

手机游戏开发 精粹

傅曦 高雷 陈博 史亚炜 编著



This is trial version
www.adultpdf.com

内 容 简 介

本书详细讲解了手机3大游戏开发平台——Symbian、Windows Mobile、J2ME的开发环境搭建、程序设计、开发调试和游戏编码，另外还提供了完整的游戏示例代码。游戏开发爱好者可以根据自己的爱好选择各平台学习。

本书适合有一定C++或Java编程基础的手机游戏编程初学者及广大手机开发人员参考使用。

前 言

写作背景

笔者从小学二年级就开始玩电子游戏了，那时候流行红白机、《坦克大战》、《魂斗罗》等，当时觉得新奇好玩。随着电子产品的迅猛发展，今天，我们随处可见笔记本电脑、PSP、手机，而且，它们的性能还在不断地提高。尤其是手机，目前中国的手机用户超过1.848亿，几乎每部手机上都有预装游戏，在等候汽车、乘坐地铁时，经常会看见人们在玩手机游戏。

如今，在手机上开发游戏的技术已经很成熟了，相对PC游戏编程来说，其入门非常简单。手机游戏开发使用的编程语言可以选择C++或Java，目前来说这两门语言在PC开发上也是很热门的。有意思的是，PC上的大型3D游戏大多是使用C++开发的，而手机上的游戏大多是使用Java开发的。

鉴于目前手机游戏市场火爆，开发人员紧缺，我们组织编写了本书，希望能对广大手机游戏开发爱好者起到抛砖引玉的作用。

本书特点

本书内容涵盖了目前主流的手机游戏开发，如Symbian开发、Windows Mobile开发、J2ME开发等；另外，手机游戏平台还包括苹果的iPhone、联通的BREW、日本手机的Doja以及Google的Android等，有些平台目前的市场占有率较小，因此本书重点讲述前3种平台。

Symbian平台在智能手机市场占有率为70%左右，其开发语言为C++。大部分大学都开设了《C++程序设计》这门课程，因此入门并不困难，而且，目前Symbian手机游戏开发人才相对紧缺，收入较高，因而值得选择学习。

虽然Windows Mobile平台目前的市场占有率较少，但手机桌面化已成不争的事实，身为微软产品的它不愁后期发展。对于有Windows编程经验的人员，这个平台是一个不错的选择。

J2ME平台的市场占有率最高，这也和它的设计理念有关系，因为只要设备上有Java虚拟机或模拟器就能运行Java程序，而且J2ME平台开发是最容易入门的手机游戏开发平台。

内容简介

本书共分4部分，全面讲解手机游戏的开发、设计及代码编写。

第1部分（第1章～第3章）：手机游戏市场前景和开发人员的职业规划。

第1章详细阐述手机游戏的市场前景。

第2章介绍手机游戏分类和技术特点。

第3章介绍手机游戏产业中的职业规划。

第2部分（第4章～第9章）：Symbian手机操作系统介绍、开发及游戏示例。

第4章讲解Symbian操作系统概述、主要特点、版本介绍以及开发基础。

第5章讲解Symbian OS开发中的异常处理、键盘事件处理、音频声音处理等。

第6章讲解Symbian OS图形程序开发，包括图形架构、基本绘图函数、字体、OpenGL ES、游戏编程中的位图技巧。

第7章介绍Symbian OS通信开发，包括通信架构、串口通信服务器、套接字服务器、游戏数据接收、串口编程范例。

第8章介绍Symbian OS开发和调试技巧、安装Symbian SDK及给Visual C++配置开发环境。

第9章介绍Symbian游戏示例。

第3部分（第10章～第13章）：Windows Mobile开发介绍，包括图形图像操作、雷电射击游戏示例。

第10章介绍Windows Mobile开发介绍，包括事件驱动和消息响应机制、开发工具和Windows Mobile程序移植。

第11章介绍Windows Mobile 开发基础，包括菜单、对话框、窗口、控件、多线程和多进程。

第12章介绍Windows Mobile图形图像操作基础，包括文本操作、基本图形图像操作和高级屏幕绘图。

第13章讲解Windows Mobile游戏示例，包括俄罗斯方块和雷电射击游戏。

第4部分（第14章～第18章）：J2ME介绍与环境搭建、程序设计、潜艇游戏示例。

第14章介绍J2ME环境搭建、开发环境和调试技巧。

第15章介绍MIDP程序开发和MIDP界面程序设计，以及List、TextBox、Form、TextField、Ticker、Display、Canvas、Graphics、Image、Font等类的使用，还介绍了MIDP数据库程序设计简介、MIDP网络程序设计。

第16章介绍MIDP2.0游戏API简介，以及Layer、LayerManager、Sprite、TiledLayer、GameCanvas等类的使用。

第17章介绍J2ME游戏设计，包括开发过程、界面设计、游戏引擎设计介绍、代码优化。

第18章讲解J2ME游戏范例，包括潜艇游戏完整代码、详尽注释及游戏中各个部分的说明。

致谢

本书由傅曦、高雷、陈博、史亚炜编著。同时，参与本书编写工作的还有刘燕祎、周晶、周丰、梅乐夫、房明浩、王亮、门店宏、吴洋、石峰、张圣亮、邱文勋、刘鯤、林远长、董前程、朱飞、汤嘉立、刘变红、刘会灯、张高煜、赵红波、张宪栋、邓志宝、刘坤、刘明辉、李鹏、白学明、步士建等。在此，对以上人员致以诚挚的谢意。

由于时间仓促，加之笔者水平有限，书中不足之处在所难免，敬请读者批评指正。本书责任编辑的电子邮箱是huangyan@ptpress.com.cn，欢迎来信交流。

编者
2009年1月

第 1 章

手机游戏的市场和发展前景

1.1 游戏市场现状

近年来，在世界范围内，随着手机的普及，手机游戏已经成为整个视频游戏领域发展最迅速的组成部分。

根据 iResearch 艾瑞市场咨询整理的国外数据显示，全球手机游戏市场收入规模在 2005 年达到了 102 亿美元，预计到 2008 年将达到 520 亿美元。

手机游戏市场是一个急速膨胀的市场，在未来几年将会成几何级数增长。近年来手机市场规模如图 1.1 所示。

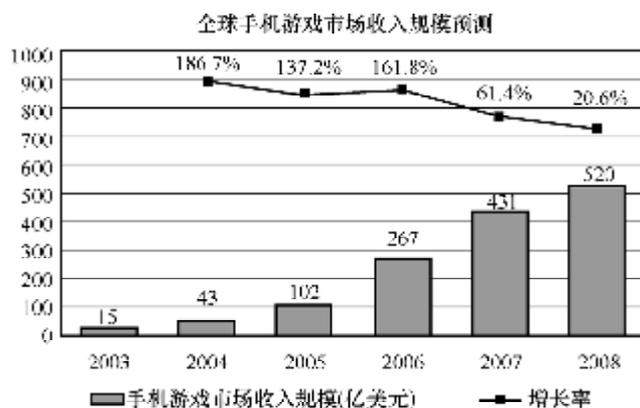


图 1.1 手机游戏市场收入规模和增长率

在这个背景下，我国的游戏也有了长足的发展，但是就其规模而言，还远远没有达到国际水平。这其中原因很多，但有一点可以肯定：我国的游戏前景是光明的。

第 2 章

手机游戏开发的技术特点

2.1 游戏的种类

从 1958 年电视游戏的诞生开始，经过不断的发展和完善，游戏的内容越来越新奇，其种类也越来越繁多。

到目前为止，已有的游戏基本分为以下几大类。

- n 角色扮演类——RPG
- n 回合制战略类
- n 益智解谜类——PUZ
- n 即时战略类——RTS
- n 动作格斗类——FTG
- n 策略类——SLG
- n 冒险类——AVG
- n 射击类——STG
- n 体育类——SPG
- n 模拟养成类——EDU

下面将分别介绍各种类型游戏的特点及其开发技术。

2.1.1 角色扮演类

角色扮演类游戏（RPG, Role-playing Game）。这类游戏提供给玩家一个充满冒险或者反映真实的世界，这个世界包含了各种角色、建筑、商店、迷宫及险峻的地形。在这个世界中，玩家所扮演的主角通过旅行、交谈、交易、打斗、成长、探险及解谜来揭开一系列的故事情节和线索，最终走向胜利的彼岸。玩家依靠自身的胆识、智慧和机敏获得一次又一次的成功，使自己扮演的主角不断发展壮大，从而得到巨大的精神满足。

单机代表作中最为经典的是国内三剑：大宇的《仙剑奇侠传》系列、《轩辕剑》系列和西山居的《剑侠情缘》系列。手机上这类游戏的代表作有《仙剑奇侠传 4》、诺基亚公司的同名游戏《Nokia》、大宇的《仙剑奇侠传 Mobile》。《仙剑奇侠传 4》的游戏界面如图 2.1 所示，《仙剑奇侠传 Mobile》的游戏界面如图 2.2 所示。



图 2.1 《仙剑奇侠传 4》游戏截图



图 2.2 《仙剑奇侠传 Mobile》游戏截图

2.1.2 回合制战略类

回合制战略类游戏（TBS, Turn-based Strategy Game）。在这类游戏中，参加战斗的几方（可以包括计算机在内）依一定顺序分别部署战略，一次部署便称作一个回合。这类游戏比角色扮演类游戏更强调战斗的场面，但不再有角色扮演游戏中令人头疼的迷宫。由于这一类游戏对战斗过程的注重，所以各类道具在游戏中非常重要。在游戏中，游戏者所要关心的就是战斗、贸易，再战斗、再贸易。

这类游戏的经典代表作有《文明》系列、《英雄无敌》系列、《天使帝国》系列等。手机上这类游戏的代表作有《大汉王朝》、《屠魔》、《三国豪侠传》等。《英雄无敌 4》的游戏界面如图 2.3 所示，《屠魔》的游戏界面如图 2.4 所示。



图 2.3 《英雄无敌 4》游戏截图



图 2.4 《屠魔》游戏截图

2.1.3 益智解谜类

益智解谜类小游戏（PUZ 也称为 PZL, Puzzle Game）。这类游戏品种繁多，老少皆宜，

是最适宜手机上玩的游戏，比如《俄罗斯方块》、《黑白棋》、《贪吃蛇》等。它们对显示、操作、声音的要求都不高，非常容易在目前的手机上实现。

这类游戏最大的优点就是体积一般较小，风格以绿色休闲为主，适用人群范围很广。手机上这类游戏的代表有《挖地雷》、《变幻线》、《推箱子》等。《挖地雷》界面如图 2.5 所示。



图 2.5 《挖地雷》游戏截图

2.1.4 即时战略类

即时战略类游戏（RTS，Realtime Strategy Game）。对应回合制战略类游戏，这类游戏中的一切都是实时发生的，要求玩家具备较好的敏捷性与宏观指挥能力。此类游戏自其开山鼻祖《沙丘》以来，就以其独特的生产和战斗模式以及掌握在游戏者手中的高度自主权而受到了广大游戏爱好者的推崇。

后来，随着多部同类游戏的面世，该类游戏也得到了不断的完善。例如，从 2D 画面向 3D 画面的过渡、随机即时生成地图、多样的兵种和武器、合理利用有限的资源以及如今风行的联机对战模式等，使“和人斗，其乐无穷”成为文明的对决。

