第 5 章　二维变换与裁剪

本章主要讲解二维基本几何变换矩阵和二维图形裁剪。二维几何变换使得二维图形运动起来，借助双缓冲技术可以生成二维图形动画。二维几何变换包括平移变换、比例变换、旋转变换、反射变换和错切变换。任何仿射变换总可以表示为这 5 种变换的组合。二维变换中，平移变换用加法处理，其余变换用乘法处理。为了将二维变换统一表示为一个矩阵，即用一种一致的乘法处理二维变换问题，需要消除矩阵的加法运算，为此引入了点的齐次坐标。

二维裁剪属于二维观察的内容。窗口建立在观察坐标系、视区建立在屏幕坐标系。为了减少窗视变换的计算量，本教材中假定窗口与视区的大小一致。通过窗口观察场景时，窗口边界会对场景进行自然裁剪，然后将可见部分绘制到视区中。常用的裁剪算法有 Cohen-Sutherland 裁剪算法、中点分割裁剪算法和 Liang-Barsky 裁剪算法。本章最后介绍 Sutherland-Hodgman 逐边裁剪算法。多边形裁剪算法要求裁剪后的图形边界是自然闭合的，也就是说，裁剪窗口的部分边界做了裁剪后多边形的边界。

5.1　知识点

(1) 几何变换：指对图形进行平移、比例、旋转、反射和错切，使得图形运动起来。其中前平移变换和旋转变换的组合是刚体变换，变换后的物体不会发生变形。平移、比例、旋转变换的组合是仿射变换，具有保持直线平行的特性，但是不保持长度和角度不变。剪切变换也是仿射变换，将正方形变为平行四边形。

(2) 齐次坐标：就是用 $n+1$ 维矢量表示 n 维矢量。我们知道，点的平移变换用加法表示，而比例变换和旋转变换用乘法表示。如果点表示为齐次坐标，则 3 种变换都用乘法处理。

(3) 二维变换矩阵：是一个 3×3 的方阵，从功能上分为 4 个子矩阵。左上角 2×2 的方阵对图形进行比例、旋转、反射和错切变换；右上角 2×1 的矩阵对图形进行平移变换；左下角 1×2 的矩阵对图形进行投影变换；右下角的 1×1 方阵对图形进行整体比例变换。

(4) 二维基本变换：是指相对坐标系原点或坐标轴进行的几何变换。

(5) 复合变换：是指图形做了一次以上的基本变换。复合变换矩阵是基本变换矩阵的组合形式。

(6) 世界坐标系：是描述其他坐标系需要的参考框架，常用于建立三维场景。

(7) 建模坐标系：描述物体几何模型的坐标系，主要用于建立物体的三维模型。

(8) 观察坐标系：是在世界坐标系中定义的直角坐标系，主要用于指定图形的输出范围。

(9) 屏幕坐标系：是整数域二维直角坐标系，用于确定图形的输出内容。

(10) 设备坐标系：显示器等图形输出设备自身都带有的一个二维直角坐标系。设备

坐标系是整数域二维坐标系，原点位于窗口客户区左上角，x 轴水平向右为正向，y 轴垂直向下为正向，基本单位为像素。

(11) 规格化设备坐标系：规格化到[0,0]~[1,1]范围的设备坐标系。

(12) 右手坐标系：各轴之间的顺序要求符合右手螺旋法则，即右手握住 Z 轴，让右手的四指从 X 轴的正向以 90°的直角转向 Y 轴的正向，这时大拇指指的方向就是 Z 轴的正向。

(13) 左手坐标系：各轴之间的顺序要求符合左手螺旋法则，即左手握住 Z 轴，让左手的四指从 X 轴的正向以 90°的直角转向 Y 轴的正向，这时大拇指指的方向就是 Z 轴的正向。

(14) 窗视变换：图形输出需要完成从窗口到视区的变换，只有位于窗口内的图形，才能在视区中输出，并且输出的形状要根据视区的大小进行适当调整。

(15) 区域码：假设窗口是标准矩形，由上、下、左、右 4 条边组成。延长窗口的 4 条边界形成 9 个区域，为每个区域分配一组 4 位的二进制编码，称为区域编码。

