Polarix

Xuyulin91@163.com

摘要

引导您搭建SimpleGUI的模拟器开发环境，帮助您能够脱离目标平台，开发通用的、可移植的人机交互接口。

SimpleGUI

模拟器环境配置

1. **环境介绍**

SimpleGUI为了方便开发者进行开发或预开发作业，特设计了PC端模拟环境用以辅助开发，模拟环境使用Codeblocks构建，图形界面基于wxWidgets建立。

1. **准备**

以下操作仅适用于WindowsXP及以上操作系统。

在搭建环境前，请先下载Codeblocks集成开发环境、wxWidgets图形库源码与编译器，三者的主页如下：

|  |  |
| --- | --- |
| Codeblocks： | http://www.codeblocks.org/ |
| wxWidgets： | http://www.wxwidgets.org/ |
| TDM-GCC： | http://tdm-gcc.tdragon.net/ |

截至此文档最后一次更新，Codeblocks最新版本为17.12，wxWidgets最新版本为开发版3.1.1，稳定版3.0.4，TDM-GCC的最新版本为5.1.0。

关于搭建SimpleGUI模拟环境，Codeblocks应使用16.01或以上版本，其他版本也可使用但操作上可能有少许出入，请自行摸索。wxWidgets请使用3.0以上版本。

如果用户从Codeblocks主页下载CodeblocksIDE，最好下载带编译器的版本，Codeblocks的最新版本17.12自带编译器版本携带的就是TDM-GCC5.1.0，如果您的电脑上已经安装了GCC编译器，请确保您的编译器是TDM-GCC4.9.1或更高版本，使用MinGW编译的wxWidgets库无法正常使用。

1. **部署CodeBlocks**

下载并解压CodeblocksIDE。

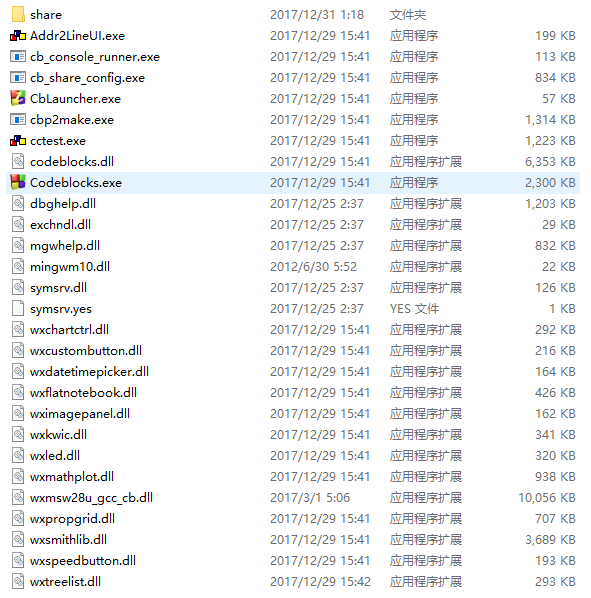


图 1 解压缩后的Codeblocks（部分）

如果是下载的带编译器版本，解压后还会有一个叫做MinGW的文件夹，这个MinGW文件夹就是TDM-GCC编译器，CodeBlocks16.01带的是4.9.2版本。17.12带的是5.1.0。

Codeblocks解压后即可运行，不需要其他的安装或部署。。

1. **部署TDM-GCC编译器**

如果下载的是纯净版不带编译器的Codeblocks，那么在部署开发环境之前，请用户自行下载好TDM-GCC5.1.0并安装，如果下载的并非安装版或直接使用Codeblocks自带的编译器，为确保后续操作能够正确执行，请按照以下操作部署TDM-GCC。

为避免编译器在部署或运行中出现一些意想不到的问题，请确保编译器被安装或解压缩在纯英文路径下。本例中，TDM-GCC被解压缩并放置在D:\Programs\GCC\TDM-GCC\_5.1.0中。

