程序设计专题——第三方图形库 Debug 报告

本学期程序设计专题课程的第三专题要求使用给定的 Graphics 库进行图形化编程，但在 Graphics 库的使用过程中发现了很多 bugs，以下将对于本组组员已发现的 bug 及其可行的解决方案进行报告。

BUG1：反复定义相同的颜色会导致内存泄漏。

触发方法：

```
for(i=1;i<=100;++i)
    DefineColor(“tempcolor”,0,0,0);
```

```
577 void DefineColor(string name,
578                  double red, double green, double blue)
579 {
580     int cindex;
581
582     InitCheck();
583     if (red < 0 || red > 1 || green < 0 || green > 1 || blue < 0 || blue > 1) {
584         Error("DefineColor: All color intensities must be between 0 and 1");
585     }
586     cindex = FindColorName(name);
587     if (cindex == -1) {
588         if (nColors == MaxColors) Error("DefineColor: Too many colors");
589         cindex = nColors++;
590     }
591     colorTable[cindex].name = CopyString(name);
592     colorTable[cindex].red = red;
593     colorTable[cindex].green = green;
594     colorTable[cindex].blue = blue;
595 }
```

造成原因：如图 591 行每次定义相同颜色代码会重新拷贝一个字符串，而没有释放原本的空间。

```
577 void DefineColor(string name,
578                  double red, double green, double blue)
579 {
580     int cindex;
581
582     InitCheck();
583     if (red < 0 || red > 1 || green < 0 || green > 1 || blue < 0 || blue > 1) {
584         Error("DefineColor: All color intensities must be between 0 and 1");
585     }
586     cindex = FindColorName(name);
587     if (cindex == -1) {
588         if (nColors == MaxColors) Error("DefineColor: Too many colors");
589         cindex = nColors++;
590     }
591     if(colorTable[cindex].name)free(colorTable[cindex].name);
592     colorTable[cindex].name = CopyString(name);
593     colorTable[cindex].red = red;
594     colorTable[cindex].green = green;
595     colorTable[cindex].blue = blue;
596 }
```

解决方案：在拷贝前释放空间,如图 591 行。

BUG2：在设置颜色 A 后重新定义颜色 A，画笔颜色不会改变。

触发方法：

```
DefineColor(“temp”,1,0,0);
SetPenColor(“temp”);
.../*draw something*/
DefineColor(“temp”,0,1,0);
```

```
SetPenColor("temp");
.../*draw something*/
```

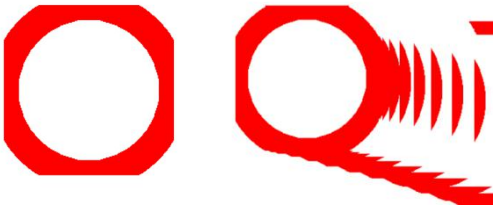
造成原因：如图 1259 行，程序在更新画笔颜色时，只判断颜色名称，不比较颜色值，如果颜色名称相同，不重复更新。

```
1246 static void PrepareToDraw(void)
1247 {
1248     int red, green, blue;
1249 /*
1250     HPEN oldPen;
1251 */
1252
1253     if (eraseMode) {
1254         DeleteObject(erasePen);
1255         erasePen = (HPEN) CreatePen(PS_SOLID, penSize, eraseColor);
1256         (void) SelectObject(osdc, erasePen);
1257         SetTextColor(osdc, eraseColor);
1258     } else {
1259         if (penColor != previousColor) {
1260             red = colorTable[penColor].red * 256 - Epsilon;
1261             green = colorTable[penColor].green * 256 - Epsilon;
1262             blue = colorTable[penColor].blue * 256 - Epsilon;
1263             drawColor = RGB(red, green, blue);
1264             previousColor = penColor;
1265         }
1266         DeleteObject(drawPen);
1267         drawPen = (HPEN) CreatePen(PS_SOLID, penSize, drawColor);
1268         (void) SelectObject(osdc, drawPen);
1269         (void) SetTextColor(osdc, drawColor);
1270     }
1271 }
```

解决方案：注释该行，取消判断。

```
static void PrepareToDraw(void)
{
    int red, green, blue;
/*
    HPEN oldPen;
*/
    if (eraseMode) {
        DeleteObject(erasePen);
        erasePen = (HPEN) CreatePen(PS_SOLID, penSize, eraseColor);
        (void) SelectObject(osdc, erasePen);
        SetTextColor(osdc, eraseColor);
    } else {
        /*if (penColor != previousColor) {*/
            red = colorTable[penColor].red * 256 - Epsilon;
            green = colorTable[penColor].green * 256 - Epsilon;
            blue = colorTable[penColor].blue * 256 - Epsilon;
            drawColor = RGB(red, green, blue);
            previousColor = penColor;
        /*}*/
        DeleteObject(drawPen);
        drawPen = (HPEN) CreatePen(PS_SOLID, penSize, drawColor);
        (void) SelectObject(osdc, drawPen);
        (void) SetTextColor(osdc, drawColor);
    }
}
```

