1. **引言**
   1. **编写目的**

本文主要对中文脚脚本编写系统——Macro进行介绍。

* 1. **系统功能**

用户可以使用中文作为脚本语言，书写自己想要的脚本。用户可以选择多种输入方式——录屏、语音输入、文字输入创建自己的脚本。系统利用简洁的语法提供市面上大部分脚本编写系统不具备的OCR功能，使得系统可以编写出功能更强的脚本。执行脚本能使计算机自动执行一定的操作，减少用户的重复的机械劳动。系统还提供两种界面样式，并提供对脚本的管理功能，用户可以对自己的脚本进行修改或删除。

* 1. **参考资料及说明**

[1] [设计模式 | 菜鸟教程 (runoob.com)](https://www.runoob.com/design-pattern/design-pattern-tutorial.html) 设计模式的学习资料

[2] [类图 - 百度文库 (baidu.com)](https://wenku.baidu.com/view/286bce04ec3a87c24028c4ed.html) 用于学习VOPC图的设计规范。

[3]Martin Fowler, Kendall Scott.UML Distilled A Brief Guide to the Standard Object Modeling Language(第二版)[M]. Addison Wesley.2003-9-15

[4] [taojy123/KeymouseGo: 类似按键精灵的鼠标键盘录制和自动化操作 模拟点击和键入 | automate mouse clicks and keyboard input (github.com)](https://github.com/taojy123/KeymouseGo) 用于学习python对window API的调用

[5] [Monaco Editor API (microsoft.github.io)](https://microsoft.github.io/monaco-editor/api/index.html) 有关编辑器API的参考文档

[6] [rishabhjaiswal3/SmartTextEditor: File Saving in Smart way and get Any time (github.com)](https://github.com/rishabhjaiswal3/SmartTextEditor) 用于学习编辑器开发

[7] [Python 基础教程 | 菜鸟教程 (runoob.com)](https://www.runoob.com/python/python-tutorial.html) Python基础语法学习

[8] [Python语言程序设计\_北京理工大学\_中国大学MOOC(慕课) (icourse163.org)](https://www.icourse163.org/course/BIT-268001) Python基础语法的学习

1. **总体设计**
   1. **需求规定**
2. 用户管理文件
3. 引言

本功能实现对用户已创建的脚本文件的管理，主要包括四个按钮，功能如下所示：

|  |  |  |
| --- | --- | --- |
| 名称 | 功能 | 描述 |
| 执行按钮 | 执行脚本 | 点击后系统执行合法脚本 |
| 修改按钮 | 修改脚本内容 | 点击后系统跳转到编辑界面以修改脚本 |
| 删除按钮 | 删除脚本 | 点击后系统按文件路径查找到对应文件并删除 |
| 搜索按钮 | 搜索脚本 | 点击后系统按文件名筛选出所有对应脚本 |

1. 输入

用户点击对应按钮

1. 加工

系统检查点击是否合法

1. 输出

系统根据按钮功能作出与上表描述中相应的操作

1. 用户编辑脚本
2. 引言

本功能实现对用户已创建的脚本文件的管理，主要包括多个按钮，功能如下所示：

|  |  |  |
| --- | --- | --- |
| 名称 | 功能 | 描述 |
| 新建按钮 | 新建文件 | 点击后系新建脚本 |
| 保存按钮 | 保存当前脚本 | 点击后系统将脚本保存到默认路径 |
| 另存为按钮 | 另存为当前脚本 | 点击后系统将脚本保存到指定路径 |
| 执行按钮 | 运行脚本 | 点击后系统执行脚本 |
| 检查按钮 | 编译脚本 | 点击后系统检查脚本内容是否合法 |
| 格式化按钮 | 格式化当前脚本 | 点击后系统将编辑器内的文本格式化 |
| 录屏按钮 | 进行录屏 | 点击后系统开始录屏，并生成脚本，再次点击按钮或按Esc停止录制 |
| 录音按钮 | 进行录音 | 点击后系统开始15秒的录音，再次点击按钮停止录音，语音识别出脚本内容 |
| 地图按钮 | 打开地图界面 | 点击后系统截图并打开图片，在图片上点击会显示点击位置的屏幕坐标 |
| 图片夹按钮 | 导入图片 | 点击后系统导入指定图片 |
| 打开按钮 | 打开脚本 | 点击后系统打开指定路径上的脚本文件 |