《帝国时代》、《魔兽争霸》、《星际争霸》是这一类游戏当中的代表作。手机上这类游戏的代表作有《炸弹王》、《海商王 2》、《家园》等。《魔兽争霸》的游戏界面如图 2.6 所示，《海商王 2》的游戏界面如图 2.7 所示。



图 2.6 《魔兽争霸》游戏截图



图 2.7 《海商王 2》游戏截图

2.1.5 动作格斗类

动作类游戏（ACT，Action Game）。这类游戏提供给玩家一个训练手眼协调及反应力的环境和功能，通常要求玩家所控制的主角（人或物）根据周围情况变化做出一定的动作，如移动、跳跃、攻击、躲避、防守等，来达到游戏所要求的效果。

此类游戏讲究逼真的形体动作、火爆的打斗效果、良好的操作手感及复杂的攻击组合等。

手机上这类游戏的代表作有《月球反击战》、《空军大战》、《坦克战》等，《月球反击战》的界面如图 2.8 所示。



图 2.8 《月球反击战》手机版游戏截图

格斗类游戏（FTG，Fighting Game）是从动作类游戏分化出来的，指两个角色一对一决斗的游戏形式。现在，此类游戏又被分化为 2D 格斗类游戏与 3D 格斗类游戏。

平面格斗类游戏的代表作有《街霸》系列和《拳皇》系列，3D 格斗类游戏的代表作是《VR 战士》系列和《铁拳》系列。手机上这类游戏的代表作有西门子公司公司的《BattleMail》（港译《功夫小子》）、日本的《街头霸王》移动版及索尼公司的摔跤游戏《斗魂列传》等。《拳皇》的游戏界面如图 2.9 所示。



图 2.9 《拳皇》手机版游戏截图

2.1.6 策略类

策略类游戏（SLG，Strategy Game）。这类游戏提供给玩家一个可以进行逻辑思考和策略、战略运用的环境，且让玩家有自由支配、管理、统领游戏中的人、事、物的权力，并通过对这种权力及谋略的运用达成游戏所要求的目标。

策略类游戏在某些方面与回合制战略类游戏有些相似，不同的是策略类游戏具有一个贯穿全局的时代或时期，而不像回合制战略游戏那样由一个个的片段来推动游戏的发展。玩家在条件真实、气氛宏大的游戏环境中充分施展智慧，克敌制胜，达到高层次的成功享受。

该类游戏中的经典是日本光荣公司旗下的 5 大代表作：《大航海》系列、《三国志》系列、《信长野望》系列、《太阁立志》系列和《大战略》系列。手机上这类游戏的代表作有《梦幻模拟战》、《龙与吸血鬼》、《垄断大亨》、《梦回唐朝》等。《太阁立志传 5》的游戏界面如图 2.10 所示，《垄断大亨》的游戏界面如图 2.11 所示。



图 2.10 《太阁立志传 5》游戏截图



图 2.11 《垄断大亨》游戏截图

2.1.7 冒险类

冒险类游戏（AVG，Adventure Game）。这类游戏在某一固定的剧情或故事下，提供给玩家一个可解谜的环境及场景，玩家必须随着故事的安排进行解谜。该类游戏的目的是借游戏主角在故事中冒险所积累的经验来解开制作者设定的谜题或疑点。

这类游戏常被设计成侦探类型的解谜游戏。冒险类游戏具有较强的逻辑推理性，是用玩家的智力挑战游戏中设下的各种类型障碍。利用游戏提供的线索，玩家在不断的探索中去发现最后的秘密，可以使自己的好奇心获得极大的满足。

此类游戏的代表作有《生化危机》系列、《神秘岛》和《寂静岭》系列。手机上这类游戏的代表作有诺基亚公司的《speed》、《超级马里奥全明星》等。《超级马里奥全明星》的游戏界面如图 2.12 所示，《生化危机 3》的游戏界面如图 2.13 所示。

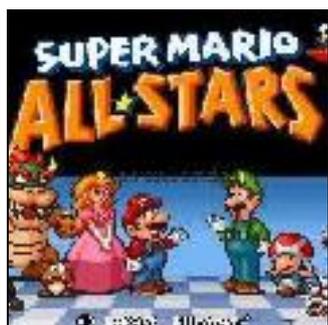


图 2.12 《超级马里奥全明星》游戏截图



图 2.13 《生化危机 3》游戏截图

2.1.8 射击类

射击类游戏（STG，Shooting Game）。这类游戏主要是指依靠远程武器，与敌人进行战斗，一般就是指飞机类型的游戏。该类游戏又分为平面型射击类游戏和 3D 型射击类游戏。

平面型射击游戏的代表作是《雷电》系列，3D 型射击游戏的代表作是《王牌空战》系列。此类游戏还包含光线枪游戏，代表作是《VR 战警》。手机上这类游戏的代表作有《核子战车》、《鬼影战机》等。《雷电 3》的游戏界面如图 2.14 所示。



图 2.14 《雷电 3》游戏截图

2.1.9 体育类

体育类游戏（SPG，Sport Game）。该类游戏内容比较广泛，但主要的还是足球、篮球、棒球等球类游戏，其他体育项目做成游戏的较少。不过，随着时代的进步，一些非主流体育项目，如今也被加入到体育游戏中来，如钓鱼、打猎等。

所有的体育游戏都有一个趋势：即，将球员的真实身份和资料完整地搬到游戏中去，并对每一个球员的各项技术指数做了量化处理。体育明星的号召力使得游戏更能吸引玩家。

这类游戏代表作有《实况足球》、《NBA》系列。手机上这类游戏的代表作有苏格兰公司 DigitalBridges 推出的《幻想足球世界》，此外，还有《高尔夫》、《篮球》、《乒乓球》、《网球》等。《实况足球 10》的游戏界面如图 2.15 所示。



图 2.15 《实况足球 10》游戏截图

2.1.10 模拟养成类

模拟养成类游戏（EDU，Education Game）。此类游戏是少有的、适合女孩子玩的游戏，它交给玩家治理的不再是一个国家或一个行业，而是一个活生生的人。玩家将一段设定的时间内照顾并规划这个孩子的衣、食、住、行，或者为这孩子选定好固定的目标，再用游戏中的指令去实现该目标。

这是一个充分表现玩家爱心的游戏类型。一般来说，此类游戏多为多种结局，所以孩子的发展方向更是五花八门，代表作品有《美少女梦工厂》系列。手机上这类游戏的代表作有《海底总动员》、《农场 CEO》等。《农场 CEO》的游戏界面如图 2.16 所示。



图 2.16 《农场 CEO》游戏截图

2.2 手机游戏的特色分类

上一节介绍了目前主流游戏的分类，这主要是从游戏本身角度来区分的。针对手机游戏这一特殊群类，又有其本身的特色分类。总的来说，手机游戏可以分成文字类游戏和图形类游戏两大类（当然，不排除还有 IVR 游戏这样的特殊类型，下面会有专门介绍）。

文字类游戏是以文字交换为游戏形式的游戏。这类游戏一般都是通过玩家按照游戏本身发给玩家手机的提示，回复相应的信息来进行的。文字类游戏主要包括短信游戏和 WAP 浏览器游戏。

图形类游戏则更接近我们熟悉的“电脑游戏”，通过动画的形式来发展情节进行游戏。由于此类游戏采用了更为直观且更为精美的画面直接表现，因此其游戏性较高，也更受玩家的欢迎。图形类游戏主要包括嵌入式游戏、Java 游戏、Brew 游戏、Uni-Java 游戏，这些游戏也可以统称为终端游戏。

2.2.1 短信游戏 (SMS)

短信被用来从一个手机向另一个手机发送简短的文字信息。短信游戏的玩法通常是：发送一条信息到某个号码，这个号码对应游戏供应商的服务器，服务器接收这条消息，执行一些操作，然后返回一条带有结果的消息到玩家手机。

短信游戏的整个游戏过程都是通过文字来表达的，所以短信游戏的娱乐性较差，但它却是兼容性最好的手机游戏之一。只要玩家的手机可以发短信，就可以畅快地享受短信游戏带来的快乐。

从本质上来说，短信是一个命令环境。短信游戏与其说是个游戏，倒不如说是个交友社

区，因为不管一款短信游戏最初的立意是什么，到最后总会慢慢演变成交友类的大型聊天室。目前，随着其他手机游戏的发展，短信游戏正在走向低谷，但彩信技术的推出，无疑将使短信游戏迎来又一个春天。短信游戏的界面如图 2.17 所示。



图 2.17 短信游戏截图

中国的游戏开发商 MiG 公司已经与诺基亚公司合作，将他们的 LUVM8 成功移植到彩信平台。LUVM8 是一种类似电视约会的短信游戏，玩法是：一组男生竞相去获得一个女性玩家的注意，或者相反；每一回合中，玩家会被问一些约会方面的问题。如果竞争者提供的答案与他或她所追求的目标玩家的答案一致，那么他们的适配分数就会增加。游戏的目的在于，获得最高分的玩家将最终成为目标玩家的“LUVM8”。彩信版在此基础上加上了动画和声效。

2.2.2 WAP 游戏

WAP 是“wireless application protocol（无线应用协议）”的英文缩写。

1997 年夏，爱立信、诺基亚、摩托罗拉和 phone.com 等通信业巨头发起了 WAP 论坛，目标是制订一套全球化的无线应用协议，以使互联网的内容和各种增值服务适用于手机用户和各种无线设备用户，并促使业界采用这一标准。

1999 年之后发售的手机大多包含一个 WAP 浏览器。所谓的 WAP 浏览器就像是简化版的 Web 浏览器，但它是静态的且支持的功能非常少，但由于其对移动设备的优化，很适合在手机这种低带宽设备上使用。

WAP 游戏的方式就是用手机访问游戏提供商的 WAP 站点，下载并浏览 WAP 页面，然后在页面的菜单上做选择，操作产生的数据将被发回 WAP 服务器，经过处理之后再发送给玩家更多的页面。

对用户来说，WAP 游戏的表现力强于短信，因为 WAP 能显示颜色和图片。但它仍然只是个静态的浏览器，在客户端几乎无法做任何处理工作，它所有的游戏运算必须依靠网络将数据传输到服务器，服务器计算结果之后再通过网络将其返回。同短信游戏一样，WAP 游戏也有不够直观的缺点。《大航海时代》WAP 版的界面如图 2.18 所示。



图 2.18 《大航海时代》WAP 版的游戏截图

据统计，目前国内免费 WAP 站点中，同时在线人数最高的

社区为 3G 泡泡免费门户 (3gpp.com.cn) 旗下的手机泡泡吧 (wapp.com), 该社区同时在线人数平均在 25 000 人左右。“WAP 天下” 号称 WAP 网站总在线用户 5 000~6 000 人, 其他站点的在线用户百人到千人不等。这些数字相比于 PC 网游几十万的数值虽然还有差距, 但是已经彰显出 WAP 游戏的强大生命力和市场。

2.2.3 IVR 游戏

IVR 是 “Interactive Voice Response (交互式语音应答)” 的英文缩写, 它是对基于手机的无线语音增值业务的统称。手机用户只要拨打指定号码, 就可根据操作提示收听、点送所需语音信息或者参与聊天、交友等互动式服务。

