第15章 抗锯齿

直线近似由一系列位于显示器坐标的像素组成，因此它们会显示锯齿，除了那些接近于水平或垂直线的直线。这种锯齿现象被称为图形失真。

抗锯齿是平滑的直线或曲线。它减少了锯齿现象，不完全是水平或垂直方向的任何直线的阶梯现象。μC/GUI 支持不同的抗锯齿质量，抗锯齿字体和高分辨率坐标。

抗锯齿的支持是一个独立的软件项目，它不包括在μC/GUI 基本的软件包中。抗锯齿软件位于子目录 GUI\AntiAlias 下。

15.1 介绍

抗锯齿通过“混合”前景色和背景色来达到平滑曲线和斜线的效果。前景色和背景色之间用到的阴影的数量越多，抗锯齿效果越佳（计算时间相应的也越长）。

抗锯齿的品质

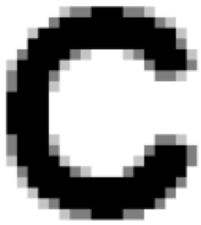
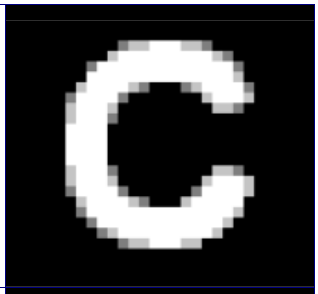
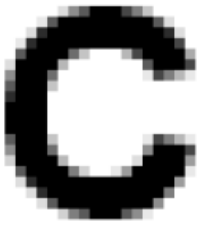
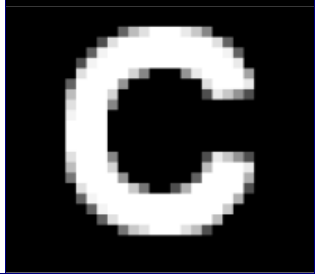
抗锯齿的品质由函数 GUI_AA_SetFactor 设定。这在本章的后面部分介绍。对于在抗锯齿和相应结果之间关系的概念，看一看所绘出的图片。

第一条直线在没有抗锯齿的情况下（抗锯齿系数为 1）绘出，第二条直线使用抗锯齿系数 2 绘出。这意思是从前景到背景的阴影数值为 2×2＝4。下一条直线使用抗锯齿系数 3 绘出，因此阴影数值为 3×3＝9，以下类推。抗锯齿系数取 4 对于大多数应用来说应该是足够了，再增大抗锯齿系数对于最终结果的影响并不显著，只会增加计算时间。

抗锯齿字体

支持两种类型的抗锯齿字体，低质量（2bpp）和高质量（4bpp）。当要使用它们的时候，程序需要显示这些字体会自动连接。下表显示没有使用抗锯齿和使用两种抗饱锯齿字体分别绘一个字符“C”的效果：