1. 输入

用户点击对应按钮

1. 加工

系统检查点击是否合法

1. 输出

系统根据按钮功能作出与上表描述中相应的操作

1. 应用设置
2. 引言

本功能允许用户对应用进行设置，包括录屏鼠标记录精度、录屏延迟记录精度、图像识别精度和界面样式。

1. 输入

用户根据需要选择不同设置

1. 加工

系统检查对精度的设置修改是否合法

1. 输出

系统根据用户需求修改相应设置

* 1. **运行环境**

系统环境：win10操作系统

* 1. **处理流程**

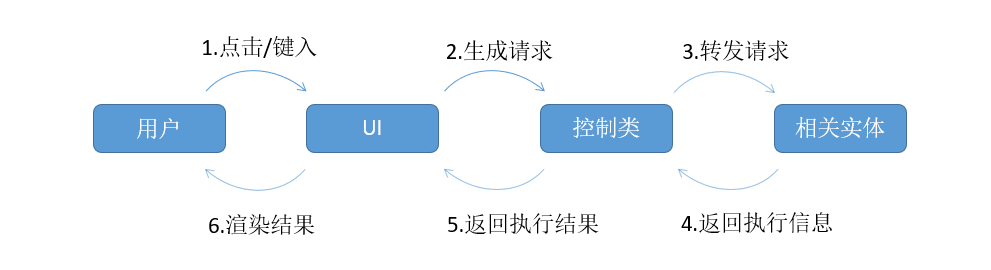


图 2-1 数据流图

* 1. **软件架构**

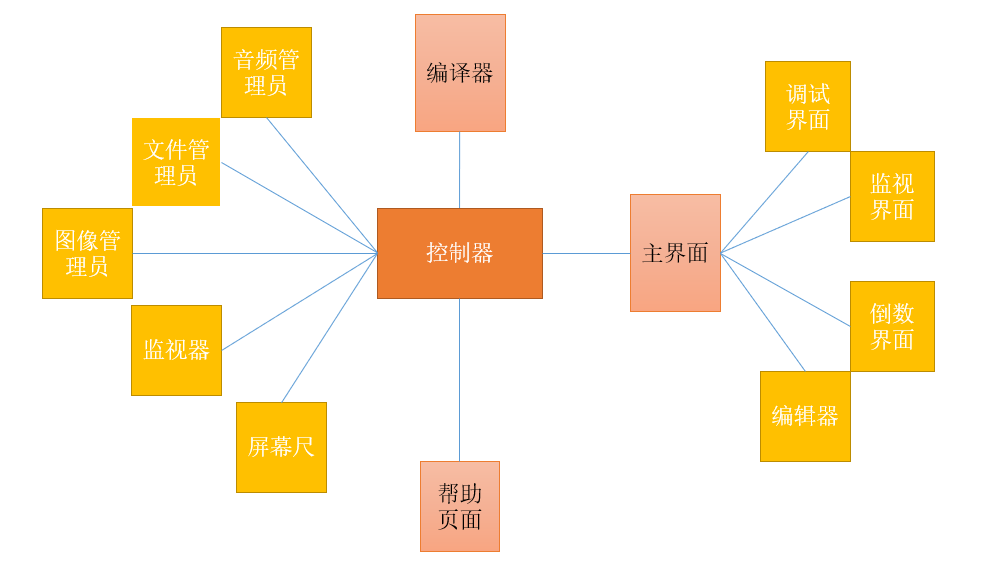
****

图 2-2 系统总体架构的概要设计图

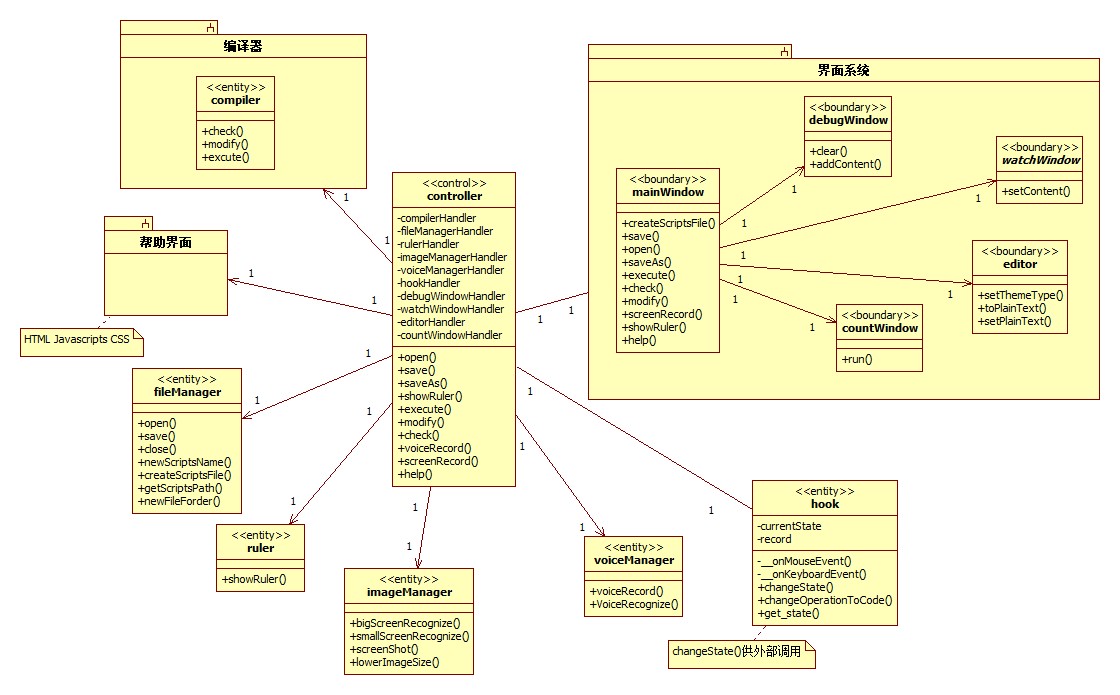


图 2-3 系统总体架构的VOPC图

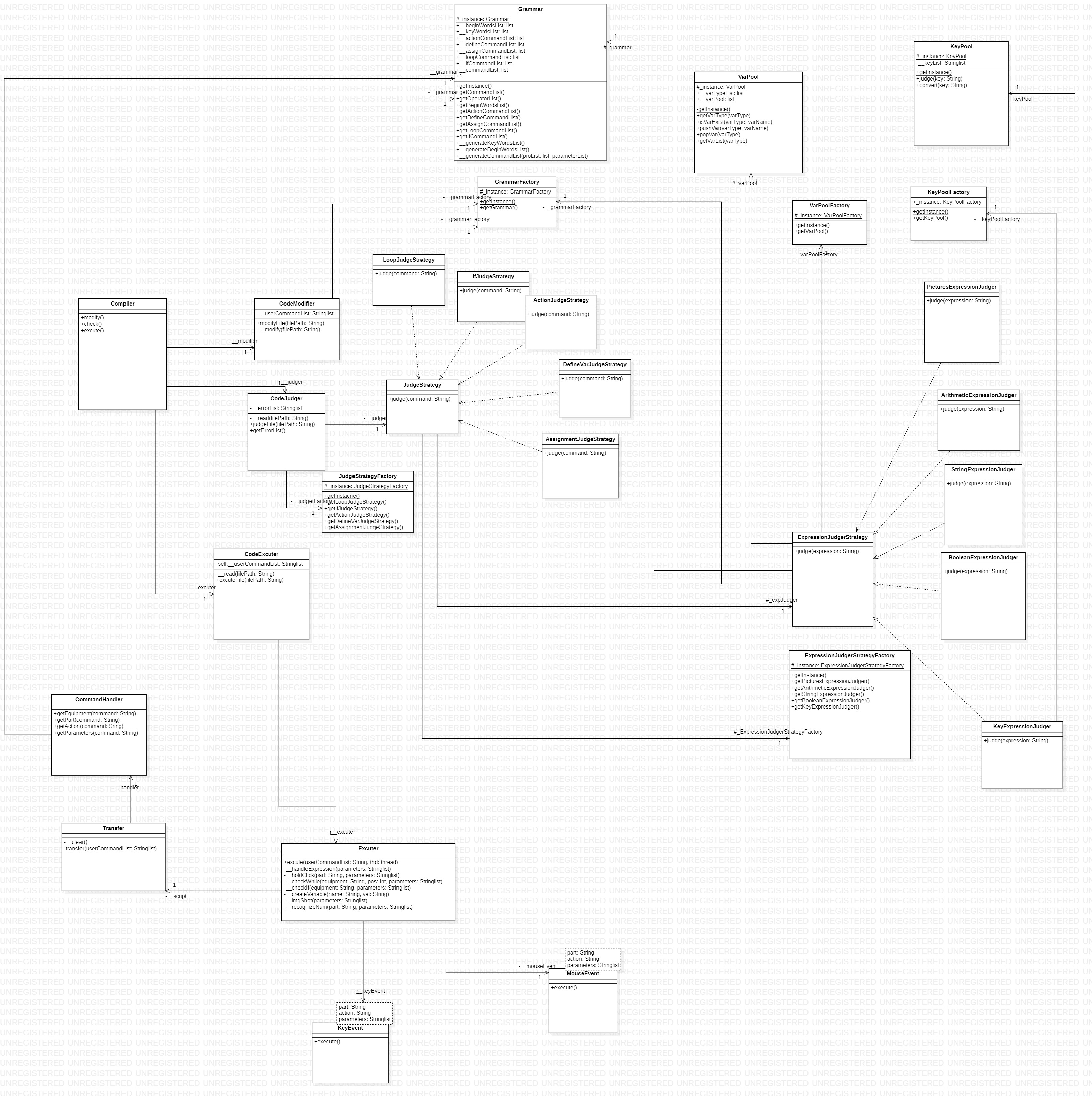


图 2-4 系统的编译器模块的类图

1. **运行设计**

|  |  |  |
| --- | --- | --- |
| 功能 | 参与者 | 功能实现逻辑简述 |
| 检索 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，向控制类请求检索功能 2. 控制类向文件管理员请求检索功能 3. 文件管理员获得目标路径下的所有文件，并利用正则匹配，匹配出所有合适的文件，回传给控制类 4. 控制类向界面回传完成消息 5. 界面依照回传信息渲染界面 |
| 删除 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，向控制类请求删除文件 2. 控制类向文件管理员请求删除文件 3. 文件管理员检测文件是否可删除（是否存在，是否已打开等）。如果可删除则进行删除操作。 4. 控制类向界面回传完成消息 5. 界面重新渲染 |