(16) 分治法：字面上的解释是“分而治之”，就是把一个复杂的问题分成两个或更多个相同或相似的子问题，再把子问题分成更小的子问题……直到最后子问题可以直接求解，原问题的解即子问题的解的合并。

5.2 教学时数

本章理论教学时数为 6 学时。详细讲解内容为：三维基本变换、二维复合变换、坐标系的分类、窗视变换、Cohen-Sutherland 裁剪算法、中点分割裁剪算法等。粗略讲解内容为：Liang-Barsky 裁剪算法和 Sutherland-Hodgman 多边形裁剪算法等。

5.3 教学目标

1. 了解齐次坐标

所谓齐次坐标，就是用 $n+1$ 维矢量表示 n 维矢量。例如，在二维平面中，点 $P(x,y)$ 的齐次坐标表示为 (Wx,Wy,W)。因此，(2,3,1)、(4,6,2)、(12,18,6)是用不同的齐次坐标三元组表示的同一个二维点(2,3)。每个二维点都有多种齐次坐标表示形式。如果 W 不等于零，就可以用 W 去除齐次坐标。如果 $W=1$，就是规范化的齐次坐标。二维点 $P(x,y)$ 的规范化齐次坐标为 $(x,y,1)$。图 5-1 中，XYW 构成了三维齐次坐标空间，X 坐标代表 Wx，Y 坐标代表 Wy。(x,y) 点是 (X,Y,W) 点的中心透视投影(投影中心为坐标系原点 O)，即 $x=X/W$，$y=Y/W$。规范化后的三维点 (X,Y,W) 形成一个被等式 $W=1$ 定义的平面，这里 $W\neq0$。如果 $W=0$，三维点 (X,Y,W) 被称为无穷远点。无穷远点不在 $(x,y,1)$ 平面上。定义齐次坐标以后，图形几何变换就可以表示为变换矩阵与图形顶点集合的齐次坐标矩阵相乘的统一形式。

2. 了解二维变换矩阵

引入齐次坐标后，二维几何变换可以表示为：变换后的齐次坐标点矩阵等于变换矩阵与变换前的齐次坐标点矩阵的乘积。这样，通过对 3×3 的变换矩阵的各元素赋值，就可以

定义平移矩阵、比例矩阵、旋转矩阵、反射矩阵和错切矩阵。

3. 实现 Cohen-Sutherland 裁剪算法

Cohen-Sutherland 裁剪算法。延长窗口的 4 条边界形成 9 个区域，为每个区域分配一组 4 位的二进制编码。Cohen 和 Sutherland 创造性地提出了直线段端点的编码规则，但这种裁剪算法需要求解直线段与窗口边界的交点。

4. 实现中点分割裁剪算法

中点分割裁剪算法避免了直线段与窗口边界的求交运算，只递归计算直线段中点坐标就可以完成直线段的裁剪，但递归计算工作量较大。

5. 了解 Liang-Barsky 裁剪算法

Liang-Barsky 裁剪算法是这 3 种算法中效率最高的算法，借助参数方程，把二维裁剪问题转化成一维裁剪问题，把直线段的裁剪问题转化为求解一组不等式问题。

6. 了解 Sutherland-Hodgman 多边形裁剪算法

多边形裁剪算法要求裁剪后的图形边界是自然闭合的，也就是说，裁剪窗口的部分边界做了裁剪后多边形的边界。多边形的裁剪使用了分治法的思想，每次用窗口的一条边裁剪多边形。

5.4 重点难点

教学重点：二维基本变换、二维复合变换、坐标系的分类、窗视变换，Cohen-Sutherland 裁剪算法、中点分割算法。教学难点：Liang-Barsky 裁剪算法、多边形裁剪算法。

5.4.1 教学重点

1. 二维基本变换

二维仿射变换表示为

$$\begin{cases} x' = ax + by + e \\ y' = cx + dy + f \end{cases} \tag{5-1}$$

写成矩阵形式为

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{5-2}$$

式中，(x,y)为变换前的二维点，(x',y')为变换后的二维点。

式(5-2)中，平移变换用加法处理，其余变换用乘法处理。更有效的方法是将二维变换统一表示为一个矩阵，即用一种一致的乘法处理二维变换问题。这需要消除矩阵的加法运算，为此引入点的齐次坐标。$P(x,y)$的齐次坐标可以简单地表示为 $P(x,y,w)$。x、y 作为点的参数用于绘制图形，w 只是参与矩阵运算。通常令 $w=1$，即使用规范化齐次坐标避免除法运算。

