

电脑报 东方工作室

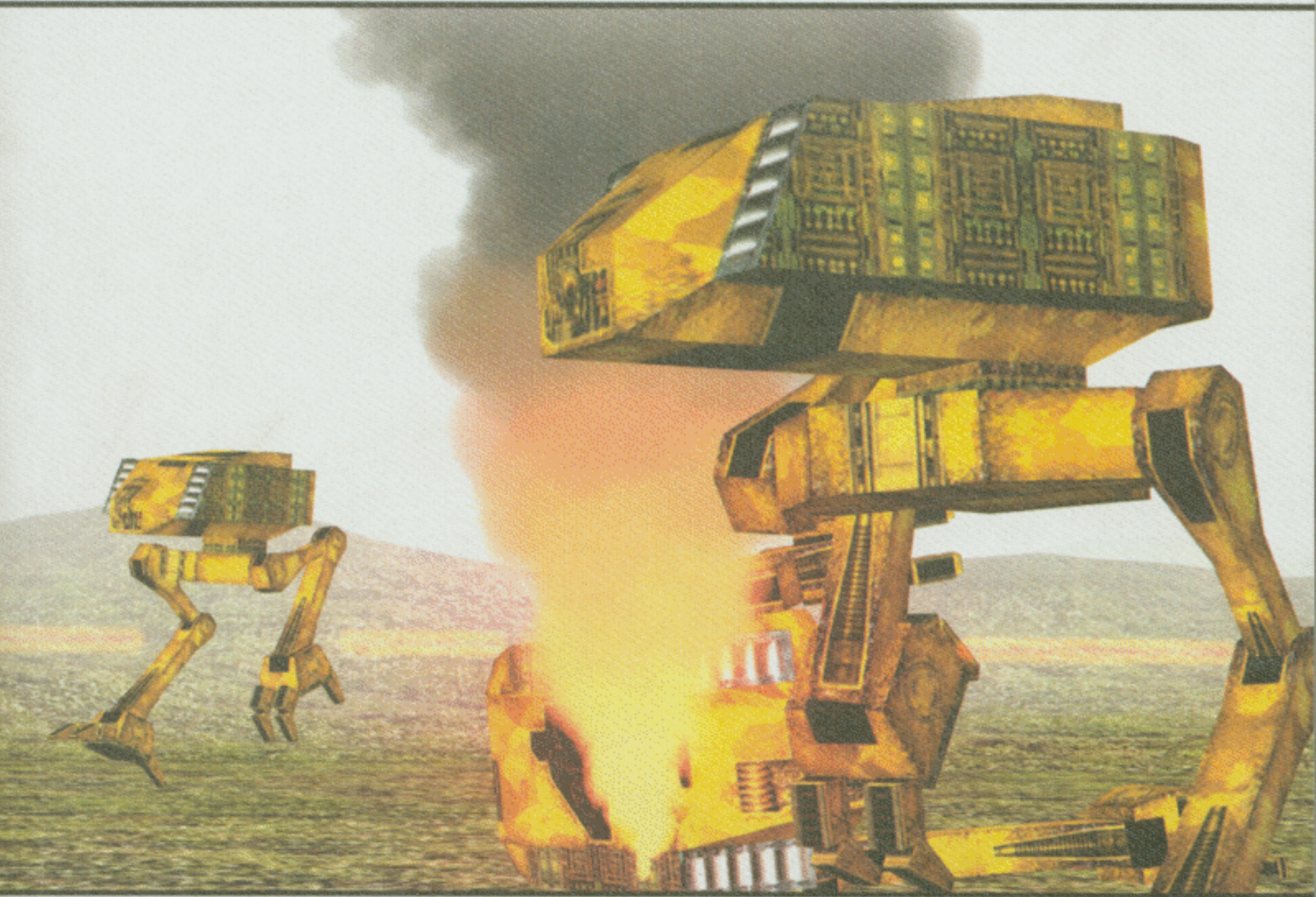
快乐写游戏
轻松学编程

PC

游戏编程 (基础篇)

