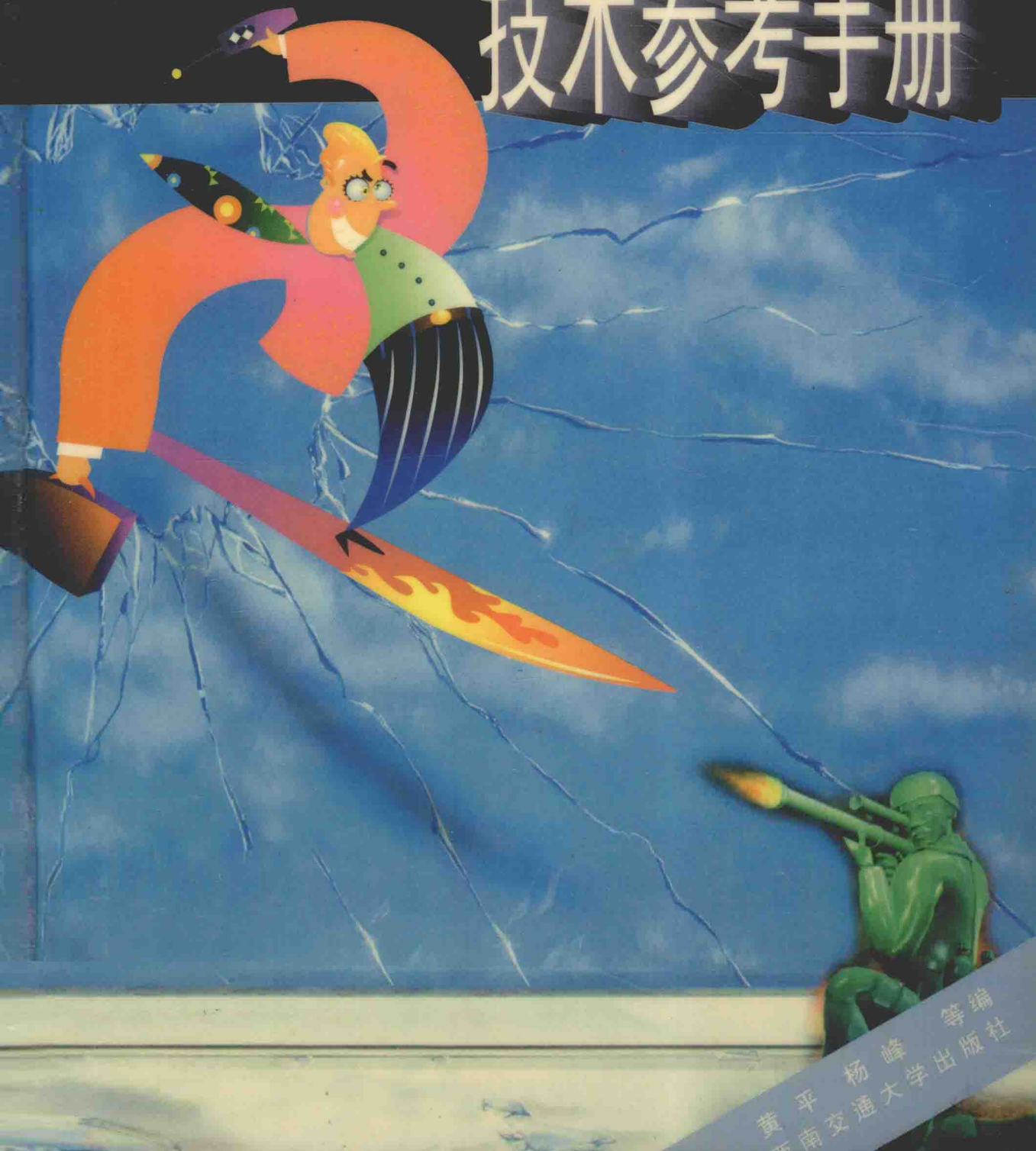
DirectDraw

技术参考手册



黄平 杨峰 等编

西南交通大学出版社

DirectDraw 技术参考手册

——————Direct X 6 技术参考手册系列丛书

黄 平 杨 峰 **等编**

西 南 交 通 大 学 出 版 社

·成 都·

内 容 提 要

DirectDraw 是 Direct X 中的一个重要成员，主要用于二维图形设计。利用 DirectDraw 可以直接访问在显示内存中的位图，利用硬件的位图移动和缓冲区页交换等技术来加速硬件和软件动画，并能够支持窗口裁剪和全屏显示。