字体类型	黑色在白色上	白色在黑色上
标准 （无抗锯齿） 1 bpp 2 级阴影		

低质量 （抗锯齿） 2 bpp 4 级阴影		
高质量 （抗锯齿） 4 bpp 16 级阴影		

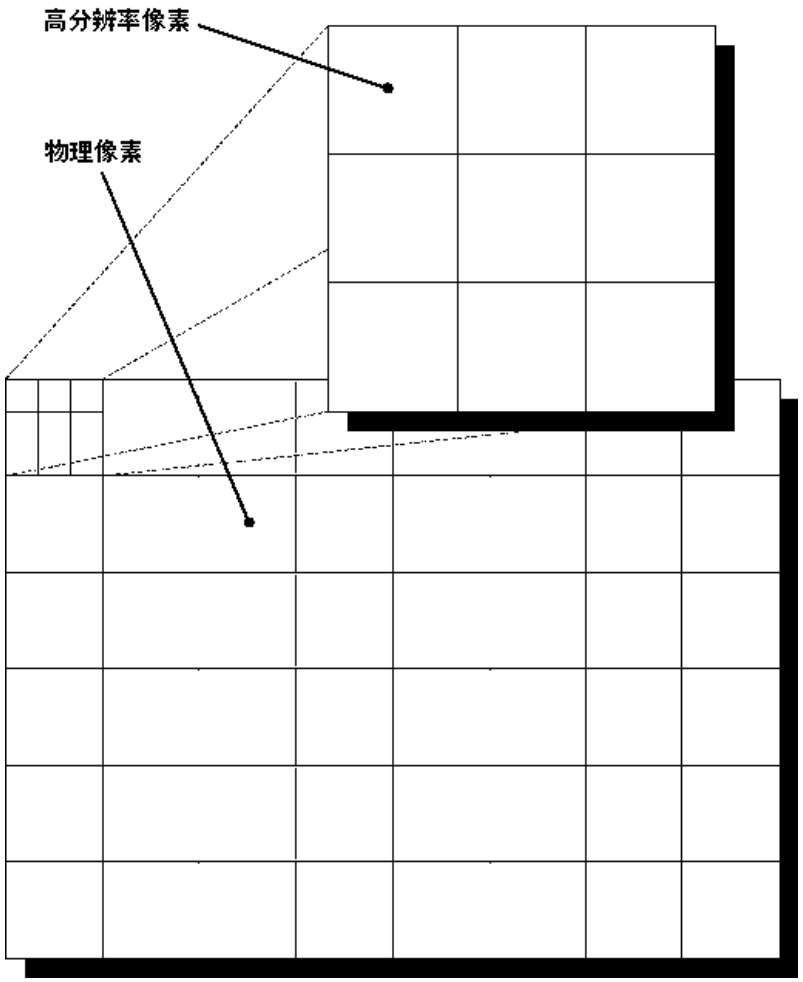
抗锯齿字体可以由μC/GUI 字体转换器建立。通常使用抗锯齿字体的目的是改善文字的外观。虽然使用高质量抗锯齿的效果比低质量的抗锯齿更具视觉上的舒适性，但是相应的，它也会占用更多的计算时间和消耗更多的内存。与非抗锯齿（1bpp）字体相比，低质量（2bpp）字体消耗的内存是其两倍，而高质量（4bpp）字体消耗的内存是其 4 倍。

高分辨率座标

当使用抗锯齿进行一个项目绘制时，使用的座标与正常（非抗锯齿）的绘图函数的一样的，这是默认模式。在函数参数中你不必考虑抗锯齿系数。例如，从（50, 100）到（100, 50）绘一条抗锯齿直线，程序应当这样写：

```
GUI_AA_DrawLine(50, 100, 100, 50);
```

μC/GUI 的高分辨率特性让你使用由抗锯齿系数和显示屏尺寸所决定的虚拟区域。高分辨率座标必须由函数 GUI_AA_EnableHiRes 启动，使用 GUI_AA_DisableHiRes 函数进行显示。这两个函数在本章的后面部分说明。使用高分辨率座标的优点是对象不仅可以放置在显示屏的物理坐标中，也可以放在这些物理坐标“中间”。下面展示了一个抗锯齿系数是 3 的高分辨率像素虚拟区域。



为了使用抗锯齿系数为 3 的高分辨率模式从像素（50,100）到（100,50）绘一条直线，程序应当这样写：

```
GUI_AA_DrawLine(150, 300, 300, 150);
```

使用高分辨率特性的范例程序请参考本章最后的范例。

15.2 抗锯齿 API 函数

下表列出了μC/GUI 的抗锯齿软件包的有效函数，所有函数在各自的类型中按字母顺序进行排列。函数的详细描述在后面列出。

控制函数	说 明
GUI_AA_DisableHiRes()	禁止高分辨率座标。
GUI_AA_EnableHiRes()	启用高分辨率座标。
GUI_AA_GetFactor()	返回当前的抗锯齿系数。
GUI_AA_SetFactor()	设置当前的抗锯齿系数。

绘图函数	说 明
GUI_AA_DrawArc()	绘一段抗锯齿圆弧。
GUI_AA_DrawLine()	绘一根抗锯齿直线。
GUI_AA_DrawPolyOutline()	绘一个抗锯齿多边形的轮廓。
GUI_AA_FillCircle()	绘一个抗锯齿的圆。
GUI_AA_FillPolygon()	绘一个填充和抗锯齿的多边形。

15.3 控制函数

GUI_AA_DisableHiRes()

描述

禁止高分辨率座标。

函数原型

```
void GUI_AA_DisableHiRes(void);
```

附加信息

默认情况下，高分辨率座标被禁止。

GUI_AA_EnableHiRes()

