WGIdemo 项目概要设计

[V0.8]

作者：赵平智

2011.08.06

文档修订记录

版本号	变更内容	日期	变更人
V0.6	基本完成概要设计。	2011.07.26	赵平智
V0.8	补充。	2011.08.06	赵平智

目录

1 前言

WGI 是用 OIOIC 技术实现的 GUI（图形用户界面）库，也是 OIOIC 技术的代表作。
OIOIC 是一种专为 C 语言设计的全新的面向对象编程机制。

WGI 下载地址：http://code.google.com/p/oic-wgi/downloads/list
OIOIC 下载地址：http://code.google.com/p/oioic/downloads/list

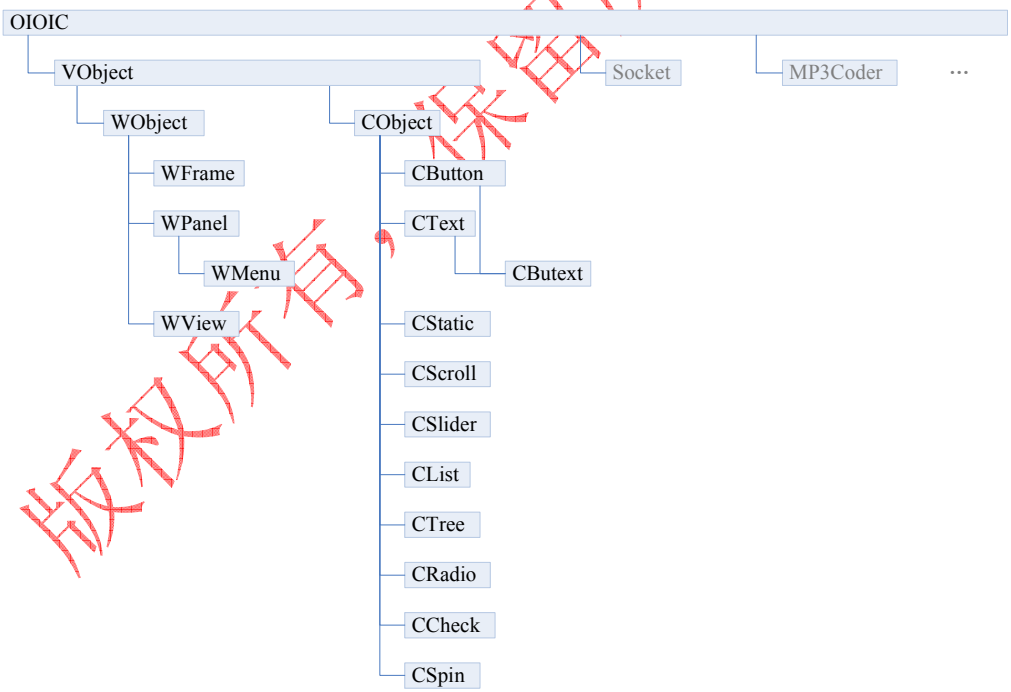
2 概要设计

2.1 哲学

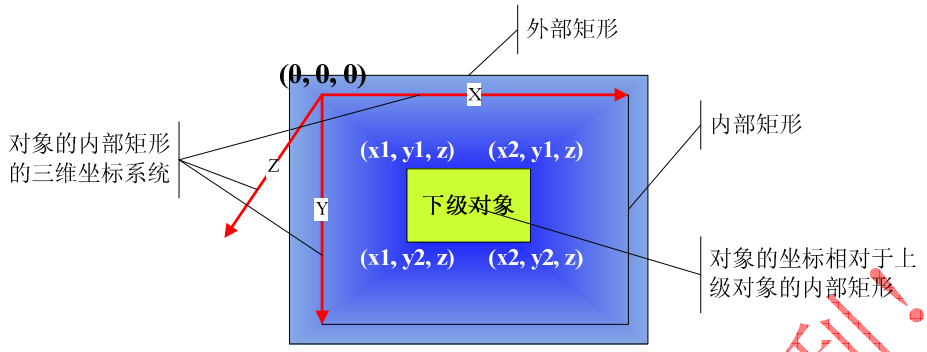
听觉感知一维空间，视觉感知二维空间，触觉感知三维空间，听觉、视觉、触觉一同感知四维空间。

2.2 WGI 层级图

WGI HIERARCHY CHART 1.0.8



2.3 可视对象的坐标系统



坐标结构体：

```
typedef struct TAG_COORDINATES
{
    RECT      rect;     // 矩形空间（rectangle space）
    int       z;        // z coordinate.
}COORDINATES;
```

2.4 父对象和子对象

批注 [微软用户1]：不需要

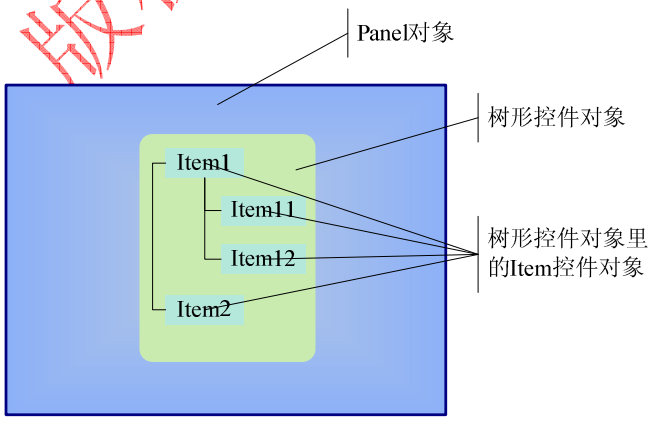
父对象和子对象是可视对象里的概念。

1. CObject 对象是它所在 WPanel 对象的子对象，这个 WPanel 对象是它的父对象。
2. WPanel 对象和 WView 对象都是它们所在的 WFrame 对象的子对象，这个 WFrame 对象是它们的父对象。

2.5 上级对象和下级对象

以下图说明上级对象和下级对象的概念：

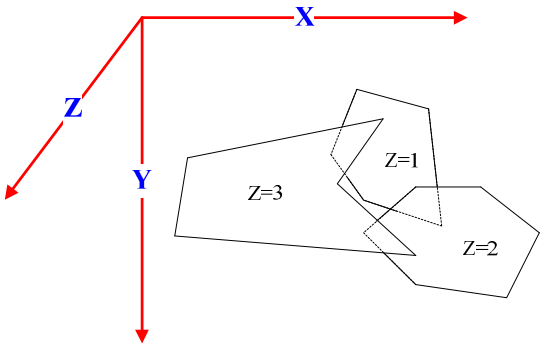
Panel 对象是树形控件对象直接上级对象，树形控件对象是它里面的 Item 控件对象的直接上级对象，Panel 对象是树形控件对象里面的 Item 控件对象的间接上级对象；树形控件对象是 Panel 对象的直接下级对象，树形控件对象里面的 Item 控件对象是这个树形控件对象的直接下级对象，树形控件对象里面的 Item 控件对象是 Panel 对象的间接下级对象。



2.6 多边形和 Z 轴多边形

多边形（polygon）是指多边形的顶点都在同一平面的多边形。Z 轴多边形就是多边形平面与 Z 轴垂直的多边形。

三个Z坐标值不同的Z轴多边形



2.7 移动和滚动

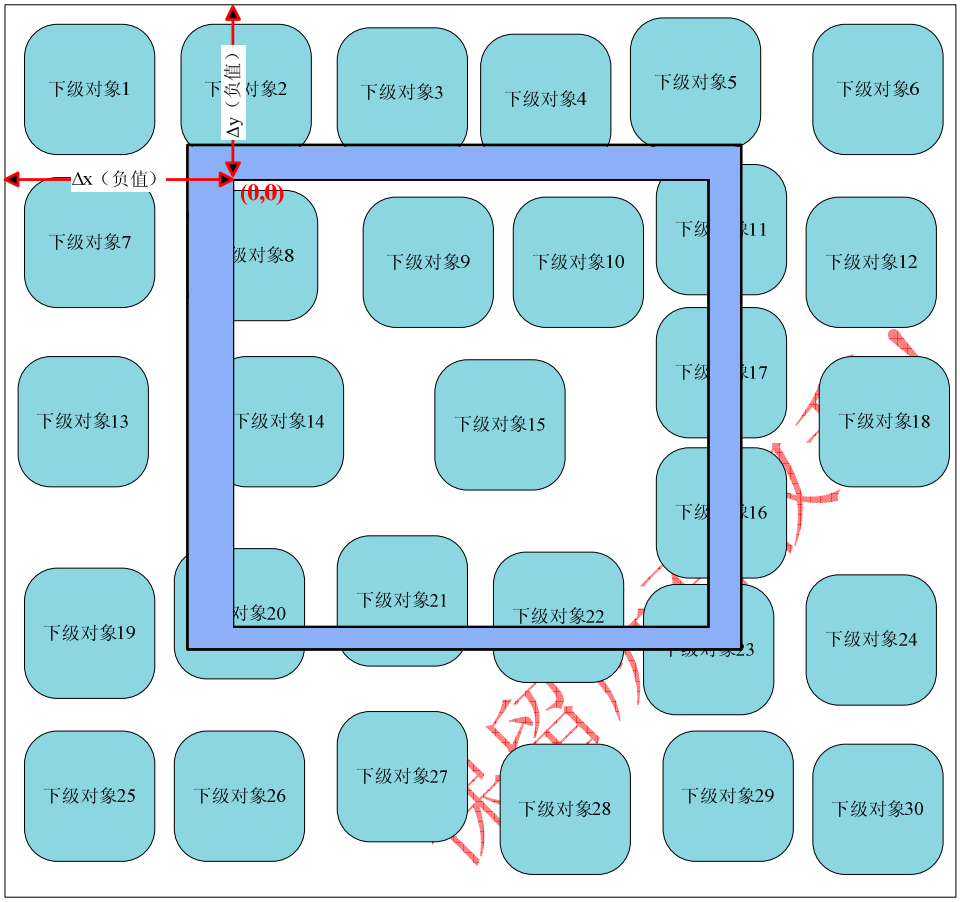
2.7.1 移动（move）

移动是针对一个对象而言，一个对象改变了矩形空间的位置或尺寸，就说这个对象移动了。

2.7.2 滚动（roll）

当内部矩形的内容区域大于显示区域时，就须要滚动内容区域才能浏览所有内容，我们把这个滚动叫做“滚动内部矩形”。内部矩形的内容区域也称为滚动区域，滚动区域是一个矩形，滚动坐标的原点是内部矩形的左上角。

