SimpleGUI

一套针对单色显示屏的开源 GUI 接口

SimpleGUI API 文档

开源，是一种态度。

目录

目录 2

文件结构 3

环境参数设置 4

数据类型定义 5

API 7

文件结构

Todo

环境参数设置

为方便用户配置和移植，SimpleGUI 在 SGUI _Config.h 文件中定义了一些列的控制和开关宏，用户可以通过打开、关闭以及修改宏定义的值来对 SimpleGUI 的一些全局属性进行修改。

1、 _SIMPLE_GUI_ENABLE_ICONV_GB2312_

此宏定义关联文字显示 API 对非 ASCII 文字的解码方式。设置值大于 0 时有效，文字相关的 API 接口将会视输入的文字为 UTF-8 格式并转换成 GB2312 格式进行解析，若此值为 0，则视输入文字为 GB2312 格式进行解析。具体操作将会在“文字显示”章节详细阐述。

2、 _SIMPLE_GUI_VIRTUAL_ENVIRONMENT_SIMULATOR_

此宏定义关联 SimpleGUI 的运行环境，设置值大于 0 时有效。当此宏定义有效，意味着 SimpleGUI 正运行于模拟环境中，用户在移植到目标平台后，请将此宏定义的值修改为 0。

3、 _SIMPLE_GUI_ENABLE_DYNAMIC_MEMORY_

此宏定义关联 SimpleGUI 移植目标平台的动态内存操作， 设置值大于 0 时有效。当此宏定义设置为有效时，意味着目标平台已实现了动态内存管理或支持动态内存管理，且用户已经做好相应的实现或移植，此时 SimpleGUI 的相关依赖内容也会被使能，如列表项目的动态增减。

4、 _SIMPLE_GUI_ENABLE_BASIC_FONT_

此宏定义关联 SimpleGUI 的内置基础字体， 设置值大于 0 时有效。SimpleGUI 内部存储了一组尺寸为 6*8 像素的可见 ASCII 字符集数据，当此宏定义有效时，此套字体数据将被支持，且文字显示 API 将可以使用此文字数据显示基本 ASCII 字符内容。

此功能设计为，如果用户使用外部字库且外部字库出现损坏或数据异常时，可以使用此字库数据输出一些警告或调试信息。

数据类型定义

1 基本数据类型定义。

为避免因平台和编译器的差异造成的对基本数据类型的定义不同，进而导致代码产生不可预知的错误，SimpleGUI 在 SGUI_Typedef.h 文件中重新定义了包括整数、字符、字符指针在内的一系列基本数据类型，在 SimpleGUI 内部的代码实现中将一致性的使用重新定义过的数据类型，用户需要在使用前明确所在平台的数据类型定义并作出相应修改，以避免因溢出或其他异常导致的错误。

具体内容请参照 SGUI_Typedef.h 文件。

2 特殊数据类型定义。

为方便 API 的实现、表达与使用，SimpleGUI 在基本数据类型的基础上，定义了一些特殊的数据结构。

2.1 矩形区域

此定义主要用于位图绘制时，标示显示区域、实际位图数据与位图数据偏移，其原型定义如下：

```
typedef struct _st_rectangle_
{
    SGUI_INT            PosX;
    SGUI_INT            PosY;
    SGUI_INT            Width;
    SGUI_INT            Height;
}SGUI_RECT_AREA;
```

PosX 为矩形左上角的 X 坐标。

PosY 为矩形左上角的 Y 坐标。

Width 为矩形宽度。

Height 为矩形高度。

由于此数据结构是为绘图设计，在实际使用中，除了以上四项参数外，可能还需要其他判断，为此，此数据类型还有相应的运算宏定义：

```
#define         RECTANGLE_X_START(ST)         ((ST).PosX)
#define         RECTANGLE_X_END(ST)           (((ST).PosX + (ST).Width - 1))
#define         RECTANGLE_Y_START(ST)         ((ST).PosY)
#define         RECTANGLE_Y_END(ST)           (((ST).PosY + (ST).Height - 1))
#define         RECTANGLE_WIDTH(ST)           ((ST).Width)
#define         RECTANGLE_HEIGHT(ST)          ((ST).Height)
#define         RECTANGLE_VALID_WIDTH(ST)
((RECTANGLE_X_START(ST)>0)?RECTANGLE_WIDTH(ST):(RECTANGLE_WIDTH(ST)+RECTANGLE_X_START(ST))
)
#define         RECTANGLE_VALID_HEIGHT(ST)
((RECTANGLE_Y_START(ST)>0)?RECTANGLE_HEIGHT(ST):(RECTANGLE_HEIGHT(ST)+RECTANGLE_Y_START(ST
)))
```