描述

启用高分辨率座标。

函数原型

```
void GUI AA EnableHiRes(void);
```

GUI_AA_GetFactor()

描述

返回当前的抗锯齿品质系数。

函数原型

```
int GUI_AA_GetFactor(void);
```

返回数值

当前的抗锯齿系数。

GUI_AA_SetFactor()

描述

设置当前的抗锯齿品质系数。

函数原型

```
void GUI_AA_SetFactor(int Factor);
```

参　数	含　意
Factor	新的抗锯齿系数

附加信息

尽管设置参数 Factor 为 1 是允许的，这将会禁止抗锯齿，导致使用一种标准字体。我们还是推荐使用抗锯齿，其品质系数范围是 2～4。默认系数是 3。

15.4 绘图函数

GUI_AA_DrawArc()

描述

在当前窗口的指定位置，使用当前画笔大小和画笔形状显示一段抗锯齿的圆弧。

函数原型

```
void GUI_AA_DrawArc(int x0, int y0, int rx, int ry, int a0, int a1);
```

参　数	含　意
x0	中心的水平坐标
y0	中心的垂直坐标

rx	水平半径

限制

现在 ry 参数无效，取而代之的是 rx 参数。

附加信息

如果工作在高分辨率模式，坐标和半径必须在高分辨率座标内否则，必须以像素值指定。

GUI_AA_DrawLine()

描述

在当前窗口的指定位置，使用当前画笔大小及画笔形状显示一条抗锯齿的直线。

函数原型

```
void GUI_AA_DrawLine(int x0, int y0, int x1, int y1);
```

参　数	含　意
x0	起始 X 轴坐标。
y0	起始 Y 轴坐标。

附加信息

如果工作在高分辨率模式，座标必须在高分辨率座标里面。否则，必须以像素值指定。

GUI_AA_DrawPolyOutline()

描述

在当前窗口的指定位置指定线宽，显示一个由一系列点组成的多边形的轮廓线。

函数原型

```
void GUI_AA_DrawPolyOutline (   const GUI_POINT* pPoint,
                                int NumPoints,
                                int Thickness,
                                int x, int y)
```

参 数	含 意
PPoint	显示的多边形的指针。
NumPoints	点的序列中指定点的数量。
Thickness	轮廓的线宽。

附加信息

绘出的折线自动连接起点和终点而形成闭合。起始点不必重复指定为终点。如果工作在高分辨率模式，座标必须在高分辨率座标里面。否则，必须以像素值指定。

范例

```
#define  countof(Array)     (sizeof(Array) / sizeof(Array[0]))
static GUI_POINT aPoints[] =
{
    {0,0},{15,30},{0,20},{-15,30}
}
void Sample (void)
{
    GUI_AA_DrawPolyOutline(aPoints, countof(aPoints), 3, 150, 40);
}
```

上面范例程序执行结果的屏幕截图

GUI_AA_FillCircle()

描述

在当前窗口的指定位置显示一个填充和抗锯齿的圆。

函数原型

```
void GUI_AA_FillCircle(int x0, int y0, int r);
```

参 数	含 意
x0	圆的中心水平坐标（在客户窗口以像素为单位）
y0	圆的中心垂直坐标（在客户窗口以像素为单位）
r	圆的半径（直径的一半）

附加信息

如果工作在高分辨率模式，座标必须在高分辨率座标里面。否则，必须以像素值指定。

GUI_AA_FillPolygon()

描述

在当前窗口指定位置，能过一个点的序列填充一个抗锯齿多边形。

函数原型

```
void GUI_AA_FillPolygon(const GUI_POINT* pPoint, int NumPoints,int x,int y)
```