利用 DirectDraw 可以容易地实现页交换，这是得到平滑无闪烁动画的关键。DirectDraw 还支持用于 3D 的 Z-Buffers（Z 缓冲）和有硬件支援的 Z 轴次序覆盖，并且支持硬件图像拉伸及同步存取标准和增强显存的特性。除了这些，DirectDraw 支持的特性还有：定制调色板和动态调色板、独占式硬件存取以及分辨率切换。

全书共分为两部分，第一部分主要介绍 DirectDraw 的一些基础知识；第二部分主要介绍 DirectDraw 中常用的函数、接口、结构和返回值等。

本书编写内容新颖、全面、通俗易懂，特别是提供了 DirectDraw 的许多新技术和编程新方法。本书非常适合作为广大用 DirectDraw 来进行游戏开发的人员的技术参考书。

DirectDraw 技 术 参 考 手 册

——Direct X 6 技术参考手册系列丛书

黄 平 杨 峰 等编

*

出 版 人 宋绍南

责任编辑 王 旻

封面设计 小 唐

西南交通大学出版社出版发行

（成都二环路北一段 111 号 邮编 610031）

成都市报华印装厂印刷

*

开本：787mm × 1092mm 1/16 印张：14.375

字数：341 千字 印数：1 ~ 5000 册

1999 年 6 月第 1 版 1999 年 6 月第 1 次印刷

ISBN7-81057-296-2/**T**·325

定价：22.00 元

目 录

第一部分 DirectDraw 基础

第二部分 DirectDraw 参考

第一部分　DirectDraw 基础

这一部份提供关于 DirectDraw 组件的信息，本部分内容分成以下几部分：

- 关于 DirectDraw
- 为什么要用 DirectDraw
- 开始——基本图形概念
- DirectDraw 组织
- DirectDraw 实质
- DirectDraw 指南
- DirectDraw 参考

第一章 关于 DirectDraw

DirectDraw 是一个 DirectX SDK (Software Develop Kot)组件，这个组件允许用户对显示内存，块传输硬件，硬件覆盖，及翻转平面直接进行操作。并且在提供这种功能的同时保持了甚于 Microsoft-Windows 应用和设备驱动程序的兼容性。

同时，DirectDraw 也是一个在保持和 Windows 图形设备界面（GDI）兼容的同时可以直接访问显示设备的软件接口。它并非高级的图形应用编辑接口（API）。DirectDraw 为游戏和 Windows 子系统软件（诸如 3-D 图形软件包，数据视频编码器等）提供了一种与设备无关的方法来获取对特殊显示设备特性的访问。

DirectDraw 可以对很多显示设备进行操作，从简单的 SVGA 显示器到高级可提供剪切、伸展和支持非 RGB 图像格式的硬件工具。DirectDraw 设计成让用户的应用可以——使用基础硬件的功能以及使用任何支持硬件加速的特点。在硬件中没有实现的特点可用 DirectX 来模拟。

DirectDraw 可以使用一种与设备无关的方法来提供对显示内存设备相关的访问。本质上，DirectDraw 在操纵显示内存。用户的应用程序只需识别一些基本的依赖设备，但这些在每种硬件工具中又是标准的，例如：RGB YUY 图像格式及在光栅线之间的峰值。用户无需调用特殊流程来使用或进行调色板登记操作。