PC Youxi Biancheng Jichupian

编著 徐丹 谭文洪
李鉴 龚敏敏



重庆大学出版社

快乐写游戏 轻松学编程

孕悦游戏编程(基础篇)

徐摇丹摇谭文洪摇李摇鉴摇龚敏敏摇编著

重庆大学出版社

内 容 提 要



摇摇本书全面介绍了游戏编程的初级入门知识,包括数据结构、算法、文件打包以及悦垣,并详细介绍了一些在游戏策划中所需要的知识。另外,作为本书的一个侧重点,书中用较大篇幅介绍了计算机图形学与阅,这些章节为读者进入游戏三维编程殿堂提供了入门知识,也能够给一些有经验的三维编程人员提供一些宝贵的经验。同时介绍了一些在实时三维游戏美术中需要注意的地方(涉及到面简化等技术)。最后一章,大部分内容是为了巩固前几章所学的理论知识所编写的例子程序,每个例子程序都能够说明一定的问题,并且每一个例子都比上一个例子涉及的知识更深入,读者可以按照顺序渐进学习。

摇图书在版编目(悦)数据

摇悦游戏编程(基础篇)徐丹等编著-重庆:重庆大学出版社,圆年猿月
摇(快乐写游戏轻松学编程系列)

摇I. 徐... II. 徐... III. 游戏—应用程序—程序设计摇IV. 摇

摇I. 徐... II. 徐... III. 游戏—应用程序—程序设计摇IV. 摇

摇中国版本图书馆悦数据核字(圆)第 号

快乐写游戏轻松学编程

悦游戏编程(基础篇)

徐丹摇谭文洪摇李摇鉴摇龚敏敏摇编著

责任编辑:陈摇其 版式设计:吴庆渝

责任校对:何建云 责任印制:张永洋

*

重庆大学出版社出版发行

出版人:张鸽盛

社址:重庆市沙坪坝正街 号重庆大学(粤区)内

邮编: 源

电话:(圆) 远

传真:(圆) 远

网址: 源

邮箱: 源 市场营销部)

全国新华书店经销

重庆科情印务有限公司印刷

*

开本: 源 摇字数: 源千

圆年 月第 员版 摇 圆年 猿月第 员次印刷

印数: 员— 员

源 定价: 猿元(赠 阅)

本书如有印刷、装订等质量问题,本社负责调换

版权所有,摇摇翻印必究

序

编写程序就像是在盖房子——看着“房子”一天一天盖起来,作为程序员的我们的心里,那份喜悦是不能用言语表达的。尤其是编写游戏程序,当你看到自己所写的游戏给大家快乐的时候,你的心里更是快乐无比。但是,这份快乐是有代价的,是与无数的不眠之夜分不开的。所以,每一个想成为游戏程序员的你们,在做这件事之前,一定要考虑清楚,你真的愿意把你的时间,你的青春奉献在这里吗?你愿意当别人在酒吧潇洒的时候,你却独自坐在计算机的面前,面对着一行行的程序吗?你愿意长时间地忍受失败,仅仅为了那一次成功吗?

如果你愿意,那么好,现在就开始,跟随这本书开始一段旅程,一段追逐梦想的旅程,你很快会发现,其实写游戏的程序很简单,只要你学会一定的规则,按照一定的规律,一个演示版很快就会出现你的面前。这也是本书的目的,这本书只给了你一把钥匙,一把进入游戏制作大门的钥匙。而只有你进了这个大门之后,你才能看到一片广阔的天地,等待你去驰骋。

游戏的程序就是一行行的代码,需要你的灵魂才能赋予它以生命,有了生命的游戏才能给大家快乐,也才能给你所想要的。你的游戏是你的热情,是你的汗水,是你的喜怒哀乐,是你的梦想,你的游戏其实就是你自己。