将二维变换公式写为 $P'=M \cdot P$

$$\begin{bmatrix} x'_0 & x'_1 & \cdots & x'_{n-1} \\ y'_0 & y'_1 & \cdots & y'_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} a & b & e \\ c & d & f \\ p & q & s \end{bmatrix} \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \\ 1 & 1 & \cdots & 1 \end{bmatrix} \tag{5-3}$$

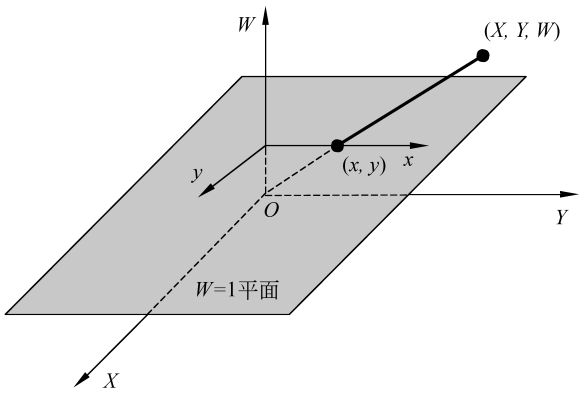

图 5-1　*XYW* 三维齐次坐标空间向 *xy* 二维空间投影

式中，$P(x,y)$为变换前的二维点，$P'(x',y')$为变换后的二维点，$\boldsymbol{M}$ 为变换矩阵。本章中的二维点采用列阵表示，这是国际上通用的做法。

基于齐次坐标表示的二维变换矩阵为

1）平移变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & 0 & T_x\\ 0 & 1 & T_y\\ 0 & 0 & 1\end{bmatrix} \tag{5-4}$$

2）比例变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}S_x & 0 & 0\\ 0 & S_y & 0\\ 0 & 0 & 1\end{bmatrix} \tag{5-5}$$

3）旋转变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}\cos\beta & -\sin\beta & 0\\ \sin\beta & \cos\beta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{5-6}$$

4）反射变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{5-7}$$

5）错切变换矩阵

$$\boldsymbol{M}=\begin{bmatrix}1 & b & 0\\ c & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{5-8}$$

式(5-4)～式(5-8)给出的是二维基本变换矩阵。所谓基本，指的是相对于坐标系原点或坐标轴进行的几何变换。

2. 二维复合变换

二维复合变换主要讲解相对于任意参考点与任意参考方向的几何变换。变换方法是：

首先将任意点平移到坐标原点，或者任意方向旋转为坐标轴方向。然后相对于任一点或任意方向做具体的二维变换，最后做任意点或任意方向的反变换，将任意点或者任意方向恢复到原始状态。在二维复合变换中，要重视相对于任意点的旋转变换与相对于任意点的比例变换。前者容易理解，后者需要作图才能说明白。主教材的例 5-1 与例 5-2 分别进行了讲解。主教材中将这 5 种变换以及复合变换编写为 CTransform2 类。将变换矩阵乘以图形顶点齐次坐标矩阵，就可以得到变换后的图形顶点的齐次坐标矩阵。擦除用变换前顶点坐标绘制的图形，绘制变换后顶点的图形，加上双缓冲技术，就实现了二维图形变换的动画。这 5 个变换矩阵中，旋转变换矩阵用得最多，要重点讲解。假如讲解金刚石图案的旋转，学生最容易想到的是改变转角，正规的做法是调用 CTransform2 的 Rotate()函数进行旋转。