RECTANGLE_X_START 意为矩形左边界 X 轴坐标。

RECTANGLE_X_END 意为矩形右边界 X 坐标。

RECTANGLE_Y_START 意为矩形上边界 Y 坐标。

RECTANGLE_Y_END 意为矩形下边界 Y 坐标。

RECTANGLE_WIDTH 意为矩形宽度。

RECTANGLE_HEIGHT 意为矩形高度。

RECTANGLE_VALID_WIDTH 意为矩形的可见宽度。

RECTANGLE_VALID_HEIGHT 意为矩形的可见高度。

2.2 实时时钟

此定义主要用于在 HMI 引擎中调用和显示实时时钟，与共通函数中的 GetNowTime 函数搭配使用，原型定义如下：

```
typedef struct
{
    SGUI_UINT16             Year;
    SGUI_UINT16             Month;
    SGUI_UINT16             Day;
    SGUI_UINT16             Hour;
    SGUI_UINT16             Minute;
    SGUI_UINT16             Second;
}SGUI_TIME;
```

其中 Year、Month、Day、Hour、Minute、Second 分别对应年、月、日时、分、秒。

API

1. 绘图 API

绘图 API 的实现文件位于 GUI 文件夹中，主要负责显示屏硬件的控制与屏幕绘图。

SimpleGUI 的绘图 API 名称全部遵从以下格式:

SGUI_分类_函数名(参数...)

所有绘图 API 均以 SGUI 开头，分类表示这个函数的用途类型，后面的函数名则是标识函数的具体用途。

1.1 共通处理

共通处理函数实现在 SGUI_Common.c 文件中，主要负责 SimpleGUI 的全局共通处理和 SimpleGUI 与外部的数据交互，函数名全部以 SGUI_Common 开始。

1.1.1 SGUI_Common_IntegerToStringWithDecimalPoint

功能描述：将一个有符号整数转化为字符串并在指定位置插入小数点。

原型声明：SGUI_SIZE SGUI_Common_IntegerToStringWithDecimalPoint (SGUI_INT iInteger, SGUI_UINT uiDecimalPlaces, SGUI_PSZSTR pszStringBuffer, SGUI_INT iAlignment, SGUI_CHAR cFillCharacter)

参数说明：

iInteger：将要被转换的数字。

uiDecimalPlaces：小数位数，如果为 0 则不插入小数点。

pszStringBuffer：转换输出字符串的输出缓存。

iAlignment：对其方式与宽度，单位为半角字符宽度，大于 0 则右对齐，小于 0 则左对齐。如果转换完的宽度大于对齐宽度，则以转换完的实际宽度为准。

cFillCharacter：若对其后还有留白位置，则以此字符填充，通常使用空格。

返回值：转换的字符串长度。

注意事项：请注意转换输出缓冲区的长度，如果发生内存越界，将产生不可预知的错误。

1.1.2 SGUI_Common_IntegerToString

功能描述：将一个有符号整数转化为字符串。

原型声明：SGUI_SIZE SGUI_Common_IntegerToString (SGUI_INT iInteger, SGUI_PSZSTR pszStringBuffer, SGUI_UINT uiBase, SGUI_INT iAlignment, SGUI_CHAR cFillCharacter)

参数说明：

iInteger：将要被转换的数字。

pszStringBuffer：转换输出字符串的输出缓存。

uiBase：转换基数，只允许 8、10 和 16 进制。

iAlignment：对其方式与宽度，单位为半角字符宽度，大于 0 则右对齐，小于 0 则左对齐。如果转换完的宽度大于对齐宽度，则以转换完的实际宽度为准。

cFillCharacter：若对其后还有留白位置，则以此字符填充，通常使用空格。

返回值：转换的字符串长度。

注意事项：请注意转换输出缓冲区的长度，如果发生内存越界，将产生不可预知的错误。

1.1.3 SGUI_Common_ConvertStringToUnsignedInteger

功能描述：将一个字符串的有效部分转换为一个无符号整数。

原型声明：SGUI_UINT SGUI_Common_ConvertStringToUnsignedInteger (SGUI_PSZSTR szString, SGUI_CHAR** ppcEndPointer, SGUI_UINT uiBase)