| 打开 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，向控制类请求打开文件 2. 控制类向文件管理员请求打开文件 3. 文件管理员打开文件，并向控制类回传文件内容 4. 控制类向界面回传文件内容 5. 界面重新渲染，将文件内容写入编辑器 |
| 新建 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，向控制类请求新建文件 2. 控制类向文件管理员请求新建文件 3. 文件管理员新建文件 4. 控制类向文件管理员请求打开新建文件 5. 文件管理员打开新建文件（详细步骤参考打开功能） |
| 保存 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，获取到编辑器内容 2. 界面向控制类请求保存内容 3. 控制类向文件管理员请求保存内容 4. 文件管理员保存内容进文件 |
| 另存为 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，获取到编辑器内容 2. 界面向控制类请求另存为内容 3. 控制类向文件管理员请求另存为内容 4. 文件管理员弹出文件对话框，获取用户想要保存的路径 5. 文件管理员将内容保存到目标路径中 |
| 图片夹 | 界面+控制类+文件管理员 | 1. 界面收到请求信息，向控制类请求打开图片夹 2. 控制类向文件管理员请求打开图片夹 3. 文件管理员打开图片夹 |
| 检查 | 界面+控制类+编译器 | 1. 界面收到请求信息，获取到编辑器内容 2. 界面向控制类请求检测内容 3. 控制类请求编译器检测内容 4. 编译器检测内容，并向控制类回传检测信息 5. 控制类向界面回传检测信息 6. 界面依据检测信息渲染界面 |