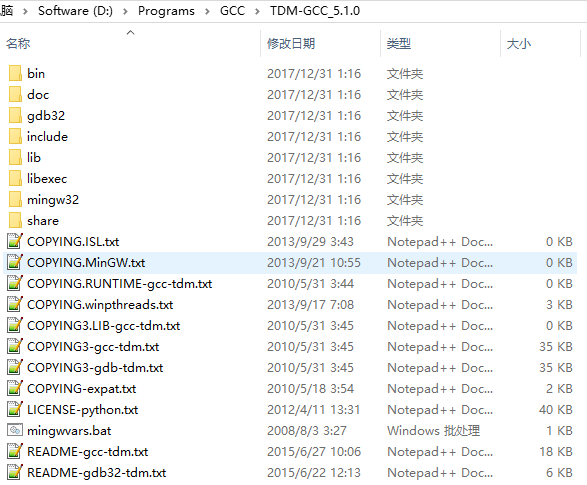


图 2 解压缩后的TDM-GCC

再次强调，为保证wxWidgets编译和使用正常，必须使用TDM-GCC4.9.1或更高版本，如果之前有安装过Mingw、Mingw-w64或其他GNU编译器的，请进行完全卸载和清理，以避免后续操作可能出现的错误。清理如果是下载的带编译器版本，解压后还会有一个叫做MinGW的文件夹，这个MinGW文件夹就是TDM-GCC编译器，CodeBlocks16.01带的是4.9.2版本。17.12带的是5.1.0。

以上操作完成后，进入系统属性设置窗口以进行环境变量的配置。不同版本的Windows进入方法稍有差别，例如WindwosXP可以在“我的电脑”上右键单击，然后在菜单中选择“属性”，Windows7中需要右键单击“计算机”，然后在显示的“系统”窗口中，单击左侧边栏中的“高级系统设置”进入系统属性设置窗口，而在Win8及以上版本中，桌面默认取消了“此电脑”图标，用户可以在文件管理器右侧的导航窗格中的“此电脑”图标上单击右键并选择“属性”菜单进入系统属性设置窗口。

此外，还可以使用一个在各个版本中共通的方式来进进入系统属性设置，按Win+R组合键，在弹出的运行对话框中输入“sysdm.cpl”并回车，系统直接打开系统属性窗口，然后选择“高级”选项卡，点击“环境变量”按钮，进入环境变量设置窗口。