显然, IVR 游戏指的就是通过语音应答进行的游戏, 这类游戏的典型代表是足球射门游戏——根据提示选择数字按键进行“射门”, 若射中则有奖品。由于这类游戏趣味性不是很强, 又需要一定的资费, 单纯依靠奖品提供的吸引力显然不能满足玩家的需要, 因此其前景不是很乐观。IVR 游戏《声色剧场》的游戏界面如图 2.19 所示。

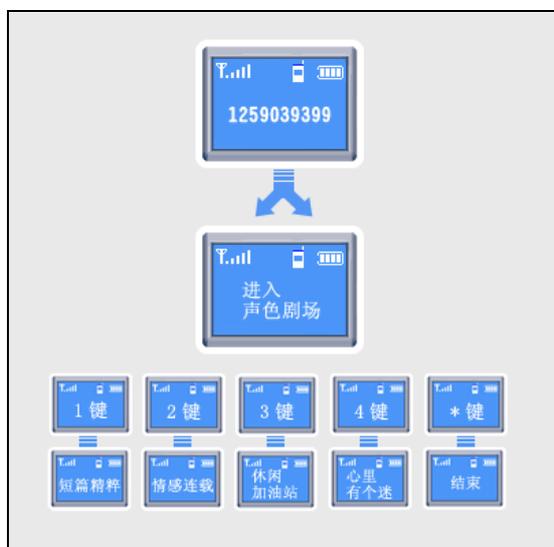


图 2.19 《声色剧场》的游戏截图

2.2.4 终端游戏

很难对终端游戏下一个定义, 一般的手机游戏分类中也不出现这个术语。这里之所以这样叫, 主要是为了区别于前述 3 种手机游戏, 也就是说, 除了通过短信、语音、浏览器进行的, 其他以手机作为终端设备进行的游戏都可以归为此类。

这类游戏类似于传统的电视电脑游戏，一般都以图形化界面出现，它是当前和未来手机游戏发展的主流。

目前，这类游戏主要分为：嵌入式游戏、Java 游戏、Brew 游戏和 Uni-Java 游戏。

1. 嵌入式游戏

嵌入式游戏是一种将游戏程序预先固化在手机芯片中的游戏。由于这种游戏的所有数据都是被预先固化在手机芯片中的，因此，对这种游戏无法进行任何修改。也就是说，既不能将其更换成其他的游戏，也不能将其删除。

诺基亚早期手机中的《贪吃蛇 1》、《贪吃蛇 2》就是嵌入式游戏的典型例子。虽然目前嵌入式游戏也发展到了相当高的技术层面，但由于游戏的不可更改和删除，使得该类游戏的发展空间大大缩小。《贪吃蛇》的游戏界面如图 2.20 所示。



图 2.20 诺基亚早期手机中的《贪吃蛇》游戏截图

2. Symbian 游戏

Symbian 是由诺基亚、摩托罗拉、索尼爱立信等国际一流大厂共同投资成立的一家公司，专门致力于智能手机操作系统 Symbian 的研发、生产和销售。

目前，诺基亚是 Symbian 最大的股东，也是其最主要的支持者，诺基亚的高端机型（以 S60 为代表）都是基于 Symbian 操作系统的。目前 Symbian 手机在智能手机方面占据了 60% 以上的市场份额。

因为 Symbian 手机功能强大，所以也出现了大量的 Symbian 游戏，其制作之精良、效果之眩目、设计之复杂、规模之宏大都堪比专业的主机游戏和 PC 游戏，可玩性非常高。Symbian 游戏是目前手机上最高端的游戏类型，也是发展最快的游戏类型。Symbian 游戏《贪吃蛇》的界面如图 2.21 所示。



图 2.21 诺基亚 S60 手机中的《贪吃蛇》游戏截图

3. Windows Mobile 游戏

Windows Mobile 是微软公司推出的手机 Windows 操作系统。凭借微软公司在操作系统和软件方面的强大实力，Windows Mobile 系统推出短短数年，便取得了非凡的成就，其市场占有率从 0 飚升到 20% 以上。

客观地说，Windows Mobile 操作系统的功能非常强大，而且对于开发者而言也更加方便，毕竟微软在操作系统和开发工具领域的实力是其他任何人都无法比拟的。

虽然目前 Windows Mobile 系统的出货量还不小，但由于有摩托罗拉、三星等一流厂商的支持，加上微软公司的鼎力推进，其前途不可小觑。对于广大游戏开发者而言，Windows Mobile 的开发环境最好，因为它和 Windows 系统开发类似，所以也不失为一个入门的好选择。Windows Mobile 游戏《反恐精英》的界面如图 2.22 所示。



图 2.22 Windows Mobile 游戏
《反恐精英》的游戏截图

4. Java 游戏

Java 是 Sun 公司开发出的一种计算机编程语言，最初是为嵌入式系统而设计的一项产品。为了区分各种不同的应用，在 Java 2 中又细分成了 Java 2 Enterprise Edition (J2EE)、Java 2 Standard Edition (J2SE) 和 Java 2 Micro Edition (J2ME) 3 种版本。

其中，J2ME 最早是在 JAVAONE 大会上被正式提出来的，是 Sun 公司专门为小型的、资源受限的消费性电子设备的应用程序开发而提供的新 Java 版本，该版本被广泛应用于移动电话、PDA、机顶盒等小型资源受限设备中。因为它的程序量仅为 K 字节的 Java 程序，所以又被称作 K-Java。

目前市场上，K-Java 类游戏占据了终端游戏的主流：一方面这类游戏代码短小精悍，占用内存不多；另一方面也是由于其进入市场早，基本已经成为通用的手机游戏开发模式。因此，这类游戏是未来手机游戏发展的趋势。Java 游戏《剑仙缘》的界面如图 2.23 所示。

5. BREW 游戏

和 Java 类似，BREW 也是一种程序语言。目前，只有 CDMA 的手机才支持 BREW。但是同时，CDMA 也支持 Java，于是为了减小成本，一般的开发商还是愿意选择基于 Java 的游戏进行开发。因此，BREW 支持的游戏还不是很多。随着手机游戏市场的进一步扩大，这类游戏的发展空间是很大的。联通 BREW 平台游戏《冒险岛》的界面如图 2.24 所示。



图 2.23 Java 游戏《剑仙缘》的截图



图 2.24 联通 BREW 平台游戏《冒险岛》截图

6. Uni-Java 游戏

Uni-Java 是中国联通为其手机准备的一个新的通用开发平台。相比于 BREW 游戏来说，它更接近于 Java 游戏的升级版。虽然现阶段来看，这类游戏受关注的程度略低于前述几个类型的游戏，目前还处于开发阶段，还没有基于 Uni-java 的手机推出，但其作为新生事物，同样不可忽视。也许不久的将来，这类游戏会和现在的 Java、BREW 游戏一样，在手机游戏市场上分一杯羹。

7. Android 手机游戏

Android 是谷歌公司于 2007 年 11 月 5 日发布的、由其自己研发的手机平台操作系统。该平台基于开源软件 Linux，由操作系统、中间件、用户界面和应用软件组成，号称是“首个为移动终端打造的真正开放和完整的移动软件”，其界面如图 2.25 所示。



图 2.25 支持 OPENGL 的 Android 操作系统

2.3 游戏设备的特点

PC 游戏刚发展起来的时候，内存和操作系统都会成为限制因素。而随着技术的发展，以后的手机在操作系统和内存上都将优化，且手机游戏的数据量并不大，所以这两项不会成为手机游戏发展的主要障碍。

与此不同，值得注意的是，手机在游戏画面的大小方面是永远都比不上 PC 机的。另外，

因为手机移动携带的缘故，也无法设置与 PC 机一样的鼠标工具。

总的来说，手机游戏与 PC 游戏有共性，也有其自身的优势和缺陷。这就决定了，手机游戏设计在尽可能保证与 PC 游戏接近的可玩性的同时，必须要扬长避短，兼顾自身的特性。

2.3.1 手机作为游戏设备的限制

1. 屏幕小

这是手机作为游戏载体永远无法克服的问题。虽然屏幕分辨率持续提高，并且彩屏成为标准，但是屏幕尺寸还是一直很小，因为没有人乐意拿着砖块一样大的手机。

还有一个相关的问题：不同手机的屏幕大小是不同的。比如说，Nokia Series 60 平台设备就提供了和 Nokia 5100 这样的 Series 40 设备不同的屏幕尺寸。

虽然各个厂商已经标准化了其产品的屏幕尺寸以避免分割市场，但是开发者仍然需要为不同的手机屏幕优化他们的游戏——玩家肯定想使用特定手机上所有可用的屏幕空间。

2. 有限的颜色和声音支持

几年前，大部分使用者手中的手机仍然是黑白的，而现在出售的手机几乎都是彩屏，应该说基本符合了大多数游戏的需求。但是，和现在华丽的 PC 游戏相比，显然手机的颜色和分辨率还遥不可及。

即使手机本来就有声音设备，但是应用程序播放声音的能力却非常有限。

J2ME 规范根本不需要硬件厂商支持声音。虽然基本的 Java 手机允许使用一些声音，并且 MIDI 支持正在成为标准，但对于高要求的玩家或专门的音乐类游戏来说，这些都远远不够。而且，通常手机中只有一个语音或一个声道可用。

3. 应用程序大小限制

大部分的 Java 手机只有很少的内存空间用于运行应用程序。在这样的限制条件下设计开发移动游戏固然非常困难，但是要知道，第一台家用电脑只有 64 KB 内存，但仍然有人热衷于在其上开发游戏软件。

一些智能手机上的内存限制就少一些，比如 Nokia 3650，它甚至可以运行几兆字节的应用程序。

4. 高等待时间

等待时间（机器发出请求和接到响应之间所花费的时间）在计算机上是以微秒计算的，在有线因特网上是以毫秒计算的，而在无线网络则要以秒计算。等待时间是网络游戏中一直存在的一个问题，开发者们总是在努力消除它带来的问题。

无线网络的等待时间非常长，因而不可能有效地开发多人快速动作移动游戏，然而基于回合制的多人游戏是相当可行的。

2.3.2 手机游戏设备的优势

手机的便携、小巧、随时随地等优点为游戏的发展提供了一个新的发展方向。对于游戏开发商来说，手机彩屏、和弦等功能的普及以及手机智能化趋势的发展，为手机成为游戏的载体提供了条件。

手机作为游戏载体的广泛普及，不但有利于游戏品牌的推广，更能够为其带来实际的经济收益。

1. 庞大的潜在用户群

现在，全球超过十亿部移动电话正在被使用，而且这个数目正在逐渐增加。在除美国之外的每个发达国家，拥有手机的人数比拥有计算机的人数更多。虽然那些手机只有一小部分是支持 Java 的，但是这个数目正在快速地增长，并且在几年内，Java 手机将要成为行业标准。移动游戏潜在的市场比其他任何平台（如 Playstation 和 GameBoy）都要大。

2. 便携性

GameBoy 比任何其他控制台游戏卖得多的一个原因就是：便携性。人们可以随时随地玩他们选择的的游戏。与现在的游戏控制台或个人电脑相比，手机可能不是一个好的游戏设备，但是人们基本上随时随刻都把它们带在身边。

外出旅游、超市排队付账甚至冬天躲在被窝里时，手机游戏都将成为其他游戏设备无所企及的首选。

3. 支持网络

因为移动电话是网络设备，所以可以实现多人游戏。就目前来说，互动类游戏与单机游戏相比其优势是巨大的，而且随着通信技术的提高，这种趋势还在进一步扩大。因此，支持网络互联成为手机战胜单机游戏设备的又一重要砝码。