滚动 dx、dy 后，显示如下：



2.8 各级控件对象都由它们的 Panel 对象创建

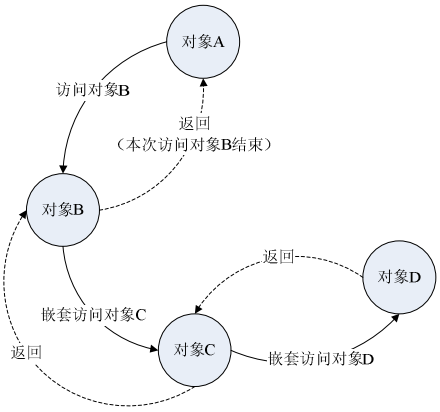
为了方便初始化新创建的控件对象，规定各级控件对象都由它们的 Panel 对象创建，并在 Panel 对象里直接初始化，控件对象里不负责创建它下面的各级控件对象。

2.9 按键激活控件只能在当前级对象群选择

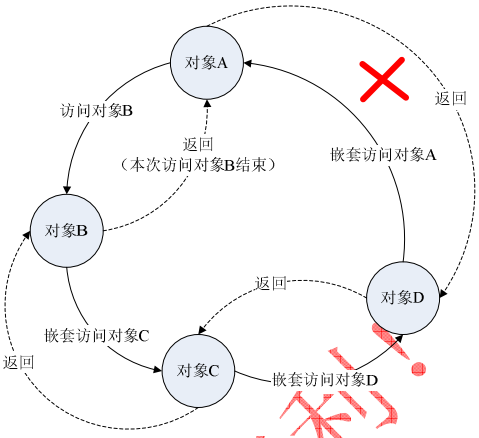
按键激活控件时，比如按 Tab 键激活下一个控件，只能在同级对象中选择控件。除用鼠标外，按键也可以把“激活状态”从一级对象群中转移到另一极对象群。

2.10 对象访问规则

1. 不允许出现“循环访问”



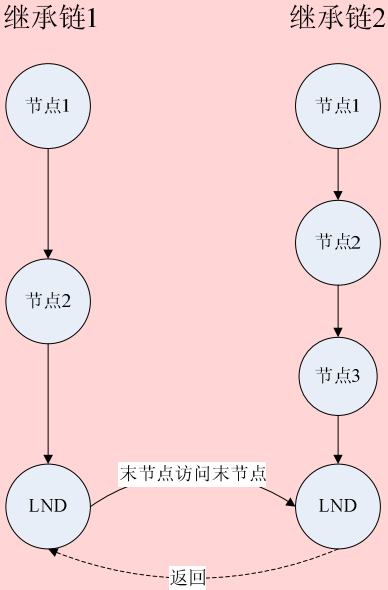
正确



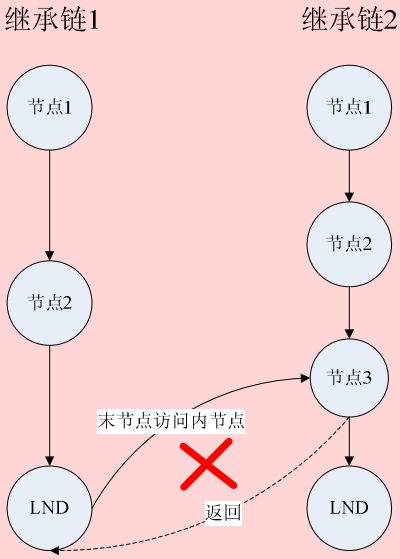
错误，不允许循环访问！

2. 不同继承链的对象间的访问规则

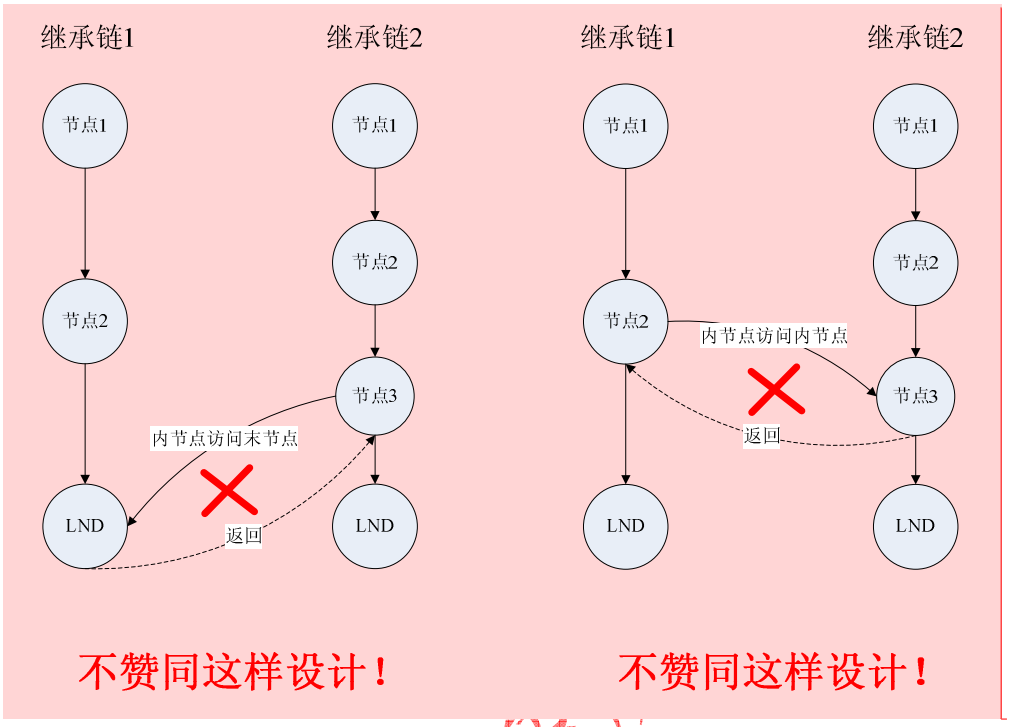
（1）对象都必须是各自继承链的末节点（LND）。



正确



不赞同这样设计！



批注 [微软用户2]：暂不要求，保留。

（2）上级对象不能跨级访问下级对象，只能访问直接下级对象；下级对象不能访问上级对象。

2.11 可视对象的矩形空间和实际空间

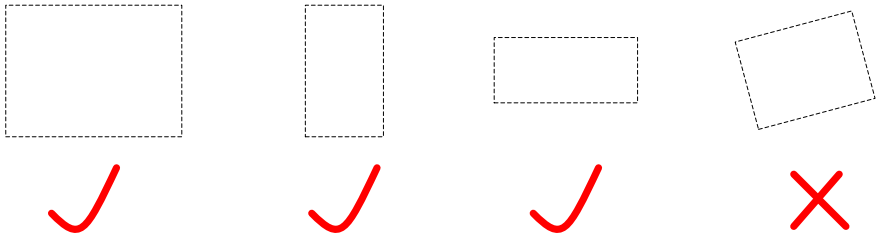
因为视觉只能感知二维空间，第三维空间是靠触觉感知出来的，所以计算机三维空间的视觉效果只能用二维空间虚拟实现，因此用平面上的“实际空间”和“矩形空间”描述视觉部分是最合理的。

1. 实际空间（actual space）

可视对象实际所占的平面空间。所有对象的实际空间内的区域边界都用多边形标识，无论边界是规则的（如圆形、弧形）还是非规则的。对于很大的圆形，如果还由美工确定它的边界多边形，工作量很大，这时可以在程序里在节点初始化时直接计算确定。对于空心的对象，空心部分不属于实际空间。

2. 矩形空间（rectangle space）

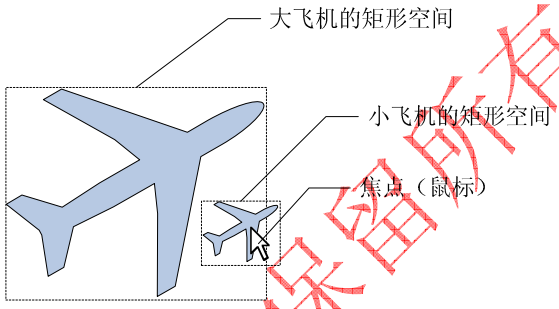
可视对象都有外部矩形和内部矩形（参加后面对窗口的描述），外部矩形所围的空间是外部矩形空间，内部矩形所围的空间是内部矩形空间，内部矩形空间属于外部矩形空间，但是在不引起歧义的情况下，通常所说的矩形空间就是指对象的外部矩形所围的空间。



当硬件焦点（如鼠标）在一个有下级对象的对象的外部矩形空间中移动时，这个对象就检查这个焦点是不是进入了他的某个直接下级对象的外部矩形空间。如果进入某个直接下级对象的外部矩形空间，就通知这个下级对象（调用这个下级对象的接口）：“有焦点进入你的外部矩形空间”，这个下级对象接到这个通知后，就确定焦点是否在自己的正上方。如果焦点在正上方，自己就处理（当然也可以不理会）这个焦点事件；如果焦点在斜上方，就返回“焦点在斜上方”（FOCUSOBLQLY,focus obliquely）的接口结果给它的这个直接上级对象。如果直接上级对象收到某直接下级对象返回“焦点在斜上方”的结果，就继续判断焦点是不是进入其他直接下级对象的外部矩形空间。

批注 [微软用户3]：原先的描述。

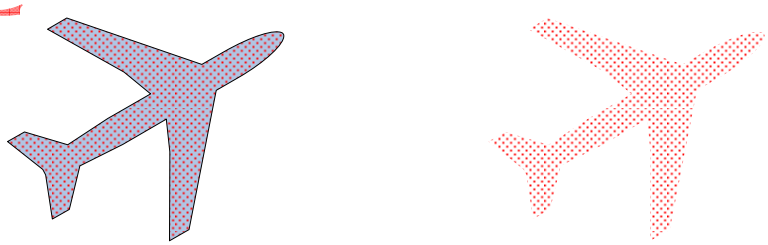
当焦点在一个有下级对象的对象的外部矩形空间中移动时，这个对象就检查这个焦点是不是进入了他的某个直接下级对象的外部矩形空间。如果进入某个直接下级对象的外部矩形空间，就通知这个下级对象（调用这个下级对象的接口）：“焦点 OVERMOVE”，这个下级对象接到这个通知后，就确定焦点是否在自己实际空间上 OVERMOVE，如果是就返回 P 结果，否则返回 N 结果。如果返回 N 结果，上级对象就继续判断焦点是不是进入其他直接下级对象的外部矩形空间。