| 格式化 | 界面+控制类+编译器+文件管理员 | 1. 界面收到请求消息，获取到编辑器内容 2. 界面向控制类请求格式化内容 3. 控制类向编译器请求格式化内容 4. 编译器格式化内容，并保存在格式化文件中 5. 控制类向文件管理者请求打开格式化文件 6. 文件管理员打开格式化文件（详细步骤参考打开功能） |
| 执行 | 界面+控制类+编译器 | 1. 界面收到请求消息，获取编辑器内容 2. 界面向控制类请求执行内容 3. 控制类向编译器请求执行内容 4. 编译器执行内容 |
| 录音 | 界面+控制类+音频管理员 | 1. 界面收到请求消息，向控制类请求录音功能 2. 控制类向音频管理员请求录音功能 3. 音频管理员录音 4. 音频管理员进行语音识别，并向控制类回传识别内容 5. 控制类向界面回传识别内容 6. 界面重新渲染，将识别内容写入编辑器 |
| 地图 | 界面+控制类+图像管理员+屏幕尺 | 1. 界面收到请求消息，向控制类请求地图功能 2. 控制类向图像管理员请求截图 3. 图像管理员截图 4. 控制类向屏幕尺请求显示截图 5. 屏幕尺创建新窗口来显示截图。用户可在新窗口完成测量工作 |
| 录屏 | 界面+控制类+监视器 | 1. 界面收到请求消息，向控制类请求录屏 2. 控制类向监视器请求录屏 3. 监视器开始录屏 4. 界面收到终止消息，向控制类请求终止录屏 5. 控制类向监视器请求终止录屏 6. 监视器停止录屏 7. 监视器将内容转化为合法代码，回传给控制类 8. 控制类将合法代码回传给界面 9. 界面重新渲染，将合法代码写入编辑器 |
| 参数修改 | 界面 | 1. 界面收到修改参数信息，获取修改后的内容 2. 界面判读修改后的内容是否合法来决定是否修改参数 3. 界面重新渲染 |
| 界面样式修改 | 界面 | 1. 界面收到修改界面样式消息，界面获取用户想要的界面样式 2. 界面重新渲染 |

