软件封面特技显示的语言实现 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/504/2021_2022__E8_BD_AF_E4_BB_B6_E5_B0_81_E9_c67_504943.htm 考试大计算机等级站整理收集： 软件编制人员都希望自己的软件能有一个漂亮的封面，如果能将图形动画技术应用到封面设计中，无疑会使封面更加美观醒目，为应用软件锦上添花。 本文提供了一种封面设计技术，能模拟摄像机推拉镜头的效果，一推一拉，极具动态。 我们知道，计算机图形处理的数据量非常大，要求的速度也很快。因此往往将图形处理软件固化成硬卡(如2.13汉字系统的神笔CAD卡)，有了图形处理卡，对图形应用的编程相对要容易些，然而，由于资金等问题，大多数的PC用户都不具备此设备，难道说就只能望“ 卡 ”兴叹了么?回答当然是否定的，本文给出的就是一种纯软件方式的图形动画技术。它不要求额外的设备，只要有EGA/VGA适配器即可。 为实现图形的动画效果，例如实现软件封面汉字标题的推拉镜头效果，只要将一幅幅不同大小的汉字画面在很短的时间内依次显示在屏幕上即可。这里涉及到一个问题，那就是在显示下一个画面之前先要清屏，然后再写下一个画面。无论二者的速度多快，都将影响动画效果，用户能感受到写屏和清屏的过程。这将大煞风景。本文采用“ 幕后组织 ”的方法解决了这一问题。在EGA/VGA的10H模式下，视频缓冲区被分成二页，一个页为当前显示页，其内容即为屏幕上显示的内容；一个为输出活动页，所有的视频输出都针对此页，它可以是当前显示页，也可以不是。在Turbo C中提供了两个函数，setactivepage和setvisualpage分别完成设置图形输出活

动页和设置图形可见页。如果我们将输出活动页设为非当前显示页，将一些不想让用户看到的处理过程放到活动页这一“幕后”去处理。然后切换当前显示页和活动输出页，那么用户看到的就是我们想让其看到的内容了。为实现汉字标题的推拉镜头效果，需要在屏幕上显示不同大小的汉字，许多汉字系统都提供了汉字的放大功能，但一则需要汉字系统的支持，且有特殊的控制命令，在C语言中不易调用；二则其放大和缩小的级差皆为整数倍，不能体现缓慢变化的过程。所以笔者编制了一个西文状态下的汉字放大和缩小的函数，可用来完成汉字的任意放大和缩小(例如0.2倍)。同时，由于无须装载汉字系统，因而可节省大量的内存。有了不同大小的汉字，只要在短时间内将之依次显示在屏幕上，就能产生动画效果。由于汉字的输出是采用在屏幕上画点的方法，所以速度很慢，简直让人无法忍受(其它的图形输出也存在这个问题)。最好是将图形整个存储，以便回显时获得较快的速度。C语言中有两个函数getimage和putimage能完成此功能，其响应速度是非常快的。但是由于这两个函数都是将图像保存在内存中，而图像的信息量往往很大，这就是说只能保存有限的图像，不能满足我们的要求。笔者在这里采用了用文件来存储图像，突破了内存的限制，从理论上讲可以存储无限大。makeface程序用来产生封面，它创建两个文件:pic为二进制图像文件，保存一幅幅图像，textpic为一辅助文件，保存每一幅图像的长度。dispface用来显示封面，将pic中的内容依次读到内存，用putimage函数写回屏幕上。dispface完成封面显示的速度很快，如果将pic和textpic放在虚拟盘中，那么效果就更好了。需要说明的是，尽管采用用文件来存储图像可以

不受内存空间的限制，但由于getimage和putimage函数现将图像放在内存中，如果图像很大，占用内存太多时，程序将会发生不可预料的结果。所以在存储较大的图像时，应采用分块存储，再分块会显的方法。一般来说一块以不超过一屏的三分之一(639*120)为好。 本文仅提供一个简单的例子，用makeface建立pic和textpic之后，执行dispface程序，可在屏幕上看到蓝地红字的“ 汉字特技显示 ”几个字由大到小逐渐推远，再由小到大逐渐拉近，最后定于屏幕中央。参照本文，加以扩充和完善，相信不难编出漂亮、醒目的动画封面。 本程序的运行环境为Turbo C 2.0，EGA/VGA显示器。 附:makeface.c,jputhzc和dispface.c /*jputhz.c*/ /*调用格式:hzdisp(X列，Y行，横扩倍数，纵扩倍数，颜色，要显示的汉字)*/ #include "stdio.h" #include "graphics.h" #include "string.h" #include "stdlib.h" void hzdisp (x,y,x-rate,y-rate,color,cstring) unsigned int x,y. float x-rate,y-rate. int color. char cstring[]. { FILE *fp. reGISter int n=0,i,j,k. register int x1,y1. char dot[73]. char sec1,sec2. unsigned long index. fp=fopen("c:\\213\\hzk24s","rb"). if(fp==NULL){printf("Cant open hzk\n").exit(1).} while(*cstring)} n . sec1=cstring-160. sec2=(cstring 1)-160. index=(sec1-16)94 sec2-1. index=72. if(fseek(fp,index,0)){ restorecrtmode(). printf("File seek error !\n"). fclose(fp). exit(1). } if((fread(dot,1,72,fp))!=72){cstring .continue.} cstring . cstring . for(i=0.i for(j=0.j for(k=0.k if(dot[i3 j] >>(7-k)&amp.1) { x1=x-rate(i 25n). y1=y-rate(j8 k). if((x-rate else{fillellipse(x1 x,y1 y,x-rate,y-rate).} } } fclose(fp). } /----------/ #include "jputhz.c" FILE fp1,text. // savescreen(int x0,int y0,int x1,int y1) { unsigned int

size. char bufs. unsigned int piclong. int ss. ss=sizeof(unsigned int). piclong=(unsigned int)malloc(ss). size=imagesize(x0,y0,x1,y1). piclong=size. fwrite(piclong,ss,1,text). bufs=(char)malloc(size). if(bufs==NULL) {puts("Memory alloc fail !/n").getch().return(0).} getimage(x0,y0,x1,y1,bufs). if(fwrite(bufs,size,1,fp1)!=1) {puts("fwrite fall").return(0).} free(bufs). } //