2.12 可视对象平面空间的两种表达方法

1. 像素坐标法

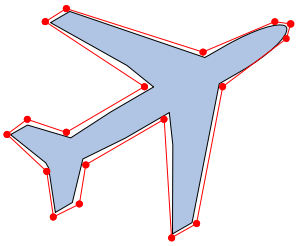
通过记住对象的每个像素的坐标来标识对象的平面空间。

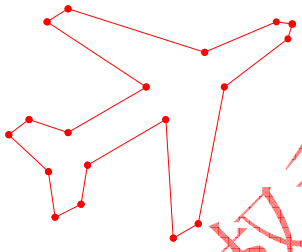


优点：逼真度最高；计算最容易。

缺点：占用空间最大；虽然容易计算，但对于有大量像素的图像，运算量很大。

2. 边界围线法





优点：占用空间最小。

缺点：逼真度差；边界复杂的图形运算量大，不容易计算。

多边形（Polygon）结构体：

```
typedef struct TAG_POLYGON
{
    const POINT*    pAPT;   // pointer to array of points. 顺时针顺序
    int             npt;        // number of points in array.
}POLYGON;
```

实际空间（actual space）多边形单向链表（Singly Linked List）节点结构体：

```
typedef struct TAG_ASPOLYGONSLLNODE
{
    POLYGON         polygon;
    ASPSLLNODE*     next;
}ASPSLLNODE;
```

区域（region）多边形单向链表（Singly Linked List）节点结构体：

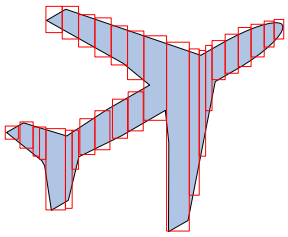
```
typedef struct TAG_RGNPOLYGONSLLNODE
{
    POLYGON         polygon;
    int             z;
    RGNPSLLNODE*    next;
}RGNPSLLNODE;
```

判断两个多边形是否相交：

```
BOOL  IsIntersectant2Polygons(const POLYGON Polygon1, const POLYGON Polygon2);
{
    多次调用 IsIntersectant2Lines
```

一次调用 IsPointInPolygon
}
判断两条线段是否相交（b：begin，e：end）：
BOOL IsIntersectant2Lines(int xb1, int yb1, int xe1, int ye1, int xb2, int yb2, int xe2, int ye2);
判断一点是否在一个多边形区域内：
BOOL IsPointInPolygon(int x, int y, const POLYGON Polygon);

3. 矩形切分法

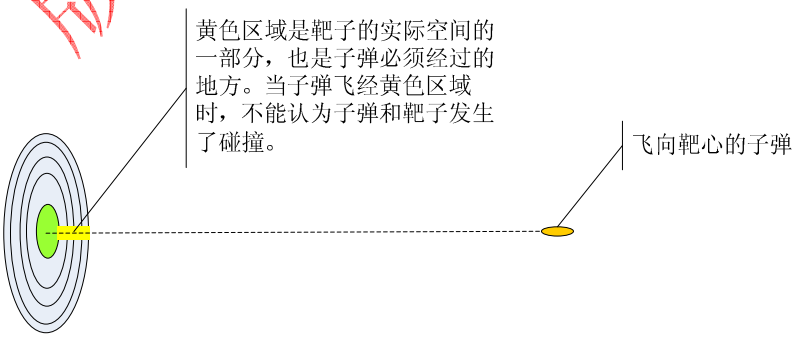


优点：逼真度较好；运算量最小，容易计算。
缺点：占用空间较大。

2.13 侵入和碰撞

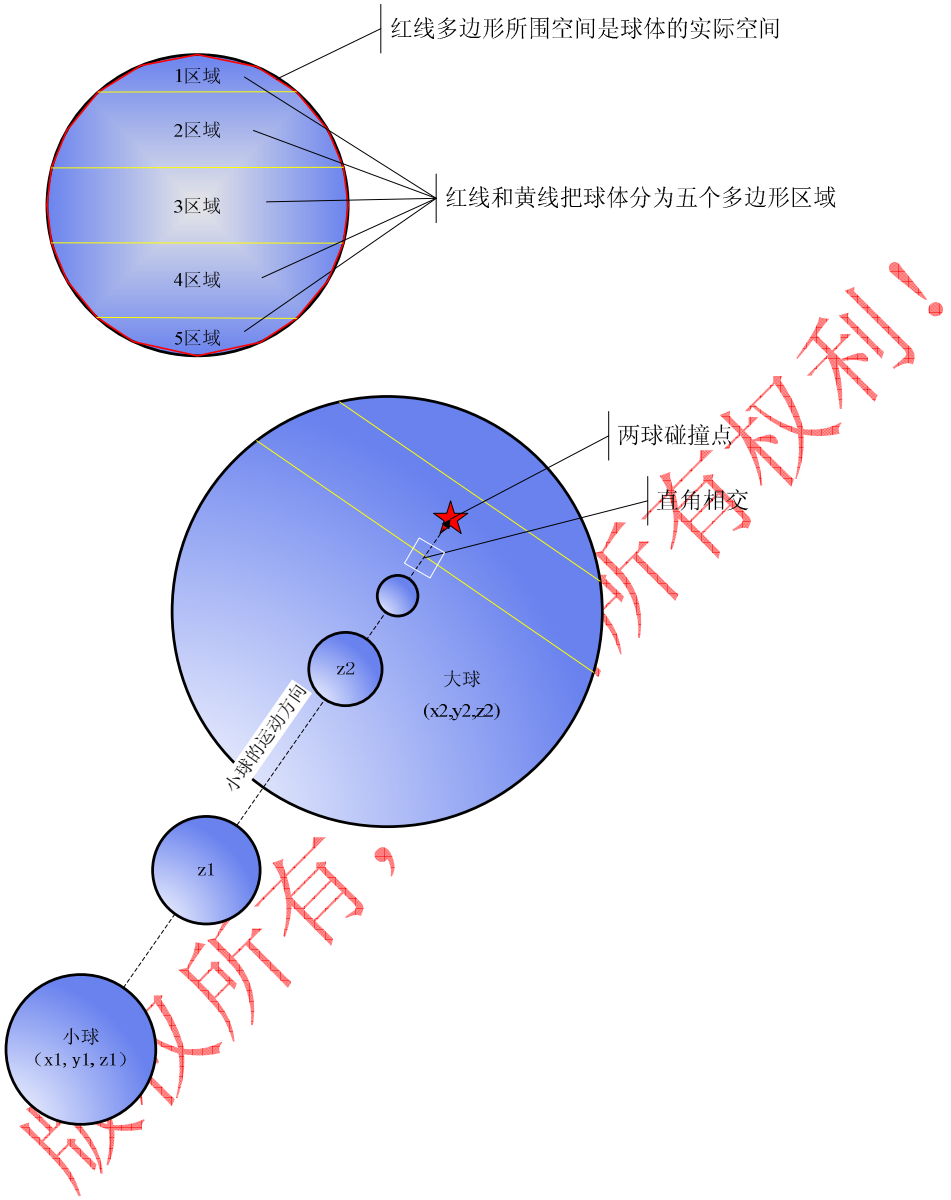
侵入：对象 A 在移动过程中，如果它的矩形空间与另一同级的对象 B 的矩形空间发生重叠，那么就说对象 A 侵入了对象 B。
碰撞：对象 A 在移动过程中，如果它的实际平面空间与另一同级的对象 B 的实际平面空间发生重叠，那么就说两个对象发生碰撞。