1. **系统出错处理**
   1. **未找到目标文件**

当系统未找到目标文件时，会依据情况选择新建文件或者在debug窗口中报错，报错的内容有“未找到目标图片”，“未找到目标文件”。

当出现这样的报错信息时，用户需要确认所给的文件名是否正确，所要打开的文件是否已被删除。

* 1. **语音识别错误**

当系统语音识别错误时，会在debug窗口中报错，报错的内容为“语音识别错误，错误类型为：……”。

当出现这样的报错信息时，用户需要检测网络连接和麦克风的可用性。

**4.3 编译器检查错误**

当系统检测编辑器内容时，如果内容存在不合语法的地方，会在debug窗口中报错，报错的内容有“第x行，表达式的括号存在错误”，“第x行，表达式运算符错误”，“第x行，表达式变量错误“，“图片变量错误”，“参数数量错误”，“表达式出现错误”，“变量不存在”，“变量名字和已有变量重复”，“变量名字和已有关键词重复” ，“没有与if语句对应的else”等。

当出现这样的报错信息时，说明当前脚本的对应行数存在对应的语法错误，用户应根据提示检查相关语句的错误。

* 1. **系统意外关闭**

系统会周期性保存脚本文件，意外关闭时，重启应用可以恢复到上次自动保存的状态。

1. **模块设计说明**
   1. **中文编译器**
2. **词法分析模块：**

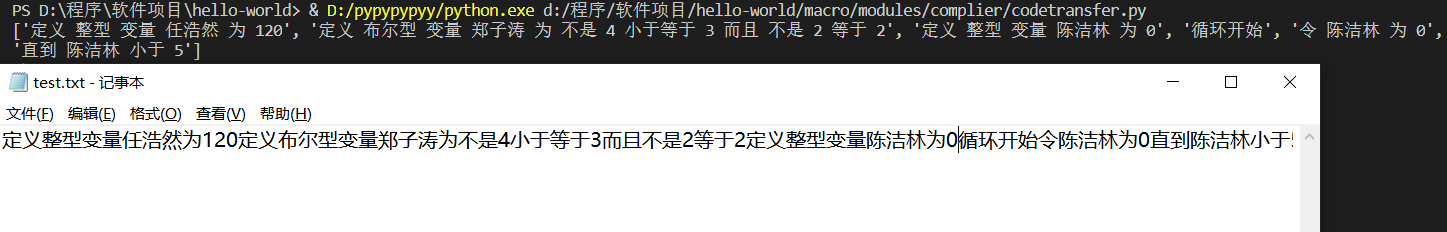
编译器内置了能够识别的词语列表，当用户通过麦克风或者键盘输入一串命令后，编译器调用modify方法，首先会将命令分行，把各行命令中的关键词分开，转化为标准格式。

图 5-1 将用户连续的输入转化为编译器能识别的形式

1. **语法分析模块**

该模块负责判断用户输入的命令是否合法。

**2.1、表达式判别模块**

**2.1.1、算术表达式判别模块**

当传入一个字符串后，该模块负责判断该字符串是不是合法的算术表达式。以空格为间隔将该字符串分割，分割的时候保留其顺序，分割完成后，第奇数个小字符串为数据，第偶数个小字符串为运算符。特别的，左括号和右括号不影响其他数据的奇偶性。

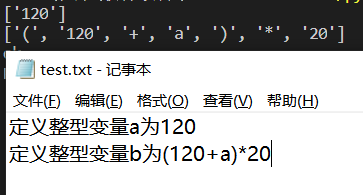


图 5-2 转化后的算术表达式

对于数据，我们要判断其是否为数字或者是否为已申请的变量。对于运算符，我们要判断该运算符是否合法。对于括号，首先我们要判断其是否出现在合法的位置，即在左括号出现在数据前，运算符后；而右括号出现在数据后，运算符前。接着判断左括号是否和右括号匹配。

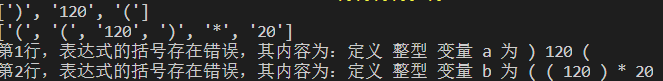


图 5-3 非法使用括号的判断

**2.1.1、布尔表达式判别模块**

当传入一个字符串后，该模块负责判断该字符串是不是合法的布尔表达式。以空格为间隔将该字符串分割，分割的时候保留其顺序，分割完成后，第奇数个小字符串为数据，第偶数个小字符串为运算符。特别的，左括号和右括号以及非运算不影响其他数据的奇偶性。

对于数据，我们要判断其是否为数字或者是否为已申请的变量。对于运算符，我们要判断该运算符是否为合法的算术运算符或者布尔运算符。对于括号，首先我们要判断其是否出现在合法的位置，即在左括号出现在数据前，运算符后；而右括号出现在数据后，运算符前。接着判断左括号是否和右括号匹配。

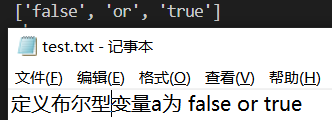


图 5-4 布尔表达式的判断

**2.2命令识别模块**

**2.2.1无参数命令识别模块**

当传入一个字符串后，该模块负责识别该命令是否为能识别的无参数命令。

**2.2.2单参数命令识别模块**

当传入一个字符串后，该模块负责识别该命令是否为能识别的单参数命令。若为单参数命令则依据该参数的类型判断该参数是否合法。如”延迟“命令，需要判断延迟后面的参数是否为合法的算术表达式。

