerrain();
            // LayerManager Paint Graphics
            layerManager.paint(g, 0, 0);
            flushGraphics();
        }
}
```

第十三章 Eliminator:场景切换

13.1 代码整理和精简

在游戏的进一步开发之前，是时候进行代码整理了。前面提到了，移动开发中，在优秀的规划和设计模式之前，需要使代码更有效率，至少使代码占用更少的空间。

首先要精简的是BaseRMS类,不通过继承一个基类而使用独立类会有一些代码上的重复，这里会将两个类合而为一。这项工作由读者自己完成，切记这样做会增加代码维护的难度。作者认为分开这些类有它的好处，因为一些游戏也许不需要*高分记录High Score*或*设置Settings*，或者*高分记录High Score*已经在服务器上实现了，而不需要存在于客户端上。独立的类可以很方便的重新利用和组装。

源代码文件见Settings.java和Score.java（第14章）。

接着要精简的是移除MainMenuScreen类并将菜单处理移到主midlet中，这会提高替换用户化的canvas菜单的难度，参考第14章的主midlet类：Eliminator.java。

第三个精简的部分是合并所有高级GUI界面为一个类，每一个屏显都对应一个switch，参考Gui.java。从而移除了独立的屏显类如*关于About*、*高分记录High Score*、*设置Setting*和*帮助Help*。

之所以进行上面的合并，是为了释放一些空间来满足游戏本身的各种附加类的需要。

此外，注意所有文字输出都使用字符串常量，如下主菜单代码片：

mainMenu.append(Utils.getStr(Utils.ID_NEW),null);

这里没有使用“New”，而是“Utils.ID_NEW”，字符串都被存储在Utils类中。Utils类可以处理本地支持，从而可以很容易地支持其它语言。更多相关内容请参考第17章Unicode和世界范围的语言支持。

13.2 场景切换

GameMap中定义了四个级别的场景，而且Leeman Cheng 和 Carlo Casimiro 制作了更为优美的图形图像。

下面是游戏中用到的新图片：

图38：游戏中的图像

第十四章 Eliminator:游戏内容

14.1 敌人

与玩家不同，敌人精灵可以在各个方向移动，也可以转向各个方向；而玩家只能从一边移动到另一边，而且总是面朝前的。由于没有兴趣重新发明轮形图(wheel)，这里将借用Jonathan Knudsen的介绍使用Game API创建2D动作游戏的动作的文章中的代码，Jonathan Knudsen在Sun公司的主页是 http://developers.sun.com/techtopics/mobility/midp/articles/game/。文章中的代码论证了图像变换和镜像的使用，强烈建议读者在继续下面的内容前先阅读他的这篇文章。

下面介绍几个不同类型的NPC：

```
public static final int NPC_CANON1 = 1;
public static final int NPC_CANON2 = 2;
public static final int NPC_CANON3 = 3;
public static final int NPC_FLYER1 = 4;
public static final int NPC_FLYER2 = 5;
public static final int NPC_FLYER3 = 6;
public static final int NPC_FLYER4 = 7;
public static final int NPC_TANK1 = 8;
public static final int NPC_TANK2 = 9;
int npcType;
```

因为所有的NPC都在一个图形表中定义，这里需要做一些调整：

```
// Hack 1:1 relation to the PNG file 3 frames of cannons first then a flyer... as
we add
//more NPCs this switch needs to be updated accordingly
switch(npcType) {
    case NPC_CANON1: this.frameOffset = 0; this.setFrame(0); break;
    case NPC_CANON2: this.frameOffset = 3; this.setFrame(3); break;
    case NPC_CANON3: this.frameOffset = 6; this.setFrame(6); break;
    case NPC_FLYER1: this.frameOffset = 9; this.setFrame(9); break;
    case NPC_FLYER2: this.frameOffset = 12; this.setFrame(12); break;
    case NPC_FLYER3: this.frameOffset = 15; this.setFrame(15); break;
    case NPC_FLYER4: this.frameOffset = 18; this.setFrame(18); break;
    case NPC_TANK1: this.frameOffset = 21; this.setFrame(21); break;
    case NPC_TANK2: this.frameOffset = 24; this.setFrame(24); break;
}
```

在NPC中加入了与主玩家的射击函数类似的射击函数，它有两个另外的约束。一个是与玩家精灵的子弹总是向正上方射击不同，我们需要敌人精灵决定子弹的射击

方向；第二个，我们需要控制子弹的数量，可以通过增加延迟，和调整调用射击方法之间的时间间隔来实现。当然可以增加更多的约束来使游戏更加现实和合理。还有一个需要考虑的是，在敌人精灵开火之前，我们要确定它是否面对着玩家精灵，如果不是，应该控制敌人精灵使它面向玩家精灵开火。

下述方法用来决定NPC所在的象限，并且决定它一个移动/转向的方向。

```
//Major hack and magic numbers here, sorry I'll try and make this more
// readable later
private int getPlayerDirection(PlayerSprite player) {
    // Beside the NPC
    if (player.getMidY() <= (getMidY() + SPC)
            && player.getMidY() >= (getMidY() - SPC)) {
        if (player.getX() < getX())
            return 12;
        else
            return 4;
    } else if (player.getMidX() >= getMidX() - SPC
            && player.getMidX() <= getMidX() + SPC) {
        if (player.getY() > getY())
            return 8;
        else
            return 0;
        // Quad 4
    } else if (player.getMidX() <= getMidX()
            && player.getMidY() <= getMidY()) {
        int qC = quadCalc(player);
        if (qC <= SPC)
            return 14;
        else if (Math.abs(player.getMidX() - getMidX()) > Math.abs(player
                .getY()
                - getMidY()))
            return 13;
        else
            return 15;
        // Quad 3
    } else if (player.getMidX() <= getMidX()
            && player.getMidY() >= getMidY()) {
        int qC = quadCalc(player);
        if (qC <= SPC)
            return 10;
        else if (Math.abs(player.getMidX() - getMidX()) > Math.abs(player
                .getY()
                - getMidY()))
            return 11;
```