参数说明：

szString：将要被转换的字符串。

ppcEndPointer：转换结束处的字符指针，如果字符串中出现了非数字字符，则会在该处终止，同时此指针指向该处，若转换至字符串尾，则该指针也指向字符串尾的 NULL。

uiBase：转换基数，只允许 8、10 和 16 进制。

返回值：转换的数字。

注意事项：若输入的字符串第一个字符即为非数字，则返回值为 0。

1.1.4 SGUI_Common_ConvertStringToInteger

功能描述：将一个字符串的有效部分转换为一个有符号整数。

原型声明：SGUI_INT SGUI_Common_ConvertStringToInteger (SGUI_PSZSTR szString, SGUI_CHAR** ppcEndPointer, SGUI_UINT uiBase)

参数说明：

szString：将要被转换的字符串。

ppcEndPointer：转换结束处的字符指针，如果字符串中出现了非数字字符，则会在该处终止，同时此指针指向该处，若转换至字符串尾，则该指针也指向字符串尾的 NULL。

uiBase：转换基数，只允许 8、10 和 16 进制。

返回值：转换的数字。

注意事项：若输入的字符串第一个字符即为非数字，则返回值为 0。

1.1.5 SGUI_Common_EncodeConvert

功能描述：字符串编码转换。

原型声明：SGUI_PSZSTR SGUI_Common_EncodeConvert
(SGUI_PCSZSTR szSourceEncode, SGUI_PSZSTR szDestinationEncode, SGUI_PSZSTR szSource)

参数说明：

szSourceEncode：源字符串编码。

szDestinationEncode：目标编码。

szSource：要转换的字符串。

返回值：转换缓冲的指针。

注意事项：此函数依赖 iconv 外部库，用于在模拟器环境中使用 UTF-8 编码格式的字符串，由于示例的字符解码使用 GB2312 格式，故需要转码。由于转码库体积庞大，所以通常情况下此函数不需要在目标单片机平台上实现，届时只需要关闭_SIMPLE_GUI_ENABLE_ICONV_GB2312_宏定义即可。

1.1.6 SGUI_Common_Allocate

功能描述：申请堆内存空间。

原型声明：SGUI_Common_Allocate(SGUI_SIZE uiSize)

参数说明：

uiSize：要申请的字节数。

返回值：申请到的内存空间头指针。

注意事项：模拟环境中，此环境是对 C 标准函数 malloc 的重新封装，移植到目标平台后如果想使用此函数，请确认平台支持或用户已自行实现 MMU 后，重写此函数，并将宏_SIMPLE_GUI_ENABLE_DYNAMIC_MEMORY_有效化。

1.1.7 SGUI_Common_Free

功能描述：释放堆内存空间。

原型声明：SGUI_Common_Free(void* pFreePointer)

参数说明：

pFreePointer：要释放的内存头指针。

返回值：无。

注意事项：模拟环境中，此环境是对 C 标准函数 free 的重新封装，移植到目标平台后如果想使用此函数，请确认平台支持或用户已自行实现 MMU 后，重写此函数，并将宏_SIMPLE_GUI_ENABLE_DYNAMIC_MEMORY_有效化。

1.1.8 SGUI_Common_MemoryCopy

功能描述：复制内存块。

原型声明：void* SGUI_Common_MemoryCopy
(void* pDest, const void* pSrc, SGUI_UINT uiSize)

参数说明：

pDest：目标内存块头指针。

pSrc：源内存块头指针。

uiSize：复制内存块的大小，单位字节。

返回值：目标内存块头指针。

注意事项：模拟环境中，此环境是对 C 标准函数 memcpy 的重新封装，用户如果不使用标准库，则需自行实现内存复制过程。

1.1.9 SGUI_Common_MemorySet

功能描述：设置内存块。

原型声明：void SGUI_Common_MemorySet
(void* pMemoryPtr, SGUI_BYTE iSetValue, SGUI_UINT uiSize)