3. 坐标系的分类

世界坐标系是固定不变的坐标系，用于建立三维场景，常分为左手系与右手系。建模坐标系用于建立物体的几何模型。坐标系原点可以建立在物体的任何位置，如圆柱的底面中心、立方体的体心或者立方体的任意一个顶点。世界坐标系相当于三维场景的舞台，定义了出场物体的相互位置，也用于确定视点位置和视向、光源位置等。将物体借助建模坐标系到世界坐标系的变换导入世界坐标系。观察坐标系定义了视点的位置和朝向。屏幕坐标系位于物体与视点之间。在屏幕坐标系内绘制物体的二维投影图。最后由规格化设备坐标系变换到设备坐标系进行图像输出，如图 5-2 所示。

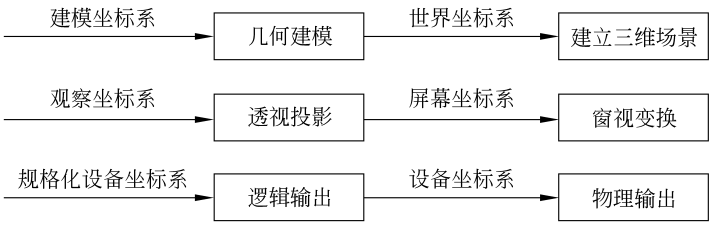

图 5-2　坐标系之间的关系

4. 窗视变换

在观察坐标系中定义的确定图形显示内容的区域称为窗口。在设备坐标系中定义的输出图形的区域称为视区。窗口和视区常为矩形，大小可以不相同。一般情况下，用户把窗口内感兴趣的图形输出到屏幕上相应的视区内。可以在屏幕上定义多个视区，用来同时显示不同窗口内的图形信息。图形输出需要进行从窗口到视区的变换，只有窗口内的图形才能在视区中输出，并且输出的形状要根据视区的大小进行调整，这称为窗视变换。

窗视变换矩阵为

$$\boldsymbol{M}=\begin{bmatrix} s_x & 0 & v_{xl}-w_{xl}s_x \\ 0 & s_y & v_{yb}-w_{yb}s_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5-9}$$

为了减少窗视变换的计算量，常假定窗口与视区的大小一致。

5. Cohen-Sutherland 裁剪算法

Cohen-Sutherland 裁剪算法是一种直线段裁剪算法，其特点是将窗口边界延长得到 9 个区域，为了判断直线段端点位于哪个区域，使用 4 位二进制数编码进行数字化处理。裁剪主要分为以下 3 步：简取、简弃、求交。

(1) 若直线段的两个端点的区域编码都为 0，即 $RC_0 | RC_1 = 0$（二者按位相或的结果为 0，即 $RC_0 = 0$ 且 $RC_1 = 0$），说明直线段的两个端点都在窗口内，应“简取”。

(2) 若直线段的两个端点的区域编码都不为 0，即 $RC_0 \& RC_1 \neq 0$（二者按位相与的结果不为 0，即 $RC_0 \neq 0$ 且 $RC_1 \neq 0$，直线段位于窗外的同一侧），说明直线段的两个端点都在窗口外，应“简弃”。

(3) 若直线段既不满足“简取”的条件，也不满足“简弃”的条件，则需要与窗口进行“求交”判断。

求交这一步需要计算窗口边界与直线段的交点。

6. 中点分割裁剪算法

Cohen-Sutherland 裁剪算法提出对直线段端点进行编码，并把直线段与窗口的位置关系划分为 3 种情况：对前两种情况进行“简取”与“简弃”的简单处理；对于第 3 种情况，需要计算直线段与窗口边界的交点。中点分割裁剪算法对第 3 种情况做了改进，不需要求解直线段与窗口边界的交点，就可以对直线段进行裁剪。

中点分割裁剪算法的原理是：简单地把起点为 P_0，终点为 P_1 的直线段等分为两段直线 PP_0 和 PP_1（P 为直线段中点），对每段直线重复“简取”和“简弃”的处理，对于不能处理的直线段，再继续等分下去，直至每段直线完全能够被“简取”或“简弃”，也就是说，直至每段直线完全位于窗口内或完全位于窗口外，就完成了直线段的裁剪工作，如图 5-3 所示。直线段中点分割裁剪算法是采用二分算法的思想逐次计算直线段的中点 P，以逼近窗口边界，设定控制常数 c 为一个很小的数（如 $c = 10^{-4}$），当 $|PP_0|$ 或 $|PP_1|$ 小于控制常数 c 时，中点收敛于直线段与窗口的交点。中点分割裁剪算法的计算过程只用到加法和移位运算，易于使用硬件实现。用硬件实现中点分割算法既快速，又高效，因为整个过程可以并行处理。硬件实现除 2 不过是将数码右移一位而已。