虽然有某些限制因素，比如无线网络相对于有线因特网的“低速”，但是不可否认，其未来的发展势头是强劲的。

2.3.3 设计实现中的扬长避短

游戏有惊人的可塑性，它们可以通过使用从石器时代到现代高科技的每一种技术来实现。当使用以前从来没用过的技术进行开发时，需要了解其优势和局限，在努力将其优势发

挥到极限的同时，回避或解决其局限性。

从手机游戏的优势和局限性的讨论中可以得出以下结论。

1. 短的游戏时间

人们迟早要打电话或接电话，他们不想把所有的电量都用来玩游戏。另一方面，当用户接听电话的时候，手机都会中断进行中的游戏。

理想的情况是，每一回合游戏应该保持在五分钟或更短时间之内，并且要求手机游戏程序必须能够暂停、保存和继续，从而不会造成游戏问题和内存溢出。因此，在设计手机游戏时要注意控制好时间，并且处理好游戏中断时发生的事情。

2. 使用网络，避免等待

网络不一定对于每个移动游戏都是必需的，但是那种与人竞争的感觉，即使只有一个排行榜，也能吸引更多的玩家。要记住，手机实际上是一种社会性设备，添加某种社会性因素到游戏中将会使它备受欢迎。

另一个值得注意的方面是避免等待。单机游戏不存在这样的问题，但在多人游戏中，这就是设计时需要解决的——没人愿意等待 3 分钟然后用剩下的 2 分钟看个片头动画。

3. 尽可能地让游戏保持小型

记住一点，人们仍然热衷于 20 世纪 80 年代的优秀游戏。在某些方面，技术的限制会强迫开发者把更多的注意力放到基本的游戏中去。而且小游戏并不代表游戏性的弱小，如果设计合理，即使是最简单的《贪吃蛇》和《推箱子》，也能成为百玩不厌的好游戏。

4. 支持多种手机和多种语言

至少，需要支持不同的屏幕尺寸，这对诺基亚的系列手机很容易做到——为 Series 40 开发一种版本，为 Series 60 开发另一种版本。

另外，全世界都在使用移动电话，每一种语言都有自己的市场。应在开发设计前做好计划，弄清楚这款游戏的市场所在——千万不要把在美国开发的全英文游戏卖到日本去。

2.4 无线游戏蓝图

当第一款手机游戏在诺基亚手机上诞生的时候，谁也没想到今天我们可以在智能手机上进行真正的无线网络游戏。2006 年，我国移动手机用户已经超过了 4 亿，再加上手机市场的持续火爆，无线手机游戏市场这一块蛋糕正在一点点地长大，一步步地趋于成熟。

来自众多研究公司的结果似乎都在表明：无线游戏将是下一个因特网金矿。Ovum 指出，到 2006 年，全球移动游戏业的价值将是 44 亿美元。Frost & Sullivan 研究公司指出，全球移

动游戏业收入在 2008 年将上升至 93.4 亿美元。

如今，从高档写字楼到青青校园，手机无处不在，已经成为都市时尚文化和消费主义的象征。起初，手机只是用来通话，后来可以发短信，而现在当人们空闲无聊的时候，手机能提供游戏和娱乐服务的功能也就显现出来了。

随着 2.5G 和 3G 的大发展，彩屏手机将会逐步取代黑白手机，以后几年的时间里，彩屏手机将具有类似 GameBoy 游戏机的功能。更重要的是，随着数据业务的成熟，寄托于移动通信网络与移动终端的无线游戏，使手机游戏由单机版迅速过渡到了类似于电脑网络游戏的时代。

无线游戏既继承了手机离线游戏即开即用、操作简便和便携的特点，又进一步被赋予了网络游戏人人交互的特点，而显得更具挑战性、刺激性和真实感，因此迅速成为目前市场上的中坚力量。

手机无线游戏的巨大市场魅力吸引了商家的目光。曾经创立 ChinaRen 的周云帆和杨宁也推出了空中网，开始拓展手机无线游戏市场。炎黄新星总裁钱中华也非常看好无线游戏的发展前景，他认为“短信游戏是一座金矿”。Magus soft 执行总裁卢勇则专注于游戏与无线互联网络的结合点，称“无线手机游戏将给中国开发商带来更好的机会”。

当然，手机无线游戏这张蓝图还有缺陷，总的来说分为以下两方面。

1. 技术问题

追根溯源，无线游戏开发商是瓶颈。开发商本身的能力与水平将成为无线手机游戏成长的关键，因为无线手机游戏现在的技术并不成熟。

另外，由于手机外形限制，屏幕不可能像 GameBoy 或 Xbox 那样适用，这就必然限制了手机游戏的活形活色。还有，若要手机的通话耳机也让手机游戏有声有色，就必然对网络游戏开发商的行业技术门槛要求更高。

据预测，在中国如果能发展很好，50~100 家游戏厂商并存不成问题。

2. 运营服务

如果网络游戏大发展，用户们最担心的问题莫过于移动运营商的综合服务，而移动运营商占据了整个价值链的枢纽地位，在手机游戏无线化的过程中理所当然地起着决定性作用。

如果要手机无线游戏得到更进一步的发展，就需要移动运营商推出统一的无线游戏标准、统一的无线游戏操作平台。运营商计费平台的建立非常重要，毕竟无线手机游戏的最大推动力来自于运营商，运营商的决策将会最大限度地引领开发商的发展方向。

另外，移动运营商的信号提供也是个在手机无线网络游戏中急待解决的问题，虽然中国移动号称具有中国覆盖最广的移动电话网络，但由于游戏的数据量相当大，已不再是几条短信所能比拟的，所以当在线用户达到一定数量时就会不堪重负，这也是制约无线游戏发展的重要因素。

要想在手机无线网络游戏市场上盈利，游戏开发商的游戏内容就必须贴紧用户，以用户需求为主，这就必然要求对无线游戏市场进一步地细分，根据不同消费需求的用户群开发游戏。手机无线游戏的未来是美好的，手机无线游戏也必然会走进我们的生活。

第 3 章

手机游戏产业中的职业规划

3.1 手机游戏项目管理的特点

手机游戏项目的开发与普通软件项目开发相比有很大的不同，主要表现为如下几个方面。

(1) 项目开发周期长。

游戏项目的开发周期一般在半年到一年，很多游戏的开发时间在一年以上。

(2) 涉及环节多。

游戏的开发涉及策划、美工（2D、3D）、程序、测试等诸多环节，特别是在资源调度上，难度很大。

(3) 求变化，多而快。

游戏是个需要和市场非常贴近的项目，市面上的游戏层出不穷，玩法推陈出新，如果不能及时赶上变化，往往在游戏推出时，就已经落后于主流游戏。因此，项目在进行过程中，经常需要根据市场变化更改需要。

这些不同给手机游戏项目经理提出了新的要求。

3.1.1 职业要求

项目管理的职业要求如下所示。

(1) 掌握项目管理相关专业知识体系，并能在实践中运用。

(2) 能够有效地把握客户需求，有很强的项目管理能力和客户引导沟通能力。

(3) 具备良好的团队精神和很好的跨部门项目管理经验（例如对程序员、美工、策划三方面的协调）。

(4) 具有成熟的项目协调执行能力和项目推进能力，能控制项目的开发时间，保障项目开发的进度。

(5) 在符合客户需求的基础上，保障项目质量。

(6) 有较强的文字功底、独立工作能力和学习能力。

3.1.2 常用工具

文本处理工具 Microsoft Office Word 是最常用的一种文字处理软件，由微软公司开发，使用它可以很方便地进行文字、文本的处理。在开发过程中，一些开发文档、说明书的撰写都需要使用 Word。

幻灯片制作工具 Microsoft Office PowerPoint 和 Word 一样，也是由微软公司开发的。它是一种幻灯片制作工具，用该工具进行产品功能的演示可以达到更好的效果。

Bug 管理工具 Mantis 是一个基于 PHP/MySQL/Web 的开源的错误追踪系统。目前其最新版本是 0.18.0rc1，安装要求 PHP 版本为 4.0.3 或更高，MySQL 版本为 3.23.2 或更高。在产品开发过程中，它以 Web 操作的形式提供项目管理和缺陷跟踪服务，能对开发过程中的 Bug 进行系统的记录。

项目管理工具 Microsoft Project 是国际上最盛行的基于网络和团队协作的项目管理软件，在 IT、工程、研发、制造、设计、广告、石油、水电等行业领域均有很好的应用。

Microsoft Project 被广泛应用于各类 IT 集成项目、采用 EPC 模式的建设工程项目，以及企业的技改和研发项目、企业的运营项目和订单项目、政府或集团的投资和多项目的管理。它能够很好地在产品开发的过程中对项目的进度进行管理，交流项目状态，以及报告项目信息。

版本控制工具 Subversion 是新一代的版本控制工具，不仅可以用于管理程序源代码，也可以应用于其他协作管理数据的工作，它能够在开发中保留下各个版本的记录，对程序的修改、调试有着很大的帮助。

3.1.3 职业建议

对项目管理人员有以下建议。

(1) 游戏项目需求管理是项目成功的关键也是难点，这种需求来自玩家的实际需求或公司自身发展和实力的情况。其中，玩家的需求即市场需求最为重要，因此必须做好市场调研并编写调研文档，以作为日后项目开发过程中的依据。

(2) 游戏项目的质量管理将会因涉及面多而显得相对复杂。要管理好美术、策划、程序三个方面，各方面的质量评测方法都不尽相同，这就需要项目管理人员掌握多方面的技能。

(3) 游戏开发项目比较难以控制预算，因此在项目立项的时候要考虑多方面的因素，特别是需求变化所带来的风险。在制定预算的时候，应该留出部分预算灵活使用。在总预算的基础上，应将项目分成若干个阶段，并根据不同阶段制定相应预算。

3.2 美术——方寸之间尽显本色

游戏美工主要负责制作美术资源。对于一款手机游戏而言，程序构建其骨骼，策划赋予其灵魂，而美工负责的是它的外表。就如同女生爱美，男生爱酷，手机游戏留给人的第一印象就是画面的品质。

现代游戏开发需要大量的美工资源，小到一个光标，大到整个屏幕，都需要游戏美工绘制、修改甚至全部重做。因此，游戏美工是一个极富创造性并且非常辛苦的工作。

3.2.1 职业要求

游戏美工的职业要求如下。

- (1) 具有丰富的美术专业知识，有良好的创意思维和理解能力，有良好的文化底蕴。
- (2) 具有良好的手绘能力和色彩感觉，能独立设计人物、场景及动画，具有较好的造型能力。
- (3) 熟悉游戏文化与主流动漫，喜爱 PC 游戏、网络游戏、手机游戏，对游戏的各个系统有一定的了解，并能把握玩家的心理。
- (4) 学习能力强，具有良好的沟通能力和团队协作精神。
- (5) 熟练掌握各种主流图形图像软件和 3D 设计制作软件。

3.2.2 常用工具

图形图像处理工具 Adobe Photoshop 是一款功能极其强大的图像编辑软件，应用它可以很轻松地进行图形图像的处理。可以对图像做各种变换，如放大、缩小、旋转、倾斜、镜像、透视等；也可进行复制、去除斑点、修补、修饰图像的残损等；还能进行图像的合成，即将几幅图像通过图层操作、工具应用合成为完整的、传达明确意义的图像。