参 数	含 意
PPoint	显示的多边形的指针。
NumPoints	点的序列中指定点的数量。
x	原点的 X 轴坐标。
y	原点的 Y 轴坐标。

附加信息

绘出的折线自动连接起点和终点而形成闭合。起始点不必重复指定为终点。

如果工作在高分辨率模式，坐标必须在高分辨率座标中。否则，它们必须以像素为单位指定。

15.5 范例

不同的抗锯齿系数

下面的范例使用非抗锯齿方式和抗锯齿方式绘斜线。源代码文件是\Misc\AntialiasedLines.c。

```
/*----------------------------------------------------------------------
```

```
文件:       AntialiasedLines.c
目的:       使用不同的抗锯齿质量显示一条直线。
----------------------------------------------------------------------*/

#include "GUI.H"

/*********************************************************************
*                    使用不同的抗锯齿质量显示一条直线                  *
*********************************************************************/

static void DemoAntialiasing(void)
{
    int i, x1, x2;
    int y = 2;
/* 设置绘图特性 */
    GUI_SetColor(GUI_BLACK);
    GUI_SetBkColor(GUI_WHITE);
    GUI_SetPenShape(GUI_PS_FLAT);
    GUI_Clear();
    x1 = 10;
    x2 = 90;
/* 绘一条没有抗锯齿的直线 */
    GUI_DispStringHCenterAt("\nNormal", (x1 + x2) / 2, 10);
    for (i = 1; i < 12; i++)
    {
        GUI_SetPenSize(i);
        GUI_DrawLine(x1, 40 + i * 15, x2, 40 + i * 15 + y);
    }
    x1 = 110;
    x2 = 190;
/* 绘一条抗锯齿品质因数为 2 的直线 */
    GUI_AA_SetFactor(2);
    GUI_DispStringHCenterAt (  "Antialiased\n\nusing factor 2",
                               (x1 + x2) / 2, 10);
    for (i = 1; i < 12; i++)
    {
        GUI_SetPenSize(i);
```

```
        GUI_AA_DrawLine(x1, 40 + i * 15, x2, 40 + i * 15 + y);
    }
    x1 = 210;
    x2 = 290;
/*  绘一条抗锯齿品质因数为 6 的直线 */
    GUI_AA_SetFactor(6);
    GUI_DispStringHCenterAt (  "Antialiased\n\nusing factor 6",
                               (x1 + x2) / 2, 10);
    for (i = 1; i < 12; i++)
    {
        GUI_SetPenSize(i);
        GUI_AA_DrawLine(x1, 40 + i * 15, x2, 40 + i * 15 + y);
    }
}

/*********************************************************************
*                                主函数                              *
*********************************************************************/

void main(void)
{
    GUI_Init();
    DemoAntialiasing();
    while(1)
        GUI_Delay(100);
}
```

上面范例执行结果的屏幕截图

正常　　使用抗锯齿系数 2　　使用抗锯齿系数 6

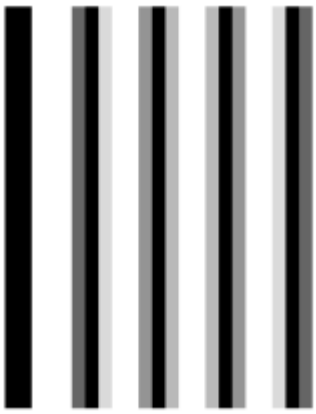
在高分辨率座标中放置的直线

这个范例展示在高分辨率座标中放置一条抗锯齿直线。源代码文件是：Sample\Misc\HiResPixel.c.

```
/*-----------------------------------------------------------------------
文件:       HiResPixel.c
目的:       高分辨率像素的示范
-----------------------------------------------------------------------*/

#include "GUI.H"

/*************************************************************************
*                        显示一条在高分辨率像素中的直线                  *
*************************************************************************/

void ShowHiResPixel(void)
{
    int i, Factor = 5;
    GUI_SetBkColor(GUI_WHITE);
    GUI_SetColor(GUI_BLACK);
    GUI_Clear();
    GUI_SetPenSize(2);
    GUI_SetPenShape(GUI_PS_FLAT);
    GUI_AA_EnableHiRes();                /* 启动高分辨率 */
```

```
    GUI_AA_SetFactor(Factor);             /* 设置品质系数 */
/* 使用虚拟分辨率绘直线 */
    for (i = 0; i < Factor; i++)
    {
        int x = (i + 1) * 5 * Factor + i - 1;
        GUI_AA_DrawLine(x, 50, x, 199);
    }
}

/*********************************************************************
*                                主函数                              *
*********************************************************************/

void main (void)
{
    GUI_Init () ;
    ShowHiResPixel();
    while(1);
}
```

