### 需求

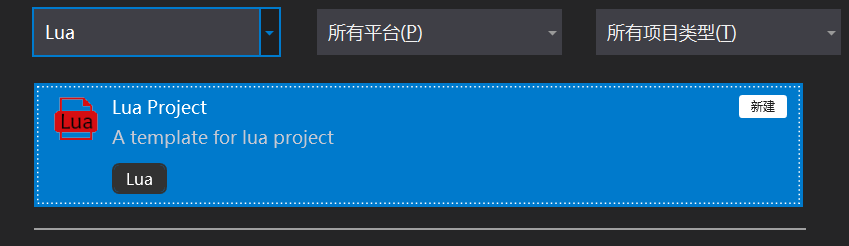
1. 插件需要Visiual Studio 2019版本（2017未经验证）
2. 插件某人配色为深色主题，首次使用请将主题改为深色。修改方法：工具->选项->环境->常规->颜色主题

### 安装

1. 将VisualStudioExtend解压到localAppdata文件夹下，该文件夹是隐藏文件夹，具体路径为c:\\ Users\\{your username}\\AppData\\Local
2. 双击vsix安装插件
3. 对于Unity，打开Untiy Editor的Package Manager界面，点击添加按钮，然后选择Add package from disk，将LuaVSHelper添加到项目中。该Package包含自动创建lua项目，与vs交互，自动定位Lua文件等功能，可以简化用户使用与调试。