```
            else
                return 9;
            // Quad 2
        } else if (player.getMidX() >= getMidX()
                && player.getMidY() >= getMidY()) {
            int qC = quadCalc(player);
            if (qC <= SPC)
                return 6;
            else if (Math.abs(player.getMidX() - getMidX()) > Math.abs(player
                    .getY()
                    - getMidY()))
                return 5;
            else
                return 7;
            // Quad 1
        } else if (player.getMidX() >= getMidX()
                && player.getMidY() <= getMidY()) {
            int qC = quadCalc(player);
            if (qC <= 5)
                return 2;
            else if (Math.abs(player.getMidX() - getMidX()) > Math.abs(player
                    .getY()
                    - getMidY()))
                return 3;
            else
                return 1;
        }
        // Should never reach here
        return -1;
    }
    private int quadCalc(PlayerSprite player) {
        return Math.abs((Math.abs(player.getMidX() - getMidX()))
                - (Math.abs(player.getY() - getMidY())));
    }
```

14.2 能力提升

第十五章 Eliminator：老怪（Boss）

第十六章 Eliminator:游戏其它

16.1 声音和振动

虽然Eliminator没有任何声音和振动，但是这并不意味着这些方面不重要。任何能提高用户体验的游戏特征都是好的！同时，切记声音和振动同时也会带来更多的资源开销，如内存和电池。同时，你也要意识到一般在玩移动游戏的时候，声音也许仅仅演变为噪音或骚扰，例如，在医院等待就诊；在图书馆时；在公共汽车上；上枯燥无味的课时……所有这些时候，最好都有个选择来打开或者关闭声音。

还需要考虑扩展声音和振动支持的特定移动设备的参数。在MIDP 2.0中有了对声音的支持，参见javax.microedition.media包。

16.2 二进制数据

将数据、地图、图像等等以二进制格式存储。

16.3 数据下载/更新

- 等级
- 图片
- …等

第三篇　更 多 技 术

- 提高可用性
- 加入时间约束
- 特征化界面
- …其它章节

第十七章 提高可用性

17.1 暂停、返回和保存

在游戏中增加暂停功能是非常必要的，因为用户在玩游戏的过程中经常会被来电、短信、彩信或者其它什么所干扰。有些时候，用户需要直接退出游戏，例如汽车到站需要找零钱的时候。也就是说，需要一个选项来暂停和返回游戏。如果用户需要退出游戏，最好可以自动保存游戏当前状态，这样用户就可以从他们上次玩到的地方开始。

当用户被像来电这样的情况所打断时，hideNotify()方法被调用，在这个方法中你可以调用暂停逻辑；同样，showNotify()调用适当的逻辑来继续游戏。首先，在游戏中被影响的两个东西是标为暂停和返回的软键或用户键，此外，主游戏循环/指针本身也是受影响的区域。当然，用户需要要被允许通过按暂停和返回键来暂停正在进行的游戏或返回被暂停的游戏。

下面的代码内容是暂停的简单实现和自动保存的简便方法。

17.1.1 使用 MIDP 2.0 实现暂停和返回

ExamplePauseResumeMIDP2.java (main midlet):

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExamplePauseResumeMIDP2 extends MIDlet {
    private Display display;

    private GameScreen gameScreen;

    public ExamplePauseResumeMIDP2() {
        display = Display.getDisplay(this);
    }

    public void startApp() {
        try {
            this.gameScreen = new GameScreen(this);
            this.gameScreen.start();
            display.setCurrent(this.gameScreen);
        } catch (Exception ex) {
            System.out.println("error: " + ex);
        }
    }
```

```
    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        gameScreen.autoSave();
    }

    public void exitGame() {
        destroyApp(false);
        notifyDestroyed();
    }
}
```

GameScreen.java:

```
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;
import java.util.*;

public class GameScreen extends GameCanvas implements Runnable,
CommandListener {
    private static final int MILLIS_PER_TICK = 50;

    private Command exitCommand = new Command("Exit",
Command.BACK, 1);

    private Command pauseCommand = new Command("Pause",
Command.SCREEN, 1);

    private Command resumeCommand = new Command("Resume",
Command.SCREEN, 1);

    private ExamplePauseResumeMIDP2 midlet;

    private boolean isPlay; // Game Loop runs when isPlay is true

    private int width; // To hold screen width

    private int height; // To hold screen height

    private Thread gameThread = null; // Main Game Thread/loop
    // Layer Manager

    private LayerManager layerManager;
```

```
    // Pause flag
    private boolean isPause;

    // Variables used for demo animation
    private int posX;

    private int posY;

    private boolean isUp;

    public GameScreen(ExamplePauseResumeMIDP2 midlet) throws
Exception {
        super(true);
        this.midlet = midlet;
        addCommand(exitCommand);
        addCommand(pauseCommand);
        setCommandListener(this);
        width = getWidth(); // get screen width
        height = getHeight(); // get screen height
        isPlay = true;
        isPause = false;
        posX = width / 2;
        posY = height / 2;
        isUp = true;
        layerManager = new LayerManager();
    }

    // Start thread for game loop
    public void start() {
        gameThread = new Thread(this);
        gameThread.start();
    }

    // Stop thread for game loop
    public void stop() {
        gameThread = null;
    }

    // Main Game Loop
    public void run() {
        Graphics g = getGraphics();
        Thread currentThread = Thread.currentThread();
        try {
            while (currentThread == gameThread) {
                long startTime = System.currentTimeMillis();
                if (isShown()) {
```

```
                    if (isPlay) {
                        tick();
                    }
                    render(g);
                }
                long timeTake = System.currentTimeMillis() - startTime;
                if (timeTake < MILLIS_PER_TICK) {
                    synchronized (this) {
                        wait(MILLIS_PER_TICK - timeTake);
                    }
                } else {
                    currentThread.yield();
                }
            }
        } catch (InterruptedException ex) {
            // won't be thrown
        }
    }

    // Handle dynamic changes to game including user input
    public void tick() {
        try {
            if (isUp) {
                if (posY - 3 > 0) {
                    posY -= 3;
                } else {
                    isUp = false;
                }
            }
            if (!isUp) {
                if (posY + 3 < height) {
                    posY += 3;
                } else {
                    isUp = true;
                }
            }
        } catch (Exception e) {
            stop();
        }
    }

    // Handle Soft-Key reponses
    public void commandAction(Command c, Displayable d) {
        if (c == exitCommand) {
            midlet.exitGame();
        }
```

```
            if (c == pauseCommand) {
                pauseGame();
            }
            if (c == resumeCommand) {
                resumeGame();
            }
        }

        // Method to Display Graphics
        private void render(Graphics g) {
            g.setColor(0x00FFFFFF);
            g.fillRect(0, 0, width, height);
            g.setColor(0x000000FF);
            g.drawString("X", posX, posY, Graphics.TOP | Graphics.LEFT);
            layerManager.paint(g, 0, 0);
            flushGraphics();
        }

        // This where you should handle game auto-save, save to either network
        // server
        // and/or local RMS
        public void autoSave() {
            // Auto Save Stuff should be here
            System.out.println("Auto Saving Logic Should Be Here");
        }

        public void pauseGame() {
            // You may or may not want to have auto-save done at Pause, that is
why
            // when you exit
            // the system console outpus and extra 2 auto saving messages, total
3
            // including the
            // the call to autosave when exiting
            autoSave();
            removeCommand(pauseCommand);
            addCommand(resumeCommand);
            isPause = true;
            stop();
        }

        public void resumeGame() {
            removeCommand(resumeCommand);
            addCommand(pauseCommand);
            isPause = false;
            start();
```

```
    }

    public void hideNotify() {
        pauseGame();
    }

    public void showNotify() {
        resumeGame();
    }
}
```

17.1.2 使用 MIDP 1.0 实现暂停和返回

ExamplePauseResumeMIDP1.java (main midlet):

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class ExamplePauseResumeMIDP1 extends MIDlet {
    private Display display;

    private GameCanvas gameCanvas;

    public ExamplePauseResumeMIDP1() {
        display = Display.getDisplay(this);
    }

    public void startApp() {
        try {
            this.gameCanvas = new GameCanvas(this);
            this.gameCanvas.start();
            display.setCurrent(this.gameCanvas);
        } catch (Exception ex) {
            System.out.println("error: " + ex);
        }
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
        gameCanvas.autoSave();
    }

    public void exitGame() {
        destroyApp(false);
```

```
            notifyDestroyed();
        }
}
```

GameScreen.java:

```
import javax.microedition.lcdui.*;
import java.util.*;

public class GameCanvas extends Canvas implements Runnable,
CommandListener {
    private static final int MILLIS_PER_TICK = 50;

    private Command backCommand = new Command("Exit",
Command.BACK, 1);

    private Command pauseCommand = new Command("Pause",
Command.SCREEN, 1);

    private Command resumeCommand = new Command("Resume",
Command.SCREEN, 1);

    private ExamplePauseResumeMIDP1 midlet;

    private boolean isPlay;

    private int width;

    private int height;

    private Thread gameThread = null;

    private Image buffer;

    // Pause flag
    private boolean isPause;

    // Variables used for demo animation
    private int posX;

    private int posY;

    private boolean isUp;

    public GameCanvas(ExamplePauseResumeMIDP1 midlet) throws
Exception {
```

```
        this.midlet = midlet;
        addCommand(backCommand);
        addCommand(pauseCommand);
        setCommandListener(this);
        width = getWidth(); // get screen width
        height = getHeight(); // get screen height
        isPlay = true;
        isPause = false;
        posX = width / 2;
        posY = height / 2;
        isUp = true;
        if (!isDoubleBuffered())
            buffer = Image.createImage(width, height);
    }