刘摇刚
于目标软件
二〇〇四年 元月 二日

目录

第一章 游戏编程基础.....	猿
1.1 游戏文化概述.....	猿
1.2 程序设计基础.....	猿
1.2.1 程序设计语言.....	猿
1.2.2 集成开发工具.....	猿
1.2.3 数据结构与算法.....	猿
1.2.4 数据的组织.....	猿
1.3 游戏程序设计基础.....	猿
1.3.1 从哪里开始.....	猿
1.3.2 选择什么语言.....	猿
1.3.3 阅读书籍.....	猿
1.3.4 计算机图形学.....	猿
1.3.5 汇编语言.....	猿
1.4 如何成为合格的游戏制作人员.....	猿
第二章 算法设计基础.....	猿
2.1 什么是算法.....	猿
2.2 伪代码.....	猿
2.3 流程图.....	猿
2.3.1 流程图概述.....	猿
2.3.2 流程图符号和使用.....	猿
2.3.3 流程图标准.....	猿
2.3.4 流程图应用实例.....	猿
2.3.5 绘制流程图的工具.....	猿
2.4 算法复杂度.....	猿
2.5 程序设计的常用算法.....	猿
2.5.1 排序.....	猿
2.5.2 递归.....	猿
2.6 游戏中的常用算法.....	猿
习题 2.....	猿
第三章 程序设计语言 悦垣.....	猿
3.1 从悦谈起.....	猿
3.2 悦垣圣战.....	猿
3.3 为什么用悦垣.....	猿
3.3.1 注释.....	猿
3.3.2 引用.....	猿
3.3.3 常量.....	猿
3.3.4 内存管理.....	猿
3.3.5 输入输出流.....	猿





猿猿猿名字空间	猿猿
猿猿猿映射	猿猿
猿猿猿重载	猿猿
猿猿猿异常	猿猿
猿猿猿悦垣基础	猿猿
猿猿猿面向对象	猿猿
猿猿猿运算符重载	猿猿
猿猿猿继承和多态	猿猿
猿猿猿模板	猿猿
猿猿猿杂项	猿猿
猿猿猿用好悦垣的几条建议	猿猿
猿猿猿标准悦垣和灾猿猿悦垣	猿猿
猿猿猿灾猿猿悦垣编译器	猿猿
猿猿猿选择灾猿猿悦垣的理由	猿猿
习题猿	猿猿
第四章猿数据结构基础	猿猿
猿猿猿线性表结构	猿猿
猿猿猿顺序存储结构	猿猿
猿猿猿链式存储结构	猿猿
猿猿猿线性表的应用实例	猿猿
猿猿猿数组	猿猿
猿猿猿二维数组的顺序存储结构	猿猿
猿猿猿二维数组的应用	猿猿
猿猿猿树与二叉树	猿猿
猿猿猿树与二叉树的存储结构	猿猿
猿猿猿树的应用实例	猿猿
习题猿	猿猿
第五章猿游戏中数据的组织	猿猿
猿猿猿使用猿猿猿动态链接库作为资源包	猿猿
猿猿猿建立一个猿猿猿工程	猿猿
猿猿猿在猿猿猿工程中加入资源文件	猿猿
猿猿猿读取资源包中的数据	猿猿
猿猿猿自定义资源包	猿猿
猿猿猿定义资源包格式	猿猿
猿猿猿管理资源包	猿猿
猿猿猿读取资源包	猿猿
习题猿	猿猿
第六章猿游戏的策划和实现	猿猿
猿猿猿游戏的类型	猿猿