图形图像设计工具 Adobe Illustrator 是 Adobe 系统公司推出的基于矢量的图形制作软件，与 Adobe Photoshop 无缝连接。它能为线稿提供无与伦比的精度和控制，适合生产任何小型到大型的复杂项目。

图像动画制作工具 Adobe ImageReady 是由 Adobe 公司开发的，以处理网络图形为主的图像编辑软件。ImageReady 与 Photoshop 之间可以进行图片的同步操作（即同时对一个图片进行处理）。只要在 Photoshop 中的工具箱下方单击图标就可以跳转到 ImageReady 界面，同样，在 ImageReady 中单击这个图标也可以进入 Photoshop 环境。

Web 动画制作工具 Macromedia Flash 是由美国 Macromedia 公司设计的一种二维动画软件，用于设计和编辑 Flash 文档。可以在程序中使用此工具增添一些 flash 动画。

网络图形编辑工具 Macromedia Fireworks 是 Macromedia 公司发布的一款专为网络图形设计的图形编辑软件，它大大简化了网络图形设计的工作难度。

无论是专业设计者还是业余爱好者，都可以很轻松地使用 Fireworks 制作出十分动感的 GIF 动画，还可以轻易地完成大图切割、动态按钮、动态翻转图等操作。

三维建模、动画制作工具 Alias Maya、AutoDesk、3ds MAX 都是当今世界上极其优秀的三维制图软件，使用它们可以在游戏中建立各种三维模型及场景，并可通程序序设计来制定它们的动作，以达到动画的效果。

3.2.3 职业建议

游戏美工的职业建议如下。

(1) 当前手机游戏开发以 Java 为主，像素级画面的游戏是主流。绘制像素级图片就是美术人员所要掌握的基本素质要求。

(2) 和绘制普通的像素画不同，绘制手机游戏的像素图片要求美工更加具有数学头脑。因为受到性能的限制，一般手机游戏的容量都是很小的，还要去除程序所用的部分，因此留给美工发挥的空间很有限。在有限的空间内完成一个精彩的游戏世界，就需要美工在绘制图片时严格地计算和把握图片的大小，这时的美工可以说更像一个头脑缜密的工程师。

(3) 作为游戏美工，在游戏制作期间与策划、程序能很好地交流也非常必要。游戏制作是一个集体的项目，要求开发团队中的每一个人都要善于交流。在游戏的制作过程中，美工要完整体现策划中所要表达的各种图片要求，配合编程人员实现各种效果，就需要与其他人员有相互的交流。

3.3 策划——有限的空间、无限的任务

在策划一个游戏之前，手机游戏策划人员首先要清楚现在市面上流行什么游戏，玩家希望玩什么样的游戏。根据玩家需要把游戏性质定出来，然后搭出一个基本的框架，再把游戏的情节、人物、场景及每个细节都设置好，最后交给技术组来实现。

从游戏的立项、草案、开发计划，到实际的工作安排、数据处理、流程编写、系统架构、公式算法，游戏开发过程中的每一步都需要策划的参与。

3.3.1 职业要求

游戏策划的职业要求如下。

(1) 热爱游戏，了解各种类型游戏，对游戏设计有一定的见解，对网络游戏、手机游戏的各个系统有深刻的认识。

- (2) 熟悉游戏制作流程，能很好地把握玩家的心理，对游戏市场有深入了解和敏锐眼光。
- (3) 具有丰富的想象力与创造力，学习能力强，善于接受新思想。
- (4) 对程序、美术、音乐、电影、漫画有一定了解，具有良好的文学功底，有良好的沟通能力和团队协作精神。
- (5) 有一定的电脑使用基础，熟练掌握文字处理等常用办公软件。

3.3.2 常用工具

手机游戏策划人员除了要使用 Microsoft Office Word 和 Microsoft Office PowerPoint 之外，还要使用流程图制作工具——Microsoft Office Visio。Microsoft Office Visio 是所有软件设计者必不可少的工具，可以用它制作的流程图包括电路流程图、工艺流程图、程序流程图、组织结构图、商业行销图、办公室布局图、方位图。

3.3.3 职业建议

游戏策划的职业建议如下。

(1) 在进行手机游戏开发之前，一个游戏策划应该先注意什么呢？必须先注意的不是游戏本身，而是游戏开发的环境，这个环境包括游戏开发平台、游戏开发的搭档、游戏开发的周期、公司的预期目标等。在没有了解清楚这些之前，根本不要想动手去做游戏。

(2) 游戏策划首先得是游戏玩家，但是，做游戏和玩游戏不是同一个概念。玩游戏的只把游戏当艺术，做游戏的必须要把游戏当技术，否则就不可能做出好游戏。这是一个心态的问题。如果不把自己的身份角度纠正过来，始终抱着只要把游戏玩好就能把游戏做好的观念做游戏，那么即使参与的游戏项目再多，也可能只是走走过场学不到任何东西。

(3) 作为一个合格的策划，不但应该全面地了解程序和美工的制作流程和基本技术，更应该学会站在程序和美工的立场思考问题。因为只有这样才会知道自己的想法最终能在游戏中如何实现，以及这样实现以后会带来怎样的问题，才会对自己的想法有更深入思考。

3.4 程序，万丈高楼平地起

正如前面所提到的，策划赋予游戏灵魂，美工装饰其外表，那么程序员要做的，就是搭起整个游戏的骨架。别说没有骨架，即便稍微缺一点少一点，整个游戏就会垮掉，由此可见程序的重要性。

同其他程序员相比，手机游戏程序员除了要有良好的基本功之外，还需要了解游戏的编程。尤其是，手机游戏作为特殊的一类游戏，在编程方面又多了许多限制，因此，要当好一个手机游戏开发人员并不容易。

3.4.1 职业要求

手机游戏程序员的职业要求如下。

- (1) 具有扎实的计算机专业知识，熟悉 Java、C/C++ 等流行开发语言，至少精通其中一种语言，有一定的编程经验。
- (2) 有良好的程序设计、分析能力，具有规范的编程风格，了解软件开发过程。
- (3) 有创新思维，学习能力强，具有良好的团队开发意识和多人协同工作经验。
- (4) 具有良好的沟通能力和理解能力，善于接受和领悟他人的想法。
- (5) 热爱游戏，渴望创造游戏，工作认真负责。

3.4.2 常用工具

J2ME 开发工具 Borland JBuilder 是 Borland 公司开发的针对 Java 的开发工具。使用 JBuilder 可以快速、有效地开发各类 Java 应用，它使用的 JDK 与 Sun 公司标准的 JDK 不同。

J2ME 经过了较多的修改，以便开发人员能够像开发 Delphi 应用那样开发 Java 应用。

J2ME 开发工具 Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，是用插件组件构建的开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括 Java 开发工具（Java Development Tools, JDT）。

C/C++ 开发工具 Microsoft Visual Studio 是微软公司推出的开发环境，是目前最流行的 Windows 平台应用程序开发环境。目前，它已经被开发到 8.0 版本，也就是 Visual Studio 2005。Visual Studio 可以用来创建 Windows 平台下的 Windows 应用程序和网络应用程序，也可以用来创建网络服务、智能设备应用程序和 Office 插件。

3.4.3 职业建议

手机游戏程序员的职业建议如下。

- (1) 手机游戏开发人员作为一类游戏开发人员，相对其他程序员，需要对图像、数值计算方面的编程更为精通。
- (2) 由于手机本身硬件（CPU 速度慢、内存小、屏幕小、分辨率低）的限制，要求开发时更加“精打细算”。
- (3) 不同于单一的团队开发，手机游戏程序员必须善于和策划、美工沟通，才能在不失实际的基础上更好地完成开发。

第 18 章

J2ME 游戏范例：潜艇游戏

18.1 代码介绍

本章将给出一个示例游戏的核心代码，这是一个著名的潜艇游戏的手机版本。它是一个良好的游戏入门范本，其中涉及精灵的使用、Tiled 的使用以及碰撞检测等运动类 2D-Tile-based 游戏常见的问题。

为了配合 API 的讲解，这里省略了大部分的游戏周边代码，而仅仅给出游戏的 GameCanvas 子类。这个类同时包含了游戏的主循环线程，非常适合用于教学演示。

如您所见，这不是一个注重产品质量的游戏，它演示了基本的内容，但不是全部，而且没有包括什么优化。本示例游戏的截图如图 18.1 所示。



图 18.1 潜艇游戏的截图

18.2 GameCanvas 子类的创建

首先要导入使用到的各个 API 包，接着创建 GameCanvas 的子类 SubCanvas，由于这个

类包含了游戏的主循环线程，因此必须实现 `Runnable` 接口。同时，为了捕获键盘操作，必须实现 `CommandListener` 接口。代码如下。

```
//创建子类 SubCanvas;
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.GameCanvas;
import javax.microedition.lcdui.game.*;
import javax.microedition.lcdui.game.LayerManager;
import java.util.*;
public class SubCanvas extends GameCanvas
    implements Runnable, CommandListener
{
    ...
}
```

类 `SubCanvas` 创建完以后，下面的代码都包含于其中，以实现游戏的主循环。为了方便理解，接下来将把代码分成几个部分逐步讲解。

18.3 变量声明及基本设定

变量声明及基本设定的代码和详细注释如下。

```
//SubCanvas 类变量声明
private Controller controller;
private Graphics graphics;
private Thread thread;
private boolean threadAlive = false;
private Command startCommand;
private Command exitCommand;
private Command pauseCommand;
//图层数据
private LayerManager layerManager;
private TiledLayer layerSeaback;
private Sprite spriteMap;
public final int GAME_INIT = 0; //游戏初始状态
public final int GAME_RUN = 1; //游戏运行状态
public final int GAME_OVER = 4; //游戏结束状态
public final int GAME_PAUSE = 5; //游戏暂停状态
public final int GAME_SUSPEND = 9; //游戏挂起状态
public boolean COMMAND_ADD_FLAG = false; //是否已经添加 Command 标识
public static final int TROOP_PLAYER = 0; //敌我标识
public static final int TROOP_ENEMY = 1;
public static int PLAYER_LEVEL = 1; //当前玩家水平
public static int ENEMY_MAX = PLAYER_LEVEL * 10; //最大敌人数量
public static int ENEMY_CURRENT = 0; //当前敌人数量
public static int ENEMY_CURRENT_LIMIT = 0; //当前敌人数量限制
protected int TRIGGER_COUNT = 0; //拖延标识，避免敌人新潜艇同一时刻全部产生
protected int TICK_COUNT = 0;
public static int mainWidth; //屏幕宽度
public static int mainHeight; //屏幕高度
public int gameState; //游戏状态
public final static int TILE_WIDTH = 10; //背景单元宽度 10px
public final static int TILE_HEIGHT = 10; //背景单元高度 10px
public final static int MILLIS_PER_TICK = 200; //动画变化频率(200ms)
private final static int WIDTH_IN_TILES=45; //游戏域宽度(以单元宽度计算)
private final static int HEIGHT_IN_TILES=24; //游戏域高度(以单元高度计算)
private final static int NUM_DENSITY_LAYERS = 4; //海面密度(背景图层)
//精灵旋转
```

```
private int[] rotations =
{
    Sprite.TRANS_NONE,
    Sprite.TRANS_MIRROR,
    Sprite.TRANS_MIRROR_ROT90,
    Sprite.TRANS_MIRROR_ROT180,
    Sprite.TRANS_MIRROR_ROT270,
    Sprite.TRANS_ROT90,
    Sprite.TRANS_ROT180,
    Sprite.TRANS_ROT270
};
// 整个游戏背景宽度 (以像素计算 : 宽度单元数 * 宽度单元像素)
public final static int WORLD_WIDTH = WIDTH_IN_TILES * TILE_WIDTH;
// 整个游戏背景高度
public final static int WORLD_HEIGHT = HEIGHT_IN_TILES * TILE_HEIGHT;
// 每一个密度层的 TILE 单元数
private final static int NUM_DENSITY_LAYER_TILES = 4;
private final static int FRACT_DENSITY_LAYER_ANIMATE = 20;
private int SEABACK_DENSITY; // 游戏初始海水密度为 0
private final Vector oceanLayersVector = new Vector();
private Vector fishCollectionVector = new Vector();
public Vector enemyCollectionVector = new Vector();
public Vector tinfishCollectionVector = new Vector();
private Sub mySub = null;
private Runtime rt = null;
private boolean userViewWindow = false; // 初始化为不使用窗口区域视野
private volatile Thread animationThread = null; // 创建不稳定的动画线程
// LayerManager 的偏移坐标
private int xViewWindow;
private int yViewWindow;
private int wViewWindow;
private int hViewWindow;
```