图 3 系统属性窗口

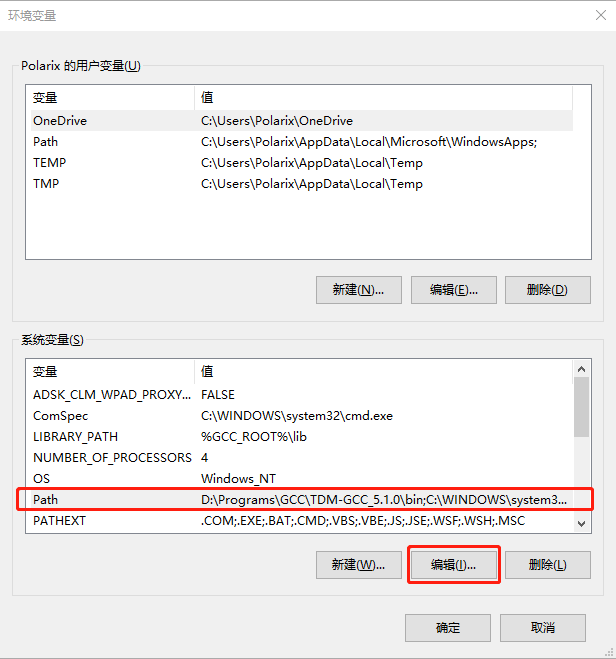


图 4 环境变量设置窗口

然后在环境变量设置窗口中选择下方“系统变量”列表中的“Path”变量，然后点击“编辑”按钮。

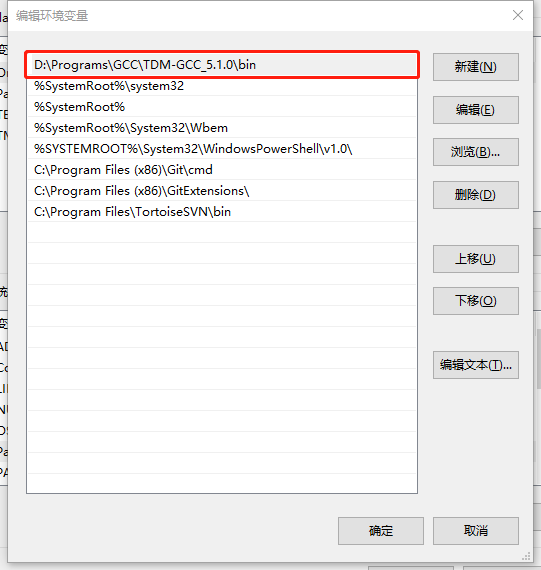


图 5 添加GCC环境变量

在此窗口中添加GCC运行用的环境变量，变量内容为GCC编译器程序的所在路径，添加完成后，点击“确定”按钮关闭环境变量编辑窗口。

此时，按Win+R键，时输入“cmd”然后回车，打开控制台，然后输入“gcc -v”控制台上将输出当前配置好的TDM-GCC的版本信息。

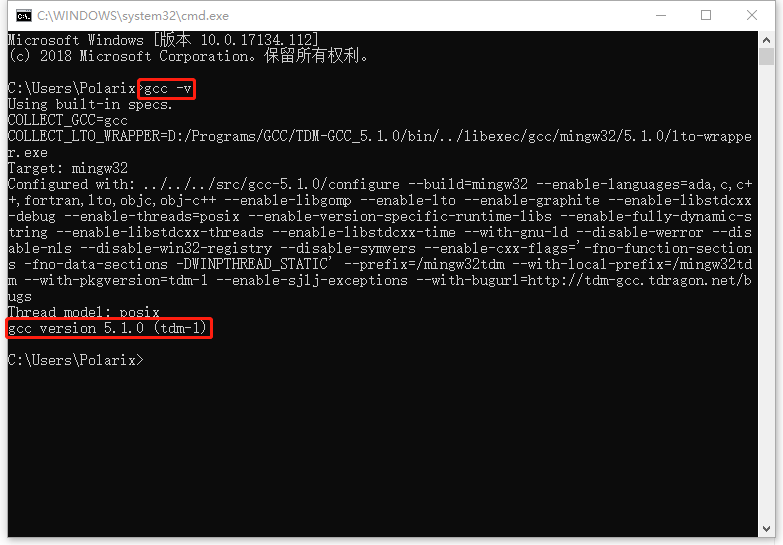


图 6 部署完成的TDM-GCC5.1.0.

如果显示的GCC版本和预计的不一致，请检查Path环境变量中保存的路径是否有其他注册的GCC编译器路径，如果有，请删除。如果使用安装的形式部署TDM-GCC，理论上安装完成后，环境变量会被自动注册，如果安装完成后，控制台中无法显示正确的GCC版本信息，请参照上述步骤重新部署TDM-GCC。

1. **部署wxWidgets库**

接下来的操作将建立在前节GCC部署成功的前提下，请确保TDM-GCC编译器已正确部署。

从wxWidgets官方网站下载wxWidgets源码并在本地解压缩。本例中解压缩至D:\wxWidgets\Development\Source文件夹下，实际操作中请以实际解压缩路径为准。为避免编译过程中出现一些意想不到的问题，请确保解压缩路径为纯英文路径。

wxWidgets解压缩后，打开控制台，进入解压缩路径。然后在进入解压缩根目录下的build\msw目录，准备编译Windows版wxWidgets库。

然后输入以下命令，开始编译wxWidgets库。

|  |
| --- |
| mingw32-make -f makefile.gcc MONOLITHIC=1 SHARED=1 UNICODE=1 BUILD=release CXXFLAGS="-fno-keep-inline-dllexport -std=gnu++11" |