使用 DirectDraw，用户可以非常容易地直接操纵显示内存，充分利用不同类型显示硬件的，解压能力而不必依赖于某一特殊硬件。

DirectDraw 可在运行 Windows 95, Windows NT 4.0 及以后版本的机子上提供高质量的游戏图形。

第二章 为什么要用 DirectDraw

DirectDraw 组件为用户带来了许多强大的功能，使用 DirectDraw，Windows 图形编程人员：

- DirectDraw 的硬件抽像层（HAL）提供一种连续接口，通过它可以直接对显示和视频内存进行操作，并从系统硬件中获取最大的功能。
- DirectDraw 可评估视频硬件能力，在任何可能的情况下使用特定硬件的特性。例如，若你的视频卡支持硬件 blit，DirectDraw 代替硬件 blit，将极大地提高性能。并且，当硬件不支持这些特点时，DirectDraw 提供一个硬件模拟层（HEL）来支持。
- Windows 95 中的 DirectDraw 从操作系统提供的 32 位内存寻址方式及平面内存模式中获取了好处。DirectDraw 以大存储块的方式而非小段方式描述视频和系统内存。如果用户使用过段/偏移量寻址方式，他们将会很快欣赏这种“平面”内存模式。
- 在全屏的应用系统中 DirectDraw 通过后备缓冲使页面翻转很容易，详细介绍请看“页面翻转和后备缓冲”。
- 支持全屏或 Windows 剪切。
- 支持 3-D Z 缓冲
- 支持硬件层的 Z 排序
- 可访问图像扩展硬件
- 可同步访问标准和增强显示设备内存区域
- 其它特性如标准和动态调色板，独占硬件访问方式及分辨率转换。

这些特性集成在一起，使得用户编写的应用程序可以很容易地在基于标准 Windows GDI 上的应用上运行，甚至是在 MS-DOS 应用也可运行。

第三章　开始——基本图形概念

这一部分描述了用 DirectDraw 进行图形编程的概况。这儿讨论的概念都是开始粗略介绍它的非技术描述，接着再给出 DirectDraw 怎样支持它的信息。

如果你是对图形有关概念很熟悉，你可以完全跳过这一部分而直接阅读包括在 DirectDraw 实质部分中更详细的内容。如果你对 Windows 下，C 和 C++编程很熟悉，对理解这一部分不含有困难，当你看完以下要点后，将对基本的 DirectDraw 图形编程概念有一个比较深刻的理解。在此将讨论以下要点：

- 设备无关的位图
- 绘图表面
- 位块传输概念
- 页面翻转与后备缓冲
- 矩形介绍
- 单色画面概念

3.1　设备无关位图

Windows 和 DirectX 使用设备无关位图作为它的默认图形文件格式。从本质上来说，一个设备无关位图文件包括以下信息：图像维数，所用颜色数，描述这些颜色的值以及描述每一像素的数据。另外，DIB 还包括一些较少使用的参数，像文件压缩信息，有效色彩（若并非全部色彩都被使用），图像的物理维数（在打印情况下）。DIB 文件通常后缀为“.bmp”，有时后缀也会为“.dib”。

因为 DIB 在 Windows 编程中很普遍，SDK 开发平台已包括许多用 DirectX 可以使用的函数。例如，以下从 SDK 开发平台的 DirectX API 中取出的函数结合了 Win 32 和 DirectX 函数从而把 DIB 装载到 DirectX 表面。

```
extern "C" IDirectDrawSurface * DDLoadBitmap(IDirectDraw *pdd,
    LPCSTR szBitmap, int dx, int dy)
{
    HBITMAP             hbm;
    BITMAP              bm;
    DDSURFACEDESC       ddsd;
    IDirectDrawSurface *pdds;
  //
  //    这是 Win32 部分。
  //    将 bitmap 作为资源装载进来，如果失败，则作为文件来试.
  //
    hbm = (HBITMAP)LoadImage(GetModuleHandle(NULL), szBitmap, IMAGE_BITMAP, dx,
```

```
dy, LR_CREATEDIBSECTION);

        if (hbm == NULL)
            hbm = (HBITMAP)LoadImage(NULL, szBitmap, IMAGE_BITMAP, dx, dy,
LR_LOADFROMFILE|LR_CREATEDIBSECTION);

        if (hbm == NULL)
            return NULL;

        //
        // 得到 bitmap 的大小。
        //
        GetObject(hbm, sizeof(bm), &bm);

        //
        // 现在返回 DirectX 函数调用。
        // 为 bitmap 产生一个 DirectDrawSurface。
        //
        ZeroMemory(&ddsd, sizeof(ddsd));
        ddsd.dwSize = sizeof(ddsd);

        ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT |DDSD_WIDTH;
        ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;
        ddsd.dwWidth = bm.bmWidth;
        ddsd.dwHeight = bm.bmHeight;

        if (pdd->CreateSurface(&ddsd, &pdds, NULL) != DD_OK)
            return NULL;

        DDCopyBitmap(pdds, hbm, 0, 0, 0, 0);

        DeleteObject(hbm);

        return pdds;
    }
```