    // Start thread for game loop
    public void start() {
        gameThread = new Thread(this);
        gameThread.start();
    }

    // Stop thread for game loop
    public void stop() {
        gameThread = null;
    }

    // Main Game Loop
    public void run() {
        Thread currentThread = Thread.currentThread();
        try {
            while (currentThread == gameThread) {
                long startTime = System.currentTimeMillis();
                if (isShown()) {
                    if (isPlay) {
                        tick();
                    }
                    repaint(0, 0, width, height);
                    serviceRepaints();
                }
                long timeTake = System.currentTimeMillis() - startTime;
                if (timeTake < MILLIS_PER_TICK) {
                    synchronized (this) {
                        wait(MILLIS_PER_TICK - timeTake);
                    }
                } else {
                    currentThread.yield();
```

```
                }
            }
            synchronized (this) {
                if (gameThread == Thread.currentThread()) {
                    gameThread = null;
                }
            }
        } catch (InterruptedException ex) {
            // won't be thrown
        }
    }

    // Handle dynamic changes to game including user input
    public void tick() {
        try {
            if (isUp) {
                if (posY - 3 > 0) {
                    posY -= 3;
                } else {
                    isUp = false;
                }
            }
            if (!isUp) {
                if (posY + 3 < height) {
                    posY += 3;
                } else {
                    isUp = true;
                }
            }
        } catch (Exception e) {
            stop();
        }
    }

    public void keyPressed(int keyCode) {
    }

    // Handle Soft-Key reponses
    public void commandAction(Command c, Displayable d) {
        if (c == backCommand) {
            midlet.exitGame();
        }
        if (c == pauseCommand) {
            pauseGame();
        }
        if (c == resumeCommand) {
```

```
            resumeGame();
        }
    }

    // Method to Display Graphics
    public void paint(Graphics g) {
        Graphics gr = g;
        try {
            if (buffer != null)
                g = buffer.getGraphics();
            g.setColor(0x00FFFFFF);
            g.fillRect(0, 0, width, height);
            g.setColor(0x000000FF);
            g.drawString("X", posX, posY, Graphics.TOP | Graphics.LEFT);
            if (g != gr)
                gr.drawImage(buffer, 0, 0, 20);
        } catch (Exception e) {
            System.out.println("Animation Error");
        }
    }

    // This where you should handle game auto-save, save to either network
    // server
    // and/or local RMS
    public void autoSave() {
        // Auto Save Stuff should be here
        System.out.println("Exiting and Auto Saving");
    }

    public void pauseGame() {
        autoSave();
        removeCommand(pauseCommand);
        addCommand(resumeCommand);
        isPause = true;
        stop();
    }

    public void resumeGame() {
        removeCommand(resumeCommand);
        addCommand(pauseCommand);
        isPause = false;
        start();
    }

    public void hideNotify() {
        pauseGame();
```

```
    }

    public void showNotify() {
        resumeGame();
    }
}
```

17.2 Unicode 和世界范围的语言支持

17.2.1 Unicode 介绍

Unicode是字符的唯一表示；这些字符包括从拉丁语到希伯来语、从日语到各种符号等所有字符。这个字符集是一个通用的标准，被称为ISO/IEC 10646。与ASCII码比较，它们都是一组数字对应一个字符，不同的是ASCII码只包含7bit，最多只能区分128个不同的字符。ASCII码仅仅覆盖了英文中的字符及一部分其它字符，但是难以包括其它语言的字符和更多的符号。随着计算机工业的发展，特别在英语不是主要字符使用集的国家，新的编码方案被创造出来。在使用不同编码方案的旧系统与新系统连接时，或者一个国家的系统与另外一个国家的系统连接时，就会存在很多潜在的显示问题。由于相互之间难以通讯，系统有可能被破坏甚至崩溃。所有这些就是Unicode出现的原因，下面摘自http://www.unicode.org的一段话很好的解释了什么是Unicode，以及使用Unicode的好处：

> *不论什么平台、什么程序、什么语言，Unicode中的每一个字符都对应唯一的数字。Unicode标准已经被Apple, HP, IBM,JustSystem, Microsoft, Oracle, SAP, Sun, Sybase, Unisys 和很多其它公司所采用。现代的标准如XML, Java, ECMAScript(JavaScript), LDAP, CORBA 3.0, WML等都需要Unicode的支持，而且Unicode已经写进官方的ISO/IEC 10646 中。Unicode被很多操作系统、所有的现代浏览器和其它产品所支持。Unicode标准的出现及支持Unicode的可用工具的开发，是当前全球软件技术的趋势。*

Unicode被大多数的语言和平台所支持，包括Java特别是J2ME。当发明Java语言的时候，很幸运地加入了对Unicode的支持，可以通过转义语句\u来进行Unicode处理。

17.2.2 使 J2ME 模拟器支持 Unicode

在显示Unicode字符之前需要确定模拟器是否支持Unicode。如果需要包含特定的字符，如日文字符，你需要MS Mincho字符集，在系统上安装了字符集之后需要编辑属性文件，这将在下一节介绍。如果需要支持各种不同的语言，操作过程是非常单调的，需要安装各种不同的属性文件。当然，一个简单的解决方法是安装一个包含多个语言支持的字体集，Arial Unicode MS就是其中之一。不幸的是，微软不在继

续提供它的免费版本，你可以购买它或者寻找一个替代的Unicode字体集。

安装新的字符集之后，进入Sun Wireless ToolKit的安装目录，再进入\wtk\devices目录，复制DefaultColorPhone到同一目录并重命名为UnicodePhone，并将UnicodePhone目录下的属性文件重命名为UnicodePhone.properties，使用任意编辑软件打开UnicodePhone.properties文件并找到与下面内容相似的部分：

```
font.default=SansSerif-plain-10
font.softButton=SansSerif-plain-11


font.system.plain.small: SansSerif-plain-9
font.system.plain.medium: SansSerif-plain-11
font.system.plain.large: SansSerif-plain-14

font.system.bold.small: SansSerif-bold-9
font.system.bold.medium: SansSerif-bold-11
font.system.bold.large: SansSerif-bold-14

font.system.italic.small: SansSerif-italic-9
font.system.italic.medium: SansSerif-italic-11
font.system.italic.large: SansSerif-italic-14

font.system.bold.italic.small: SansSerif-bolditalic-9
font.system.bold.italic.medium: SansSerif-bolditalic-11
font.system.bold.italic.large: SansSerif-bolditalic-14

font.monospace.plain.small: Monospaced-plain-9
font.monospace.plain.medium: Monospaced-plain-11
font.monospace.plain.large: Monospaced-plain-14

font.monospace.bold.small: Monospaced-bold-9
font.monospace.bold.medium: Monospaced-bold-11
font.monospace.bold.large: Monospaced-bold-14

font.monospace.italic.small: Monospaced-italic-9
font.monospace.italic.medium: Monospaced-italic-11
font.monospace.italic.large: Monospaced-italic-14

……
```

现在，如果你使用的是Arial Unicode MS字体集，用Arial\ Unicode\ MS替换文件中的SansSerif 和 Monospaced，更改后的文件内容如下：