当然，如果有其他的发布需求，也可以编译其他形式的wxWidgets库，具体的编译参数及含义请自行在网络上搜索。

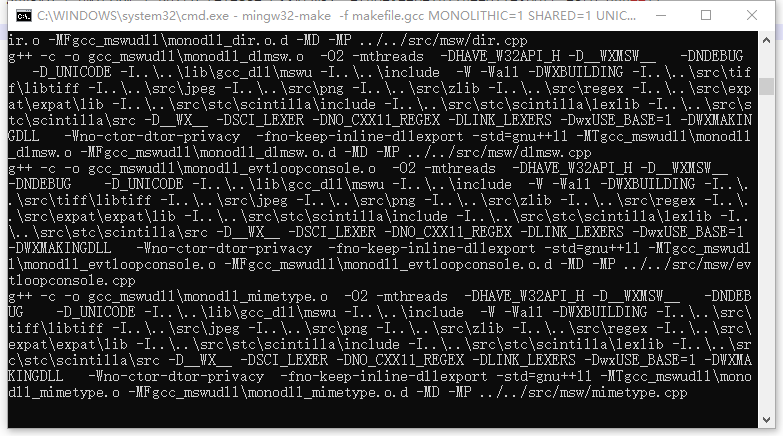


图 7 编译wxWidgets

编译全程视电脑性能而定，耗时约30分钟。编译过程中可能出现因类型转换导致的警告，可以忽略不计。

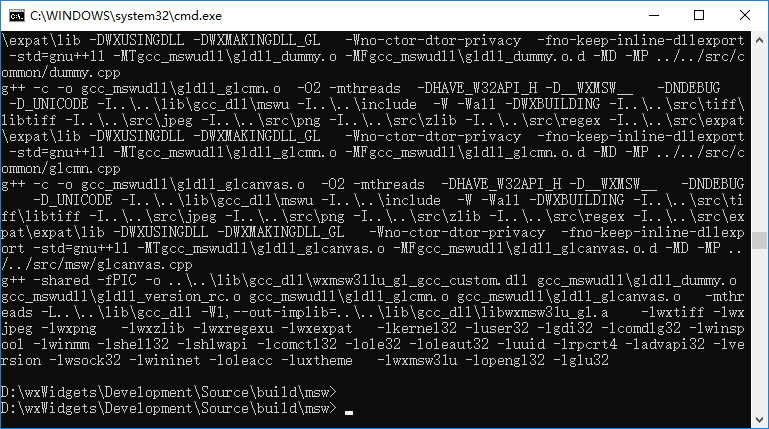


图 8 编译完成

1. **配置环境**

编译完wxWidgets环境后，启动Condeblocks。

选择“Settings”（设置）下的“Global variables”（全局变量）项目。

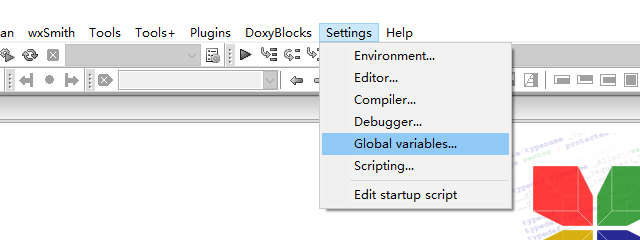


图 9 Global variables选项

在打开的全局变量编辑窗口中，点击“New”按钮，新建一个全局变量。

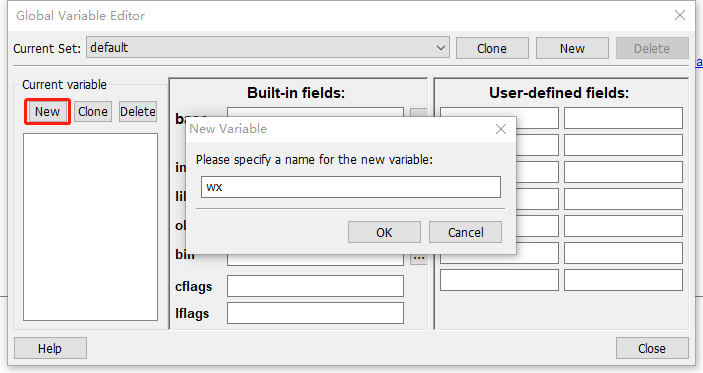


图 10 创建新的全局变量

然后将全局变量的根路径、包含路径和库路经分别设置为wxWidgets的根路径include路径和lib路径，其他留空即可，参考下图。

