第 11 章

图形的保存和重绘

在前面章节的学习中，我们发现这样的情况会经常发生：当程序窗口发生重绘时（例如窗口尺寸发生变化后），已在窗口中绘制的图形会消失无踪。本章将介绍如何保存窗口中已绘制的图形，并当窗口发生重绘时，如何再现窗口中先前的内容。

11.1 坐标空间和转换

Microsoft Windows 下的程序运用坐标空间和转换来对图形输出进行缩放、旋转、平移、斜切和反射。

11.1.1 坐标空间

一个坐标空间是一个平面的空间，通过使用两个相互垂直并且长度相等的轴来定位二维对象，如图 11.1 所示。

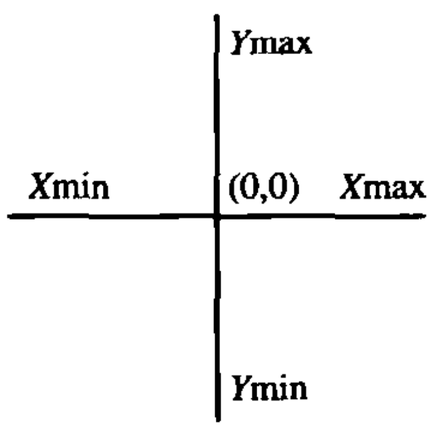

图 11.1　坐标空间示意图

Win32 应用程序编程接口（API）使用四种坐标空间：世界坐标系空间、页面空间、设备空间和物理设备空间。应用程序运用世界坐标系空间对图形输出进行旋转、斜切或者反射。**Win32 API 把世界坐标系空间和页面空间称为逻辑空间。**最后一种坐标空间（物理设备空间）通常指应用程序窗口的客户区，但是它也包括整个桌面、完整的窗口（包括框架、标题栏和菜单栏）或打印机的一页或绘图仪的一页纸。物理设备的尺寸随显示器、打

印机或绘图仪所设置的尺寸而变化。

11.1.2 转换

如果要在物理设备上绘制输出，Windows 就把一个矩形区域从一个坐标空间复制到（或映射到）另一个坐标空间，直至最终完整的输出呈现在物理设备（通常是屏幕或打印机）上。

如果应用程序调用了 SetWorldTransform 函数，那么映射就从应用程序的世界坐标系空间开始；否则，映射在页面空间开始进行。在 Windows 把矩形区域的每一点从一个空间复制到另一个空间时，它采用了一种被称作转换的算法，**转换是把对象从一个坐标空间复制到另一个坐标空间时改变（或转变）这一对象的大小、方位和形态**，尽管转换把对象看成一个整体，但它也作用于对象中的每一点或每条线。如图 11.2 所示是运用 SetWorldTransform 函数而进行的一个典型转换。

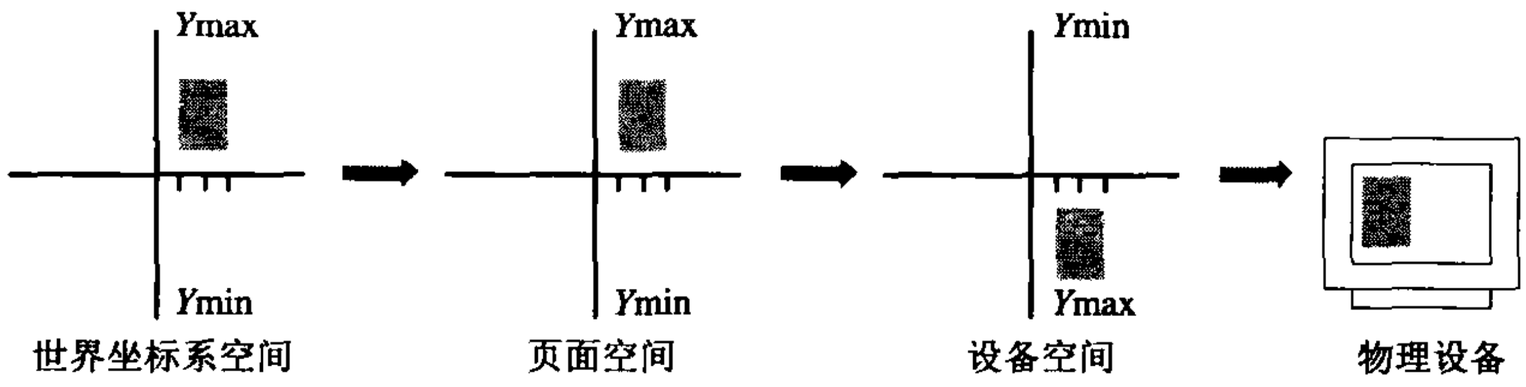

图 11.2 调用 SetWorldTransform 函数后程序进行的转换过程

在实际绘图时，世界坐标系空间中的一个区域要先被映射到页面空间，然后再由页面空间映射到设备空间。对设备空间来说，通常它的左上角是坐标点（0,0），向右是 x 增加的方向，向下是 y 增加的方向。然后再由设备空间映射到物理设备空间（通常就是屏幕），于是，图形就在计算机的屏幕上显示出来了。我们平时在开发程序时，如果需要把图形的某个局部区域放大，就可以利用世界坐标系和转换来完成，读者可以参看如例 11-1 所示的由 MSDN 提供的这个例子。