远猿猿动作游戏	猿园
远猿猿策略游戏	猿员
远猿猿角色扮演游戏	猿猿
远猿猿模拟游戏	猿缘
远猿猿解谜游戏	猿苑
远猿猿游戏的表现形式	猿愿
远猿猿顶视角	猿愿
远猿猿斜视角	猿园
远猿猿第一人称视角	猿员
远猿猿第三人称视角	猿猿
远猿猿策划简单的游戏	猿匹
习题 远	猿延
第七章 远猿猿计算机图形学与三维美术基础	猿苑
远猿猿进入 猿猿的世界	猿愿
远猿猿什么是 猿猿	猿愿
远猿猿三维图元	猿怨
远猿猿一般的三维渲染技术	猿怨
远猿猿光栅图形和三维变换	猿圈
远猿猿光栅图形的显示原理	猿圈
远猿猿点与直线的光栅化	猿猿
远猿猿直线、圆、椭圆	猿猿
远猿猿平移	猿延
远猿猿旋转	猿苑
远猿猿缩放	猿怨
远猿猿图形裁减	猿圈
远猿猿点裁减	猿猿
远猿猿线裁减	猿猿
远猿猿多边形裁减	猿猿
远猿猿可见面	猿源
远猿猿可见面检测的算法	猿源
远猿猿隐藏线的消除	猿源
远猿猿背面剔除	猿延
远猿猿画家算法	猿苑
远猿猿在 猿猿云 中的算法	猿愿
远猿猿二叉空间分割树	猿员
远猿猿八叉树	猿猿
远猿猿光线跟踪方法	猿缘
远猿猿明暗效果	猿匹
远猿猿猿猿	猿员



第7章 摇匀学配学	页远
第8章 计算机动画	页苑
第8章 计算机动画的功能	页愿
第8章 关键帧系统	页愿
第8章 游戏美术初步	页怨
习题 苑	页怨
第八章 图形学基础知识	页员
第8章 顶点	页圆
第8章 可变顶点格式(云云)	页圆
第8章 顶点缓冲区	页猿
第8章 转换和照明管线(裁圆)	页源
第8章 概述	页源
第8章 世界转换	页缘
第8章 观察转换	页远
第8章 照明	页远
第8章 投影转换	页远
第8章 裁剪	页苑
第8章 除以憎——非均匀化	页愿
第8章 视口缩放	页愿
第8章 建立裁圆管线矩阵	页愿
第8章 世界矩阵	页愿
第8章 观察矩阵	页怨
第8章 投影矩阵	页圆
第8章 照明	页员
第8章 环境光	页圆
第8章 直射光	页圆
第8章 光的颜色和材质颜色	页猿
第8章 光源类型	页猿
第8章 光的属性	页缘
第8章 设置和获取光的属性	页远
第8章 启用和禁止照明	页远
第8章 启用和禁止某种光	页苑
第8章 渲染	页苑
第8章 渲染引擎和裁圆	页苑
第8章 索引点	页愿
第8章 渲染引擎	页怨
第8章 图元类型	页圆
第8章 设置渲染状态	页圆
第8章 纹理	页远



愿爱摇纹理坐标	愿远
愿爱摇建立纹理	愿苑
愿爱摇纹理压缩	愿苑
愿爱摇纹理过滤	愿愿
愿爱摇纹理寻址模式	愿园
愿爱摇纹理包装	愿猿
愿爱摇粤猿猿混合	愿缘
愿爱摇概念	愿缘
愿爱摇粤猿猿的例子	愿远
愿爱摇粤猿猿测试	愿苑
愿爱摇自左乘	愿愿
习题 愿	愿愿
第九章 摇组 装 起 来	愿怨
愿园摇不基于 阅裁框架的 阅裁应用程序	愿园
愿园摇阅裁框架	愿园
愿猿摇阅裁框架的生成	愿园
愿源摇用向导生成一个 阅裁框架	愿猿
愿缘摇第 员 个例子——绘制一个矩形	愿苑
愿缘愿摇定义顶点格式	愿苑
愿缘园摇定义点缓冲	愿愿
愿缘猿摇建立点缓冲	愿愿
愿缘源摇对点缓冲填充数据	愿怨
愿缘缘摇渲染点缓冲	愿园
愿缘园摇清除顶点缓冲	愿猿
愿缘苑摇编译运行	愿园
愿缘愿摇回忆一下	愿园
愿远摇第 圆 个例子——给矩形添加纹理	愿猿
愿远愿摇修改顶点格式	愿猿
愿远园摇定义纹理	愿猿
愿远猿摇给点缓冲添加纹理坐标	愿源
愿远源摇初始化纹理	愿源
愿远缘摇渲染时设置纹理	愿远
愿远园摇清理纹理	愿苑
愿远苑摇编译运行	愿苑
愿苑摇第 猿 个例子——渲染游戏场景	愿怨
愿苑园摇制作地面	愿怨
愿苑猿摇制作天空	愿源
愿愿摇第 源 个例子——渲染树木	愿员
愿怨摇第 缘 个例子——使用摄影机	愿缘