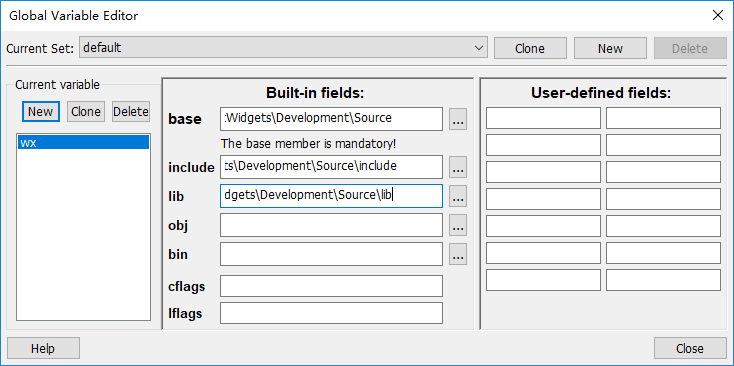


图 11 wx环境变量的设定

然后点击“Close”关闭环境变量编辑窗口，至此，SimpleGUI模拟环境开发需要使用的Codeblocks+wxWidgets环境搭建完成。

1. **编译工程**

进入SimpleGUI根目录，各子目录存放内容说明如下：

|  |  |
| --- | --- |
| DemoProc | SimpleGUI应用演示代码 |
| DemoProject | 移植演示代码，目前只配置了STM32F103正在EmBitz下的演示环境。 |
| Documents | 说明文档 |
| GUI | 绘图引擎接口实现 |
| HMI | 交互引擎接口实现 |
| SimulatorEnv | 模拟环境工程 |

进入SimulatorEnv\Project\CodeBlocks目录下，SimpleGUI.cbp文件夹即为模拟环境的工程文件，使用Codeblocks打开。如果用户下载的是nosetup（绿色版）的Codeblocks，需要手动建立关联才能直接用双击的方式打开cbp文件，否则，用户只能先打开Codeblocks，然后从Codeblocks中执行打开操作以打开工程文件。