2.14 打靶问题



子弹和靶子的 Z 坐标相同时

2.15 两球碰撞问题



如果 A 球和 B 球水平相向运动并碰撞，可以定义五种碰撞结果：

1. A 球 1 区域和 B 球 5 区域碰撞

碰撞后，A 球逆时针快速旋转并缓慢向下偏左的方向移动；B 球逆时针快速旋转并缓慢向上偏右的方向移动。

2. A 球 2 区域和 B 球 4 区域碰撞

碰撞后，A 球逆时针缓慢旋转并快速向左下方移动；B 球逆时针缓慢旋转并快速向右上方移动。

3. A 球 3 区域和 B 球 3 区域碰撞

碰撞后，A 球向左方快速反弹并且不旋转；B 球向右方快速反弹并且不旋转。

4. A 球 4 区域和 B 球 2 区域碰撞

碰撞后，A 球顺时针缓慢旋转并快速向左上方移动；B 球顺时针缓慢旋转并快速向右下方移动。

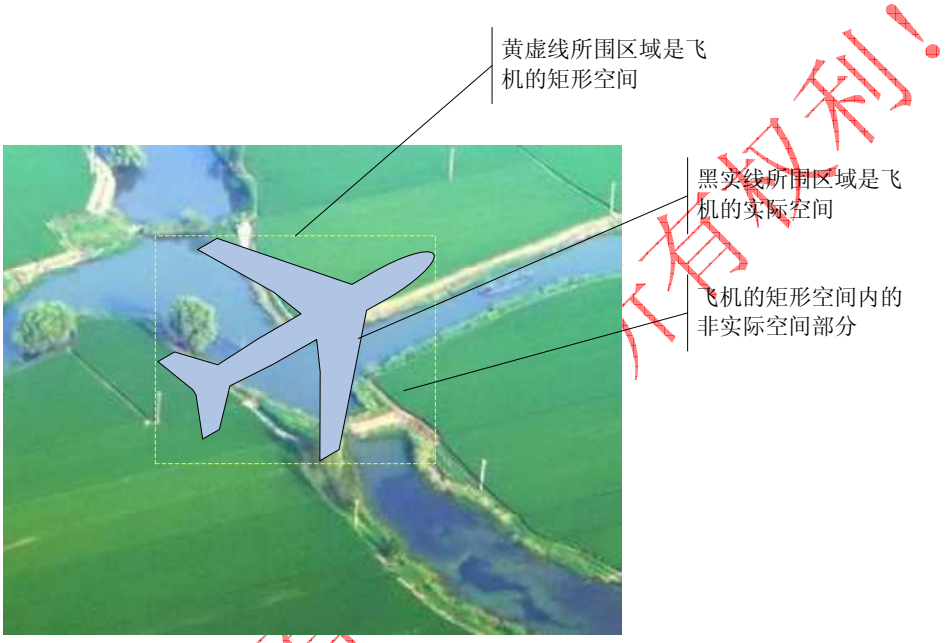
5. A 球 5 区域和 B 球 1 区域碰撞

碰撞后，A 球顺时针快速旋转并缓慢向上偏左的方向移动；B 球顺时针快速旋转并缓慢向下偏右的方向移动。

2.16 导弹击中飞机的不同部位时要有不同的爆炸效果

如何实现？

2.17 如使对象的矩形空间内的非实际空间部分透明？



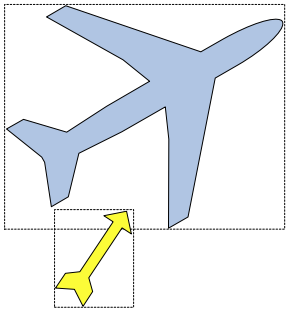
2.18 可视对象的 3D 显示效果

可视对象的 3D 显示效果可以通过改变对象的图片内容和矩形空间的位置及尺寸来实现。

2.19 控件对象的移动

Panel 对象和控件对象决定其直接下级对象是否移动。如果移动，就调用这个下级对象的“移动”行为（AC2，CMD_MOVE，输入流是 MOVE）。如果移动后与其他直接下级对象发生碰撞，还要调用这个被撞的下级对象的“被动碰撞”行为（AC2，EVT_PSVCOLLISION，EVT – event（事件），输入流是 PSVCOLLISION）。

导弹追击飞机



主动碰撞（active collision）单向链表节点结构体：

```
typedef struct TAG_ATVCOLLISIONSLLNODE
{
    const  OBJECT*          pObj;    // 被碰撞的对象
    const  RGNPSLLNODE*     pRgn;     // 碰撞的区域
    ATVCLNSLLNODE*          next;
}ATVCLNSLLNODE;
```

被动碰撞（passive collision）结构体：

```
typedef struct TAG_PSVCOLLISION
{
    const  OBJECT*          pObj;   // 主动碰撞的对象
    const  RGNPSLLNODE*     pRgn; // 被碰撞的区域
}PSVCOLLISION;
```

“移动”行为输入流：

```
typedef struct TAG_MOVE
{
    const COORDINATES       newcs;
    const ATVCLNSLLNODE*    pacsn;
}MOVE;
```


2.20 下级对象面貌更新或者矩形空间发生变化，上级对象重画这个下级对象的处理方案

谁让对象改变面貌或矩形空间？两种情况：

1. 上级对象直接让下级对象更新面貌或改变矩形空间。
2. 对象自己更新面貌或改变矩形空间。

2.20.1 方案 1：一个线程管理一个对象

优点：是一个最理想的方案，如果是上述“情况 2”，下级对象直接通知上级对象重画即可，不会出现循环访问的问题。

缺点：方案最理想，但是因为硬件资源永远是有限的，一个对象对应一个线程不现实。另外，如果是上述的“情况 1”，也会出现循环访问的问题。

2.20.2 方案 2：只有一个线程（传统方法），下级对象直接通知上级对象重画

此方案不可行，容易出现循环访问的问题。

2.20.3 方案 3：只有一个线程（传统方法），下级对象用窗口消息通知面板对象重画

虽然此方案可以避免循环访问的问题，但还没想好解决先让界面更新再处理其他事务的方法。

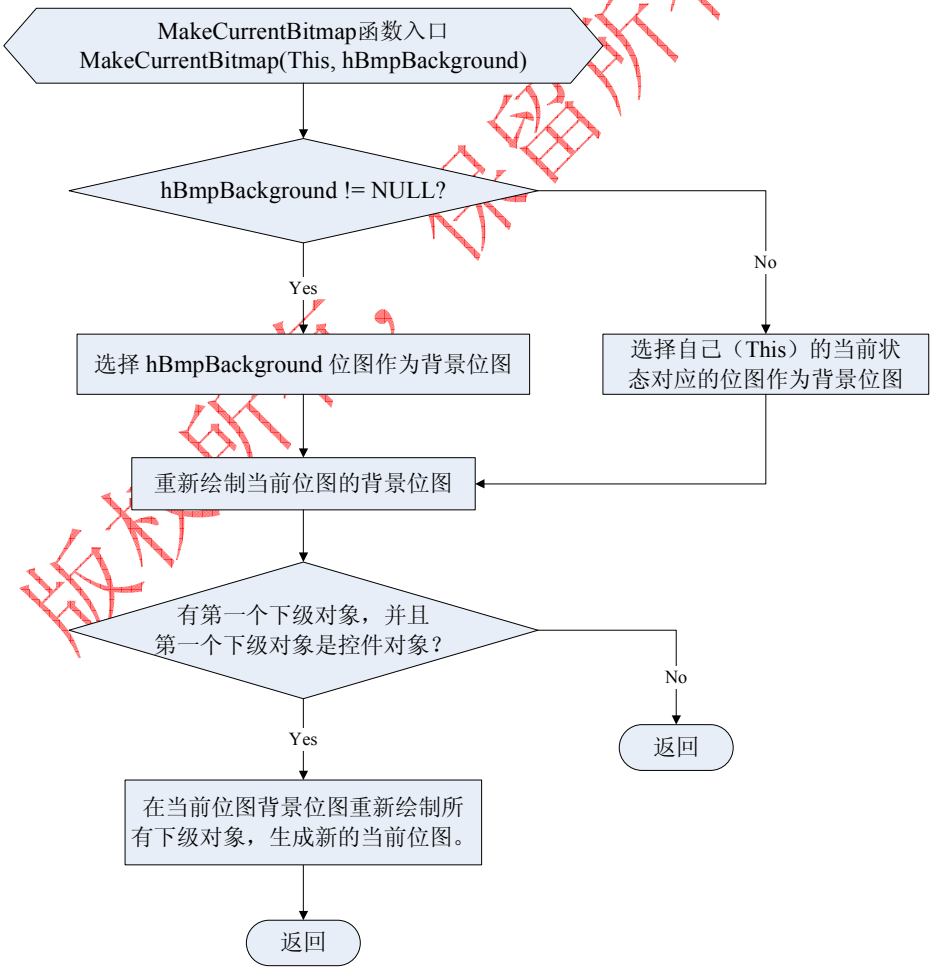
2.20.4 方案 4：Paint 函数递归调用，直到完成在 Panel 对象上的绘制

这个方案可行，采用这个方案。

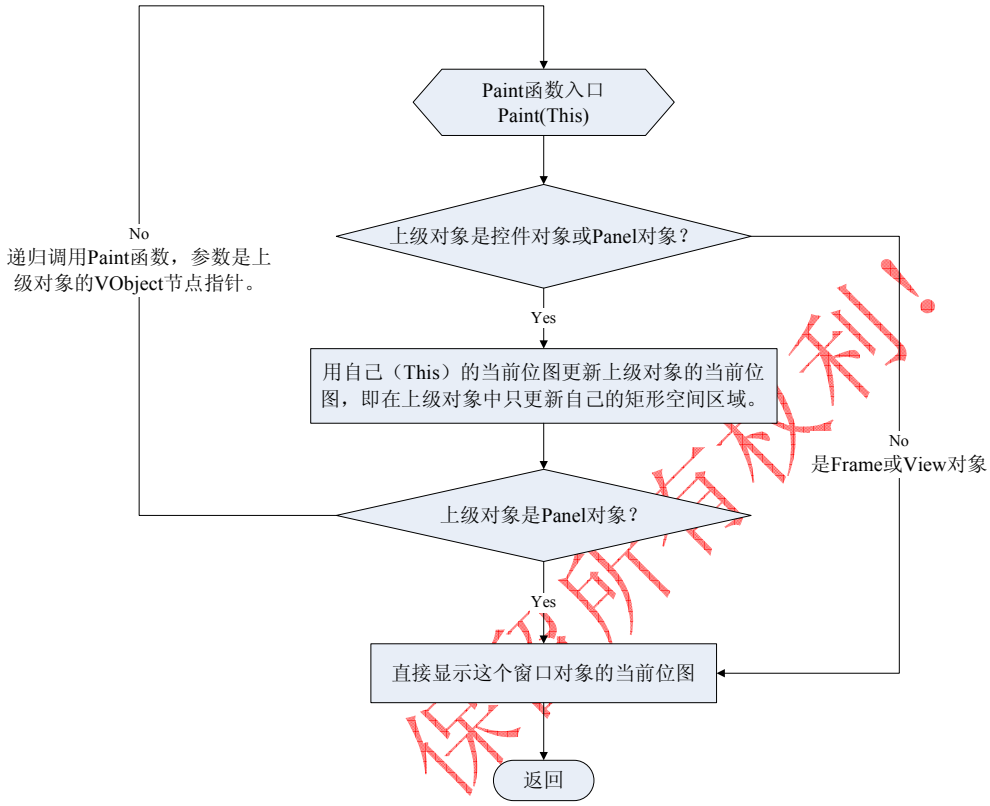
VObject 节点的三个重要的函数：

1. void　MakeCurrentBitmap(OBJECT* This, const HBITMAP hBmpBackground, const BOOL bRoll)
2. void　Paint(OBJECT*　This)
3. void　RepaintSuperior(OBJECT*　This)