**2.2.3双参数命令识别模块**

当传入一个字符串后，该模块负责识别该命令是否为能识别的双参数命令。若为双参数命令则依据该参数的类型判断该参数是否合法。如”定义 变量“命令，需要判断变量的类型是否合法且变量的名字是否合法。

**2.2.4三参数命令识别模块**

当传入一个字符串后，该模块负责识别该命令是否为能识别的三参数命令。若为三参数命令则依据该参数的类型判断该参数是否合法。如”定义 变量 为“命令，需要判断变量的类型是否合法且变量的名字是否合法，且“为”后面是否是合法的表达式。

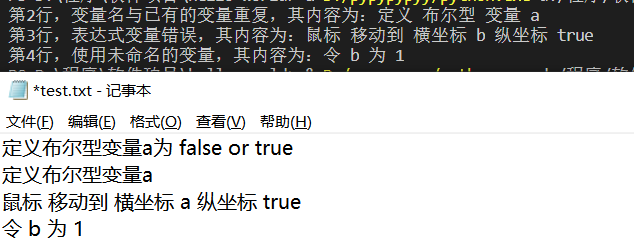


图 5-5 各类命令的判断

**2.3 if，循环语句的判断模块**

**2.3.1 if语句的判断模块**

当传入一个字符串后，若为if语句的起始语句，则if层数计数器+1，若为if语句的结束语句，则if层数计数器-1。在一层if中，只能允许先出现一次“那么”（then），再出现一次“否则”(else)。

当退出该if层，则利用栈清空在这if层申请的变量。

**2.3.2循环语句的判断模块**

当传入一个字符串后，若为循环语句的起始语句，则循环层数计数器+1，若为循环语句的结束语句，则循环层数计数器-1。当退出该循环层，则利用栈清空在这循环层申请的变量。

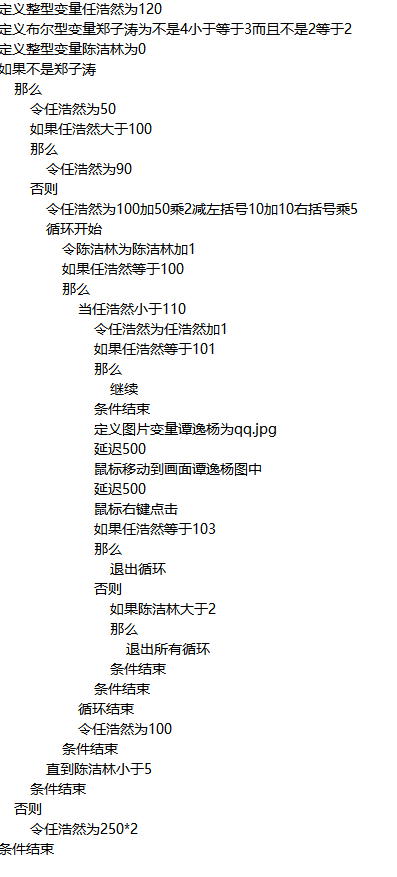


图 5-6 循环和if语句嵌套

**3.传递关键参数模块**

该模块负责将正确的命令分解为特定的几个参数，传入执行层，让编译器正确的执行命令。本模块将传递4个列表给执行层，前三个列表分别存储着每行命令的类型、命令操作对象、命令的行为，而后一个列表则是一个列表型列表，存储着每行命令所需要的参数，如各类表达式。

**3.1命令识别模块**

该模块负责识别命令，借助正则表达式来实现匹配，然后正确地将命令的类型、命令操作对象、命令的行为和参数存储在正确的列表中。

**3.2 表达式分解模块**

该模块负责将表达式分解，然后正确地存储到参数列表中。



图 5-7 将命令转化为4个列表

**4.语义分析执行模块**

该模块负责对上一个模块传递过来的参数进行分析并执行。上一模块传递过来的参数在这一模块中被看作一条条指令，每条指令都有特定的功能以及相应的格式。在本模块中，我们做的就是分析这些指令的功能以及指令的执行顺序，最后执行所有的指令。由于我们做的是一种较为简单的脚本语言，因此中间代码生成以及目标代码生成模块我们进行了省略和合并，在语义分析完毕后，用自研的方法，通过python语言执行所有的指令。

**4.1变量**

**4.1.1变量类型**

在本语言中，用户可以声明以及使用变量。变量目前分为4种类型：整型、布尔型、字符串、图片。前三种类型与一般的计算机高级语言没什么区别，不同的是第四种图片类型。图片类型是一种特殊的字符串，它的值是图片在计算机中存储的绝对路径。