BUG3：图形擦不干净、粗笔画图形显示不全（被矩形区域截断）



触发方法：

使用尺寸大于等于 2 的画笔，在各类回调函数进行擦除（主函数中此 BUG 不会暴露）。

```
Void Timer_recall(int id)
{
    SetPenSize(3);
    ...
    SetEraseMode(TRUE);
    ...
    SetEraseMode(FALSE);
    ...
}
```

造成原因：

程序擦除原理是用白色画笔在缓冲里覆盖，然后刷新到屏幕，但是设置刷新区域时忘记考虑画笔粗细。因为通常 void Main()中都有 InitGraphics()函数，该函数会将整个窗口设置为更新区域，所以该 bug 不会再主程序中暴露，需要等到 WM_PAINT 消息后，无效矩形被清空，此 BUG 才会出现。

```
static void SetLineBB(RECT *rp, double x, double y, double dx, double dy)
{
    int x0, y0, x1, y1;

    x0 = ScaleX(x);
    y0 = ScaleY(y);
    x1 = ScaleX(x + dx);
    y1 = ScaleY(y + dy);
    rp->top = Min(y0, y1);
    rp->bottom = Max(y0, y1) + 1;
    rp->left = Min(x0, x1);
    rp->right = Max(x0, x1) + 1;
}
```

```
    xmax = xmin + PixelsX(2 * rx);
    ymax = ymin + PixelsX(2 * ry);
    if (sweep < 0) {
        start += sweep;
        sweep = -sweep;
    }
    if (sweep >= 360) {
        SetRect(rp, xmin, ymin, xmax, ymax);
        return;
    }
    if (start < 0) {
        start = 360 - fmod(-start, 360);
    } else {
        start = fmod(start, 360);
```

```
    if (start + sweep > 360) {
        xl = xmin;
    } else {
        xl = Min(ix0, ix1);
    }
    start = fmod(start + 270, 360);
    if (start + sweep > 360) {
        yb = ymax;
    } else {
        yb = Max(iy0, iy1);
    }
    SetRect(rp, xl, yt, xr, yb);
}
```

```
    polygonBounds.left = Min(polygonBounds.left, x);
    polygonBounds.right = Max(polygonBounds.right, x);
    polygonBounds.top = Min(polygonBounds.top, y);
    polygonBounds.bottom = Max(polygonBounds.bottom, y);
```

Remarks

The invalidated areas accumulate in the update region until the region is processed when the next **WM_PAINT** message occurs or until the region is validated by using the **ValidateRect** or **ValidateRgn** function.

The system sends a **WM_PAINT** message to a window whenever its update region is not empty and there are no other messages in the application queue for that window.

If the *bErase* parameter is **TRUE** for any part of the update region, the background is erased in the entire region, not just in the specified part.

解决方法：在计算更新区域时计算画笔大小，如下图。

```
static void SetLineBB(RECT *rp, double x, double y, double dx, double dy)
{
    int x0, y0, x1, y1;

    x0 = ScaleX(x);
    y0 = ScaleY(y);
    x1 = ScaleX(x + dx);
    y1 = ScaleY(y + dy);
    rp->top = Max(0,Min(y0, y1)-penSize-penSize);
    rp->bottom = Max(y0, y1) + penSize+penSize;
    rp->left = Max(0,Min(x0, x1)-penSize-penSize);
    rp->right = Max(x0, x1) + penSize+penSize;
}
```

```
        start += sweep;
        sweep = -sweep;
    }
    if (sweep >= 360) {
        SetRect(rp, Max(0,xmin-penSize), Max(0,ymin-penSize), xmax+penSize, ymax+penSize);
        return;
    }
    if (start < 0) {
```

```
        yb = ymax;
    } else {
        yb = Max(iy0, iy1);
    }
    SetRect(rp, Max(0,xl-penSize), Max(0,yt-penSize), xr+penSize, yb+penSize);
}

/*
```

```
        polygonPoints = newPolygon;
    }
    polygonBounds.left = Min(polygonBounds.left, x);
    polygonBounds.right = Max(polygonBounds.right, x);
    polygonBounds.top = Min(polygonBounds.top, y);
    polygonBounds.bottom = Max(polygonBounds.bottom, y);
    polygonPoints[nPolygonPoints].x = x;
    polygonPoints[nPolygonPoints].y = y;
    nPolygonPoints++;
}
```

```
        polygonPoints = newPolygon;
    }
    polygonBounds.left = Max(Min(polygonBounds.left, x-penSize),0);
    polygonBounds.right = Max(polygonBounds.right, x+penSize);
    polygonBounds.top = Max(Min(polygonBounds.top, y-penSize),0);
    polygonBounds.bottom = Max(polygonBounds.bottom, y+penSize);
    polygonPoints[nPolygonPoints].x = x;
    polygonPoints[nPolygonPoints].y = y;
    nPolygonPoints++;
}
```