这一部分基本上没什么可以说的，各个变量也都有注释说明。只有几个要注意的地方：首先是 `final` 和 `static` 的使用，`final` 的数据值不能被改变，而 `static` 数据是类所有，不是平时对象中的数据，这两种数据在初始化一个游戏中的数据时常被用到；其次，游戏域宽/高度和整个游戏背景宽/高度是不同的，注意区分；最后，代码中将 `sprite` 的旋转定义成一个 `int` 数组以方便使用，是很可取的方法，值得借鉴。

18.4 构造函数和地图显示

构造函数和地图显示的代码及详细注释如下。

```
// SubCanvas 构造函数
public SubCanvas(Controller controller)
{
    super(false); // 不屏蔽键盘事件(潜艇运动采用主动轮询，而开火则采用捕获方式)
    this.controller = controller;
    this.graphics = getGraphics();
    this.layerManager = new LayerManager();
    init();
    // 画布构造时即建立玩家潜艇，初始位置为屏幕的(1/3, 1/3)处
    mySub = new Sub(this, SubMIDlet.createImage("/res/sub.png"),
        mainWidth / 3, mainHeight / 3, layerManager);
    rt = Runtime.getRuntime(); // 监测运行时，以便及时运行垃圾回收
    // 添加菜单按钮
```

```
startCommand = new Command("Start", Command.OK, 1);
pauseCommand = new Command("Pause", Command.OK, 1);
exitCommand = new Command("Exit", Command.EXIT, 2);
this.gameState=this.GAME_INIT;//初始化游戏状态, 游戏处于demo 画面状态
//启动应用程序
threadAlive = true; thread = new Thread(this);
thread.start();
}
```

上面这段代码是类 `SubCanvas` 的构造函数, 除了使用基本的 `new` 语句添加图层和按钮等以外, 还启动了应用程序, 因此游戏的主循环也在这个类中得以实现。`Super (false)` 语句使得键盘事件未被屏蔽, 另一个要注意的细节是 `rt = Runtime.getRuntime();`, 垃圾的及时回收对运行速度有一定的影响。

```
//SubCanvas 构造函数中调用的 Init()
private void init()
{
    this.clearData();//清理数据
    //根据显示设备,设置合适的最大区和显示视野
    mainWidth = getWidth();
    mainHeight = getHeight();
    this.xViewWindow = 0;
    //现有设备不能容纳所有游戏区域
    if(WORLD_WIDTH > mainWidth)
    {
        userViewWindow = true;
        this.wViewWindow = mainWidth;
    }
    Else //现有设备可以容纳所有游戏区域
    {
        this.wViewWindow = WORLD_WIDTH;
    }
    this.yViewWindow = 0; //相同方法处理高度
    if(WORLD_HEIGHT > mainHeight)
    {
        userViewWindow = true;
        this.hViewWindow = mainHeight;
    }else
    {
        this.hViewWindow = WORLD_HEIGHT;
    }
    //设定图层显示方式
    if(userViewWindow)
    {
        this.layerManager.setViewWindow(xViewWindow,
                                        yViewWindow,wViewWindow, hViewWindow);
    }
}
//清理数据函数
protected void clearData()
{
    PLAYER_LEVEL = 1;
    ENEMY_MAX = PLAYER_LEVEL * 10;
    ENEMY_CURRENT = 0;
    TRIGGER_COUNT = 0;
    SEABACK_DENSITY = 0;
    ENEMY_CURRENT_LIMIT = PLAYER_LEVEL * 2;
}
```

这一段代码有两个函数, 主要涉及地图数据清理和地图显示窗口的设置, 其中 `init ()`

函数是在 SubCanvas 类的构造函数中被调用的，然后在 init 函数中又调用了 clearData () 函数清理地图上的数据。函数运行的次序值得注意。

18.5 主线程及各物体控制函数

主线程及各物体控制函数的代码及详细注释如下：

```
//程序主线程，每 50ms 运行刷新一次
public void run()
{
    //利用条件驱动线程
    while(threadAlive)
    {
        try
        {
            Thread.sleep(25);
        }
        catch (InterruptedException e) {e.printStackTrace();}
        //分离对玩家潜艇和普通物体的响应速度
        if(gameState == GAME_RUN)
        {
            mySub.movePosition(getKeyStates());
        }
        if((TICK_COUNT % 2) == 0)
        {
            this.paintCanvas(graphics); // 重画事件
            this.tick();
        }
        TICK_COUNT ++;
    }
}
//用于控制开火的函数
protected synchronized void keyPressed(int keyCode)
{
    int action = getGameAction(keyCode);
    if(action == Canvas.FIRE && gameState == GAME_RUN)
    {
        if(mySub != null) //玩家潜艇开火
        {
            mySub.fire();
        }
    }
}
//秒触发器函数
public synchronized void tick()
{
    //主动查询状态
    int keyState = getKeyStates();
    if(gameState != GAME_OVER)
    {
        this.tickFishes();//执行鱼类图形运动
        this.tickTinfish();//执行鱼雷触发
        if(gameState == GAME_RUN)
        {
            //替代 Canvas 中的 KeyPressed() 捕获事件
            mySub.movePosition(keyState);
            //创建并执行敌人潜艇行为
            this.tickSub();
        }
    }
}
```

```
        this.tickEnemySub();
        if(ENEMY_CURRENT ==0 && ENEMY_MAX == 0)
        {
            gameState = GAME_SUSPEND; //挂起线程
        }
    }
}
```

以上代码是游戏主线程控制函数 `run()` 和两个用于输入捕获的函数 `keyPressed` 和 `tick`，程序语句并不难理解，也有注释参考。值得强调的是，这个程序混和使用了主动轮询和键盘捕获两种获得输入的方法，读者可以细细琢磨这样的好处和坏处。

```
//鱼类图形运动函数
private void tickFishes()
{
    for(int i = 0; i < fishCollectionVector.size(); i++)
    {
        FishCollection collection =
            (FishCollection)fishCollectionVector.ElementAt(i);
        collection.tick();
    }
}
//执行鱼雷触发函数
protected void tickTinfish()
{
    for(int j = 0; j < tinfishCollectionVector.size(); j++)
    {
        Tinfish tinfish=(Tinfish)tinfishCollectionVector.ElementAt(j);
        if(!tinfish.isLifeState())
        { //如果某个鱼雷元素已经结束生命周期，则将其置 null，并从数组中删除
            tinfishCollectionVector.removeElementAt(j);
            this.layerManager.remove(tinfish);
            tinfish = null;
        }else
        {
            tinfish.tick();
        }
    }
}
//玩家潜艇运动及生命状态函数
private void tickSub()
{
    if(!mySub.isLifeState())
    {
        gameState = GAME_OVER;
    }
}
//敌人潜艇的运行操作函数
protected void tickEnemySub()
{
    //当敌人剩余最大数量大于 0，并且敌人当前数量小于并行敌人上限时
    //可以添加新的敌人潜艇
    if(ENEMY_MAX >= 0 && ENEMY_CURRENT <= ENEMY_CURRENT_LIMIT
        && ENEMY_CURRENT< 10)
    {
        int iLeft = ENEMY_MAX - ENEMY_CURRENT;
        if(iLeft > 0) //当剩余敌人量(最大量 - 当前量)大于 0 的时候
        {
            int n = SubMIDlet.createRandom(iLeft) + 1;
        }
    }
}
```

```
//创建敌人图片
Image image=SubMIDlet. CreateImage("/res/enemysub_f.png");
int xPosition = 0;
int yPosition=(SubMIDlet.createRandom(5)* WORLD_HEIGHT)/5;
//拖延标识, 避免敌人新潜艇同一时刻全部产生
for(int i = 0; i < n; i++)
{
    if(TRIGGER_COUNT >= 20)
    {
        yPosition = (WORLD_HEIGHT * (i % 5)) / 5;
        if(i % 2 == 0)
        {
            xPosition = 0;
        }else
        {
            xPosition = WORLD_WIDTH - image.getWidth();
        }
        //创建一艘敌人潜艇, 同时更新监听数组
        EnemySub enemySub = new EnemySub(this, image,
            xPosition,yPosition,PLAYER_LEVEL);
        ENEMY_CURRENT++;
        layerManager.insert(enemySub, 0);
        this.enemyCollectionVector.addElement(enemySub);
        TRIGGER_COUNT = 0;
    }else
    {
        TRIGGER_COUNT++;
    }
}
image = null;
}
//对所有已经存在的敌人潜艇进行 tick 触发
Image imageDestroyed = null;
for(int j = 0; j < enemyCollectionVector.size(); j++)
{
    EnemySubenemySub=(EnemySub)enemyCollection Vector.ElementAt(j);
    int iCount = 0;
    //当生命周期结束
    if(!enemySub.isLifeState())
    {
        imageDestroyed=SubMIDlet. CreateImage("/res/enemysub_die.png");
        enemySub.setImage(imageDestroyed, imageDestroyed.getWidth(),
            imageDestroyed.getHeight());
        if(enemySub.getVx() >= 0)
        {
            enemySub.setTransform(Sprite.TRANS_NONE);
        }else
        {
            enemySub.setTransform(Sprite.TRANS_MIRROR);
        }
        enemyCollectionVector.removeElementAt(j);
        //消灭一艘敌人潜艇, 同时更新监听数组
        SpriteChanged spriteChanged =
            new SpriteChanged (enemySub, layer Manager);
        spriteChanged.start();
        spriteChanged = null;
        enemySub = null;
        ENEMY_CURRENT--;
        ENEMY_MAX--;
    }else
```

```
        {  
            enemySub.tick();  
        }  
    }  
    imageDestroyed = null;  
}
```

这一部分代码是对游戏中各种物体的控制，包括鱼类图形、鱼雷、玩家潜艇、敌人潜艇。函数的结构并不复杂，难度不大，如果读者理解有困难可以参考注释和 API 文档，这里不再一一赘述。要注意的是，敌人潜艇函数中涉及的变量较多，控制方法也相对复杂，每次创建或消灭一艘敌人潜艇时，不要忘了更新监听数组。