第 1 章 第 1 个例子——使用模型	1
第 2 章 第 2 个例子——组合起来	15
2.1 添加顶点格式的定义	15
2.2 添加成员变量	20
2.3 构造函数	23
2.4 初始化 阙设备的成员变量	25
2.5 恢复模型,并设置摄影机矩阵和透视矩阵	28
2.6 接收用户操作,刷新成员变量	31
2.7 渲染	33
2.8 最后一步,清理工作	35
2.9 编译运行	37
附录 计算机图形学(游戏)编程书籍	38



第一章

游戏编程基础





孕悦游戏文化概述

孕悦游戏已经有十几年的发展历史了,而游戏真正成为产业却只是在这几年。这些年来,新的软、硬件技术不断地推动这个行业的发展,并使之越来越蓬勃。而且,随着网络的出现,又为这个产业注入了新鲜的血液。

回顾人类文化的发展历史,任何一种新的娱乐方式的兴起,都不可避免地包含着想像力和技术这两种因素。

幻想是人类永恒的需要,它从不放过每一个展开翅膀的机会。一种为实用而创造出来的工具往往会被人们转变为一种想像的工具,人类的想像力面对种种不同的技术手段,似乎始终保持着惊人的适应和创造潜能。顽石化为庄严凝重的雕像,这是一种想像力创造的奇迹,那么当由“园”与“员”构成的枯燥无味的计算机语言转化为屏幕上变幻无穷的、曲折往返的电子游戏时,这不也是一种奇迹吗?不妨说,电子游戏是用最逻辑的东西创造感性,是用最抽象的东西创造直观。而实现这对矛盾的转换的正是人类的想像力。

然而,在赞叹人类想像力的同时,应当注意技术的因素在人类文化发展中的位置。正是技术手段更新带来的信息载体的更新,为新的幻想方式的展开,为新的艺术门类与娱乐形式的兴起提供了可能。而这一点往往是最容易被研究者所忽视的。

比如说史诗,它为什么会繁盛在早期人类社会,恰恰也是基于特定的技术因素。史诗繁盛的时代是一个口语的时代,在那个时代,文字虽然已经出现,但由于书写工具的限制,它不可能大量的记录,传播人类文献、文明依赖的是口耳相传,依赖的是人的记忆。所以口语的时代是一个韵文的时代,有韵的文字可以歌唱或吟诵,这对于人的记忆来说要容易一些。

而长篇小说的兴起无疑同印刷术的突破性进展有关。印刷术从技术上解决了限制文献大规模复制、传播的难题,为长篇大论的写作创造了条件。书写取代了吟诵,散文(广义的散文,指所有无韵的文字)取代了韵文。小说,这种更为自由随意的文学形式在文学创作中成为了最重要的门类。