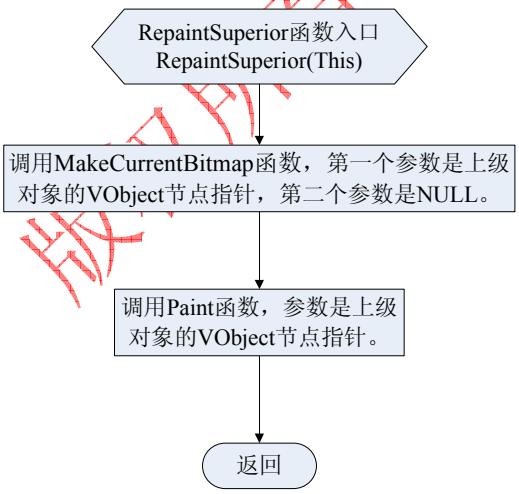
MakeCurrentBitmap 函数流程图



Paint 函数流程图



RepaintSuperior 函数流程图



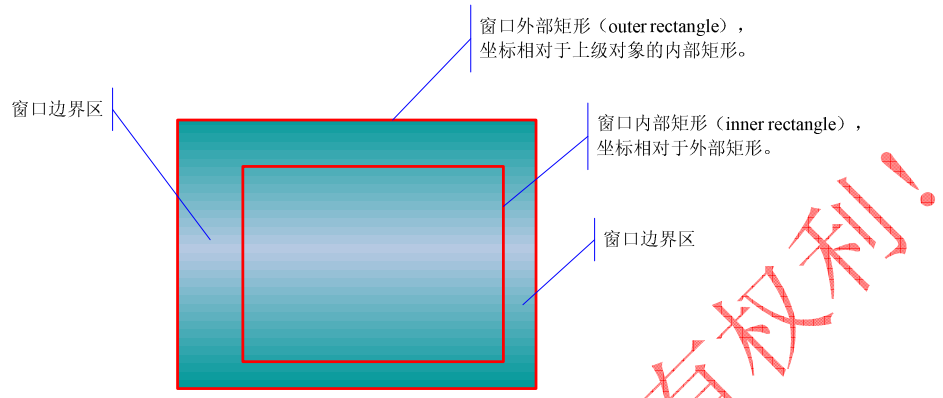
2.21 多个对象结合为一个对象，一个对象分解为多个对象

比如人和车两个对象，当人进入车内时，就把车这个对象销毁，人转换为坐车状态。相反，

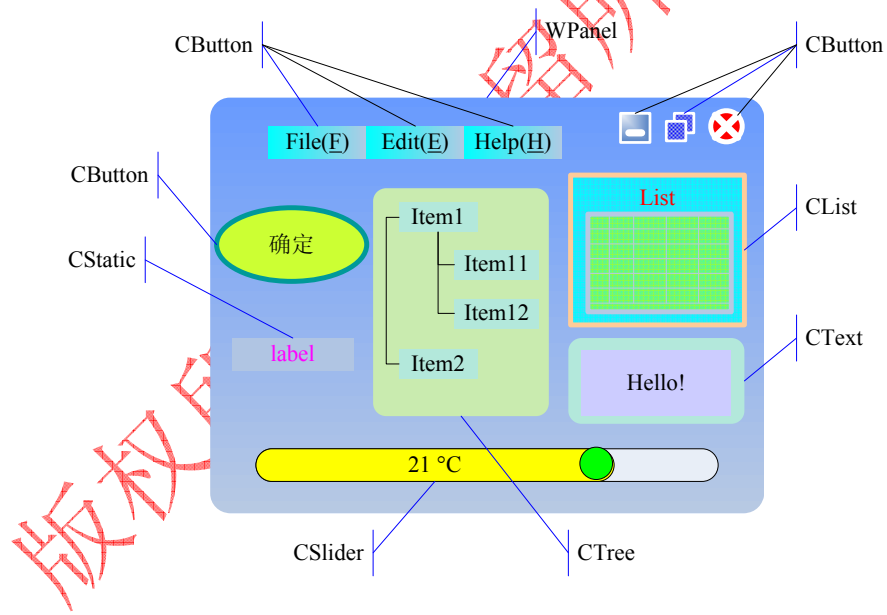
当人走出车外时，就创建一个车的对象，人转换为步行状态。

2.22 窗口

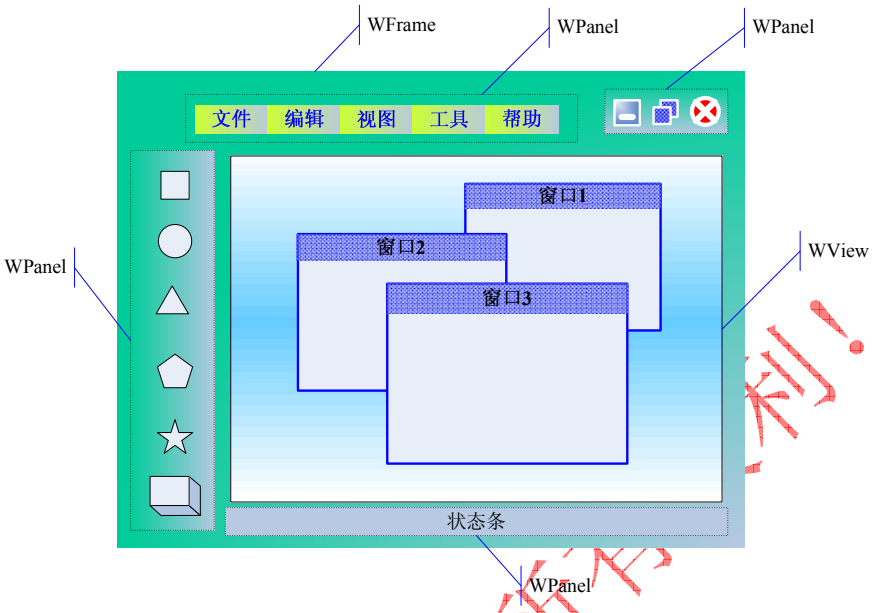
广义来讲，控件也是窗口。窗口对象矩形空间都有外部矩形和内部矩形。窗口内容都放在内部矩形里，窗口内容的坐标都相对于内部矩形。外部矩形和内部矩形之间的区域就是窗口边界区。



2.22.1 对话框窗口



2.22.2 多文档窗口



2.23 与焦点有关的概念

1. 焦点 OVER-MOVE（focus over move）：焦点在空中移动。
2. 焦点 DOWN（focus down）：焦点落地。
3. 焦点 UP（focus up）：焦点升起。
4. 焦点 ON-MOVE（focus on move）：焦点在地面移动。
5. 活动状态（Active）
   对象处于活动状态
6. 选中状态（Selected）
   比如鼠标左键在某控件点击时，这个控件就被选中了。在同级对象中，最多只能有一个对象处于激活状态，一个对象被激活，之前处于激活状态的对象就转为非激活状态。

批注 [微软用户4]：扩展的，保留。

2.24 与焦点有关的消息

```
MSG_WGI_FOCUSONMOVE     // 焦点在上面移动
MSG_WGI_FOCUSUP         // 焦点 UP
MSG_WGI_FOCUSDOWN       // 焦点 DOWN
MSG_WGI_FOCUSDOWNMOVE     // 焦点 DOWN 并且移动
MSG_WGI_FOCUSCLICK        // 焦点 CLICK
MSG_WGI_FOCUSDOUBLECLICK      // 焦点 DOUBLE-CLICK
MSG_WGI_ACTIVATE              // 激活
MSG_WGI_SELECT                // 选中
```

2.25 窗口消息处理流程

窗口消息由用户窗口对象的窗口处理函数接收。

1. 有些窗口消息，用户窗口对象收到后，先不处理而直接通过窗口处理函数转发给基对象处理。
2. 有些窗口消息，用户窗口对象先处理后再转发给基对象处理。
3. 有些窗口消息，用户窗口对象自己处理，不再发给基对象。