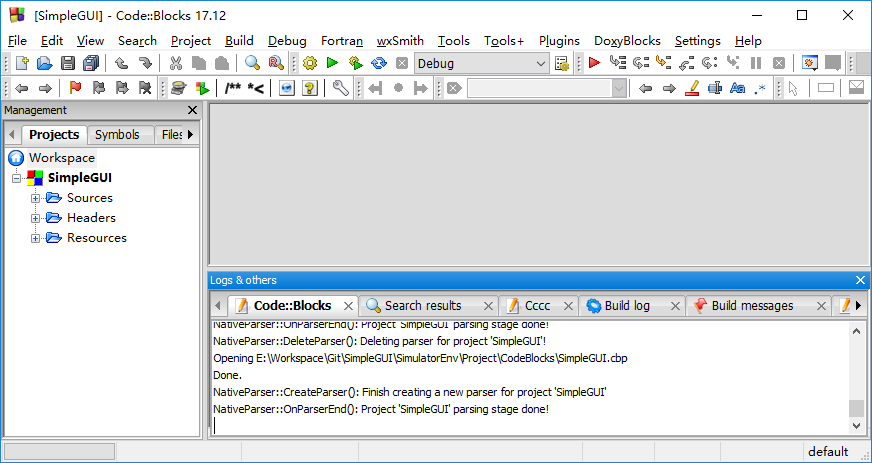


图 12 SimpleGUI模拟环境工程

点击工具栏上的编译按钮或按快捷键Ctrl+F9，开始编译模拟环境演示工程。

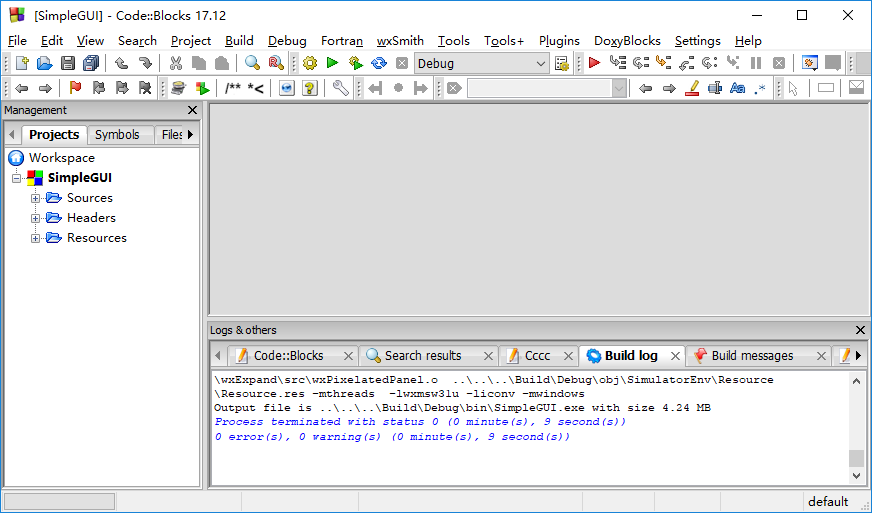


图 13 编译完成

编译完成，显示无错误和警告，现在点击工具栏上的运行按钮或按快捷键Ctrl+F10就可以看到模拟环境的运行效果了。

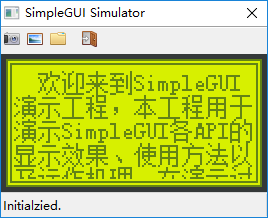


图 14 SimpleGUI模拟环境演示程序

1. **环境配置**

模拟环境中的虚拟显示屏幕的配色方案为黄底黑字的LCD点阵显示屏，为最大程度上模拟真实情况下的使用场景，模拟环境的虚拟LCD面板可以通过修改配置定义修改颜色和尺寸。

进入工程根目录，打开工程跟目录下的SimulatorEnv\Common\inc文件夹（对应工程路径/SimpleGUI/Headers/SimulatorEnv/Common/inc），打开Common.h文件，找到以下宏定义：

|  |  |
| --- | --- |
| LCD\_COLOR\_OBJ\_PIX | 逻辑有效像素点颜色（前景颜色） |
| LCD\_COLOR\_OBJ\_BKG | 逻辑无效像素点颜色（背景颜色） |
| LCD\_COLOR\_OBJ\_EDGE | 边框颜色 |
| LCD\_COLOR\_OBJ\_GRID | 网格颜色 |
| PARAM\_DEFAULT\_PIXEL\_NUM\_H | 横向像素点数量 |
| PARAM\_DEFAULT\_PIXEL\_NUM\_V | 纵向像素点数量 |
| PARAM\_DEFAULT\_PIXEL\_SIZE | 像素尺寸 |
| PARAM\_DEFAULT\_EDGE\_WIDTH | 边框宽度 |

工程默认情况下为128\*64，黄黑色屏幕，不显示像素网格。如果想改变外观，可以同通过修改宏定义的值，颜色为RGBA格式小端对其的32位数，可以通过屏幕取色工具获取RGB值。

例如，想修改成192\*64，黑底蓝字的OLED效果，可以按照如下方式修改：

1. 使用一张目标屏幕的照片，使用取色工具获取屏幕背景、像素以及边框颜色的RGB值，比如淘宝上的照片，就可以直接使用。



图 15 淘宝上的商品照片，可以直接从画红圈的位置获取需要的颜色

1. 将颜色的RGBA值分别更新入三个宏定义中，例如上图中屏幕背景色、像素色和边框色分别设定为0xFF070707、0xFFF1FA4F和0xFFC9652F，由于不需要显示网格，所以不需要修改网格颜色。