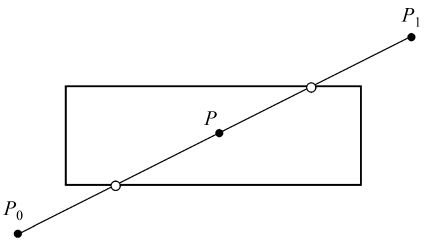

图 5-3　中点分割算法

5.4.2　教学难点

1. Liang-Barsky 裁剪算法

梁友栋和 Barsky 提出了比 Cohen-Sutherland 裁剪算法速度更快的直线段裁剪算法。该算法是以直线的参数方程为基础设计的，把直线与窗口边界求交的二维裁剪问题转化为通过求解一组不等式确定直线段参数的一维裁剪问题。Liang-Barsky 算法将直线段与窗口的相互位置关系划分为两种情况：平行于窗口边界的直线段不平行于窗口边界的直线段。

2. 多边形裁剪算法

多边形裁剪算法 Sutherland-Hodgman 又称为逐边裁剪算法，基本思想是：用裁剪窗口的 4 条边依次对多边形进行裁剪，并将一部分窗口边界加入裁剪后的多边形中，作为裁剪后的多边形边界。窗口边界的裁剪顺序无关紧要，这里采用左、右、下、上的顺序。多边形裁剪算法的输出结果为裁剪后的多边形顶点序列。在算法的每一步中，仅考虑窗口的一条边以及延长线构成的裁剪线，该线把平面分为两部分：一部分包含窗口，称为可见侧；另一部分

落在窗口外，称为不可见侧。

对于裁剪窗口的每条边，多边形的任一顶点只有两种相对位置关系，即位于裁剪窗口的外侧(不可见侧)或内侧(可见侧)，共有 4 种情形。设边的起点为 P_0，终点为 P_1，边与裁剪窗口的交点为 P。图 5-4(a)中，P_0和 P_1都位于裁剪窗口内侧。将 P_1加入输出列表。图 5-4(b)中，P_0位于裁剪窗口内侧，P_1位于裁剪窗口外侧。将交点 P 加入输出列表。图 5-4(c)中，P_0位于裁剪窗口外侧，P_1位于裁剪窗口内侧。将交点 P 和 P_1加入输出列表。图 5-4(d)中，P_0和 P_1都位于裁剪窗口外侧。输出列表中不加入任何顶点。Sutherland-Hodgman 裁剪算法可用于裁剪任意凸多边形。

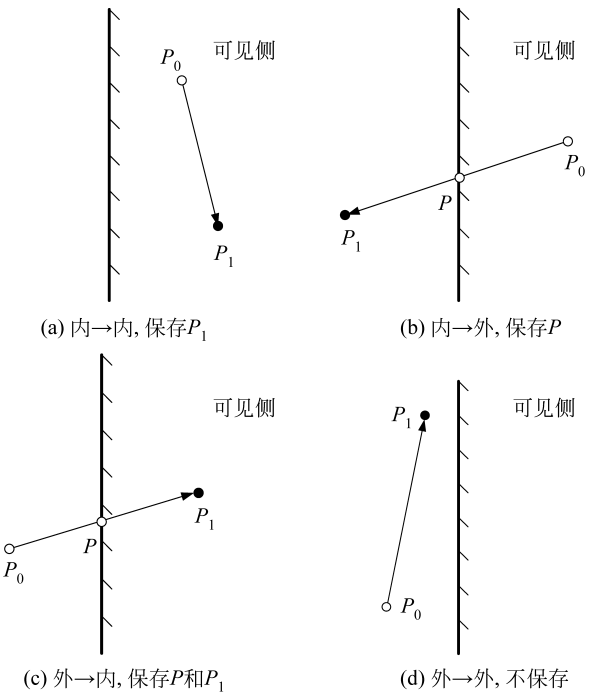

图 5-4　边与裁剪窗口的位置关系

5.5　教学案例建议

重点讲解二维变换的算法。二维变换可以让二维图形动起来。可以讲解正六边形线框与窗口客户区的碰撞动画，也可以讲解金刚石图案与窗口客户区的碰撞动画。这两个图形在与窗口客户区边界的碰撞过程中一边平移、一边绕自身中心旋转。