参数说明：

pMemoryPtr：要设置的内存块头指针。

iSetValue：要设置的每一个字节的值。

uiSize：内存块的大小，单位字节。

返回值：目标内存块头指针。

注意事项：模拟环境中，此环境是对 C 标准函数 memcpy 的重新封装，用户如果不使用标准库，则需自行实现内存复制过程。

1.1.10 SGUI_Common_StringLength

功能描述：测量字符串的长度。

原型声明：SGUI_SIZE SGUI_Common_StringLength
(SGUI_PCSZSTR szString)

参数说明：

szString：字符串头指针。

返回值：字符串长度。

注意事项：模拟环境中，此环境是对 C 标准函数 strlen 的重新封装，用户如果不使用标准库，则需自行实现字符串长度计算的过程。

1.1.11 SGUI_Common_StringCopy

功能描述：复制字符串。

原型声明：SGUI_PSZSTR SGUI_Common_StringCopy(SGUI_PSZSTR szDest, SGUI_PCSZSTR szSrc)

参数说明：

szDest：复制的字符串缓存。

szSrc：被复制的字符串缓存。

返回值：复制的字符串缓存头指针。

注意事项：模拟环境中，此环境是对 C 标准函数 strcpy 的重新封装，用户如果不使用标准库，则需自行实现字符串复制的过程。

1.1.12 SGUI_Common_StringLengthCopy

功能描述：复制不超过特定长度的字符串。

原型声明：SGUI_PSZSTR SGUI_Common_StringLengthCopy (SGUI_PSZSTR szDest, SGUI_PCSZSTR szSrc, SGUI_SIZE uiSize)

参数说明：

szDest：复制的字符串缓存。

szSrc：被复制的字符串缓存。

uiSize：复制字符串的长度，单位为字节。

返回值：复制的字符串缓存头指针。

注意事项：模拟环境中，此环境是对 C 标准函数 strncpy 的重新封装，用户如果不使用标准库，则需自行实现字符串复制的过程。

1.1.13 SGUI_Common_GetNowTime

功能描述：获取当前特定系统时间。

原型声明：void SGUI_Common_GetNowTime (SGUI_TIME* pstTime)

参数说明：

pstTime：保存时间数据的结构体。

返回值：复制的字符串缓存头指针。

注意事项：此函数需绑定系统 RTC 的处理，需要用户更具目标平台自行实现，读取 RTC 时间并赋值到参数指定的 RTC 结构体中。

1.1.14 SGUI_Common_RefreshScreen

功能描述：刷新屏幕显示。

原型声明：void SGUI_Common_RefreshScreen(void)

参数说明：无。

返回值：无。

注意事项：用于更新屏幕显示的接口，需要用户自行实现，通常用于使用了显示缓存的情况下。

1.1.15 SGUI_Common_ReadFlashROM

功能描述：读取外部存储中的数据。

原型声明：void SGUI_Common_ReadFlashROM(SGUI_ROM_ADDRESS uiAddressHead, SGUI_SIZE uiDataLength, SGUI_BYTE* pBuffer)

参数说明：

uiAddressHead：读取的首地址。

uiDataLength：读取数据的长度。

pBuffer：存放读取数据的缓冲区的首地址。

返回值：无。

注意事项：此函数需根据实际系统平台，实现从内部或外部 Flash 中读取数据的操作。在 SimpleGUI 中，此函数通常用于读取字模、图片或图标数据等。

1.1.16 SGUI_Common_Delay

功能描述：延时等待。

原型声明：void SGUI_Common_Delay(SGUI_UINT32 uiTimeMs)

参数说明：

uiTimeMs：等待的毫秒数。

返回值：无。

注意事项：根据实际系统平台实现的延时函数，用于 HMI 引擎中，绘图引擎不使用此函数。

1.2 坐标系定义

在 SimpleGUI 的绘图操作中，以屏幕有效显示区域的左上角为坐标原点，X 轴向右为正方向，Y 轴向下为正方向，起始坐标为(0,0)，以屏幕像素为单位向正方向增长。