上面范例执行结果的屏幕截图放大后的效果



使用高分辨率抗锯齿绘运动的的指针

这个范例通过绘一个旋转的指针（每步旋转 0.1 度）展示高分辨率抗锯齿的应用。本例没有屏幕截图，因为高分辨率抗锯齿效果只有在指针运动时才可见。没有使用高分辨率的情况下，指针的运动是以短暂的“跳动”形式显示。而在高分辨率下，不会出现跳动。

范例源代码文件是：\Misc\HiResAntialiasing.c。

```
/*-----------------------------------------------------------------------
文件:       HiResAntialiasing.c
目的:       高分辨率抗锯齿展示
-----------------------------------------------------------------------*/

#include "GUI.H"

/*********************************************************************
*                                数据                                *
*********************************************************************/

#define countof (Obj)   (sizeof (Obj) / sizeof (Obj[0] ) )
static const GUI_POINT aPointer[] =
{
    {0,3}, {85,1}, {90,0}, {85,-1}, {0,-3}
};
static GUI_POINT aPointerHiRes[countof(aPointer)];
typedef struct
{
    GUI_AUTODEV_INFO AutoInfo;
    GUI_POINT aPoints[countof(aPointer)];
    int Factor;
} PARAM;

/*********************************************************************
*                                绘图函数                            *
*********************************************************************/

static void DrawHiRes(void * p)
{
    PARAM * pParam = (PARAM *)p;
    if (pParam->AutoInfo.DrawFixed)
    {
        GUI_ClearRect(0, 0, 99, 99);
    }
    GUI_AA_FillPolygon( pParam->aPoints,
```

```
                        countof(aPointer),
                        5 * pParam->Factor,
                        95 * pParam->Factor);
}

static void Draw(void * p)
{
    PARAM * pParam = (PARAM *)p;
    if (pParam->AutoInfo.DrawFixed)
    {
        GUI_ClearRect(100, 0, 199, 99);
    }
    GUI_AA_FillPolygon(pParam->aPoints, countof(aPointer), 105, 95);
}

/*********************************************************************
*                    通过绘旋转的指针来展示高分辨率                    *
*********************************************************************/

static void ShowHiresAntialiasing(void)
{
    int i;
    GUI_AUTODEV aAuto[2] ;
    PARAM Param;
    Param.Factor = 3;
    GUI_DispStringHCenterAt("Using\nhigh\nresolution\nmode", 50, 120);
    GUI_DispStringHCenterAt("Not using\nhigh\nresolution\nmode", 150, 120);
/* 建立 GUI_AUTODEV 物体 */
    for (i = 0; i < countof(aAuto); i++)
    {
        GUI_MEMDEV_CreateAuto(&aAuto[i]);
    }
/* 计算指针的高分辨率坐标 */
    for (i = 0; i < countof(aPointer); i++)
    {
        aPointerHiRes[i].x = aPointer[i].x * Param.Factor;
        aPointerHiRes[i].y = aPointer[i].y * Param.Factor;
```

```
    }
    GUI_AA_SetFactor(Param.Factor);                    /* 设置抗锯齿系数 */
    while(1)
    {
        for (i = 0; i < 1800; i++)
        {
            float Angle = (i >= 900) ? 1800 - i : i;
            Angle *= 3.1415926f / 1800;
    /* 在高分辨率中绘指针 */
            GUI_AA_EnableHiRes();
            GUI_RotatePolygon ( Param.aPoints,
                                aPointerHiRes,
                                countof(aPointer),
                                Angle);
            GUI_MEMDEV_DrawAuto(&aAuto[0],
                                &Param.AutoInfo,
                                DrawHiRes,
                                &Param);
    /* 在非高分辨率中绘指针 */
            GUI_AA_DisableHiRes();
            GUI_RotatePolygon ( Param.aPoints,
                                aPointer,
                                countof(aPointer),
                                Angle);
            GUI_MEMDEV_DrawAuto(&aAuto[1],
                                &Param.AutoInfo,
                                Draw,
                                &Param);
            #ifdef WIN32
                GUI_Delay(2);
            #endif
        }
    }
}

/*********************************************************************
*                          主函数                                    *
```

```
*********************************************************************/

void main (void)
{
    GUI_Init();
    ShowHiresAntialiasing();
}
```