```
font.default=Arial\ Unicode\ MS-plain-10
font.softButton=Arial\ Unicode\ MS-plain-11


font.system.plain.small: Arial\ Unicode\ MS-plain-9
font.system.plain.medium: Arial\ Unicode\ MS-plain-11
font.system.plain.large: Arial\ Unicode\ MS-plain-14

font.system.bold.small: Arial\ Unicode\ MS-bold-9
font.system.bold.medium: Arial\ Unicode\ MS-bold-11
font.system.bold.large: Arial\ Unicode\ MS-bold-14

font.system.italic.small: Arial\ Unicode\ MS-italic-9
font.system.italic.medium: Arial\ Unicode\ MS-italic-11
font.system.italic.large: Arial\ Unicode\ MS-italic-14

font.system.bold.italic.small: Arial\ Unicode\ MS-bolditalic-9
font.system.bold.italic.medium: Arial\ Unicode\ MS-bolditalic-11
font.system.bold.italic.large: Arial\ Unicode\ MS-bolditalic-14

font.monospace.plain.small: Arial\ Unicode\ MS-plain-9
font.monospace.plain.medium: Arial\ Unicode\ MS-plain-11
font.monospace.plain.large: Arial\ Unicode\ MS-plain-14

font.monospace.bold.small: Arial\ Unicode\ MS-bold-9
font.monospace.bold.medium: Arial\ Unicode\ MS-bold-11
font.monospace.bold.large: Arial\ Unicode\ MS-bold-14

font.monospace.italic.small: Arial\ Unicode\ MS-italic-9
font.monospace.italic.medium: Arial\ Unicode\ MS-italic-11
font.monospace.italic.large: Arial\ Unicode\ MS-italic-14

font.monospace.bold.italic.small: Arial\ Unicode\ MS-bolditalic-9
font.monospace.bold.italic.medium: Arial\ Unicode\ MS-bolditalic-11
font.monospace.bold.italic.large: Arial\ Unicode\ MS-bolditalic-14

……
```

更多详细的安装指导请参考Qusay H. Mahmoud的优秀指南的参考部分。

17.2.3Unicode 测试

现在，在所有字符都是Unicode的前提下，闻名于世的Hello World程序如下：

源代码：

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class SimpleUnicodeTest extends MIDlet {
    Display display;

    Form form = null;

    StringItem msg = null;

    public SimpleUnicodeTest() {
    }

    public void startApp() {
        display = Display.getDisplay(this);
        msg = new StringItem("'Hello World' in japanese",
                "\u3053\u3093\u306B\u3061\u306F\u4E16\u754C");
        form = new Form("Unicode Test");
        form.append(msg);
        display.setCurrent(form);
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean unconditional) {
    }
}
```

输出:

图39：Unicode实例

通过参考在线网络工具，经常可以将“Hello World”转换为“こんにちは世界”，又从“こんにちは世界”转换为“\u3053\u3093\u306B\u3061\u306F\u4E16\u754C”。

17.2.4 读取 Unicode 文件

接下来要做的就是从外部加载定义好的语言包到程序中，一个选择是从文本文件（Text File）读取语言定义，更为合适的选择是使用Unicode文件。下面是赌钱Unicode文件要用到的方法。

可以使用http://www4.vcnet.ne.jp/~klivo/sim/simeng.htm上的免费应用程序Simredo创建测试文件。

```
public String readUnicodeFile(String filename) {
    StringBuffer buffer = null;
    InputStream is = null;
    InputStreamReader isr = null;
    try {
        Class c = this.getClass();
        is = c.getResourceAsStream(filename);
        if (is == null)
            throw new Exception("File Does Not Exist");
        isr = new InputStreamReader(is, "UTF8");
        buffer = new StringBuffer();
        int ch;
        while ((ch = isr.read()) > -1) {
            buffer.append((char) ch);
        }
        if (isr != null)
            isr.close();
    } catch (Exception ex) {
        System.out.println(ex);
    }
    return buffer.toString();
}
```

17.2.5 读取包含 Unicode 代码的文本文件

上一节介绍了读取Unicode文件的方法，另一个方法是读取包含Unicode代码的Text文件。回顾前面的Hello World实例，在文本文件中包含\u3053\u3093\u306B\u3061\u306F\u4E16\u754C这段文字。当文件被程序读取的时候，每个字符都被当做字符串对待。如\u3053被以‘\’‘u’‘3’‘0’‘5’‘3’形式读入，现在只要找到并分析出‘\u’，就可以确定随后的四个字符表示一个Unicode字符。下面方法可以将字符串转换成正常的Unicode字符，返回有效的四个字符组成的Unicode，如3053。

```
private String convertStrToUnicode(String value) {
```

```
        short valueAsShort = Short.parseShort(value.trim(),16);
        return String.valueOf((char)valueAsShort);
    }
```

17.2.6 地区化实例 I

在理解了Unicode之后，我们可以编写一个在游戏或者应用程序中实现地区化的完整实例。

这个例子将尝试去抽象和分离出地区化数据的方法，虽然这不是移动开发中最有效率的方法，在下一节会提出改进和提高的本实例的建议。这里最先要解决的是，如何在不重新编译程序的条件下添加和移除语言支持包。也许在大多数人看来这是无关紧要的，经常有人说“简单！只有整加它然后重新编译一下，没有什么大不了的!”。但是，这的确会有“什么大不了”，每一次对代码的更改都很有可能带来新的Bug，反之，如果只是编译了文本文件而产生了错误，则可以确定是文本文件的错误。为了减少人员输入的错误，可以考虑开发一个格式校验工具，从而可以判断地区化的文件是否包含正确的格式。例如，你有一个在英国发布的支持法语和英语的游戏，运营商现在想支持德语和西班牙语，而你需要做的只是给他们发送适当的新文件，或者将支持地区化文件的应用程序重新打包，从而降低了服务的次数。

Locale类将读入基于手机的地区或者决定设置成的地区的三个文件，在测试过程中，它将从locale.support中读出所有的地区并回放到控制台中。在控制台，每个地区的主键通过locale.keys文件一对一的映射。

Test Driver Midlet:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class LocaleTest extends MIDlet {
    private LocaleLib localeLib;

    private Display display;

    TextBox box = null;

    public void startApp() {
        try {
            localeLib = LocaleLib.getInstance();
            //localeLib = LocaleLib.getInstance("en-us");
            //localeLib = LocaleLib.getInstance("de");
            String[] localeList = localeLib.getSupportedLocales();
            for (int i = 0; i < localeList.length; i++) {
```

```
                    localeLib.setLocaleKey(localeList[i]);
                    localeLib.loadLocale();
                    System.out.println("\n\n==== Locale " + localeList[i]
                            + " ================");

    System.out.println(localeLib.getLocaleString("ID_HELLOWORLD"));
                    System.out.println(localeLib.getLocaleString("ID_NEW"));

    System.out.println(localeLib.getLocaleString("ID_SETTINGS"));

    System.out.println(localeLib.getLocaleString("ID_HISCORE"));

    System.out.println(localeLib.getLocaleString("ID_HELP"));

    System.out.println(localeLib.getLocaleString("ID_ABOUT"));
                    System.out.println(localeLib.getLocaleString("ID_EXIT"));
                }
                String                              temp2                              =
localeLib.getLocaleString("ID_HELLOWORLD");
                box = new TextBox("", temp2, 500, 0);
                display = Display.getDisplay(this);
                display.setCurrent(box);
            } catch (Exception ex) {
                System.out.println(ex);
            }
    }

    public void destroyApp(boolean b) {
    }

    public void pauseApp() {
    }
}
```

LocaleTest.java:

```
import java.io.*;
import java.util.*;

public class LocaleLib {
    private static LocaleLib instance = null;

    private static String localeKey = null;

    private static String[] idKeys = null;
```

```
    private static String[] supportedLocales = null;

    private static String[] translationText = null;

    private static Hashtable translation = new Hashtable();

    private LocaleLib() {
    }

    private LocaleLib(String localeKey) throws Exception {
        try {
            supportedLocales = loadTextFile("/locale.support");
            idKeys = loadTextFile("/locale.keys");
            setLocaleKey(localeKey);
            loadLocale();
        } catch (Exception e) {
            throw new Exception("Error Loading Locale Resources::" + e);
        }
    }

    public static LocaleLib getInstance() throws Exception {
        return getInstance(System.getProperty("microedition.locale"));
    }

    public static LocaleLib getInstance(String localKey) throws Exception {
        if (instance == null)
            instance = new LocaleLib(localKey);
        return instance;
    }

    public String getLocaleKey() {
        return this.localeKey;
    }

    public String getLocaleString(String stringKey) {
        return this.parseUnicode((String) translation.get(stringKey));
    }

    public void setLocaleKey(String localeKey) throws Exception {
        this.localeKey = localeKey.toLowerCase();
        if (!validateLocaleKey())
            throw new Exception("Invalid Locale Key");
    }

    public String[] getSupportedLocales() {
        return this.supportedLocales;
```

```
    }

    public void loadLocale() throws Exception {
        translationText = loadTextFile("/locale."
                + this.localeKey.toLowerCase());
        if (translationText.length == idKeys.length) {
            for (int i = 0; i < idKeys.length; i++) {
                translation.put(idKeys[i], translationText[i]);
            }
        } else {
            throw new Exception("Invalid Locale Files, data not matching
keys");
        }
    }

    /**
     * Ensure the current Locale key exists in the supported local list as
     * listed in locale.support *
     *
     * @param none
     * @return false is invalid and true is valid
     */
    private boolean validateLocaleKey() {
        boolean result = false;
        for (int i = 0; i < supportedLocales.length; i++) {
            if (localeKey.equals(supportedLocales[i].toLowerCase())) {
                result = true;
                break;
            }
        }
        return result;
    }

    /**
     * Read Text file deliminted by comma *
     *
     * @param filename
     * @return Array with delimimted values
     */
    private String[] loadTextFile(String filename) throws Exception {
        String[] result = null;
        int resultIdx = 0;
        StringBuffer sb = null;
        Vector strings = null;
        Enumeration enum = null;
        try {
```

```
            Class c = this.getClass();
            InputStream is = c.getResourceAsStream(filename);
            if (is == null)
                throw new Exception("File Does Not Exist");
            sb = new StringBuffer();
            strings = new Vector();
            byte b[] = new byte[1];
            while (is.read(b) != -1) {
                if (b[0] != 44 && b[0] > 31 && b[0] < 255) {
                    sb.append(new String(b));
                } else {
                    if (sb.toString().trim().length() > 0) {
                        strings.addElement(sb.toString().trim());
                    }
                    sb.delete(0, sb.length());
                }
            }
            is.close();
            if (sb.toString().trim().length() > 0) {
                strings.addElement(sb.toString().trim());
            } else {
                sb = null;
            }
            result = new String[strings.size()];
            enum = strings.elements();
            while (enum.hasMoreElements()) {
                result[resultIdx] = (String) enum.nextElement();
                resultIdx++;
            }
        } catch (Exception e) {
            throw new Exception("ERROR: " + e);
        } finally {
            sb = null;
            strings = null;
            enum = null;
        }
        return result;
    }

    /**
     * Parse and translate unicode codes to the appropriate unicode symbol *
     *
     * @param String
     *                value contain possibly both ASCII characters and
unicode codes
     *               Unicode codes are signified by back slash then the
```

```
character u
     * @return String value with unicode codes translated to thier
appropriate
     *              unicode symbols
     */
    private String parseUnicode(String value) {
        String result = "";
        // Ensures value has at least one character
        if (value == null || value.length() <= 0)
            return result;
        int idx = 0;
        int idxFound = -1;
        int findLength = 2;
        StringBuffer sb = new StringBuffer();
        while ((idxFound = value.indexOf("\\u", idx)) != -1) {
            sb.append(value.substring(idx, idxFound));
            sb.append(convertStrToUnicode(value.substring(idxFound + 2,
                    idxFound + 6)));
            idx = idxFound + 6;
        }
        if (idx < value.length()) {
            sb.append(value.substring(idx));
        }
        return sb.toString();
    }

    /**
     * Converts a unicode string code to the unicode symbol *
     *
     * @param String
     *                  value representating a unicode code ie: 0669
     * @return unicode symbol as a String
     */
    private String convertStrToUnicode(String value) {
        short valueAsShort = Short.parseShort(value.trim(), 16);
        return String.valueOf((char) valueAsShort);
    }
}
```

17.2.7 地区化实例 II——建议

在了解了地区化如何实现之后，这里有一些相关的建议。由于具体情况不同，这些建议也许不符合你的情况，请自行斟酌使用。

1. 加入另外一个XML剖析器做为抽象层并将所有文件存储为XML格式。这种方式比较适用于那些建立一个通用的游戏服务器来给不同买主提供各种游戏的情况。Xadra通过他们的游戏服务器API实现了这种功能，参考 www.xadra.com。
2. 你也许希望通过从服务器取回地区化数据来代替直接在本地读取文件。这里有很多可供选择的通讯层，如SOAP、二进制数据等等。相关知识将在其它章节中讲述。
3. 假如你要制作一个非联网游戏并希望提高代码效率，可以考虑移除所有数据文件并将地区化数据以静态字符串的形式存储在LocaleLib.java中。同时，将原来用String定义的主键改为用Integer定义，这样也可以减少游戏所使用的内存空间。你需要自己积累这方面的经验。

第十八章 加入时间约束

18.1 概述

加入时间约束，即我们所说的试用版，就是让原始客户先使用我们功能不完全的版本，不管是J2ME游戏或者应用程序，如果满意的话需要付费才能使用到功能完全的版本。又很多试用版的形式，这里介绍三种最常见的：

- **日期：**使用具体的日期；到达这一时间游戏就会过期。通常假定正确时间就是用户手机设置的时间，故玩家可以将手机上的时间重新设置成较早的时间，从而可以继续游戏。一般用户会感觉这样很困扰，并且考虑购买你的游戏，特别是那些一直访问Internet的手机，它们会自动将时间更新到正确的时间。

- **期限：**用户第一次使用游戏时，当前日期被记录；从那之后，到达游戏设定的期限长度后，游戏过期。

- **使用次数限制：**用户每玩一次游戏都会被记录下来，一旦游戏的使用次数达到了设定的最大值，游戏将过期。

后两种方式需要记录使用的时间或者使用的次数。一种方法是使用本地的RMS系统，但是一些手机有移除所有RMS记录的功能。即使在不具有这种功能的手机上，用户也可以通过获得或更改他们本身的RMS来移除甚至修改游戏的RMS记录。替代的方法是通过网络读取服务器上的使用信息。简单起见，这里使用本地RMS，真的庆幸的是，一般的游戏玩家并不具有足够的知识来自己更改或移除RMS。

18.2 实现方法

在上节列出的三个方式中，涉及的数据有过期日期、最长使用期限、最大使用次数。有几种存储这些初始数据的选择，可以直接在代码中设定，也可以通过运行时要读取的文本文件设定。考虑到需要重新编译和打包，前一种方法并不可取；甚至第二种方法也要编辑文本文件同时重新打包JAR文件。如果游戏开发者完全负责游戏的发布，这二种方法还可以接受，但是一般情况下，游戏开发者开发出的游戏都是有第三方（如运营商、集团及其它发布渠道）进行部署，让他们进行重新编译和打包游戏就不是很现实了。这时，考虑到JAD文件经常要被分发者自己更改下载的URL路径，通过JAD文件来实现这几种数据的存储不失为一个好的选择。当然可以设定默认的试用期的值，避免分发者再去更改相关参数。

这里列出了在JAD文件中新加入的参数：

- Expire-Date: 20031201
- Expire-Days: 30
- Expire-Plays: 10

加入了过期日期后的JAD文件将是如下形式：