所有的图片都保存在指定文件夹中，用户在声明图片变量时，只需要输入图片名称，本模块会通过路径查找算法，定位到图片所在的绝对路径，并将路径保存在图片变量中。调用该图片变量时，本模块会通过图片的绝对路径对图片进行访问。

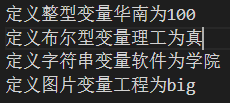


图 5-8 声明四种类型变量

**4.1.2变量存储方式**

用户声明的每一个变量都会以二元列表的方式存储在一个字典中。其中变量名作为字典的索引，二元列表的第一个参数存储的是变量的类型名称，二元列表的第二个参数存储的是变量的值。

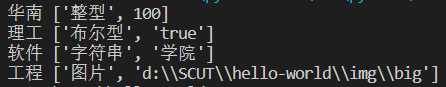


图 5-9 变量的存储形式

**4.1.3变量作用域**

为了方便没有接触过编程的用户理解和使用我们的脚本语言，本语言中变量一旦被声明，便可以在全局范围中使用。

**4.1.4变量的运算**

本语言设置的运算符有优先级，因此表达式不能从左往右直接运算。本模块在接收到用户输入的中缀表达式后，通过中缀转后缀表达式算法，将中缀表达式转化为后缀表达式，最后再对该后缀表达式进行求解。

**4.2条件语句**

对于条件语句的处理，本模块采用了自研的条件指令处理算法。该算法需要预先创建两个布尔类型的栈，一个名为if\_flag，一个名为if\_state。If\_flag存放当前执行到的指令所在的条件嵌套中的所有条件的布尔值。If\_state存放当前执行的指令所在的条件嵌套中的位置。每执行一条指令时，都会调用该条件指令处理算法。算法描述如下：

（1）若当前执行的指令为“如果”指令：如果条件表达式为真，那么将true值压入if\_flag中，否则将false压入if\_flag中。

（2）若当前执行的指令为“那么”指令：将true值压入if\_state中。

（3）若当前执行的指令为“否则”指令：弹出if\_flag栈顶元素，然后将false值压入if\_flag中。

（4）若当前执行的指令为“条件结束”指令：弹出if\_flag和if\_state的栈顶元素。

（5）若当前执行的指令均不是（1）~（4）的指令，那么判断if\_flag的值是否完全等价与if\_state的值。若完全等价，则执行该指令；否则，跳过该指令的执行。

算法基本思想：在一个嵌套的条件语句中，对于某个子条件语句，如果条件判断表达式的值为真，就会执行“那么”后面的语句；为假，就会执行“否则”后面的语句。因此，if\_flag存放所有条件判断表达式的值的状态，而if\_state存放当前执行到的指令位于条件嵌套的位置的状态。如果两个状态相同，则证明当前指令所在位置是正确的，即可以执行该条指令。

**4.3循环语句**

对于循环语句的处理，本模块采用了自研的循环指令处理算法。该算法需要预先创建两个布尔类型的栈，一个名为while\_pos，一个名为while\_state。while\_state存放当前执行到的指令所在的循环嵌套中的所有循环条件的布尔值。while\_pos存放当前执行的指令的序号。并且创建一个continue\_flag变量，用于存放当前循环是否被continue。每执行一条指令时，都会调用该循环指令处理算法。以“当……循环结束”为例，算法描述如下：

（1）若当前执行的指令为“当”指令：如果循环判断表达式为真，那么将true值压入while\_state中，否则将false压入while\_state中。然后将当前指令序号压入while\_pos中。

（2）若当前执行的指令为“循环结束”指令：如果while\_state中存在false元素，那么将while\_state和while\_pos栈顶元素弹出；否则将下一条要执行的指令序号标记为while\_pos的栈顶元素的值。然后将continue\_flag标记为false。

（3）若当前执行的指令为“退出循环”指令：将while\_flag栈顶元素标记为false。

（4）若当前执行的指令为“继续”指令：将continue\_flag标记为true。

（5）若当前执行的指令为“退出所有循环”指令：将while\_flag所有元素标记为false。

（6）若当前执行的指令是（1）（3）（4）（5）之外的指令：如果continue\_flag为true或者while\_state中存在false值，那么跳过该指令的执行；否则执行该指令。

算法基本思想：在一个嵌套的循环语句中，对于某个子循环语句，当循环判断表达式的值为真，就会循环执行在“当”和“循环结束”的语句；否则就会跳过这部分的语句。因此，我们只需要判断当前指令所在的循环嵌套中是否所有的循环判断表达式都为真。如果都为真，则说明该语句应该被执行。