今天的游戏,已经从之前的单纯用于娱乐的程序,发展成为一种“互动艺术”的表现形式。游戏本身并不预示着任何一种文化,它可以表达东方的东西,也可以表达西方的东西,换句话说游戏已经和音乐、建筑一样具备了作为文化传播媒介的一切特征。对文化传播媒介比较敏感的人或许会有这样的体会:音乐那种从听觉上令人心旷神怡,思想随之飞扬的感觉,就和欣赏建筑那种从线条上带来的美感不一样,王小波先生将这种感觉称之为韵律。我借用一下这个比喻,把话说下去——不同的媒介之间的区别,在于其韵律的不一。音乐的韵律是贴近听觉的韵律,绘画的韵律是贴近视觉的韵律,而游戏也拥有它自己的韵律,且它的韵律最贴近这个经典表达形式与科技相互冲击、交融的时代。就根据韵律,贯注以心灵的声音,人们创作出各种不同的游戏节奏,再让不同的节奏相互交错成曲,成为一个个优秀的作品,例如《轩辕剑》系列(图员源)。不难发现,脱离了游戏韵律,游戏节奏以及游戏中的人文材料便会顿失它们原本在游戏里面的意义,甚至沦为乏味的聒噪。回过头来看看许多人眼中的游戏文化,大抵是局部提取游戏题材,异化游戏韵律,刻意要它与另一种媒介的韵律相结合,不过这种异化以后的东西将不再属于游戏。



图 员猿猿 摇摇游戏已经成为了一种文化

猿猿 摇摇程序设计基础

无疑,程序设计是一个软件的灵魂。而且随着计算机程序设计语言的越来越智能化,程序设计也变得越来越容易,也有越来越多的人在尝试用一行一行的代码来实现自己的梦想。同样,也有越来越多的人准备投身到这个行业中来。

一直以来,有一个这样的观念,即“软件=数据结构+算法+代码”,虽然随着技术的进步以及不断扩大的软件规模,这样的观念受到了很大的冲击。但不可否认的是,目前大多数软件的设计还是没有突破这样一个公式。

猿猿 摇摇程序设计语言

程序设计语言是随着计算机的诞生而诞生的。最初的程序就是打孔机在纸上打的一个一个小孔,它们代表了计算机能够识别的二进制的代码。为了抛弃这种不直观而且容易出错的方法,人们逐渐的使用一些符号来代替难于记忆的二进制编码,并随之产生了程序设计语言。

早期的开发方法主要是面向过程,或者说面向处理的。这种对过程的强调是由于当时使用的程序设计语言主要是面向过程的语言,主要应用是批量的和基于文件的应用。开始于 20 世纪 50 年代的结构化方法是最有代表性的方法。再如,魏士和 悦士提出的符合软件工程模块化原则的结构化设计方法,至今仍在使用。

面向对象是一种新兴的程序设计方法,或者说它是一种新的程序设计模型,其基本思想是使用对象、类、继承、封装、消息等基本概念来进行程序设计。



它是从现实世界中客观存在的事物(即对象)出发来构造软件系统,并在系统构造中尽可能运用人类的自然思维方式,强调直接以问题域(现实世界)中的事物为中心来思考问题、认识问题。并根据这些事物的本质特点,把它们抽象地表示为系统中的对象,作为系统的基本构成单位(而不是用一些与现实世界中的事物相关比较远,并且没有对应关系的其他概念来构造系统)。这可以使系统直接地映射问题域,保持问题域中事物及其相互关系的本来面貌。

从程序设计的角度来看,面向对象的程序设计语言必须有描述对象及其相互之间关系的语言成分。这些程序设计语言可以归纳为以下几类:系统中一切皆为对象,对象是属性及其操作的封装体,对象可按其性质划分为类,对象成为类的实例,实例关系和继承关系是对象之间的静态关系,消息传递是对象之间动态联系的惟一形式,也是计算的惟一形式,方法是消息的序列。

下面是 C++ 语言的一些片断。C++ 作为一门新兴的、高度面向对象的语言,受到了越来越多用户的欢迎。

```

class Circle {
public:
    Circle(int r):radius(r) {}
    Circle(int r, int a):radius(r), angle(a) {}
    Circle() {}
private:
    int radius;
    int angle;
};

class Point {
public:
    Point(int x, int y):x(x), y(y) {}
private:
    int x;
    int y;
};

class Line {
public:
    Line(Point p1, Point p2):p1(p1), p2(p2) {}
private:
    Point p1;
    Point p2;
};

class Shape {
public:
    Shape():area(0) {}
    virtual int area() const = 0;
};

class Circle:public Shape {
public:
    Circle(int r):radius(r) {}
private:
    int radius;
};

class Point:public Shape {
public:
    Point(int x, int y):x(x), y(y) {}
private:
    int x;
    int y;
};

class Line:public Shape {
public:
    Line(Point p1, Point p2):p1(p1), p2(p2) {}
private:
    Point p1;
    Point p2;
};
    
```