```
MIDlet-1: TimeTrialDemo, , TimeTrialDemo
MIDlet-Jar-Size: 49006
MIDlet-Jar-URL: TimeTrialDemo.jar
MIDlet-Name: TimeTrialDemo
MIDlet-Vendor: Sun Microsystems
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.0
MicroEdition-Profile: MIDP-1.0
Expire-Date: MjAwMzEwMTU=
```

Expire-Date的值是不可读的，它是使用Base64加密算法加密来提高安全性。虽然Base64不算很强的加密算法，但是阻止通过访问JAD文件来更改Expire-Date值的人还是够用了。这有可能对分发者造成困扰，但是开发者仅仅需要重新发布JAD文件，这显然好过重新编译和打包。

可以使用MIDlet类的getAppProperty(String key)方法来获取JAD文件中的参数值，而不是System类的静态getProperty(String key)。例如：

```
String expirePlays = getAppProperty("Expire-Date");
```

获得这个值之后就可以验证游戏是否到期了。而且要验证Expire-Date的合法性，包括是否都是数字、合法日期、在可接受的范围中，如果不符合设定的规则，说明Expire-Date被用户私自更改，作为惩罚可以直接使游戏过期。

18.3 增强对游戏的保护

保护游戏还有一些方法：

- 应用服务器端组件实现试用期功能
- 用混淆器混淆代码是明智的
- 增加数字权利管理（Digital Rights Management,DRM）。上面提到的方法并不能阻止任何人将游戏从一部手机传到另一部手机。使用DRM则可以防止没有权限的人分发游戏，一旦原始游戏被使用与特定手机相关的授权码加密后，只有相应的手机才可以执行游戏。
- 考虑移除一些不想让用户在demo或试用版中使用的特征，如升级、特定的能力提升等。这会降低游戏的体验，鼓励用户购买完全版

- 使用强度加密算法，如Bouncy Castle 或者 Kerberos
- 通过OTA下载时，设定一个缓存区，使用户只能下载一次，或者在一定时间后才能再一次下载
- OTA只对会员开放

18.4 价值和价格

在介绍了游戏试用版的发布后，我们来看看手机游戏一般价格，在美国一般平均为$3.00到$9.00，这通常是由市场决定的。然而，运营商会帮游戏定价，一些运营商也会需要试用版或demo版本。

18.5 实例

18.5.1 主 Midlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class TimeTrialDemo extends MIDlet implements CommandListener {
    private Display display;

    private Command cmdExit = new Command("Exit", Command.EXIT,
1);;

    private Form formMain;

    private StringItem stringItem;

    private TimeTrial timeTrial;

    private static final String rmsName = "TimeTrialDemoRMS";

    private boolean result;

    public TimeTrialDemo() {
        try {
            // Manual Encrypt - You may need this to change the values in
the
            // JAD file
            //System.out.println("Manual Encryption: " +
            // Base64.encodeToString("20031015"));
            // Time Trial - By Date
            //timeTrial = new TimeTrial(rmsName,TimeTrial.TT_DATE);
            //result = TimeTrial.isValid(getAppProperty("Expire-Date"));
```

```
                // Time Trial - Number of Days
                //timeTrial = new TimeTrial(rmsName,TimeTrial.TT_DAYS);
                //result = TimeTrial.isValid(getAppProperty("Expire-Days"));
                // Time Trial - Number of Plays
                timeTrial = new TimeTrial(rmsName, TimeTrial.TT_PLAYS);
                result = timeTrial.isValid(getAppProperty("Expire-Plays"));
                if (result)
                    System.out.println("Game NOT Expired");
                else
                    System.out.println("Game Expired");
            } catch (Exception ex) {
                System.out.println(ex);
            }
            display = Display.getDisplay(this);
            stringItem = new StringItem("", "See Console");
            formMain = new Form("Time Trial Demo");
            formMain.addCommand(cmdExit);
            formMain.append(stringItem);
            formMain.setCommandListener(this);
        }

        public void startApp() {
            display.setCurrent(formMain);
        }

        public void pauseApp() {
        }

        public void destroyApp(boolean unconditional) {
        }

        public void commandAction(Command c, Displayable s) {
            if (c == cmdExit) {
                destroyApp(false);
                notifyDestroyed();
            }
        }
}
```

18.5.2 试用期类

```
import javax.microedition.rms.*;
import java.util.*;
import java.io.*;

public class TimeTrial extends BaseRMS {
```

```
public static final int TT_DATE = 0;

public static final int TT_DAYS = 1;

public static final int TT_PLAYS = 2;

public static final int MAX_NUM_DAYS = 30;

public static final int MAX_NUM_PLAYS = 10;

Calendar currentCalendar;

Date currentDate;

int currentDateAsInt; // YYYYMMDD

private int timeTrialType;

private String value;

private String decryptedValue;

private int numDays;

private int numPlays;

/**
 * Constructor
 */
public TimeTrial(String rmsName, int timeTrialType) throws Exception {
    super(rmsName);
    setTimeTrialType(timeTrialType);
    currentDate = new Date();
    currentCalendar = Calendar.getInstance();
    currentCalendar.setTime(currentDate);
    currentDateAsInt = getCurrentDateAsInt();
}

/**
 * Change TimeTrial Method
 */
public void setTimeTrialType(int timeTrialType) throws Exception {
    if (timeTrialType == TT_DATE || timeTrialType == TT_DAYS
            || timeTrialType == TT_PLAYS) {
        this.timeTrialType = timeTrialType;
    } else {
```

```
            throw new Exception("Invalid Time Trial Option");
        }
    }

    /**
     * Check if App is Exired
     */
    public boolean isValid(String value) throws Exception {
        boolean result = false;
        this.value = value;
        decryptedValue = Base64.decodeToString(value);
        switch (this.timeTrialType) {
        case TT_DATE:
            result = isValidDate();
            break;
        case TT_DAYS:
            result = isValidDays();
            break;
        case TT_PLAYS:
            result = isValidPlays();
            break;
        }
        ;
        return result;
    }

    /**
     * Time Trial by Date
     */
    private boolean isValidDate() throws Exception {
        boolean result = false;
        Date expireDate = new Date();
        Calendar expireCalendar = Calendar.getInstance();
        expireCalendar.set(Calendar.YEAR, Integer.parseInt(decryptedValue
                .substring(0, 4)));
        expireCalendar.set(Calendar.MONTH, Integer.parseInt(decryptedValue
                .substring(4, 6)));
        expireCalendar.set(Calendar.DAY_OF_MONTH, Integer
                .parseInt(decryptedValue.substring(6, 8)));
        expireDate = expireCalendar.getTime();
        if (currentDate.getTime() < expireDate.getTime())
            result = true;
        return result;
    }

    /**
```

```
     * Time Trial By Number of Days
     */
    private boolean isValidDays() throws Exception {
        boolean result = false;
        loadTimeTrialData();
        if (currentDateAsInt - numDays < Integer.parseInt(decryptedValue)) {
            result = true;
        }
        return result;
    }

    /**
     * Time Trial By Number of Plays
     */
    private boolean isValidPlays() throws Exception {
        boolean result = false;
        loadTimeTrialData();
        System.out.println("Play: " + decryptedValue + " Plays RMS: "
                + numPlays);
        if (numPlays <= Integer.parseInt(decryptedValue)) {
            updatePlays(++numPlays);
            result = true;
        }
        return result;
    }

    public void loadTimeTrialData() throws Exception {
        try {
            // Will call either loadData() or createDefaultData()
            this.open();
            if (this.getRecordStore() != null)
                this.close();
        } catch (Exception e) {
            throw new Exception("Error loading Settings" + e);
        }
    }

    private void updatePlays(int numPlays) throws Exception {
        try {
            // load current scores
            this.open();
            // Update
            this.numPlays = numPlays;
            updateData();
            // close
            if (this.getRecordStore() != null)
```

```
                this.close();
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::updateTimeTrial::" +
e);
            }
        }

        private int getCurrentDateAsInt() {
            StringBuffer sb = new StringBuffer();
            sb.append(String.valueOf(currentCalendar.get(Calendar.YEAR)));
            if (currentCalendar.get(Calendar.MONTH) < 10)
                sb
                        .append("0"
                                + String.valueOf(currentCalendar
                                        .get(Calendar.MONTH)));
            else
                sb.append(String.valueOf(currentCalendar.get(Calendar.MONTH)));

        sb.append(String.valueOf(currentCalendar.get(Calendar.DAY_OF_MONTH)));
            return Integer.parseInt(sb.toString());
        }

        ////////////////////////////////////////////////////////
        // RMS Related methods (loadData,createDefaultData, updateData) inherited
        // from BaseRMS
        protected void loadData() throws Exception {
            try {
                byte[] record = this.getRecordStore().getRecord(1);
                DataInputStream istream = new DataInputStream(
                        new ByteArrayInputStream(record, 0, record.length));
                numDays = istream.readInt();
                numPlays = istream.readInt();
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::loadData::" + e);
            }
        }

        /**
         * Default values for numDays will be currentDay (first day the user plays
         * the game * numPlays is defaulted to zero number times played
         */
        protected void createDefaultData() throws Exception {
            try {
                numDays = currentDateAsInt;
                numPlays = 0;
                ByteArrayOutputStream bstream = new ByteArrayOutputStream(12);
```

```
                DataOutputStream ostream = new DataOutputStream(bstream);
                ostream.writeInt(numDays);
                ostream.writeInt(numPlays);
                ostream.flush();
                ostream.close();
                byte[] record = bstream.toByteArray();
                this.getRecordStore().addRecord(record, 0, record.length);
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::createDefaultData::" +
e);
            }
        }

        protected void updateData() throws Exception {
            try {
                ByteArrayOutputStream bstream = new ByteArrayOutputStream(12);
                DataOutputStream ostream = new DataOutputStream(bstream);
                ostream.writeInt(numDays);
                ostream.writeInt(numPlays);
                ostream.flush();
                ostream.close();
                byte[] record = bstream.toByteArray();
                this.getRecordStore().setRecord(1, record, 0, record.length);
            } catch (Exception e) {
                throw new Exception(this.getRMSName() + "::updateData::" + e);
            }
        }
}
```

