在VB中进行各种图形切换的方法 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/459/2021_2022__E5_9C_A8VB_E4_B8_AD_E8_BF_c97_459709.htm 图形的切换在有关于多媒体的应用中是经常见到的。常见的切换方式可以概括为以下几种。 切换(Cut)：当前图象快速的被另一幅图象所取代 淡入(Fade In)：当前图象缓缓变黑至消失 淡出(Fade Out)：一幅图象缓缓的从黑色的屏幕中出现 隐现(Dissolve)：一幅图象缓缓的变为另一幅图象 滑入(Wipe)：一幅图象逐渐穿过并覆盖当前图象 拉进(Slide)：一幅图象从屏幕一边匀速滑入 弹进弹出(Pop on/off)：一幅图象立即出现或消失 上拉下拉(Pull Down/Up)：一幅图象象窗帘一样从屏幕项部拉下，并以相同方式返回 翻转(Flip)：当前图象翻转，在其反面显示另一图象 旋转(Spin)：图象以旋转方式出现 VB和Windows API为生成切换效果提供了有力的工具。图形切换的实质是通过快速组合或拷贝来建立分离的图象。在某些情况下，建立切换效果只不过将当前图象的属性由Visible转变True，如上述效果中的弹进弹出、切换等，而多数情况下，切换效果需要调用Windows的GDI函数来完成。 图形的切换和弹进弹出是最容易实现的例子，前面已经解释过这两种切换效果的含义了。在Form中拉出Image1和Image2，注意，Image1和Image2的坐标和大小应完全一样。并设置好事先准备好的两幅图形。将Image1的Visible设为True，再将Image2的Visible设为false，双击Form，输入如下程序代码： Private Sub Form_Click() Image1.Visible=False Image2.Visible=True End Sub 在程序运行时，应只能看到Image1。这时，点击鼠标，即可看Image1快速

的被Image2所代替。形成图形的切换效果。这时再将Image1和Image2的坐标错开，不要让两个图形有重叠部分。这时再运行程序，点击鼠标，Image1消失的同时， Image2在另一个地方出现。形成图形的弹进弹出效果。 拉进和滑入是稍微复杂点的切换方式。在Form1中加入Timer组件，设interval为1，再在Form1中加入PictureBox1，假设PictureBox1的Left为-3000，Width为3000；Form1的Width为9000，双击Timer组件，加入下列程序代码： Private Sub Timer1_Timer() If Picture1.Left > 6000 Then Picture1.Left = 6000 Else Picture1.Move Picture1.Left 100 End If End Sub 运行程序时，由于Pictrue1在边界外，所以看不见，随着Picture.Left 100，Pictrue1由边界外缓缓滑入Form中，当Picture1距左边界大约6000时，也就大致在Form1中间位置时，Picture1停止运动。形成了图形从屏幕边或一角平稳滑入的拉入效果。而如果在Picture1静止后的位置上事先设置好和Picture1一样大小的Picture2，这样又形成下一图形逐渐穿过并覆盖当前图形的滑入效果。 上述算法都很简单，也能满足一般的编程需求，不过在进行一些高要求的编程时，这些效果则略显粗糙了，但调用Windows本身的GDI函数则可满足一些近似苛求的效果。下面我们通过调用GDI函数重写上面的滑入效果。 在Form1中设两个尽可能大小相同的picCurrentPicture和picNextPictrue，并设picCurrentPicture的Visible为True，picNextPicture的Visible设为False；Autosize设为True，ScaleMode设为Pixel，再在Form1中加入一个Command和一个Timer。程序编写过程如下： 先对公共部分进行说明，可以直接从VB中的Text API View中将有关BitBlt的声明复制过来，具体内容如下： Declare Function BitBlt Lib

"GDI" (ByVal hDestDC As _Integer, ByVal X As Integer, ByVal Y As Integer, _ByVal nWidth As Integer, ByVal nHeight As Integer, _ByVal hSrcDC As Integer, ByVal XSrc As Integer, _ByVal YSrc As Integer, ByVal dwRop As Long) As IntegerConst lngSRCCORY=&amp.HCC0020Const intSTEPS=64Dim intCurrentPictrueWidth As Integer, _intWidthChange As Integer, intPieceToAdd As Integer, intNextPiece As Integer 双击Timer组件，输入下列代码：Private Sub Timer1_Timer() Dim intI As Integer IntNextPiece=BitBlt(picCurrentPicture.hDC,0,0, _ intPieceToAdd,PicCurrentPicture.ScaleHeight, _ picNextPicture.hDC,0,0,lngSRCCOPY) IntPieceToAdd=intPieceToAdd intWidthChang IntI=intI 1 If(intI-intSteps) >0 Then Timer1.Enabled=FalseEnd Sub 双击Command1，输入如下代码：Private Sub Command1_Click() intCurrentPictureWidth=picCurrentPicture.ScaleWidth intPieceToAdd=intWidthChange Timer1.Enabled=TrueEnd Sub 这时再运行程序，点击Command1，怎么样，这次滑入的效果是不是比用前一种方法要精细的多？ 另外常用到的还有翻转、旋转以及淡入淡出。限于篇幅，这里只作简要介绍。 在翻转效果中，显示每一图形的反面并不重要，关键在于需要用到一块空间来改变它。同时也必须考虑图象大小不同的情况。另外，通过跟踪某些关键事件并使用它们检测图象应该变大还是变小，可以使程序更加简练。 假设一矩形，中心画出一条垂直线段，切换从矩形上最宽时开始，当切换的前一半完成时，矩形中心线两边的部分都缩小为窄长条，在这一过程中其宽度按一固定量(设为TempWidth)减小，在切换的后

半过程中，矩形将会扩展到新图象的尺寸。 这种缩小和扩大的完成方式将决定切换过程的光滑程度。在程序中，矩形的宽度减小某一增量(TempWidth),图形将在新矩形中重画(通过伸缩使之适合新的大小)，再用Move 方法将图形向左或向右移动增量大小的一半。通过调整Timer控件的速度和增量的大小，就能生成预期的效果。 至于淡入淡出，相对来说有些复杂，不过原理只是把所显示的图形的各个像素随机变黑或是将随机的将变黑的象素变回原色，也需要调用GDI函数，详细说明需用较大的篇幅，在这里就不再过多的叙述了。