1.3 基础绘图

基础绘图函数的实现位于 SGUI_Basic.c 文件中，主要负责 SimpleGUI 中点、线、面和位图等基本图形的绘制功能，函数名全部以 SGUI_Basic 开始。

1.3.1 数据类型定义

为方便基础绘图 API 的实现与使用，SimpleGUI 在此部分定义了两种数据类型分别用于像素点颜色的说明和绘图方式的说明。

像素点颜色类型为 SGUI_COLOR，其原型定义如下：

```
typedef enum
{
    SGUI_COLOR_BKGCLR    = 0,
    SGUI_COLOR_FRGCLR    = 1,
    SGUI_COLOR_TRANS     = 2,
}SGUI_COLOR;
```

SGUI_COLOR_BKGCLR 为背景色，通常指显示屏幕上未被点亮（或未被有效化）的点的颜色。

SGUI_COLOR_FRGCLR 为前景色，通常指显示屏幕上被点亮（或被有效化）的点的颜色。

SGUI_COLOR_TRANS 为透明。指忽略当前点的绘制，保持当前点（或区域）的显示状态不变。

绘图模式型为 SGUI_DRAW_MODE，其原型定义如下：

```
typedef enum
{
    SGUI_DRAW_NORMAL    = 0,
    SGUI_DRAW_REVERSE   = 1,
}SGUI_DRAW_MODE;
```

SGUI_DRAW_NORMAL 为正常绘制，指根据上述 SGUI_COLOR 类型中对点颜色的定义进行绘制。

SGUI_DRAW_REVERSE 为反色绘制，指在绘制是将上述 SGUI_COLOR 类型中的 SGUI_COLOR_BKGCLR 与 SGUI_COLOR_FRGCLR 两种定义互换，绘制反色图形，此定义通常用于绘制反色位图。

1.3.2 SGUI_Basic_ClearScreen

功能描述：清空屏幕显示。

原型声明：void SGUI_Basic_ClearScreen(void)

参数说明：无。

返回值：无。

注意事项：因需要适配的硬件平台不同，有的屏幕设备支持清空操作，有的则必须手动填充，所以此函数需要用户根据实际使用的设备自行实现。

1.3.3 SGUI_Basic_DrawPoint

功能描述：在屏幕上绘制点。

原型声明：void SGUI_Basic_DrawPoint(SGUI_UINT uiPosX, SGUI_UINT uiPosY, SGUI_COLOR eColor)

参数说明：

uiPosX：要绘制点的 X 坐标。

uiPosY：要绘制点的 Y 坐标。

eColor：绘制点的颜色。

返回值：无。

注意事项：SimpleGUI 只针对单色屏幕设计，绘制点的颜色只有“黑”和“白”两种，分别对应像素的设置状态和清空状态，详情请参考 SGUI_COLOR_BKGCLR 数据类型的定义。

1.3.4 SGUI_Basic_DrawLine

功能描述：在屏幕上绘制直线段。

原型声明：void SGUI_Basic_DrawLine (SGUI_INT uiStartX, SGUI_INT uiStartY, SGUI_INT uiEndX, SGUI_INT uiEndY, SGUI_COLOR eColor)

参数说明：

uiStartX：线段起始点的 X 坐标。

uiStartY：线段起始点的 Y 坐标。

uiStartX：线段终止点的 X 坐标。

uiStartY：线段终止点的 Y 坐标。

eColor：绘制线段的颜色。

返回值：无。

注意事项：线段起止点的坐标值可以为负值，当为负值时意为坐标位于屏幕显示区域上侧或左侧以外的区域。超出屏幕显示区域部分的线段将不会被绘制。

1.3.5 SGUI_Basic_DrawRectangle

功能描述：在屏幕上绘制封闭矩形。

原型声明：void SGUI_Basic_DrawRectangle (SGUI_UINT uiStartX, SGUI_UINT uiStartY, SGUI_UINT uiWidth, SGUI_UINT uiHeight, SGUI_COLOR eEdgeColor, SGUI_COLOR eFillColor)