18.5.3 BaseRMS 类

见第十章的BaseRMS.java文件。

18.5.4 Base64 类

下面代码不是原作者本人写的，而是从网上找到的，不幸的是不记得代码的编写者是谁了。

```
import java.io.*;

public class Base64 {
    protected static final char[] alphabet = { 'A', 'B', 'C', 'D', 'E', 'F',
            'G', 'H', // 0 to 7
            'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', // 8 to 15
            'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', // 16 to 23
```

```
            'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', // 24 to 31
            'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', // 32 to 39
            'o', 'p', 'q', 'r', 's', 't', 'u', 'v', // 40 to 47
            'w', 'x', 'y', 'z', '0', '1', '2', '3', // 48 to 55
            '4', '5', '6', '7', '8', '9', '+', '/' // 56 to 63
    };

    protected static final int[] decodeTable = { -1, -1, -1, -1, -1, -1, -1,
            -1, -1, -1, // 0 to 9
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, // 10 to 19
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, // 20 to 29
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, // 30 to 39
            -1, -1, -1, 62, -1, -1, -1, 63, 52, 53, // 40 to 49
            54, 55, 56, 57, 58, 59, 60, 61, -1, -1, // 50 to 59
            -1, -1, -1, -1, -1, 0, 1, 2, 3, 4, // 60 to 69
            5, 6, 7, 8, 9, 10, 11, 12, 13, 14, // 70 to 79
            15, 16, 17, 18, 19, 20, 21, 22, 23, 24, // 80 to 89
            25, -1, -1, -1, -1, -1, -1, 26, 27, 28, // 90 to 99
            29, 30, 31, 32, 33, 34, 35, 36, 37, 38, // 100 to 109
            39, 40, 41, 42, 43, 44, 45, 46, 47, 48, // 110 to 119
            49, 50, 51 // 120 to 122
    };

    public static String encodeToString(String s) {
        char[] encoded = encode(s);
        StringBuffer encodedSB = new StringBuffer();
        for (int i = 0; i < encoded.length; i++) {
            encodedSB.append(encoded[i]);
        }
        return encodedSB.toString();
    }

    public static char[] encode(String s) {
        return encode(s.getBytes());
    }

    public static char[] encode(byte[] bytes) {
        int sixbit;
        char[] output = new char[((bytes.length - 1) / 3 + 1) * 4];
        int outIndex = 0;
        int i = 0;
        while ((i + 3) <= bytes.length) {
            sixbit = (bytes[i] & 0xFC) >> 2;
            output[outIndex++] = alphabet[sixbit];
            sixbit = ((bytes[i] & 0x3) << 4) + ((bytes[i + 1] & 0xF0) >> 4);
            output[outIndex++] = alphabet[sixbit];
```

```
                sixbit = ((bytes[i + 1] & 0xF) << 2) + ((bytes[i + 2] & 0xC0) >>
6);
                output[outIndex++] = alphabet[sixbit];
                sixbit = bytes[i + 2] & 0x3F;
                output[outIndex++] = alphabet[sixbit];
                i += 3;
            }
            if (bytes.length - i == 2) {
                sixbit = (bytes[i] & 0xFC) >> 2;
                output[outIndex++] = alphabet[sixbit];
                sixbit = ((bytes[i] & 0x3) << 4) + ((bytes[i + 1] & 0xF0) >> 4);
                output[outIndex++] = alphabet[sixbit];
                sixbit = (bytes[i + 1] & 0xF) << 2;
                output[outIndex++] = alphabet[sixbit];
                output[outIndex++] = '=';
            } else if (bytes.length - i == 1) {
                sixbit = (bytes[i] & 0xFC) >> 2;
                output[outIndex++] = alphabet[sixbit];
                sixbit = (bytes[i] & 0x3) << 4;
                output[outIndex++] = alphabet[sixbit];
                output[outIndex++] = '=';
                output[outIndex++] = '=';
            }
            return output;
        }

        public static String decodeToString(String encoded) {
            byte[] decoded = decode(encoded);
            StringBuffer decodedSB = new StringBuffer();
            for (int i = 0; i < decoded.length; i++) {
                decodedSB.append((char) decoded[i]);
            }
            return decodedSB.toString();
        }

        public static byte[] decode(String encoded) {
            byte[] decoded = null;
            int decodedLength = (encoded.length() / 4 * 3);
            int invalid = 0;
            if (encoded.length() % 4 != 0) {
                System.err.println("It's not BASE64 encoded string.");
                return null;
            }
            if (encoded.charAt(encoded.length() - 2) == '=') {
                invalid = 2;
            } else if (encoded.charAt(encoded.length() - 1) == '=') {
```

```
            invalid = 1;
        }
        decodedLength -= invalid;
        decoded = new byte[decodedLength];
        int i = 0, di = 0;
        int sixbit0, sixbit1, sixbit2, sixbit3;
        for (; i < encoded.length() - 4; i += 4) {
            sixbit0 = decodeTable[encoded.charAt(i)];
            sixbit1 = decodeTable[encoded.charAt(i + 1)];
            sixbit2 = decodeTable[encoded.charAt(i + 2)];
            sixbit3 = decodeTable[encoded.charAt(i + 3)];
            decoded[di++] = (byte) ((sixbit0 << 2) + ((sixbit1 & 0x30) >>
4));
            decoded[di++] = (byte) (((sixbit1 & 0xF) << 4) + ((sixbit2 &
0x3C) >> 2));
            decoded[di++] = (byte) (((sixbit2 & 0x3) << 6) + sixbit3);
        }
        switch (invalid) {
        case 0:
            sixbit0 = decodeTable[encoded.charAt(i)];
            sixbit1 = decodeTable[encoded.charAt(i + 1)];
            sixbit2 = decodeTable[encoded.charAt(i + 2)];
            sixbit3 = decodeTable[encoded.charAt(i + 3)];
            decoded[di++] = (byte) ((sixbit0 << 2) + ((sixbit1 & 0x30) >>
4));
            decoded[di++] = (byte) (((sixbit1 & 0xF) << 4) + ((sixbit2 &
0x3C) >> 2));
            decoded[di++] = (byte) (((sixbit2 & 0x3) << 6) + sixbit3);
            break;
        case 1:
            sixbit0 = decodeTable[encoded.charAt(i)];
            sixbit1 = decodeTable[encoded.charAt(i + 1)];
            sixbit2 = decodeTable[encoded.charAt(i + 2)];
            decoded[di++] = (byte) ((sixbit0 << 2) + ((sixbit1 & 0x30) >>
4));
            decoded[di++] = (byte) (((sixbit1 & 0xF) << 4) + ((sixbit2 &
0x3C) >> 2));
            break;
        case 2:
            sixbit0 = decodeTable[encoded.charAt(i)];
            sixbit1 = decodeTable[encoded.charAt(i + 1)];
            decoded[di++] = (byte) ((sixbit0 << 2) + ((sixbit1 & 0x30) >>
4));
            break;
        }
        return decoded;
```

```
    }

    static boolean bytesEquals(byte[] b1, byte[] b2) {
        if (b1.length != b2.length) {
            return false;
        }
        for (int i = b1.length - 1; i >= 0; i--) {
            if (b1[i] != b2[i]) {
                return false;
            }
        }
        return true;
    }
}
```