就目前来说,比较流行的语言有 C++、C#、Python、Lua 和 JavaScript 等。它们中的大部分都是面向对象的语言。非面向对象的语言目前主要应用在 游戏引擎 环境中,在其他的领域已经应用得不多了。

集成开发工具

软件开发发展到今日,已经不完全是单纯的源代码的积累了,为了更好地进行程序设计,人们开发了一系列的辅助工具。而集成开发工具正是把这些工具集合起来,成为一个单独的软件包。一般的集成开发工具都集合了语言编译器、代码编辑器、程序调试器等等有利于程序设计的工具,如图 1-1 所示。

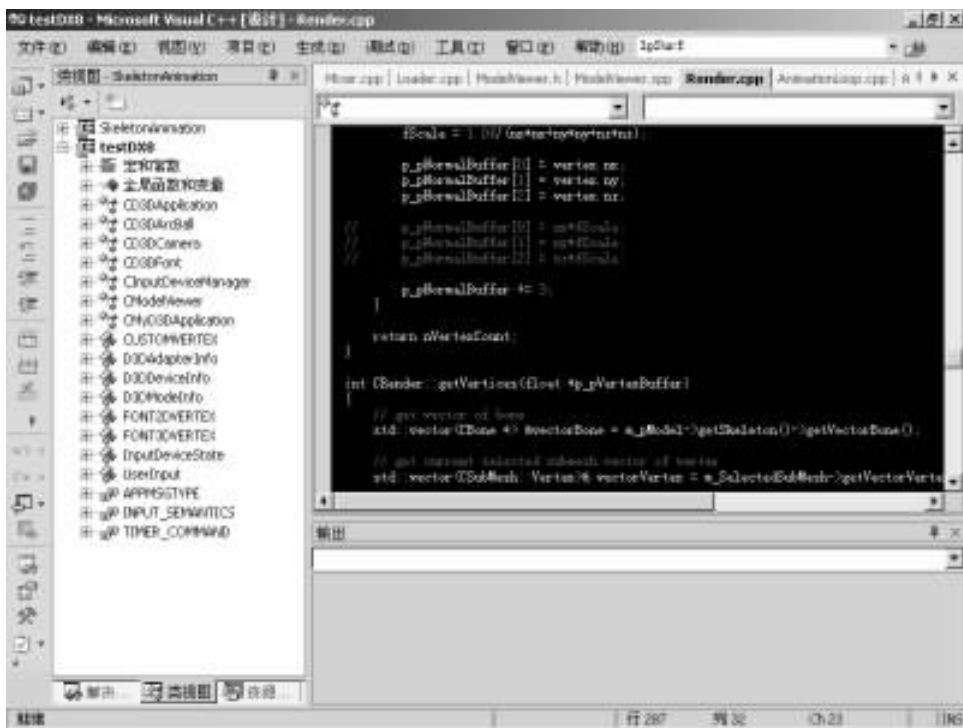


图 1-1 集成开发环境

随着集成开发工具的出现,程序设计的效率大大地提高了。以下列出了流行语言的集成开发工具。

- C++: Visual Studio, Code::Blocks, Eclipse, NetBeans
- C#: Visual Studio, Mono
- Python: PyCharm, Anaconda, IDLE
- Lua: Roblox Studio, Luau
- JavaScript: Visual Studio Code, WebStorm