BUG4：反复重画导致闪烁

触发方法：反复重画一个图形。

造成原因：程序在设置无效矩形（更新区域）时将最后一个参数设为了 TRUE，导致每次重画后更新时程序会用背景色白色对区域进行填充，导致闪烁。

```
    InvalidateRect(graphicsWindow, NULL, 1);
```

```
    InvalidateRect(graphicsWindow, NULL, FALSE);
```

解决方法：将程序所有的函数 InvalidateRect(..., ..., TRUE);的最后一个参数改为 FALSE。(出现位置：797,1235,1243,1302,1330,1406,1651).

BUG5：字号设置异常

触发方法: SetPointSize 通常不能正确地设置字号。

造成原因：变量含义混淆，代码笔误。

解决方法：如图。

```
+-- 45 lines: * Function: DisplayFont----------
                fontTable[fontIndex].font = newFont;
                fontTable[fontIndex].ascent = metrics.tmAscent;
                fontTable[fontIndex].descent = metrics.tmDescent;
                fontTable[fontIndex].height =
                  metrics.tmHeight + metrics.tmExternalLeading;
                fontTable[fontIndex].points =
                  metrics.tmHeight - metrics.tmInternalLeading;
                currentFont = fontIndex;
                textFont = CopyString(font);
                pointSize = fontTable[fontIndex].points;
                textStyle = style;
            } else {
                (void) SelectObject(osdc, oldFont);
```

```
+-- 45 lines: * Function: DisplayFont----------
                fontTable[fontIndex].font = newFont;
                fontTable[fontIndex].ascent = metrics.tmAscent;
                fontTable[fontIndex].descent = metrics.tmDescent;
                fontTable[fontIndex].height =
                  metrics.tmHeight + metrics.tmExternalLeading;
                fontTable[fontIndex].points =
                  size;
                currentFont = fontIndex;
                textFont = CopyString(font);
                pointSize = fontTable[fontIndex].size;
                textStyle = style;
            } else {
                (void) SelectObject(osdc, oldFont);
```

BUG6：抗锯齿字体无法擦除干净

触发方法：用 SetFont 设置一个带有抗锯齿的字体。

造成原因：导致擦不干净的具体原因尚不清楚，但知道如果使用默认带有抗锯齿的字体，擦除后悔留下痕迹。

解决方法：在加载字体时声明不使用抗锯齿。

NONANTIALIASED_QUALITY Font is never antialiased, that is, font smoothing is not done.

```
    fontIndex = FindExistingFont(fontName, size, style);
    if (fontIndex == -1) {
        newFont =
          CreateFont(-size, 0, 0, 0,
                     (style & Bold) ? FW_BOLD : FW_NORMAL,
                     (style & Italic) != 0,
                     0, 0, 0, 0, 0, NONANTIALIASED_QUALITY, FF_DONTCARE, fontName);
        if (newFont != NULL) {
```

```
    fontIndex = FindExistingFont(fontName, size, style);
    if (fontIndex == -1) {
        newFont =
          CreateFont(-size, 0, 0, 0,
                     (style & Bold) ? FW_BOLD : FW_NORMAL,
                     (style & Italic) != 0,
                     0, 0, 0, 0, 0, 0, 0, fontName);
        if (newFont != NULL) {
```

BUG7：部分字体的在特定字号下擦不干净

触发方法：已发现的，如 Courier New 字体，30 号字。

造成原因：同 BUG3

解决方法：原理同 BUG3，如图。

```
static void SetTextBB(RECT *rp, double x, double y, string text)
{
    SIZE textSize;
    int ix, iy;

    if (!GetTextExtentPoint(osdc, text, strlen(text), &textSize)) {
        Error("Internal error: Text size calculation failed");
    }
    ix = ScaleX(x);
    iy = ScaleY(y);
    SetRect(rp, Max(ix-1,0), Max( iy - textSize.cy + fontTable[currentFont].descent - 1, 0),
            ix + textSize.cx + 1, iy + fontTable[currentFont].descent + 1);
}
```

程设专题 2018 春夏学期 47 组

郭书廷

2018.5.27