第十九章 特征化界面

本章将介绍通过GameCanvas或Canvas创建特征化用户界面的方法，有关建立特征化用户界面的优缺点这里不再赘述，请参考第八章。

一般认为创建特征化界面就是编写新的输入方法，其实不然，特征化界面不但包括菜单选择，还包括像*设置Settings*等一系列相关内容。创建自己的单选框、多选框等是比较困难的，可以考虑符号化*高分记录*High Score的输入，登录游戏的验证、跳入下一关的秘笈等等。请继续阅读后面章节。

19.1 主菜单

19.1.1 简单高亮菜单

现在考虑将游戏主菜单设计得更有创造性也更像一款游戏。首先，进行一些简单的改变,将五个菜单项(开始新游戏New Game,高分记录High Score,设置Settings,帮助Help，关于About）设为居中显示。被选中的选项将会增大字体、改变颜色并且在后面显示高亮条，从而突出显示。视图如下：

图40：简单高亮菜单实例

现在来看一下程序的源代码。

主 Midlet – SimleCustomMenu.java:

与一般的Midlet差不多。

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;

public class SimpleCustomMenu extends MIDlet implements CommandListener {
    Display display;

    Display pauseDisplay;

    boolean isSplash = true;

    MenuScreen menuScreen;

    public SimpleCustomMenu() {
        MenuScreen menuScreen = new MenuScreen();
        display = Display.getDisplay(this);
        display.setCurrent(menuScreen);
    }

    protected void startApp() throws MIDletStateChangeException {
    }

    protected void pauseApp() {
    }

    protected void destroyApp(boolean flag) throws MIDletStateChangeException {
    }

    public void commandAction(Command cmd, Displayable dis) {
    }
}
```

MenuScreen.java

高亮菜单的实现不需要太大代码，仅仅是通过菜单项的个数和字体的大小来使菜单项居中。由于代码非常直观，更多信息请阅读代码：

```
import javax.microedition.lcdui.*;
```

```
public class MenuScreen extends Canvas implements Runnable {
    // Set Fonts
    static final Font lowFont = Font.getFont(Font.FACE_MONOSPACE,
            Font.STYLE_PLAIN, Font.SIZE_SMALL);

    static final Font highFont = Font.getFont(Font.FACE_MONOSPACE,
            Font.STYLE_BOLD, Font.SIZE_MEDIUM);

    // Set Color
    static final int lowColor = 0x000000FF; // Not Highlighted
    static final int highColor = 0x00FF0000; // Highlighted
    static final int highBGColor = 0x00CCCCCC;// Highlighted Background

    static int width; // screen width

    static int height; // screen height

    static int startHeight; // height where the menu starts

    static final int spacing = highFont.getHeight() / 2; // spacing between
menu
                                                         // items
    // Menu Item Labels

    static final String[] mainMenu = { "New Game", "High Score",
"Settings",
            "Help", "About" };

    // To hold the current highlighted menu option
    static int menuIdx;

    // Menu Thread
    Thread menuThread;

    // Constructor
    public MenuScreen() {
        // Get Width and Height of Canvas
        width = getWidth();
        height = getHeight();
        // Calculate Start Height of Menu
        startHeight = (highFont.getHeight() * mainMenu.length)
                + ((mainMenu.length - 1) * spacing);
        startHeight = (height - startHeight) / 2;
        // Set Selected Menu Item to the first menu item
        menuIdx = 0;
        // Create Thread and Start
```

```
        menuThread = new Thread(this);
        menuThread.start();
    }

    // Simple Run -- Should be modified for better performance/efficiency
    public void run() {
        while (true) {
            repaint();
        }
    }

    // Paint Main Menu
    public void paint(Graphics g) {
        g.setColor(0x00FFFFFF);
        g.fillRect(0, 0, width, height);
        for (int i = 0; i < mainMenu.length; i++) {
            if (i == menuIdx) {
                g.setColor(highBGColor);
                g.fillRect(0, startHeight + (i * highFont.getHeight())
                        + spacing, width, highFont.getHeight());
                g.setFont(highFont);
                g.setColor(highColor);
                g.drawString(mainMenu[i], (width - highFont
                        .stringWidth(mainMenu[i])) / 2, startHeight
                        + (i * highFont.getHeight()) + spacing, 20);
            } else {
                g.setFont(lowFont);
                g.setColor(lowColor);
                g.drawString(mainMenu[i], (width - lowFont
                        .stringWidth(mainMenu[i])) / 2, startHeight
                        + (i * highFont.getHeight()) + spacing, 20);
            }
        }
    }

    // Capture Keypresses for Menu Selection
    protected void keyPressed(int code) {
        if (getGameAction(code) == Canvas.UP && menuIdx - 1 >= 0) {
            menuIdx--;
        } else if (getGameAction(code) == Canvas.DOWN
                && menuIdx + 1 < mainMenu.length) {
            menuIdx++;
        }
    }
}
```

19.1.2 改进高亮菜单

这里不可能介绍所有的改善菜单的方法，因为你可以很容易的通过图像的组合来获得大量的不同的却是吸引人的菜单。例如，增加一个背景图片可以带来很大的改观：

图41：改进后的主菜单

19.2 配件输入

19.3 字符化输入

其它章节

网格计算机
聚集（Rendezvous）
P2P

比较
Java Vs MS Vs Symbian Vs Palm Vs Brew

可能的章节
- 2D变换（2D Transformations）
- 等距贴砖（Isometric Tiles）
- 人工智能（AI）
- 多用户（Multiplayer）

附　录

- 运行书中实例
- 编译工具
- OTA

附录 A：运行书中实例

附录 B：编译工具

B.1 使用 Ant

B.2 使用 Ant&Antenna

附录 C：OTA