18.6 重画事件

重画事件函数的代码及详细注释如下。

```
//画布重画事件,替代 Canvas 中的 paint()事件  
//根据不同的游戏状态 (gameState) 画出游戏画面  
//本方法由 thread 每次重新启动,最后执行 flushGraphics()重画缓冲区  
public synchronized void paintCanvas(Graphics g)  
{  
    if(gameState == GAME_INIT)           //游戏第一次启动  
    {  
        this.setFullScreenMode(true);    //设置为全屏模式并清屏  
        g.setColor(255, 255, 255);  
        g.fillRect(0, 0, mainWidth, mainHeight);  
        if(!COMMAND_ADD_FLAG)           //添加响应命令及监听器  
        {  
            this.addCommand(pauseCommand);  
            this.addCommand(exitCommand);  
            this.setCommandListener(this);  
            COMMAND_ADD_FLAG = true;  
        }  
        //参数调整  
        if(fishCollectionVector != null)  
        {  
            fishCollectionVector = null;  
            fishCollectionVector = new Vector();  
        }  
        if(this.layerManager != null)  
        {  
            this.layerManager = null;  
            this.layerManager = new LayerManager();  
            if(userViewWindow)  
            {  
                this.layerManager.setViewWindow(xViewWindow, yView Window,  
                                                wViewWindow, hViewWindow);  
            }  
        }  
        if(mySub != null)  
        {  
            mySub = null;  
            mySub=new Sub(this,SubMIDlet. CreateImage ("/res/sub.png " ),  
                          getWidth()/3,getHeight() / 3, layer Manager);  
        }  
    }  
}
```

```
//创建背景图层
this.createSandBackground();
this.createSunkenBoat();
this.createFishCollection(0, FishCollection.NUM_FISH_TYPES);
this.createMysub();
this.createSeaBackground();
mysub.setPosition(getWidth() / 3, getHeight() / 3);
gameState = GAME_RUN;
}
else if(gameState == GAME_RUN) //游戏处于运行状态
{
//在性能过耗(可用内存不到当前内存总量的 4/5 时),进行垃圾回收 GC
if(rt.freeMemory() < (rt.totalMemory() * 4 / 5))
{
rt.gc();
}
}
else if(gameState == GAME_SUSPEND) //下一轮游戏
{
//更新数据
mysub.setHpLife(15);
mysub.setPosition(mainWidth / 3, mainHeight / 3);
//目前游戏只设计了 4 级关卡
if(PLAYER_LEVEL >= 4)
{
gameState = GAME_OVER;
}
else
{
PLAYER_LEVEL ++;
controller.EventHandler(Controller.EVENT_NEXTROUND);
ENEMY_MAX = PLAYER_LEVEL * 10;
ENEMY_CURRENT = 0;
TRIGGER_COUNT = 0;
TICK_COUNT = 0;
ENEMY_CURRENT_LIMIT = PLAYER_LEVEL * 2;
Layer layer = null;
//暂时删除 layerManager 中所有文件
//清空鱼雷与敌人潜艇数据,为更新图层做准备
this.tinfishCollectionVector.removeAllElements();
this.enemyCollectionVector.removeAllElements();
this.fishCollectionVector.removeAllElements();
for(int i = 0; i < layerManager.getSize(); i++)
{
layer = layerManager.getLayerAt(i);
layerManager.remove(layer);
}
layer = null;
//更新海底图层数据
this.SEABACK_DENSITY = (SEABACK_DENSITY + 1) % 4;
gameState = GAME_INIT;
this.unActive();
}
}
else if(gameState == GAME_OVER) //游戏结束
{
threadAlive = false;
controller.EventHandler(Controller.EVENT_MENU_GAMEOVER);
gameState = this.GAME_INIT;
}
```

```
    }  
    //最后, 在缓冲区重画  
    this.layerManager.paint(g, 0, (getHeight() - WORLD_HEIGHT) / 2);  
    this.flushGraphics();  
}  
//游戏状态动作监听函数  
public void commandAction(Command command, Displayable display)  
{  
    if(command == startCommand) //开始游戏  
    {  
        if(this.gameState == GAME_OVER)  
        {  
            gameState = GAME_INIT;  
        }  
        else  
        {  
            gameState = GAME_RUN;  
        }  
        this.removeCommand(this.startCommand);  
        this.addCommand(pauseCommand);  
    }  
    else if(command == pauseCommand) //暂停游戏  
    {  
        gameState = GAME_PAUSE;  
        this.removeCommand(this.pauseCommand);  
        this.addCommand(startCommand);  
    }  
    else if(command == exitCommand) //退出游戏  
    {  
        gameState = GAME_OVER;  
    }  
}
```

重画事件函数是继主线程函数之后的又一重要函数, 它控制了每一次游戏状态的变化, 无论游戏的开始、暂停、结束, 还是进入下一个关卡, 都离不开这个函数。它是通过状态监听函数 `commandAction()` 来获取游戏的当前状态并执行相应操作的, 并在函数体的最后通过调用 `layerManager.paint()` 来刷新游戏画面。

18.7 背景图层

背景图层的代码及详细注释如下。

```
//创建海底沙地背景图层  
protected void createSandBackground()  
{  
    Image bottomTitles = SubMIDlet.createImage("/res/bottom.png");  
    //将图片 bottomTitles 切成指定大小(TILE_WIDTH, TILE_HEIGHT)  
    //并创建一个指定维数的背景数组  
    TiledLayer layer = new TiledLayer(WIDTH_IN_TILES, 1, bottomTitles,  
        TILE_WIDTH, TILE_HEIGHT);  
    //将海底图层数组中的每个小格用原始图片的第 i 块来填充  
    for(int column = 0; column < WIDTH_IN_TILES; column++)  
    {  
        int i = SubMIDlet.createRandom(NUM_DENSITY_LAYER_TILES) + 1;  
        layer.setCell(column, 0, i);  
    }  
}
```

```
layer.setPosition(0, WORLD_HEIGHT - bottomTitles.getHeight());
layerManager.append(layer); //添加图层
bottomTitles = null;
}
//创建海底水层背景图片
protected void createSeaBackground()
{
    if(this.SEABACK_DENSITY >= 4){SEABACK_DENSITY = 0;}
    Image image = SubMIDlet.createImage("/res/densityLayer"
        +this.SEABACK_DENSITY + ".png ");
    layerSeaback = new TiledLayer(WIDTH_IN_TILES, HEIGHT_IN_TILES,
        image, TILE_WIDTH, TILE_HEIGHT);
    for(int row = 0; row < HEIGHT_IN_TILES; row++)
    {
        for(int column = 0; column < WIDTH_IN_TILES; column++)
        {
            layerSeaback.setCell(column, row,
                SubMIDlet.createRandom(NUM_DENSITY_LAYER_TILES)+1);
        }
    }
    layerSeaback.setPosition(0, 0);
    layerManager.append(layerSeaback);
    image = null;
}
//创建沉船图层
protected void createSunkenBoat()
{
    Image imageSunkenBoat=SubMIDlet.createImage("/res/sunkenBoat.png");
    //出现的沉船数量
    int numSunkenBoats = 1+(WORLD_WIDTH/(3*imageSunkenBoat.getWidth()));
    Sprite sunkenBoat;
    int bx = 0;
    for(int i = 0; i < numSunkenBoats; i++)
    {
        sunkenBoat = new Sprite(imageSunkenBoat,
            imageSunkenBoat.getWidth(),
            imageSunkenBoat.getHeight());
        sunkenBoat.SetTransform(this.rotations[SubMIDlet.CreateRandom(
            this.rotations.length)]);
        bx = (WORLD_WIDTH-imageSunkenBoat.getWidth())/numSunkenBoats;
        bx = (i * bx) + SubMIDlet.createRandom(bx); //随机定义沉船位置
        sunkenBoat.setPosition(bx,WORLD_HEIGHT-imageSunkenBoat.getHeight());
        this.layerManager.append(sunkenBoat);
        mySub.addCollideable(sunkenBoat);
    }
    imageSunkenBoat = null;
}
//创建玩家潜艇
protected void createMySub()
{
    this.layerManager.append(mySub);
}
//创建鱼群背景
protected void createFishCollection(int startId, int endId)
{
    for(int id = startId; id < endId; id++)
    {
        int w = WORLD_WIDTH / 4;
        int h = WORLD_HEIGHT / 4;
        int x = SubMIDlet.createRandom(WORLD_WIDTH - w);
```

```
int y = SubMIDlet.createRandom(WORLD_HEIGHT - h);
int vx = FishCollection.randomVelocity(TILE_WIDTH / 4);
int vy = FishCollection.randomVelocity(TILE_HEIGHT / 4);
//初始化鱼类图层, 同时把图层添加到图层管理器上
FishCollection fishCollection = new FishCollection (layerManager,
            id,x,y,w,h,x,vy,0, 0, WORLD_WIDTH, WORLD_HEIGHT);
this.fishCollectionVector.addElement(fishCollection);
    }
}

//重新设置图层显示区域
public void adjustViewWindow(int xSub, int ySub, int width, int height)
{
    if (this.userViewWindow)
    {
        xViewWindow = xSub + (width / 2) - (wViewWindow / 2);
        if (xViewWindow < 0)
        {
            xViewWindow = 0;
        }
        if (xViewWindow > (WORLD_WIDTH - wViewWindow))
        {
            xViewWindow = WORLD_WIDTH - wViewWindow;
        }
        yViewWindow = ySub + (height / 2) - (hViewWindow / 2);
        if (yViewWindow < 0)
        {
            yViewWindow = 0;
        }
        if (yViewWindow > (WORLD_HEIGHT - hViewWindow))
        {
            yViewWindow = WORLD_HEIGHT - hViewWindow;
        }
        layerManager.setViewWindow(xViewWindow,
            yViewWindow,
            wViewWindow,
            hViewWindow);
    }
}

public Sub getMySub() //获取玩家潜艇
{
    return mySub;
}

public boolean isThreadAlive() //判断线程是否激活
{
    return threadAlive;
}

public void setThreadAlive(boolean threadAlive) //设置线程
{
    this.threadAlive = threadAlive;
}

public void active()
{
    this.showNotify();
}

public void unActive()
{
    this.hideNotify();
}
}
```

最后一部分代码比较简单，包括各个图层的设置和一些小函数。关于图层设置，每个图层的方法都差不多，看懂一个基本就能举一反三了，因此后面不再详细注释。

看完整个程序后，作一个简单的回顾，请着重注意代码中的以下几点。

(1) `run` 方法用于游戏的主循环。

(2) `paintCanvas` 方法用于渲染。

(3) 关于输入捕获的一点说明：这个游戏并没有屏蔽键盘事件，它混合使用了主动轮询用于潜艇运动，而开火则采用捕获方式。

目 录

第1部分 手机游戏行业概况

第1章 手机游戏的市场和发展前景	2
1.1 游戏市场现状	2
1.1.1 国际游戏市场现状	3
1.1.2 国内游戏市场现状	3
1.2 游戏发展趋势	4
1.3 游戏中的心理学	6
第2章 手机游戏开发的技术特点	8
2.1 游戏的种类	8
2.1.1 角色扮演类	8
2.1.2 回合制战略类	9
2.1.3 益智解谜类	9
2.1.4 即时战略类	10
2.1.5 动作格斗类	10
2.1.6 策略类	11
2.1.7 冒险类	12
2.1.8 射击类	13
2.1.9 体育类	13
2.1.10 模拟养成类	14
2.2 手机游戏的特色分类	14
2.2.1 短信游戏 (SMS)	14
2.2.2 WAP游戏	15
2.2.3 IVR游戏	16
2.2.4 终端游戏	16
2.3 游戏设备的特点	19
2.3.1 手机作为游戏设备的限制	20
2.3.2 手机游戏设备的优势	21
2.3.3 设计实现中的扬长避短	21
2.4 无线游戏蓝图	22
第3章 手机游戏产业中的职业规划	24
3.1 手机游戏项目管理的特点	24
3.1.1 职业要求	24
3.1.2 常用工具	25
3.1.3 职业建议	25
3.2 美术—方寸之间尽显本色	26
3.2.1 职业要求	26