图 16 修改宏定义的颜色信息

1. 修改PARAM\_DEFAULT\_PIXEL\_NUM\_H宏定义的值为192。
2. 重新编译工程即可看到效果。

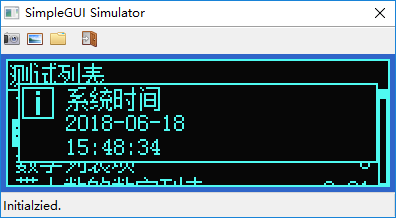


图 17 修改配色和尺寸后的效果

上述的宏用于SimulatorEnv\Common\src\Common.c（对应工程路径/SimpleGUI/Sources/SimulatorEnv/Common/src/Common.c）文件中,对默认参数设定函数SetDefaultParameterData功能能定义，此函数将在编译好的模拟器启动时对像素面板进行初始化时调用，届时此处设定的参数将会被用来设置像素面板的显示。

void SetDefaultParameterData(PixelPanelParameter\* pstParameter)

{

if(**NULL** != pstParameter)

{

// Appearance

pstParameter->HorizontalPixelNumber = PARAM\_DEFAULT\_PIXEL\_NUM\_H;

pstParameter->VerticalPixelNumber = PARAM\_DEFAULT\_PIXEL\_NUM\_V;

pstParameter->PixelSize = PARAM\_DEFAULT\_PIXEL\_SIZE;

pstParameter->EdgeWidth = PARAM\_DEFAULT\_EDGE\_WIDTH;

pstParameter->EnableGrid = PARAM\_DEFAULT\_GRID\_DISABLE;

// ScreenColor

pstParameter->PanelColor.RGBA = LCD\_COLOR\_OBJ\_BKG;

pstParameter->PixelColor.RGBA = LCD\_COLOR\_OBJ\_PIX;

pstParameter->EdgeColor.RGBA = LCD\_COLOR\_OBJ\_EDGE;

pstParameter->GridColor.RGBA = LCD\_COLOR\_OBJ\_GRID;

}

}

如果想要使能像素网格，可以将

pstParameter->EnableGrid = PARAM\_DEFAULT\_GRID\_DISABLE;

修改为

pstParameter->EnableGrid = PARAM\_DEFAULT\_GRID\_ENABLE;

重新编译后，效果如下：

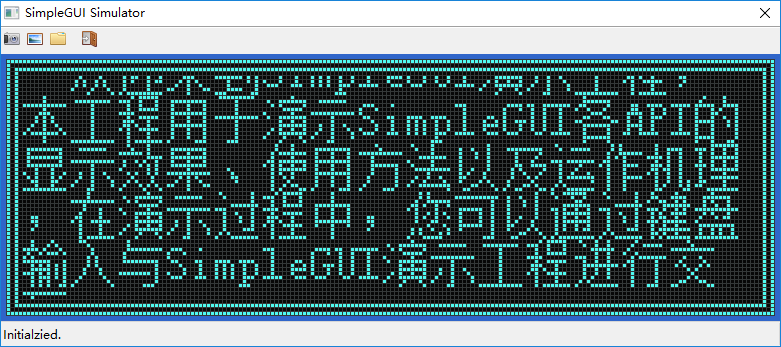


图 18 使能网格线后的效果

注意，为了避免画面显示失调，模拟器只有在像素点尺寸（也就是宏定义PARAM\_DEFAULT\_PIXEL\_SIZE）的值**大于4**时，网格设定才会有效，在像素点尺寸小于等于2时，网格线将被强制关闭。上图为像素尺寸为4时，开启预览的效果。

1. **开发接口**

模拟环境的开发用接口在工程目录下的SimulatorEnv\Interface\src文件夹下（工程路径），共有两个文件：

|  |  |
| --- | --- |
| UserActionInterface.cpp | 用户操作接口，用于环境初始化、用户按键操作做、定时器动作等。 |
| VirtualDeviceInterface.cpp | 虚拟设备接口，用于模拟器显示面板操作，通常用于更新屏幕显示，刷新屏幕、清空屏幕等。 |