参数说明：

uiStartX：矩形左上角点的 X 坐标。

uiStartY：矩形左上角点的 Y 坐标。

uiWidth：矩形的宽度，以像素为单位。

uiStartY：矩形的高度，以像素为单位。

eEdgeColor：矩形边框的颜色。

eFillColor：矩形内部的填充颜色。

返回值：无。

注意事项：如果只需要绘制矩形边框，不希望矩形内部被填充，则 eFillColor 参数传入 SGUI_COLOR_TRANS 即可。超出屏幕显示区域的部分将不会被绘制。

1.3.6 SGUI_Basic_DrawCircle

功能描述：在屏幕上绘制封闭圆形。

原型声明：void SGUI_Basic_DrawCircle(SGUI_UINT uiCx, SGUI_UINT uiCy, SGUI_UINT uiRadius, SGUI_COLOR eEdgeColor, SGUI_COLOR eFillColor)

参数说明：

uiCx：圆形圆心的 X 坐标。

uiCy：圆形圆心的 Y 坐标。

uiRadius：圆的半径，以像素为单位。

eEdgeColor：圆的边框的颜色。

eFillColor：圆的内部的填充颜色。

返回值：无。

注意事项：如果只需要绘制圆形边框，不希望矩形内部被填充，则 eFillColor 参数传入 SGUI_COLOR_TRANS 即可。

1.3.7 SGUI_Basic_ReverseBlockColor

功能描述：反色矩形区域。

原型声明：void SGUI_Basic_ReverseBlockColor(SGUI_UINT uiStartX, SGUI_UINT uiStartY, SGUI_UINT uiWidth, SGUI_UINT uiHeight)

参数说明：

uiStartX：矩形左上角点的 X 坐标。

uiStartY：矩形左上角点的 Y 坐标。

uiWidth：矩形的宽度，以像素为单位。

uiStartY：矩形的高度，以像素为单位。

返回值：无。

注意事项：无。

1.3.8 SGUI_Basic_DrawBitMap

功能描述：绘制位图。

原型声明：void SGUI_Basic_DrawBitMap(SGUI_RECT_AREA* pstDisplayArea, SGUI_RECT_AREA* pstDataArea, SGUI_BYTE* pDataBuffer, SGUI_DRAW_MODE eDrawMode)

参数说明：

pstDisplayArea：指定位图的显示区域。

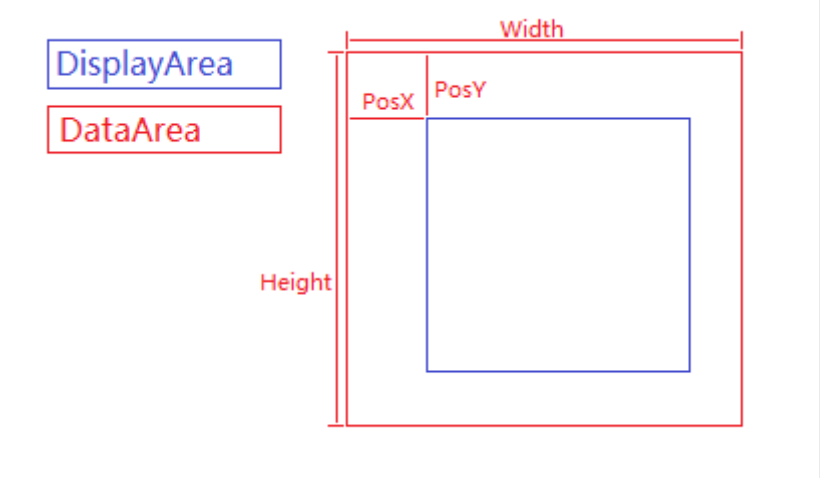
pstDataArea：位图大小以及显示的偏移量，以像素为单位。

pDataBuffer：位图数据。

eDrawMode：绘制方式（正常或反色）。

返回值：无。

注意事项：传入参数中 pstDisplayArea 指向的数据为显示位图的矩形区域，超出区域的部分将不被显示，pstDataArea 指向的数据标明了位图的实际大小以及在显示区域内的偏移量，具体逻辑关系如下图所示：



图中红色代表位图的实际尺寸，蓝色代表指定的显示区域尺寸，则实际显示时，只有蓝色区域的中的位图被显示出来。

1.4 文字显示

1.5 消息框

1.6 输入框

1.7 进度条

1.8 滚动条

1.9 列表框
1.10 实时曲线
2. HMI 引擎