例 11-1

```
void TransformAndDraw(int iTransform, HWND hWnd)
{
    HDC hDC;
    XFORM xForm;
    RECT rect;

// Retrieve a DC handle for the application's window.

hDC = GetDC(hWnd);

// Set the mapping mode to LOENGLISH. This moves the
// client area origin from the upper left corner of the
// window to the lower left corner (this also reorients
// the y-axis so that drawing operations occur in a true
// Cartesian space). It guarantees portability so that
```

```
// the object drawn retains its dimensions on any display.

SetGraphicsMode(hDC, GM_ADVANCED);
SetMapMode(hDC, MM_LOENGLISH);

// Set the appropriate world transformation (based on the
// user's menu selection).

switch (iTransform)
{
    case SCALE:       // Scale to 1/2 of the original size.
         xForm.eM11 = (FLOAT) 0.5;
         xForm.eM12 = (FLOAT) 0.0;
         xForm.eM21 = (FLOAT) 0.0;
         xForm.eM22 = (FLOAT) 0.5;
         xForm.eDx  = (FLOAT) 0.0;
         xForm.eDy  = (FLOAT) 0.0;
         SetWorldTransform(hDC, &xForm);
         break;

    case TRANSLATE:   // Translate right by 3/4 inch.
         xForm.eM11 = (FLOAT) 1.0;
         xForm.eM12 = (FLOAT) 0.0;
         xForm.eM21 = (FLOAT) 0.0;
         xForm.eM22 = (FLOAT) 1.0;
         xForm.eDx  = (FLOAT) 75.0;
         xForm.eDy  = (FLOAT) 0.0;
         SetWorldTransform(hDC, &xForm);
         break;

   case ROTATE:     // Rotate 30 degrees counterclockwise.
         xForm.eM11 = (FLOAT) 0.8660;
         xForm.eM12 = (FLOAT) 0.5000;
         xForm.eM21 = (FLOAT) -0.5000;
         xForm.eM22 = (FLOAT) 0.8660;
         xForm.eDx  = (FLOAT) 0.0;
         xForm.eDy  = (FLOAT) 0.0;
         SetWorldTransform(hDC, &xForm);
         break;

   case SHEAR:       // Shear along the x-axis with a
                     // proportionality constant of 1.0.
         xForm.eM11 = (FLOAT) 1.0;
         xForm.eM12 = (FLOAT) 1.0;
         xForm.eM21 = (FLOAT) 0.0;
         xForm.eM22 = (FLOAT) 1.0;
         xForm.eDx  = (FLOAT) 0.0;
         xForm.eDy  = (FLOAT) 0.0;
```

```
            SetWorldTransform(hDC, &xForm);
            break;

        case REFLECT:     // Reflect about a horizontal axis.
            xForm.eM11 = (FLOAT) 1.0;
            xForm.eM12 = (FLOAT) 0.0;
            xForm.eM21 = (FLOAT) 0.0;
            xForm.eM22 = (FLOAT) -1.0;
            xForm.eDx  = (FLOAT) 0.0;
            xForm.eDy  = (FLOAT) 0.0;
            SetWorldTransform(hDC, &xForm);
            break;

        case NORMAL:      // Set the unity transformation.
            xForm.eM11 = (FLOAT) 1.0;
            xForm.eM12 = (FLOAT) 0.0;
            xForm.eM21 = (FLOAT) 0.0;
            xForm.eM22 = (FLOAT) 1.0;
            xForm.eDx  = (FLOAT) 0.0;
            xForm.eDy  = (FLOAT) 0.0;
            SetWorldTransform(hDC, &xForm);
            break;
    }

    // Find the midpoint of the client area.

    GetClientRect(hWnd, (LPRECT) &rect);
    DPtoLP(hDC, (LPPOINT) &rect, 2);

    // Select a hollow brush.

    SelectObject(hDC, GetStockObject(HOLLOW_BRUSH));

    // Draw the exterior circle.

    Ellipse(hDC, (rect.right / 2 - 100), (rect.bottom / 2 + 100),
        (rect.right / 2 + 100), (rect.bottom / 2 - 100));

    // Draw the interior circle.

    Ellipse(hDC, (rect.right / 2 -94), (rect.bottom / 2 + 94),
        (rect.right / 2 + 94), (rect.bottom / 2 - 94));

    // Draw the key.

    Rectangle(hDC, (rect.right / 2 - 13), (rect.bottom / 2 + 113),
        (rect.right / 2 + 13), (rect.bottom / 2 + 50));
```