有关 DIB 文件的详细内容，请看 SDK 开发平台。

3.2 绘图表面

绘图表面接收视频数据最终以图像（确切地说是位图）显示在屏幕上。在大多数 Windows 程序中，用户用像 GetDC 这种函数访问绘图表面。GetDC 代表获取设备内容（DC）。在获取了设备内容后，可以开始在屏幕上绘图。但是，Win 32 的图形函数在系统的另一部分提供，即在图形设备界面（GDI）GDI 是一个系统组件，这个组件提供一个抽像层使得标准

Windows 应用程序可以在屏幕上绘图。

GDI 的弊端是它并不是为高性能的多媒体软件设计的。它是为商业应用软件如字处理器，电子表格等设计的。GDI 提供对系统内存中视频缓冲的访问，而非视频内存，并且没有充分利用视频卡所提供的特点。总而言之，GDI 对于大部分商业软件来说是优秀的，但对于多媒体或游戏软件来说，则执行太慢了。

在另一方面，DirectDraw 可以给用户提供一个代表实际视频内存。这意味着当用户使用 DirectDraw 时，可以直接在视频卡的内存上写数据，使用户的图形例程运行非常快。这些表面用相邻的内存块来代表，这使得在其中寻址非常容易。

详细内容请见“表面”。

3.3 Blitting 概念

“blit”是位块传输的缩写。所谓位块传输是从内存中的一个位置把数据块传到另一位置的过程。图形编程人员利用位块传输把图形从内存中的一个位置传到另一个位置。位块传输经常用于子画面的动画，这个将在后面进行讨论。详细内容请见“子图形概念”。

用户可以用 IDirectDrawSurface3::Bit 和 IDirectDrawSurface3::BitFast 方法进行块传输。

3.4 页面翻转和后备缓冲

页面翻转是多媒体、动画、游戏软件中的关键。页面翻转是卡通艺术家把他们的想像生活化的模拟。例如，艺术家在一张纸上画了一幅图。然后把它放在一边接着画另一幅图，对每一幅图，艺术家只是作了轻微的改动，因此当在两个页面间翻转时，图像看起来就像动画一样。

软件中的页面翻转和上述过程非常相似。首先，用户设置一系列 DirectDraw 平面，这些平面将像艺术工作人员翻转页面一样翻转。第一个平面称为开始平面，后面的平面称为后备缓冲。应用程序往后备缓冲中写数据，然后翻转主（开始）平面后备缓冲中的数据将显示在屏幕。当系统显示图像时，应用软件将重新向缓冲中写数据。只要在显示动画，这个过程就一直持续。使动画显示快速、有效。

DirectDraw 很容易设置翻转方案，从一种相对简单的双缓冲方案（一个开始页面和一个后备缓冲）到更复杂的方案，这一种复杂的方案有许多附加缓冲。详细内容请参看翻转平面。

3.5 矩形介绍

在整个 DirectDraw 和 Windows 编程中，提到屏幕上对象时一般用约束矩形这个术语。一个约束矩形用两点描述：左上角和右下角的两点。大部分应用程序用 RECT 这个结构来描述这个约束矩形。当作位块传输到屏幕或作命中检测。RECT 结构定义如下：