3.2.2	常用工具	26
3.2.3	职业建议	27
3.3	策划—有限的空间、无限的任务	27
3.3.1	职业要求	27
3.3.2	常用工具	28
3.3.3	职业建议	28
3.4	程序, 万丈高楼平地起	28
3.4.1	职业要求	29
3.4.2	常用工具	29
3.4.3	职业建议	29

第2部分 Symbian游戏开发

第4章	Symbian操作系统	32
4.1	Symbian OS	32
4.1.1	Symbian OS概述	32
4.1.2	Symbian的版本介绍	33
4.1.3	Symbian操作系统的主要特点	34
4.2	Symbian OS开发基础	35
4.2.1	命名约定	36
4.2.2	命名空间	38
4.2.3	基本数据类型	38
4.2.4	Symbian OS应用程序类型	39
第5章	Symbian OS开发	42
5.1	异常 (Exception)	42
5.1.1	异常处理	42
5.2	Symbian OS的受限部分	48
5.3	定时器 (Timer)	50
5.4	键盘事件处理	52
5.5	声音	54
5.5.1	音频声音	55
5.5.2	预备声音	56
5.5.3	使用观察者对象	57
5.5.4	程序代码	57
5.5.5	声音编程中的技巧	59
5.6	安装	60
第6章	Symbian OS图形开发	62
6.1	图形架构	62
6.2	基本绘图函数	64
6.2.1	绘制文本函数	64
6.2.2	点	64
6.2.3	线	65
6.2.4	绘制形状	65
6.2.5	填充图形	67

6.3	字体和位图服务器	67
6.3.1	位图格式	68
6.3.2	遮罩 (mask)	70
6.3.3	读取位图	70
6.3.4	显示位图	71
6.3.5	位图操作中的像素级处理	73
6.3.6	字体	74
6.4	Windows服务器	76
6.4.1	客户端缓冲	77
6.4.2	窗口	78
6.4.3	控制环境	79
6.4.4	UI库	80
6.4.5	精灵 (sprite)	81
6.4.6	双缓冲 (double buffering)	83
6.4.7	直接绘制 (direct draw)	85
6.5	OpenGL ES	88
6.5.1	OpenGL ES概述	88
6.5.2	OpenGL ES框架	89
6.5.3	Series 60平台中的 OpenGL ES	90
6.5.4	OpenGL ES功能	90
6.6	视频剪辑	91
6.7	游戏编程中的位图技巧	93
6.7.1	降低色彩深度	93
6.7.2	像素处理	102
6.7.3	贴图	104
6.7.4	直接写屏与系统特殊 事件	106
第7章	Symbian OS通信开发	108
7.1	通信架构	108
7.2	串口通信服务器	109
7.3	套接字服务器 (Socket server)	111
7.4	游戏数据接收	113
7.5	串口编程范例	114
第8章	Symbian OS开发和调试技巧	132
8.1	安装Symbian SDK以及给 Visual C++配置开发环境	132
8.1.1	安装SDK	132
8.1.2	配置Visual C++	133
8.1.3	编译	134
8.1.4	打包	135
8.1.5	手机测试	136
8.2	在Series 60硬件上调试	136
8.2.1	安装系统	136
8.2.2	开始调试	137

8.2.3	使用蓝牙进行设备调试	137
第9章	Symbian游戏示例	141
第3部分 Windows Mobile游戏开发		
第10章	Windows Mobile开发介绍	160
10.1	事件驱动和消息响应机制	160
10.2	Windows Mobile程序和 Windows程序的不同点	161
10.2.1	Windows CE APIs和 Win32 API的不同	161
10.2.2	Windows CE MFC和 标准MFC的不同	161
10.2.3	存储器的限制	162
10.2.4	电源管理	162
10.2.5	硬件特性	162
10.2.6	测试和调试	162
10.3	Embedded Visual C++开发 工具介绍	163
10.3.1	Embedded Visual C++的 特性	163
10.3.2	建立应用程序	164
10.3.3	类及文件说明	168
10.3.4	Hello Windows CE 程序	169
10.3.5	辅助开发工具介绍	169
10.4	VS.NET Compact Framework 开发工具	172
10.5	Windows程序向Windows Mobile程序移植的关键因素	175
10.5.1	移植使用Windows CE API	175
10.5.2	管理Windows CE的 存储器	176
10.5.3	管理可用的电量	176
10.5.4	移植图形用户接口	177
10.5.5	调整位图和图标	177
10.5.6	使用Unicode	177
10.5.7	创建和管理窗口	178
10.5.8	使用Windows CE 对话框	178
10.5.9	移植用户接口控件	178
10.5.10	管理Windows CE 线程	178
10.5.11	更改用户接口	179
10.5.12	支持Windows CE 通信	179
10.5.13	最小化使用注册表	180
第11章	Windows Mobile 开发基础	181

11.1	Windows Mobile菜单	181
11.1.1	概要	181
11.1.2	和菜单有关的主要消息及其响应函数	181
11.1.3	CMenu类	183
11.1.4	上下文菜单	185
11.1.5	菜单操作技巧	186
11.2	对话框	188
11.2.1	使用资源编辑器编辑对话框	188
11.2.2	对话框的数据交换和数据检查	188
11.2.3	模态和非模态对话框	189
11.2.4	通用对话框	191
11.3	窗口	192
11.3.1	产生CWnd对象	192
11.3.2	消息映射	194
11.3.3	关闭窗口	195
11.3.4	CWnd和句柄	195
11.3.5	SDMV应用中的窗口切换函数	196
11.4	控件	198
11.4.1	Windows标准控件和通用控件	198
11.4.2	MFC控件类介绍及使用范例	236
11.5	Windows Mobile的多线程和多进程	256
11.5.1	Windows Mobile的多任务机制	257
11.5.2	多线程	259
11.5.3	多进程	267
第12章	Windows Mobile 图形图像操作基础	273
12.1	基本文本操作	273
12.1.1	文本输出函数	273
12.1.2	文本属性	277
12.1.3	字符属性	279
12.1.4	字体	280
12.1.5	文本显示特殊技巧	283
12.2	基本图形图像操作	286
12.2.1	绘图函数	286
12.2.2	画笔	289
12.2.3	画刷	293
12.2.4	位图画刷	294
12.2.5	方便实用的画笔和画刷类	296

12.2.6	绘图模式	298
12.2.7	位图	299
12.2.8	位操作	301
12.2.9	图标	302
12.3	高级屏幕绘图	303
12.3.1	DIB类	303
12.3.2	未公开的图像API	307
12.3.3	IMGDECMP.dll和 VOImage类	308
12.3.4	IJG JPEG库	310
12.3.5	Windows CE高速 图形库	312
12.3.6	二维实时图形	315
12.3.7	分析图表类库	317
第13章	Windows Mobile游戏示例	324
13.1	俄罗斯方块游戏的代码分析	324
13.1.1	程序结构分析	324
13.1.2	程序的主界面	325
13.1.3	程序入口分析	325
13.1.4	程序的主窗口代码 分析	326
13.1.5	游戏的主要逻辑控制	330
13.1.6	工具类代码分析	333
13.2	雷电射击游戏的代码分析	341
13.2.1	游戏的主界面	341
13.2.2	主要函数介绍	342
13.2.3	程序的主处理模块	342
13.2.4	背景绘制函数	345
13.2.5	游戏中敌机的控制和 处理函数	346
13.2.6	游戏中我机的控制 代码	351
第4部分 J2ME 游戏开发		
第14章	J2ME介绍与环境搭建	356
14.1	J2ME介绍	356
14.1.1	J2ME的基本概念	356
14.1.2	MIDP的使用范围	357
14.1.3	J2ME开发的特点	357
14.2	J2ME开发环境和调试技巧	358
14.2.1	J2ME开发环境的 准备	358
14.2.2	调试技巧	364
14.2.3	代码优化	365
第15章	MIDP程序开发	368
15.1	MIDlet介绍	368
15.1.1	什么是MIDlet	368

15.1.2	MIDlet的生命周期	368
15.2	MIDP界面程序设计	369
15.2.1	MIDP界面类函数库	369
15.2.2	List类Gauge	370
15.2.3	Alert与AlertType	373
15.2.4	TextBox类	376
15.2.5	Form类概述	379
15.2.6	StringItem与 ImageItem	379
15.2.7	TextField与DateField	382
15.2.8	Gauge与ChoiceGroup	384
15.2.9	CustomItem	386
15.2.10	Ticker类	391
15.3	MIDP图形处理	392
15.3.1	MIDP图形处理 函数库	392
15.3.2	Display类	392
15.3.3	Canvas类	394
15.3.4	Graphics类	395
15.3.5	Image类	398
15.3.6	Font类	400
15.4	MIDP数据库程序设计简介	401
15.5	MIDP网络程序设计	402
15.5.1	通用连接框架	402
15.5.2	使用HTTP进行连接	404
15.5.3	使用Socket进行连接	407
第16章	MIDP 2.0游戏API	410
16.1	游戏API简介	410
16.2	Layer类和LayerManager类	411
16.3	Sprite类	412
16.3.1	帧	412
16.3.2	帧序列	413
16.3.3	引用像素	414
16.3.4	变换	415
16.3.5	碰撞检测	415
16.4	TiledLayer类	416
16.4.1	图块	416
16.4.2	单元格	417
16.5	GameCanvas类	418
16.5.1	屏幕缓冲	419
16.5.2	按键查询	419
16.6	MIDP 2.0对游戏开发有用的 其他特性	420
第17章	J2ME游戏设计	422
17.1	游戏开发过程	422
17.1.1	提案	422
17.1.2	设计	423

17.1.3	实现	424
17.1.4	评审	426
17.1.5	完成	426
17.1.6	设计流程图.....	427
17.2	界面设计	427
17.2.1	游戏界面设计概述.....	427
17.2.2	界面设计的原则、 步骤和文档格式.....	428
17.2.3	界面设计实例—欢迎 界面和功能选择界面...	429
17.3	游戏引擎设计介绍.....	432
17.4	J2ME程序测试.....	435
17.4.1	介绍	435
17.4.2	搭建开发环境.....	435
17.4.3	编写测试类.....	436
17.4.4	调试运行.....	437
17.5	J2ME代码优化建议.....	437
第18章	J2ME游戏范例：潜艇游戏	439
18.1	代码介绍	439
18.2	GameCanvas子类的创建	439
18.3	变量声明及基本设定.....	440
18.4	构造函数和地图显示.....	441
18.5	主线程及各物体控制函数.....	443
18.6	重画事件	446
18.7	背景图层	448