### 创建工程

1. 打开Visiual Studio，创建新项目，在语言下拉框中选择Lua

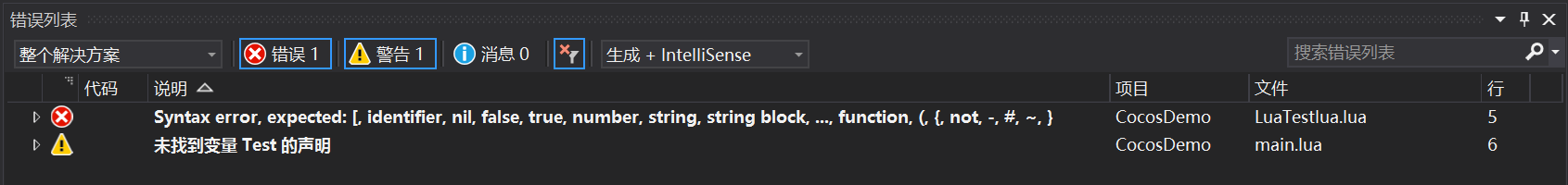


1. 选中Lua Project，点击下一步，填写工程名称，选着路径后，点击创建。

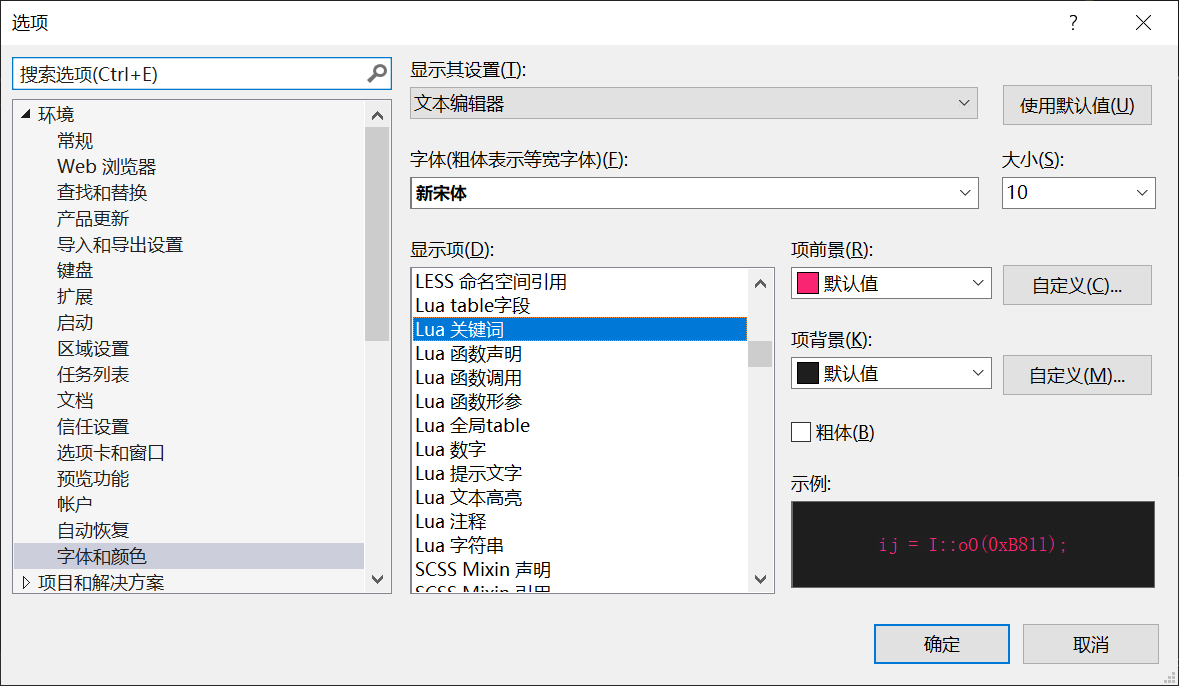


### 功能

1. 语法解析：插件在编写代码的过程中会自动解析Lua语法，对于语法错误会以红色错误的形式显示出来；同时插件也会检查变量名，对于不存在的变量，会以黄色警告的形式显示出来。对于在宿主程序中定义的变量，可以在API文件中定义，以避免产生警告，具体定义方式可参考智能提示。



1. 颜色分类器：插件默认将Lua分为关键词，函数声明，函数调用，函数形参，数字，字符串，table字段，注释，全局Table等9种颜色，默认颜色采用深色主题，如果喜欢浅色主题可自行修改颜色。修改方法：工具->选项->环境->字体和颜色，显示其设置中选择文本编辑器，在显示项中寻找Lua开头的相关项。

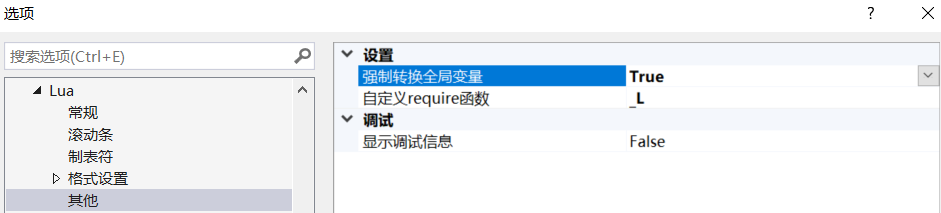


1. 智能提示
2. 所有全局变量和chuck中最后return 的变量都会注入到全局代码提示中
3. 支持自定义提示。自定义提示包含以下关键字
4. param 用于声明函数形参类型 格式--@param {#class} name {comment}，大括号表示可选，在function的声明上方添加
5. type 用于声明变量的类型 格式--@type #class {comment} 在变量上方或同行添加
6. return 用于声明函数返回值 格式 --@return {#class } {comment} ，在function的声明上方添加
7. extend 用于声明table的继承关系 格式 --@extend #class，在table的声明上方添加
8. module 用于声明一个table 格式 --@module name，仅在API文件夹下的注释文件里生效
9. function 用于声明一个函数 格式--@function name(param1,…)，仅在API文件夹下的注释文件里生效
10. field 用于声明一个table的字段 格式--@field {#class} name，仅在API文件夹下的注释文件里生效
11. overload 用于声明函数重载 格式--@overload name(param1,…)，仅在api文件夹下的注释文件里生效
12. #class表示类型，也可以自定义类型 格式为 #{x=1,y=2}。对于重名的函数，可在#和class之间声明源，源为文件相对于工程的路径，用.分割 格式#(source)class
13. 具体用法可参见示例Intellisense
14. 工程属性

工程属性是右键点击工程文件，打开属性窗口->通用属性里显示的内容。工程属性修改后，必须手动刷新或重启后才能生效。

1. 搜索路径：Lua宿主程序通常会有多级搜索路径，为了可以正确解析require方法以及调试器可以正确的添加断点，请设置合适的搜索路径，保证其优先级。
2. 加载API：插件自带的默认API提示，可以选择Cocos2dx,Unity3D和None
3. API文件夹：用于自定义的API提示，只有在API文件夹下的lua文件才会产生api解析
4. 刷新：右键点击工程文件在上下文菜单里会有刷新菜单，点击刷新菜单会自动将刷新属性配置的文件类型添加到工程中，并重新解析工程。多个文件类型使用;分隔，所有类型使用\*.\*表示，
5. 自定义require函数