5.6　教学程序

使用静态切分视图，将窗口分为左、右窗格。左窗格为继承于 CFormView 类的表单视图类 CLeftPortion，右窗格为一般视图类 CTestView。左窗格提供代表“图形顶点数”(5、10、15 和 20)、“平移变换”(x 方向和 y 方向)、“旋转变换”(逆时针和顺时针)和“比例变换”(放大和缩小)的滑动条，用于控制右窗格内的图形变化。右窗格内以窗口客户区中心为图

形的几何中心，绘制不同顶点数的金刚石图案。基于双缓冲技术，金刚石图案在右窗格内无闪烁运动，形成三维动画。设定背景色为黑色，图形用白色线条绘制。使用客户区边界检测技术，图形在右窗格内与客户区边界碰撞后改变运动方向，如图 5-5 所示。

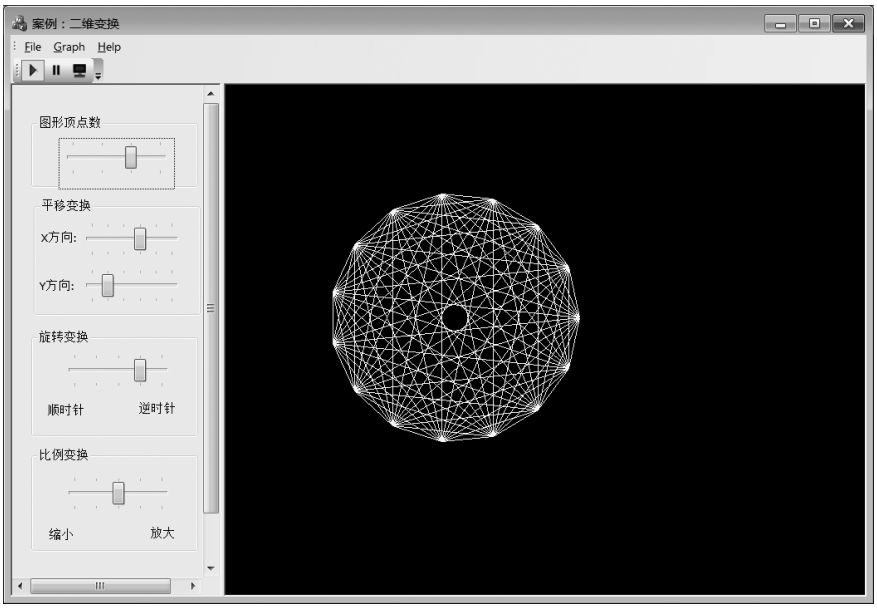

图 5-5　二维几何变换效果图

5.6.1　程序描述

程序由 3 部分构成：第一部分是建立划分左、右窗格的静态切分视图框架，左窗格指定为表单视图，右窗格指定为一般视图；第二部分是为左侧窗格滑动条控件添加消息映射函数；第三部分是通过文档类在右窗格内绘制动态二维图形，使用定时器技术控制图形在右窗格内按照左窗格设置的滑动条控件的位置值进行运动。二维图形与右窗格客户区边界发生碰撞后改变运动方向。

5.6.2　静态切分视图框架

所谓“静态切分”，是指文档窗口在第一次被创建时，窗格的次序和数目就已经被切分好了，不能再被改变，但是可以缩放窗格大小。每个窗格通常代表不同的视图类对象。本案例中，左窗格代表表单视图类 CLeftPortion，用于控制图形，右窗格代表一般视图类 CTestView，用于显示图形。静态切分视图框架的创建分为以下 6 个步骤。

(1) 在 ResourceView 面板中，新建默认标识符为 IDD_DIALOG1 的对话框资源。打开对话框属性，设置 Style 为 Child，Border 为 None，如图 5-6 所示。