2.26 控件对象事件处理流程

控件事件只能通知给控件的父对象。

批注 [微软用户5]：不需要，删除

2.27 VObject

2.27.1 属性

2.27.1.1 父对象

批注 [微软用户6]：不需要

OBJECT*

2.27.1.2 第一个下级对象

OBJECT*

2.27.1.3 上一个同级对象

OBJECT*

2.27.1.4 下一个同级对象

OBJECT*

2.27.1.5 当前焦点在上面的下级对象

OBJECT*

2.27.1.6 当前焦点按住的下级对象

OBJECT*

2.27.1.7 第一个选中的下级对象

OBJECT*

2.27.1.8 下一个选中的同级对象

OBJECT*

2.27.1.9 禁止使用时的图片

HBITMAP

2.27.1.10 正常时的图片

HBITMAP

2.27.1.11 注视焦点时的图片

HBITMAP

2.27.1.12 抓住焦点时的图片

HBITMAP

2.27.1.13 处于激活状态时的图片

HBITMAP

2.27.1.14 处于激活状态但上级对象失去焦点时的图片

HBITMAP

2.27.1.15　实际空间

```
ASPSLLNODE*　　pAS; //actual space
```

第一个 ASPSLLNODE 是对象外边界的多边形，其他 next 指针都是指向对象的空心的多边形。

2.27.1.16　实际空间区域划分

```
RGNPSLLNODE*　　pRGN; // region
```

指向第一个多边形区域，next 指针指向下一个多边形区域。

2.27.1.17　是否显示

比特位，0：显示，1 不显示。不显示时不响应外界事件。

2.27.1.18　是否可以看见焦点

比特位，0：可以看见焦点，1：不可以看见焦点。

2.27.1.19　是否可以抓住焦点

比特位，0：可以抓住焦点，1：不可以抓住焦点。

2.27.1.20　是否可以激活

比特位，0：可以激活，1：不可以激活。

2.27.1.21　是否接受 Tab 键

比特位，0：接受 Tab 键，1：不接受 Tab 键。

2.27.1.22　是否可以使用控件

比特位，0：可以使用（Enabled），1：不可以使用。

2.27.1.23　是否可以移动

比特位，0：可以移动，1：不可以移动。

2.27.2 行为

2.27.2.1 CMD_MOVE（移动）

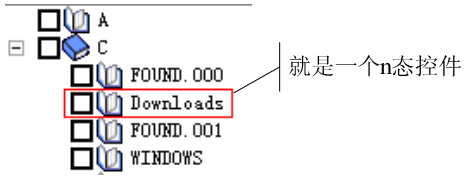
AC2 行为，由直接上级对象调用，输入流是 MOVE。

2.28 WObject

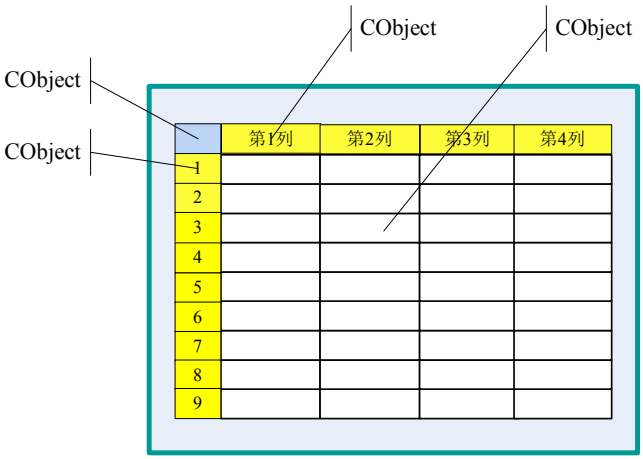
2.29 CObject

2.29.1 属性

2.30 CTree



2.31 CList



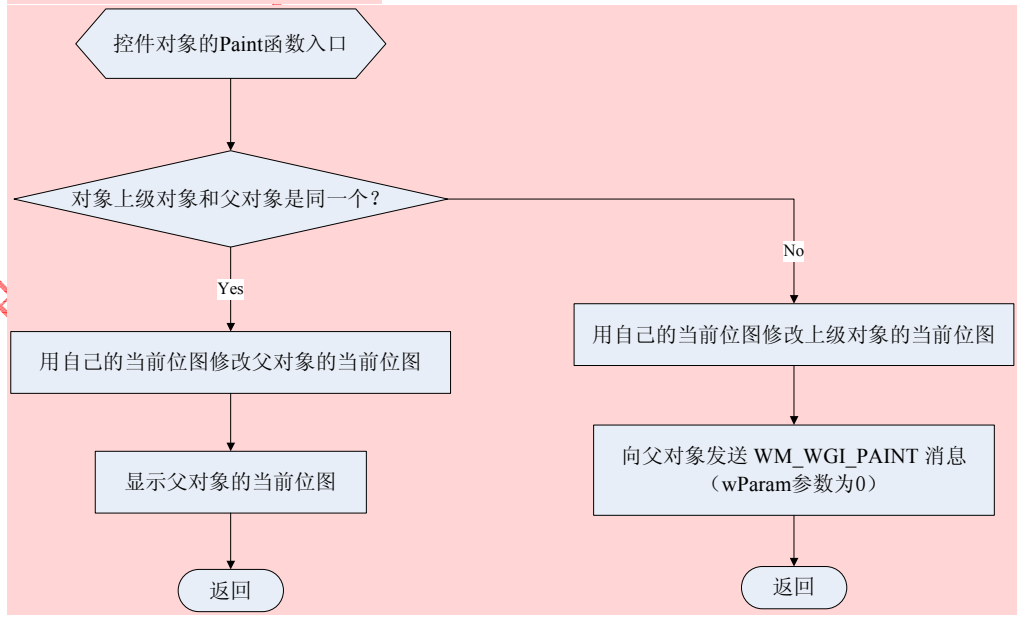
2.32 窗口消息

窗口消息是发送给窗口对象的消息，在 Windows 系统中都宏定义为：WM_USER + N（N 是正整数值）。

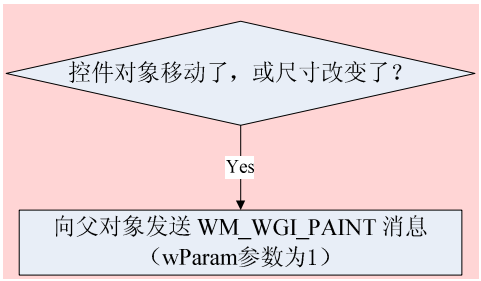
2.33 WM_WGI_PAINT（显示当前位图）

控件对象在需要刷新显示的时候用 PostMessage 函数（wParam 参数为 0 或 1（0 – 控件的矩形空间没变化，1 – 控件的矩形空间已变化），lParam 参数是发送此消息的对象指针）把这个消息发送给其父对象。MakeCurrentBitmap 函数生成当前位图，Paint 函数绘画显示当前位图，两个函数在 WFrame，WPanel，WView，CObject 节点中各自实现。

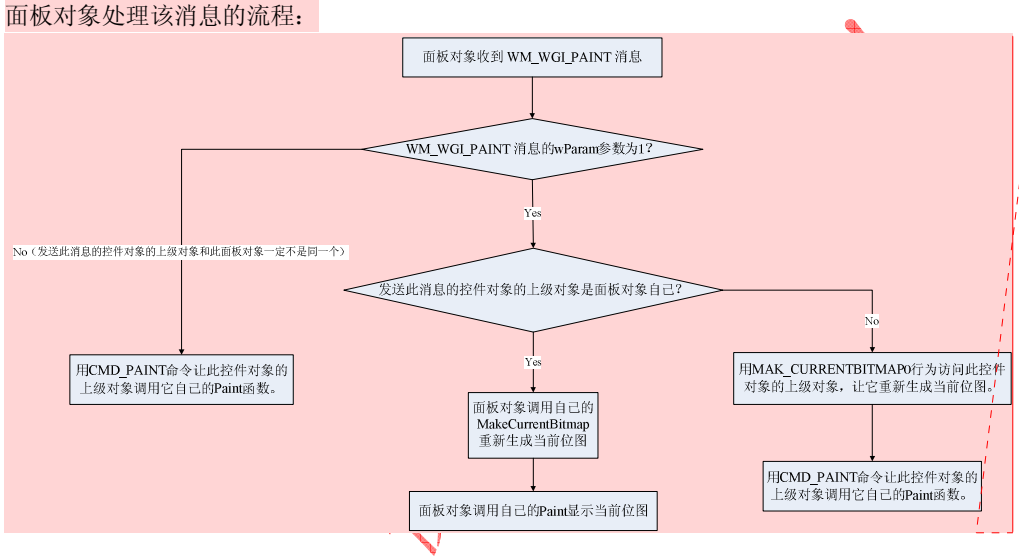
发送 wParam 参数为 0 的消息流程：



发送 wParam 参数为 1 的消息流程：



面板对象处理该消息的流程：



批注 [微软用户7]：不需要

2.34 函数

2.35 Paint（显示当前位图）

此函数在 CObject 节点中的实现流程：

2.36 简化接口行为的分类定义

2.37 在 oioic.h 文件中

2.37.1 原定义

```
/*
*
* 描述：对象交互行为定义。
* 备注：=== 定义模板 ===
*
*       /-** 非一个对象模专用的 ACO 交互行为：**-/
```

```
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_  + 0))
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_  + x - 1))
*       #define     AC1_PVT_            (AC0_PVT_ + x)  /-* 非一个对象模专用的 AC1 交互
行为起点。*-/
*
*       /-** 非一个对象模专用的 AC1 交互行为：**-/
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_  + 0))
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_  + x - 1))
*       #define     AC2_PVT_            (AC1_PVT_ + x)  /-* 非一个对象模专用的 AC2 交互
行为起点。*-/
*
*       /-** 非一个对象模专用的 AC2 交互行为：**-/
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_  + 0))
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_  + x - 1))
*       #define     AC3_PVT_            (AC2_PVT_ + x)  /-* 非一个对象模专用的 AC3 交互
行为起点。*-/
*
*       /-** 非一个对象模专用的 AC3 交互行为：**-/
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_  + 0))
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_  + x - 1))
*       #define     AC0_PVT_XXX_        (AC3_PVT_ + x)  /-* X1 的私有 AC0 交互行为起点。
*-/
*
*
*       /-** X1 对象模的私有 AC0 交互行为：**-/
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_XXX_  + 0))
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_XXX_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC0 | (AC0_PVT_XXX_  + x - 1))
*       #define     AC1_PVT_XXX_        (AC0_PVT_XXX_ + x)  /-* X1 的私有 AC1 交互行为起
点。*-/
*
*       /-** X1 对象模的私有 AC1 交互行为：**-/
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_XXX_  + 0))
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_XXX_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC1 | (AC1_PVT_XXX_  + x - 1))
```

```
*       #define     AC2_PVT_XXX_        (AC1_PVT_XXX_ + x)  /-* X1 的私有 AC2 交互行为起
点。*-/
*
*       /-** X1 对象模的私有 AC2 交互行为：**-/
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_XXX_  + 0))
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_XXX_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC2 | (AC2_PVT_XXX_  + x - 1))
*       #define     AC3_PVT_XXX_        (AC2_PVT_XXX_ + x)  /-* X1 的私有 AC3 交互行为起
点。*-/
*
*       /-** X1 对象模的私有 AC3 交互行为：**-/
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_XXX_  + 0))
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_XXX_  + 1))
*       ...     ...     ...     ...     ...     ...
*       #define     XXX_YYYYY           (AC3 | (AC3_PVT_XXX_  + x - 1))
*       #define     AC0_PVT_YYY_        (AC3_PVT_XXX_ + x)  /-* X2 的私有 AC0 交互行为起
点。*-/
*
*
*       /-** X2 对象模的私有 AC0 交互行为：**-/
*       ...     ...     ...     ...     ...     ...
*
*       ...     ...     ...     ...     ...     ...
*
*/
#define     AC0_                0

/** AC0 交互行为：**/
#define     ASK_SHARED          (AC0 | (AC0_  + 0))   /* 问对象共享么。*/
#define     SET_SHARED          (AC0 | (AC0_  + 1))   /* 设置对象共享。*/
#define     SET_UNSHARED        (AC0 | (AC0_  + 2))   /* 设置对象非共享。*/
#define     SET_BLOCKI          (AC0 | (AC0_  + 3))   /* 设置对象阻断输入。*/
#define     SET_UNBLOCKI        (AC0 | (AC0_  + 4))   /* 设置对象非阻断输入。*/
#define     SET_BLOCKO          (AC0 | (AC0_  + 5))   /* 设置对象阻断输出。*/
#define     SET_UNBLOCKO        (AC0 | (AC0_  + 6))   /* 设置对象非阻断输出。*/
#define     CMD_RESET           (AC0 | (AC0_  + 7))   /* 重置。*/
#define     CMD_PREPARE         (AC0 | (AC0_  + 8))   /* 准备。*/
#define     AC1_                (AC0_ + 9)  /* AC1 交互行为起点。*/

/** AC1 交互行为：**/
#define     GET_IQCTY           (AC1 | (AC1_  + 0))   /* 获取 IQ 容量。*/
#define     GET_OQCTY           (AC1 | (AC1_  + 1))   /* 获取 OQ 容量。*/
#define     AC2_                (AC1_ + 2)  /* AC2 交互行为起点。*/
```

```
/** AC2 交互行为：**/
#define    SET_IQCTY          (AC2 | (AC2_  + 0))   /* 设置 IQ 容量。*/
#define    SET_OQCTY          (AC2 | (AC2_  + 1))   /* 设置 OQ 容量。*/
#define    CMD_INITIALIZE     (AC2 | (AC2_  + 2))   /* 初始化 */
#define    AC3_               (AC2_ + 3)  /* AC3 交互行为起点。*/

/** AC3 交互行为：**/
/* ...     ...     ...     ...     ...     ... */
#define    AC0_PVT_           (AC3_  + 0)    /* <外部私有 AC0 交互行为起点> */
```