以上列举的工具都是同类语言中开发工具的佼佼者,具有一定的普遍性。



线性表数据结构与算法

毫不夸张地说,数据结构的选择很大程度上决定了程序的复杂度。对于初学者来说,不应该被大学教材上那些看似高深的知识而吓倒。很多人认为只有掌握了树、图这样复杂的数据结构才算真正掌握了数据机构。其实不然,实际上在常规的程序设计中(不涉及到复杂的底层)树和图应用得很少。而且,深入地了解基础知识对理解树和图也是大有帮助的。

一般来说,复杂的数据结构都能被相对简单的数据结构所模拟,这样会带来一些灵活度,但也会导致控制难度的增大。

最基本的数据结构就是线性表、堆栈以及队列。而根据在内存中存放的方式不同也分为顺序存储和链式存储两种方式。学习数据结构,一定要对这些基础的东西有一个很深入的了解,才能在应用中游刃有余。

而算法的好坏是衡量一个程序性能的重要标准。尽管计算机硬件技术的飞速发展使得算法优化所能够起到的作用越来越小,但良好的算法素养一定能够在开发大型程序,特别是游戏中使你获益颇多。

最通用的两类算法就是排序和搜索,在实际的编程中,这两类算法的应用也比较广泛。根据实际情况的不同选择不同的算法能够达到最好的效率。

值得注意的是,目前的很多程序设计语言都加入了类似的算法库。悦垣甚至把这类的算法库加入到标准中来,称为 `std::vector`。下面就是一段 `std::vector` 的程序,它能够在 `Visual Studio` 中编译通过。

```

#include <vector>
using namespace std;

int main()
{
    vector<int> v;
    v.push_back(1);
    v.push_back(2);
    v.push_back(3);
    v.push_back(4);
    v.push_back(5);
    v.push_back(6);
    v.push_back(7);
    v.push_back(8);
    v.push_back(9);
    v.push_back(10);
    v.push_back(11);
    v.push_back(12);
    v.push_back(13);
    v.push_back(14);
    v.push_back(15);
    v.push_back(16);
    v.push_back(17);
    v.push_back(18);
    v.push_back(19);
    v.push_back(20);
    v.push_back(21);
    v.push_back(22);
    v.push_back(23);
    v.push_back(24);
    v.push_back(25);
    v.push_back(26);
    v.push_back(27);
    v.push_back(28);
    v.push_back(29);
    v.push_back(30);
    v.push_back(31);
    v.push_back(32);
    v.push_back(33);
    v.push_back(34);
    v.push_back(35);
    v.push_back(36);
    v.push_back(37);
    v.push_back(38);
    v.push_back(39);
    v.push_back(40);
    v.push_back(41);
    v.push_back(42);
    v.push_back(43);
    v.push_back(44);
    v.push_back(45);
    v.push_back(46);
    v.push_back(47);
    v.push_back(48);
    v.push_back(49);
    v.push_back(50);
    v.push_back(51);
    v.push_back(52);
    v.push_back(53);
    v.push_back(54);
    v.push_back(55);
    v.push_back(56);
    v.push_back(57);
    v.push_back(58);
    v.push_back(59);
    v.push_back(60);
    v.push_back(61);
    v.push_back(62);
    v.push_back(63);
    v.push_back(64);
    v.push_back(65);
    v.push_back(66);
    v.push_back(67);
    v.push_back(68);
    v.push_back(69);
    v.push_back(70);
    v.push_back(71);
    v.push_back(72);
    v.push_back(73);
    v.push_back(74);
    v.push_back(75);
    v.push_back(76);
    v.push_back(77);
    v.push_back(78);
    v.push_back(79);
    v.push_back(80);
    v.push_back(81);
    v.push_back(82);
    v.push_back(83);
    v.push_back(84);
    v.push_back(85);
    v.push_back(86);
    v.push_back(87);
    v.push_back(88);
    v.push_back(89);
    v.push_back(90);
    v.push_back(91);
    v.push_back(92);
    v.push_back(93);
    v.push_back(94);
    v.push_back(95);
    v.push_back(96);
    v.push_back(97);
    v.push_back(98);
    v.push_back(99);
    v.push_back(100);
}
    
```