以上两个文件中定义的接口函数将完成用户代码和模拟器之间的交互、这在逻辑上，相当于目标平台上的驱动程序。

UserActionInterface.cpp文件中定义了以下接口：

|  |  |
| --- | --- |
| UAIF\_OnInitialize | 模拟器程序启动时调用，通常用于初始化操作。 |
| UAIF\_OnKeyPress | 用户键盘事件响应函数。 |
| UAIF\_OnTimerEventProcess | 模拟定时器函数，每1ms触发一次。 |
| UAIF\_OnRTCUpdateEventProcess | 实时时钟函数，每秒触发一次用于更新系统时间。 |

此文件中定义的函数将被模拟环境中相应的处理函数调用，分别对应启动、用户按键（交互）、定时器三种事件，其他需求可以在定时器中模拟或按照需求增加接口。

VirtualDeviceInterface.cpp文件中定义了以下接口：

|  |  |
| --- | --- |
| VDIF\_SetPixel | 设定像素点值。 |
| VDIF\_GetPixel | 获取像素点值。 |
| VDIF\_RefreshDisplay | 更新屏幕显示，设定像素点值后，需要调用 此函数以更新屏幕显示。 |
| VDIF\_ClearDisplay | 清空屏幕显示。 |

此文件中定义的函数使用于GUI函数的Basic模块中，用于控制屏幕。

以上接口的使用和原理可以参照模拟环境工程中相应的示例代码。

1. **文件编码**

在SimpleGUI中，有关文字的处理都使用一个名为ENCODE的宏进行转码，此宏定义声明在SGUI\_Common.h中。

#if (\_SIMPLE\_GUI\_VIRTUAL\_ENVIRONMENT\_SIMULATOR\_ > 0) &&

(\_SIMPLE\_GUI\_ENABLE\_ICONV\_GB2312\_ > 0)

// Only used when running in simulated environment and text encode need convert to GB2312.

SGUI\_PSZSTR SGUI\_Common\_EncodeConvert(SGUI\_PCSZSTR szSourceEncode,

SGUI\_PSZSTR szDestinationEncode, SGUI\_PSZSTR szSource);

#define \_SIMPLE\_GUI\_ENCODE\_BUFFER\_SIZE (512)

#define ENCODE(S) (SGUI\_Common\_EncodeConvert("UTF-8", "GB2312", (char \*)S))

#else

#define ENCODE(S) (S)

#endif

为了避免在不同语言环境或不同编码下产生乱码，SimpleGUI所有的源代码文件均以UTF-8格式进行编码，但为了方便中文显示，程序运行时对中文字符的解码全部按照GB2312编码规则进行，所以需要将UTF-8字符串转码为GB2312字符串，转码函数定义在SGUI\_Common.c中，依赖iconv外部库。

在实际使用中，开发者可以根据自身系统对字符的解码算法或使用的编码集，重写SGUI\_Text.c中的SGUI\_Text\_GetCharacterTableIndex函数以适应相应的编码规则，进而实现正常的文字显示。

1. **联系开发者**

首先，感谢您对SimpleGUI的赏识与支持。

虽然最早仅仅作为一套GUI接口库使用，但我最终希望SimpleGUI能够为您提供一套完整的单色屏GUI及交互设计解决方案，如果您有新的需求、提议亦或想法，可以联系QQ 326684221或电子邮件xuyulin91@163.com，也可以在以下地址留言：

SimpleGUI@开源中国：<https://www.oschina.net/p/simplegui>

SimpleGUI@码云：<https://gitee.com/Polarix/simplegui>

本人并不是全职的开源开发者，依然有工作及家庭的琐碎事务要处理，所以对于大家的需求和疑问反馈的可能并不及时，多有怠慢，敬请谅解。

最后，再次感谢您的支持。