在工具->选项->文本编辑器->Lua->其他 中设置自定义require函数，配合强制转换全局变量可以对一些特殊的写法产生比较好的提示。参考示例CustumRequire



1. 结构视图

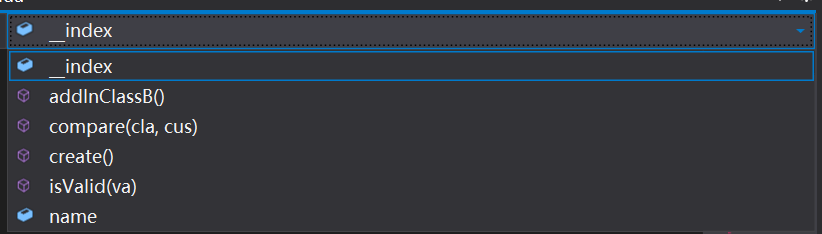
对于table，function，if，for等语句都会产生一个结构，可以点击折叠按钮将其折叠，可以使用 --@region 和--@endregion自定义结构

1. 忽略属性

文件夹可以设置为忽略，当设置为忽略后，该文件夹下的所有lua文件将不再解析。

1. 文件大纲

在编辑器界面上面有两个下拉框，左侧为该文件定义的变量，右侧为左侧选中的变量内的字段，点击下拉框可以显示所有的字段，点击选中并跳转到所在行



1. 智能缩进

当输入回车进入下一行时，智能缩进根据上下文确定下一行的缩进等级。

智能缩进的开启方式：工具->选项->文本编辑器->Lua->制表符->缩进，选择智能。

1. 格式化

代码格式化，VS默认快捷键 Ctrl+K,Ctrl+D 格式化文件，Ctrl+K,Ctrl+F 格式化选中。格式化内容时如果有语法错误格式化会失败。格式化相关设置在工具->选项->文本编辑器->Lua->格式设置，可以自定义一些格式化规则。也可以在制表符里修改制表符大小，缩进大小等内容。

1. 代码片段

默认支持的代码片段包括do,elseif,for,fori,forn,forp,function,if,repeat,while，输入上述关键字后，输入Tab即可自动生成代码片段。

1. 转到定义

按住Crtl,然后鼠标左键点击可以跳转到相关定义

1. 快速信息提示

鼠标悬浮在变量上可以查看提示

1. 参数名称提示

输入函数后，输入(和,会自动提示函数参数，如果有overload表示的方法，还会有函数重载的选择

1. 注释代码

注释代码VS默认快捷键Ctrl+K,Ctrl+C 取消注释Ctrl+K,Ctrl+U

1. 括号匹配
2. 选中高亮

### 调试

1. 调试属性
2. 调试类型：分为CreateProcess 和AttachProcess两种。
3. 命令：对于CreateProcess为启动程序的具体路径；对于AttachProcess为附加的进程名，如Unity.exe
4. 命令参数：对于CreateProcess为启动程序的参数；对于AttachProcess没有任何意义
5. 工作目录：对于CreateProcess为启动程序的工作目录，对于AttachProcess没有任何意义
6. Lua错误监视：开启该功能后，调试器会重写pcall和xpcall两个函数，以便监听所有的lua执行错误，当发生错误后，会自动中断程序执行，以便查看错误信息。对于AttachProcess类型，当停止调试后，会产生一个隐藏问题，如果pcall和xpcall函数被保存到了其他变量里，再次执行时就会因无法访问的内存导致进程崩溃。

为了避免崩溃可以采用以下方法：

1. 避免将pcall和xpcall保存到其他变量中
2. 在停止调试前，先停止Lua运行
3. 将Lua错误监视 关闭
4. 自定义生成步骤：在调试器启动之前执行的命令，目前仅支持xcopy
5. xcopy：格式 xcopy sourcePath destPath，将sourcePath的文件或者文件夹拷贝到destPath下
6. 支持立即窗口 监视窗口 自动窗口 悬浮提示 及局部变量窗口
7. 支持条件断点（条件及命中次数），支持操作。
8. 远程调试
   1. 打开工程文件属性窗口，选择配置属性->调试，要启动的调试器选择Remote Debugger。
   2. 输入要调试的ip地址和端口
   3. 将LuaRemoteDebug放入lua代码中，在入口文件处，require该文件
   4. 注意LuaRemoteDebug文件里的DebugPort要与调试属性窗口的端口号保持一致