2.37.2 新定义

```
/*
*
* 描述：对象交互行为定义。
* 备注：=== 定义模板 ===
*
*       /-** AC0 交互行为：**-/
*       #define     XXX_YYYYY           (AC0_ + 0)
*       #define     XXX_YYYYY           (AC0_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*       /-** AC1 交互行为：**-/
*       #define     XXX_YYYYY           (AC1_ + 0)
*       #define     XXX_YYYYY           (AC1_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*       /-** AC2 交互行为：**-/
*       #define     XXX_YYYYY           (AC2_ + 0)
*       #define     XXX_YYYYY           (AC2_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*       /-** AC3 交互行为：**-/
*       #define     XXX_YYYYY           (AC3_ + 0)
*       #define     XXX_YYYYY           (AC3_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*/

/** AC0 交互行为：**/
#define    ASK_SHARED         (AC0 + 0)    /* 问对象共享么。*/
#define    SET_SHARED         (AC0 + 1)    /* 设置对象共享。*/
#define    SET_UNSHARED       (AC0 + 2)    /* 设置对象非共享。*/
#define    SET_BLOCKI         (AC0 + 3)    /* 设置对象阻断输入。*/
#define    SET_UNBLOCKI       (AC0 + 4)    /* 设置对象非阻断输入。*/
```

```
#define     SET_BLOCKO          (ACO + 5)    /* 设置对象阻断输出。*/
#define     SET_UNBLOCKO        (ACO + 6)    /* 设置对象非阻断输出。*/
#define     CMD_RESET           (ACO + 7)    /* 重置。*/
#define     CMD_PREPARE         (ACO + 8)    /* 准备。*/
#define     ACO_                (ACO + 9)    /* <其它 ACO 交互行为的起点> */

/** AC1 交互行为：**/
#define     GET_IQCTY           (AC1 + 0)    /* 获取 IQ 容量。*/
#define     GET_OQCTY           (AC1 + 1)    /* 获取 OQ 容量。*/
#define     AC1_                (AC1 + 2)    /* <其它 AC1 交互行为的起点> */

/** AC2 交互行为：**/
#define     SET_IQCTY           (AC2 + 0)    /* 设置 IQ 容量。*/
#define     SET_OQCTY           (AC2 + 1)    /* 设置 OQ 容量。*/
#define     CMD_INITIALIZE      (AC2 + 2)    /* 初始化 */
#define     AC2_                (AC2 + 3)    /* <其它 AC2 交互行为的起点> */

/** AC3 交互行为：**/
/* ...     ...     ...     ...     ...     ... */
#define     AC3_                (AC3 + 0)    /* <其它 AC3 交互行为的起点> */
```

2.38 简化接口执行结果的分类定义

2.39 在 oioic.h 文件中

2.39.1 原定义

```
/*
*
* 描述：对象接口执行结果定义。
* 备注：=== 定义模板 ===
*
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*       #define     IR_N_PVT_YYY_           (IR_N_PVT_XXX_ - x)  /-* X2 的私有 N 结果
起点。*-/
*       #define     IR_N_XXX            (IR_N_PVT_XXX_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_N_XXX            (IR_N_PVT_XXX_ -1)
*       #define     IR_N_XXX            (IR_N_PVT_XXX_ -0)
*       /-** X1 对象模的私有 N 结果：^ **-/
*
*       #define     IR_N_PVT_XXX_           (IR_N_PVT_ - x)  /-* X1 的私有 N 结果起点。
*-/
*       #define     IR_N_XXX            (IR_N_PVT_ - x + 1)
*       ...     ...     ...     ...     ...     ...
```

```
*       #define     IR_N_XXX            (IR_N_PVT_ -1)
*       #define     IR_N_XXX            (IR_N_PVT_ -0)
*       /-**
*       ** 非一个对象模专用的 N 结果：^
*       **-/
*
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*       #define     IR_0_PVTN_YYY_          (IR_0_PVTN_XXX_ - x)  /-* X2 的私有负 0
结果起点。*-/
*       #define     IR_0_XXX            (IR_0_PVTN_XXX_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_0_XXX            (IR_0_PVTN_XXX_ -1)
*       #define     IR_0_XXX            (IR_0_PVTN_XXX_ -0)
*       /-** X1 对象模的私有负 0 结果：^ **-/
*
*       #define     IR_0_PVTN_XXX_          (IR_0_PVTN_ - x)  /-* X1 的私有负 0 结果
起点。*-/
*       #define     IR_0_XXX            (IR_0_PVTN_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_0_XXX            (IR_0_PVTN_ -1)
*       #define     IR_0_XXX            (IR_0_PVTN_ -0)
*       /-**
*       ** 非一个对象模专用的负 0 结果：
*       **
*       ** 非一个对象模专用的正 0 结果：
*       **-/
*       #define     IR_0_XXX            (IR_0_PVTP_ + 0)
*       #define     IR_0_XXX            (IR_0_PVTP_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_0_XXX            (IR_0_PVTP_ + x - 1)
*       #define     IR_0_PVTP_XXX_          (IR_0_PVTP_ + x)  /-* X1 的私有正 0 结果
起点。*-/
*
*       /-** X1 对象模的私有正 0 结果：**-/
*       #define     IR_0_XXX            (IR_0_PVTP_XXX_ + 0)
*       #define     IR_0_XXX            (IR_0_PVTP_XXX_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_0_XXX            (IR_0_PVTP_XXX_ + x - 1)
*       #define     IR_0_PVTP_YYY_          (IR_0_PVTP_XXX_ + x)  /-* X2 的私有正 0
结果起点。*-/
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*
*       /-**
```

```
*       ** 非一个对象模专用的 P 结果：
*       **-/
*       #define     IR_P_XXX            (IR_P_PVT_ + 0)
*       #define     IR_P_XXX            (IR_P_PVT_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_P_XXX            (IR_P_PVT_ + x - 1)
*       #define     IR_P_PVT_XXX_           (IR_P_PVT_ + x)  /-* X1 的私有 P 结果起点。
*-/
*
*       /-** X1 对象模的私有 P 结果：**-/
*       #define     IR_P_XXX            (IR_P_PVT_XXX_ + 0)
*       #define     IR_P_XXX            (IR_P_PVT_XXX_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     IR_P_XXX            (IR_P_PVT_XXX_ + x - 1)
*       #define     IR_P_PVT_YYY_           (IR_P_PVT_XXX_ + x)  /-* X2 的私有 P 结果
起点。*-/
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*
*/
#define     IR_N_PVT_           (IR_N_PRIVATE_ - 50)  /* <外部私有 N 结果的起点> */
#define     IR_N_UNSHARED       (FR_N_PRIVATE_ - 1)  /* Open，对象是非共享的。*/
#define     IR_N_FULL           (FR_N_PRIVATE_ - 0)  /* Open，对象引用已满，不能再打
开。*/
/* >#define     IR_N_PRIVATE_    (IR_N - x)  /-* <内部私有 N 结果的起点> *-/ */

#define     IR_N_PRIVATE_    (IR_N - 10)  /* <内部私有 N 结果的起点> */
#define     IR_N_INVALIDARG       (IR_N -1) /* 无效参数 */
#define     IR_N          __FR_ON__       /* N 结果（否定性结果）的起点。*/
#define     __IR_ON__       (IR_O_PVTN_ - 1000)  /* O 结果域的 N 界。*/

#define     IR_O_PVTN_      (IR_O - 10)  /* <外部私有负 O 结果起点> */

#define     IR_O_NOPEN      (IR_O - 2)  /* Close，对象还没有被打开。*/
#define     IR_O_VAINLY     (IR_O - 1)  /* 空操作 */
#define     IR_O            0           /* O 结果（中立性结果）域的中点。*/
#define     IR_O_OBSOIX      (IR_O + 1)  /* OBS 顺序非接力访问模式专用执行结果。*/
#define     IR_O_SBOOIX      (IR_O + 2)  /* SBO 顺序非接力访问模式专用执行结果。*/
#define     IR_O__SBOOIX     (IR_O + 3)  /* SBO 顺序接力访问模式专用执行结果。*/

#define     IR_O_PVTP_      (IR_O + 10)  /* <外部私有正 O 结果起点> */

#define     __IR_OP__       (IR_O_PVTP_ + 1000)  /* O 结果域的 P 界。*/
#define     IR_P          __FR_OP__       /* P 结果（肯定性结果）的起点。*/
#define     IR_P_PRIVATE_     (IR_P + 10) /* <内部私有 P 结果的起点> */
```

```
/* >#define     IR_P_PRIVATE_     (IR_P + x)  /-* <内部私有 P 结果的起点> *-/ */
#define     IR_P_RCZERO       (IR_P_PRIVATE_ + 0) /* Close，对象引用计数变为 0。*/
#define     IR_P_RCONE        (IR_P_PRIVATE_ + 1) /* Open，对象引用计数为 1。*/
#define     IR_P_PVT_         (IR_P_PRIVATE_ + 50) /* <外部私有 P 结果的起点> */
```