(2) 在添加新的静态文本框前，先看 Toolbox 视图是否已显示，如果没有显示，在菜单栏上单击 View→Toolbox 即可，如图 5-7 所示。为对话框添加 4 个 Group Box 控件、7 个 StaticText 控件和 5 个 Slider 控件，如图 5-8 所示。滑动条控件的标识符从上至下依次为 IDC_SLIDER1～IDC_SLIDER5，分别代表图形顶点数、x 方向平移参数、y 方向平移参数、旋转角度和比例系数。为了在每个滑动条上都显示刻度线，可以选中 Tick Marks 和 Auto

Ticks 选项，如图 5-9 所示。将 Caption 为“三角形面片数：8”的静态文本的标识符设置为 IDC_CURFACE，用于响应顶点数变化的通知消息。

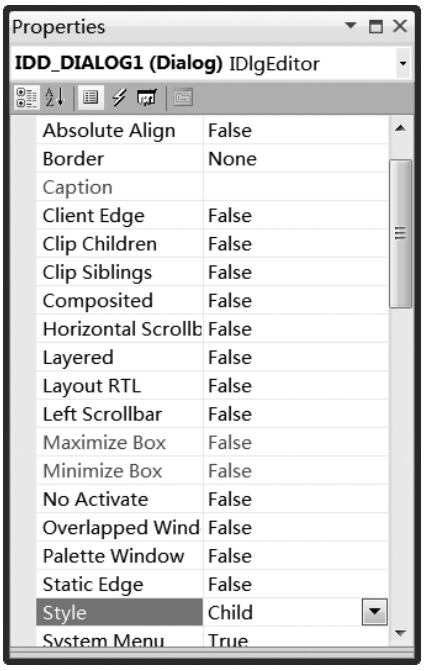

图 5-6　对话框 Style 属性设置

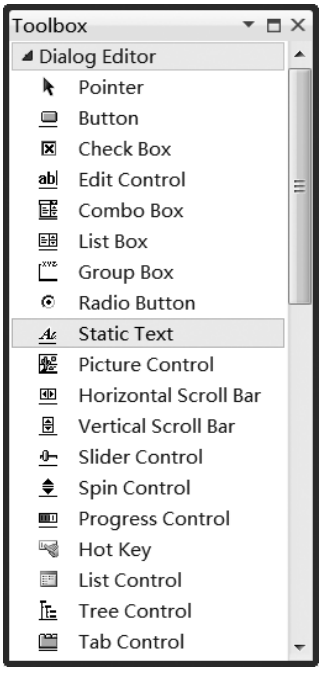

图 5-7　Toolbox

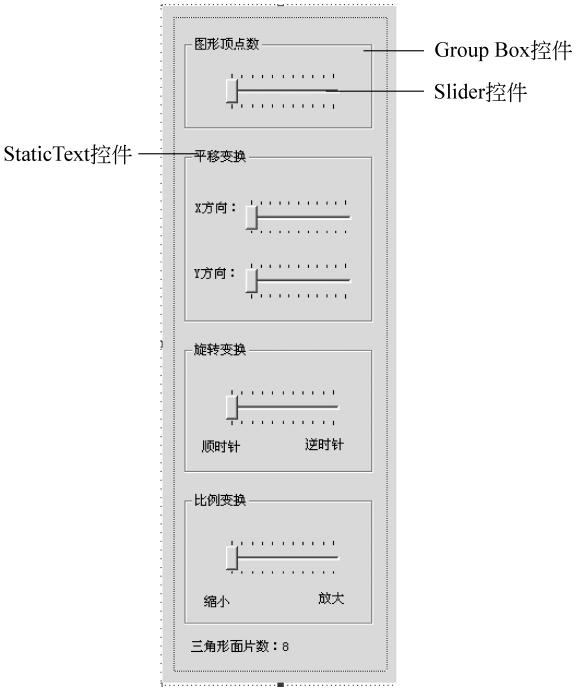

图 5-8　控件的设置

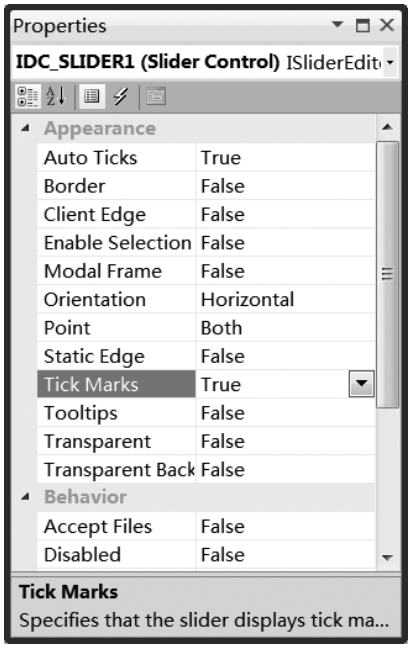

图 5-9　滑动条 Style 属性设置

(3) 双击对话框,创建继承于 CFormView 类的 CLeftPortion 类,如图 5-10 所示。CFormView 类具有许多无模态对话框的特点,并且可以包含控件。

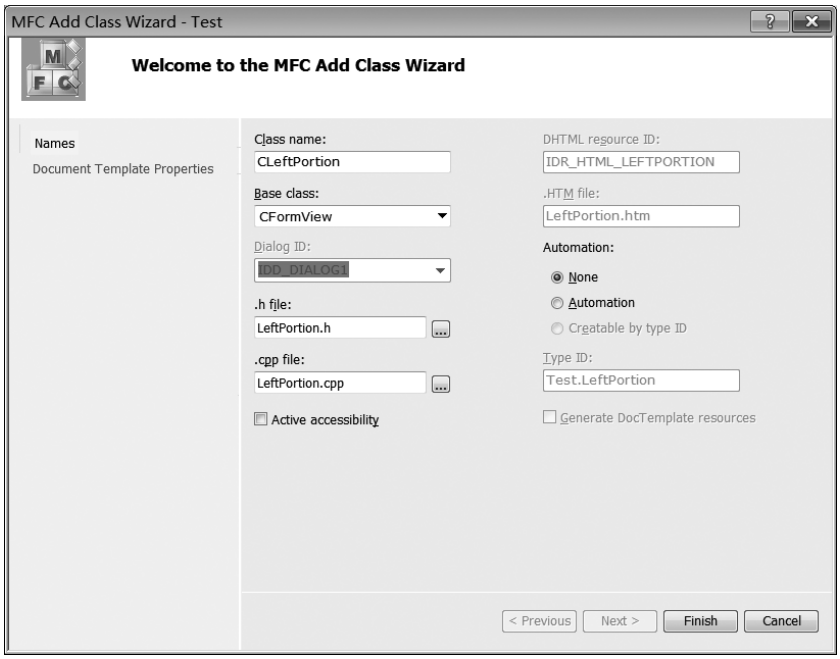

图 5-10 继承于表单类的对话框

(4) 在 CMainFrame 框架窗口类中声明一个 CSplitterWnd 类的成员变量 m_wndSplitter,定义如下。

```
protected:     // control bar embedded members
    CMFCMenuBar            m_wndMenuBar;
    CMFCToolBar            m_wndToolBar;
    CMFCStatusBar          m_wndStatusBar;
    CMFCToolBarImages      m_UserImages;
   CSplitterWnd            m_wndSplitter;      // 分割器
```

(5) 使用 ClassWizard 向导为 CMainFrame 类添加 OnCreateClient()函数。这里是使用 ClassWizard 重写父类的虚函数,而不是添加消息处理,如图 5-11 所示。

(6) 在 OnCreateClient()函数中调用 CSplitterWnd 类的成员函数 CSplitterWnd::CreateStatic()创建静态切分窗格,并调用 CSplitterWnd::CreateView()为每个窗格创建视图窗口。在主框架显示静态切分窗格口前,每个窗格的所有视图都必须已被创建好。

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext * pContext)
{
    // TODO: Add your specialized code here and/or call the base class
    m_wndSplitter.CreateStatic(this,1,2);          //产生 1×2 的静态切分窗格
    m_wndSplitter.CreateView(0,0,RUNTIME_CLASS(CLeftPortion),CSize(220,600),
```