2.39.2 新定义

```
/*
*
* 描述：对象接口执行结果定义。
* 备注：=== 定义模板 ===
*
*       ...     ...     ...     ...     ...     ...
*       #define     IR_N_XXX            (IR_N_ - 1)
*       #define     IR_N_XXX            (IR_N_ - 0)
*       /-** 接口的 N 结果（上）：**-/
*       ...     ...     ...     ...     ...     ...
*       #define     IR_O_XXX            (IR_ON_ - 1)
*       #define     IR_O_XXX            (IR_ON_ - 0)
*       /-**
*       ** 接口的负 0 结果（上）：
*       **
*       ** 接口的正 0 结果（下）：
*       **-/
*       #define     IR_O_XXX            (IR_OP_ + 0)
*       #define     IR_O_XXX            (IR_OP_ + 1)
*       ...     ...     ...     ...     ...     ...
*       /-** 接口的 P 结果（下）：**-/
*       #define     IR_P_XXX            (IR_P_ + 0)
*       #define     IR_P_XXX            (IR_P_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*/
#define     IR_N_               (IR_N - 4)    /* <其它 N 结果的起点> */
#define     IR_N_UNSHARED       (FR_N - 3)    /* Open，对象是非共享的。*/
#define     IR_N_FULL           (FR_N - 2)    /* Open，对象引用已满，不能再打开。*/
#define     IR_N_INVALIDARG     (IR_N - 1)    /* 无效参数 */
#define     IR_N        __IR_ON__             /* N 结果（否定性结果）的起点。*/
#define     __IR_ON__       (IR_ON_ - 1000) /* O 结果域的 N 界。*/
#define     IR_ON_          (IR_O - 3)      /* <其它负 0 结果的起点> */
#define     IR_O_NOPEN      (IR_O - 2)      /* Close，对象还没有被打开。*/
#define     IR_O_VAINLY     (IR_O - 1)      /* 空操作 */
#define     IR_O            0               /* O 结果（中立性结果）域的中点。*/
#define     IR_O_OBSOIX     (IR_O + 1)    /* OBS 顺序非接力访问模式专用执行结果。*/
#define     IR_O_SBOOIX     (IR_O + 2)    /* SBO 顺序非接力访问模式专用执行结果。*/
```

```
#define     IR_O__SBOOIX     (IR_O + 3)     /* SBO 顺序接力访问模式专用执行结果。*/

#define     IR_OP_           (IR_O + 4)     /* <其它正 O 结果的起点> */
#define     __IR_OP__        (IR_OP_ + 1000)  /* O 结果域的 P 界。*/
#define     IR_P          __IR_OP__         /* P 结果（肯定性结果）的起点。*/
#define     IR_P_RCZERO      (IR_P + 0)     /* Close，对象引用计数变为 0。*/
#define     IR_P_RCONE       (IR_P + 1)     /* Open，对象引用计数为 1。*/
#define     IR_P_            (IR_P + 2)     /* <其它 P 结果的起点> */
```

2.40 简化函数执行结果的分类定义

2.41 在 oioic.h 文件中

2.41.1 原定义

```
/*
*
* 描述：函数执行结果定义，建议功能复杂或多返回值的函数使用。
* 备注：=== 定义模板 ===
*
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*       #define     FR_N_PVT_YYY_          (FR_N_PVT_XXX_ - x)  /-* X2 的私有 N 结果
起点。*-/
*       #define     FR_N_XXX               (FR_N_PVT_XXX_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_N_XXX               (FR_N_PVT_XXX_ -1)
*       #define     FR_N_XXX               (FR_N_PVT_XXX_ -0)
*       /-** X1 函数的私有 N 结果：^ **-/
*
*       #define     FR_N_PVT_XXX_          (FR_N_PVT_ - x)  /-* X1 的私有 N 结果起点。
*-/
*       #define     FR_N_XXX               (FR_N_PVT_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_N_XXX               (FR_N_PVT_ -1)
*       #define     FR_N_XXX               (FR_N_PVT_ -0)
*       /-**
*       ** 非一个函数专用的 N 结果：^
*       **-/
*
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*       #define     FR_O_PVTN_YYY_         (FR_O_PVTN_XXX_ - x)  /-* X2 的私有负 O
结果起点。*-/
*       #define     FR_O_XXX               (FR_O_PVTN_XXX_ - x + 1)
```

```
*       ...     ...     ...     ...     ...     ...
*       #define     FR_0_XXX            (FR_0_PVTN_XXX_ -1)
*       #define     FR_0_XXX            (FR_0_PVTN_XXX_ -0)
*       /-** X1 函数的私有负 0 结果：^ **-/
*
*       #define     FR_0_PVTN_XXX_          (FR_0_PVTN_ - x)  /-* X1 的私有负 0 结果
起点。*-/
*       #define     FR_0_XXX            (FR_0_PVTN_ - x + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_0_XXX            (FR_0_PVTN_ -1)
*       #define     FR_0_XXX            (FR_0_PVTN_ -0)
*       /-**
*       ** 非一个函数专用的负 0 结果：^
*       **
*       ** 非一个函数专用的正 0 结果：
*       **-/
*       #define     FR_0_XXX            (FR_0_PVTP_ + 0)
*       #define     FR_0_XXX            (FR_0_PVTP_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_0_XXX            (FR_0_PVTP_ + x - 1)
*       #define     FR_0_PVTP_XXX_          (FR_0_PVTP_ + x)  /-* X1 的私有正 0 结果
起点。*-/
*
*       /-** X1 函数的私有正 0 结果：**-/
*       #define     FR_0_XXX            (FR_0_PVTP_XXX_ + 0)
*       #define     FR_0_XXX            (FR_0_PVTP_XXX_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_0_XXX            (FR_0_PVTP_XXX_ + x - 1)
*       #define     FR_0_PVTP_YYY_          (FR_0_PVTP_XXX_ + x)  /-* X2 的私有正 0
结果起点。*-/
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*
*       /-**
*       ** 非一个函数专用的 P 结果：
*       **-/
*       #define     FR_P_XXX            (FR_P_PVT_ + 0)
*       #define     FR_P_XXX            (FR_P_PVT_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_P_XXX            (FR_P_PVT_ + x - 1)
*       #define     FR_P_PVT_XXX_           (FR_P_PVT_ + x)  /-* X1 的私有 P 结果起点。
*-/
*
*       /-** X1 函数的私有 P 结果：**-/
*       #define     FR_P_XXX            (FR_P_PVT_XXX_ + 0)
```

```
*       #define     FR_P_XXX            (FR_P_PVT_XXX_ + 1)
*       ...     ...     ...     ...     ...     ...
*       #define     FR_P_XXX            (FR_P_PVT_XXX_ + x - 1)
*       #define     FR_P_PVT_YYY_           (FR_P_PVT_XXX_ + x)  /-* X2 的私有 P 结果
起点。*-/
*       ...     ...     ...     ...     ...     ...
*       ...     ...     ...     ...     ...     ...
*
*/
#define     FR_N_PVT_           (FR_N_PRIVATE_ - 50)  /* <外部私有 N 结果的起点> */
#define     FR_N_LACKOID        (FR_N_PRIVATE_ - 2)  /* __CreateObject__，没有足够
的 OID 供分配。*/
#define     FR_N_HASVR          (FR_N_PRIVATE_ - 1)  /* CallerCome，已经有这个 VR 了。
*/
#define     FR_N_LOST           (FR_N_PRIVATE_ - 0)  /* Enqueue/Dequeue，已丢失了
Byte。*/
/* >#define     FR_N_PRIVATE_       (FR_N - x)  /-* <内部私有 N 结果的起点> *-/ */

#define     FR_N_PRIVATE_       (FR_N - 10)  /* <内部私有 N 结果的起点> */
#define     FR_N_INVALIDARG     (FR_N -1) /* 无效参数。*/
#define     FR_N        __FR_ON__   /* N 结果（否定性结果）的起点。*/
#define     __FR_ON__       (FR_O_PVTN_ - 1000)  /* O 结果域的 N 界。*/

#define     FR_O_PVTN_      (FR_O - 10)  /* <外部私有负 O 结果的起点> */

#define     FR_O_VAINLY         (FR_O - 1)  /* 空操作 */
#define     FR_O            0           /* O 结果（中立性结果）域的中点。*/

#define     FR_O_PVTP_      (FR_O + 10)  /* <外部私有正 O 结果的起点> */

#define     __FR_OP__       (FR_O_PVTP_ + 1000)  /* O 结果域的 P 界。*/
#define     FR_P        __FR_OP__   /* P 结果（肯定性结果）的起点。*/
#define     FR_P_PRIVATE_       (FR_P + 10)  /* <内部私有 P 结果的起点> */

/* >#define     FR_P_PRIVATE_       (FR_P + x)  /-* <内部私有 P 结果的起点> *-/ */
#define     FR_P_PVT_           (FR_P_PRIVATE_ + 0)  /* <外部私有 P 结果的起点> */
```

2.41.2 新定义

```
/*
*
* 描述：函数执行结果定义，建议功能复杂或多返回值的函数使用。
* 备注：=== 定义模板 ===
*
*       ...     ...     ...     ...     ...     ...
*       #define     FR_N_XXX            (FR_N_ - 1)
```

```
*       #define     FR_N_XXX            (FR_N_ - 0)
*       /-** 函数的 N 结果（上）：**-/
*       ...     ...     ...     ...     ...     ...
*       #define     FR_0_XXX            (FR_ON_ - 1)
*       #define     FR_0_XXX            (FR_ON_ - 0)
*       /-**
*       ** 函数的负 0 结果（上）：
*       **
*       ** 函数的正 0 结果（下）：
*       **-/
*       #define     FR_0_XXX            (FR_OP_ + 0)
*       #define     FR_0_XXX            (FR_OP_ + 1)
*       ...     ...     ...     ...     ...     ...
*       /-** 函数的 P 结果（下）：**-/
*       #define     FR_P_XXX            (FR_P_ + 0)
*       #define     FR_P_XXX            (FR_P_ + 1)
*       ...     ...     ...     ...     ...     ...
*
*/
#define     FR_N_           (FR_N - 5)  /* <其它 N 结果的起点> */
#define     FR_N_LACKOID    (FR_N - 4)  /* __CreateObject__，没有足够的 OID 供分配。*/
#define     FR_N_HASVR          (FR_N - 3)  /* CallerCome，已经有这个 VR 了。*/
#define     FR_N_LOST           (FR_N - 2)  /* Enqueue/Dequeue，已丢失了 Byte。*/
#define     FR_N_INVALIDARG     (FR_N -1) /* 无效参数。*/
#define     FR_N        __FR_ON__   /* N 结果（否定性结果）的起点。*/
#define     __FR_ON__       (FR_ON_ - 1000)  /* 0 结果域的 N 界。*/
#define     FR_ON_          (FR_0 - 2)  /* <其它负 0 结果的起点> */
#define     FR_0_VAINLY     (FR_0 - 1)  /* 空操作 */
#define     FR_0            0           /* 0 结果（中立性结果）域的中点。*/
#define     FR_OP_          (FR_0 + 1)  /* <其它正 0 结果的起点> */
#define     __FR_OP__       (FR_OP_ + 1000)  /* 0 结果域的 P 界。*/
#define     FR_P        __FR_OP__   /* P 结果（肯定性结果）的起点。*/
#define     FR_P_           (FR_P + 0)  /* <其它 P 结果的起点> */
```

3 xxx