为了方便用户编写代码，我们添加了“退出所有循环”关键字，作用是可以在嵌套循环内部一次性退出所有的循环。

**4.4键盘与鼠标的控制**

对于键盘与鼠标的控制，我们运用了多线程的技术。比如“按下鼠标左键持续5秒“的指令，执行这条指令时会创建一个子线程计时器，初始时处于睡眠状态。5秒后该子线程被唤醒，自动执行鼠标左键松开指令。通过该方法，可以实现鼠标键盘同步操作。

**5.2图像识别**

根据软件设计需要，我们制定了两种不同的语法。为了实现两种语法的功能，制作了两种不同的图像识别。

**1.在一个大图中寻找某个特定的小图，返回图片中心坐标**

小图需要用户传入，一般为屏幕上某个位置的截图。大图默认是用户的实时屏幕截图，不需要用户传入，不过也提供相关语法允许用户自己传入大图进行识别。用户可以设置识别精度参数，满足不同用户对精度的需求。

实现方式为：先将两张图片转化为灰度图像，利用numpy进行像素识别。附上为该模块的测试时的截图。

传入小图，大图不传入，使用默认的实时屏幕截图。绿色方框框住的地方为识别出的图像位置。

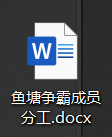


图 5-10 传入的小图

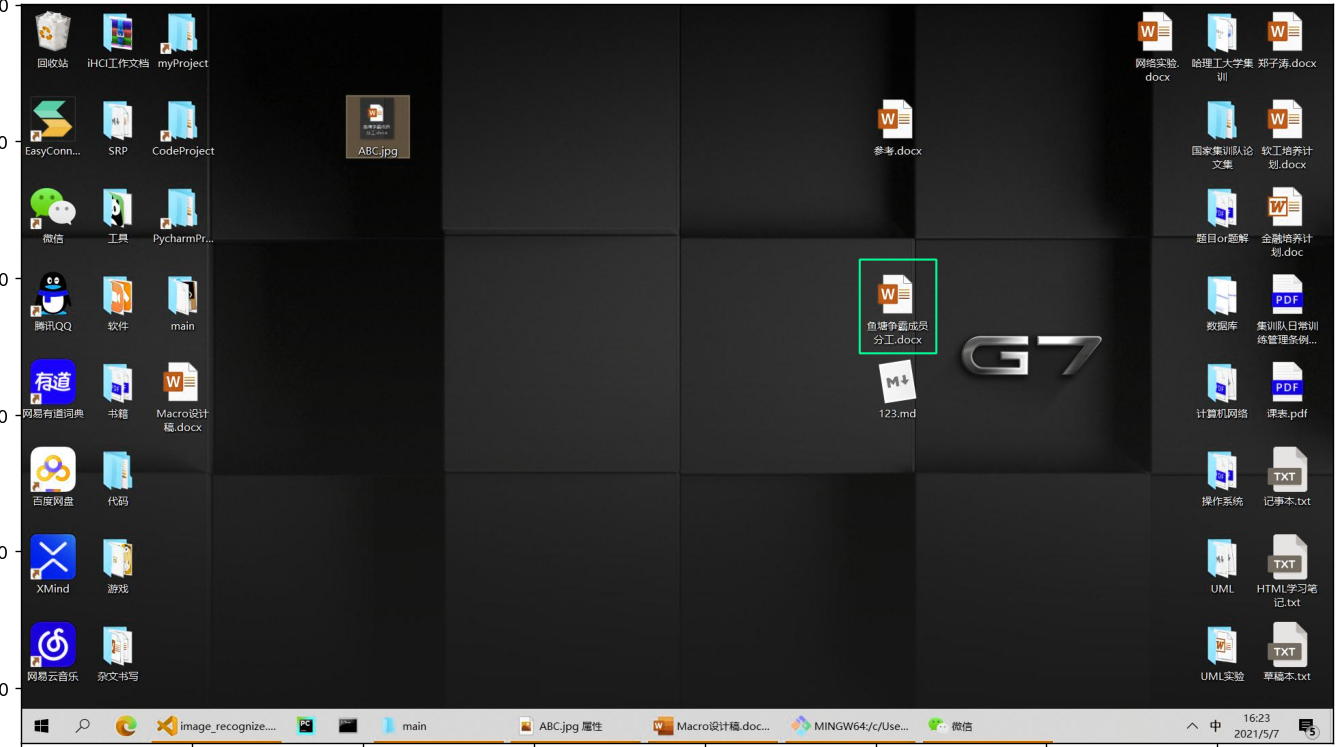
****

图 5-11 图像识别结果

**2.识别图片里的特定文字，返回文字中心坐标**

图片默认是用户的实时屏幕截图，不需要用户传入，不过也运行用户自己传入大图进行识别。用户传入需要寻找的文字，将会得到这个文字的坐标。

该技术采用百度的OCR技术，该技术已经成熟，可以精准地识别图像的文字。附上为该模块的测试时的截图。

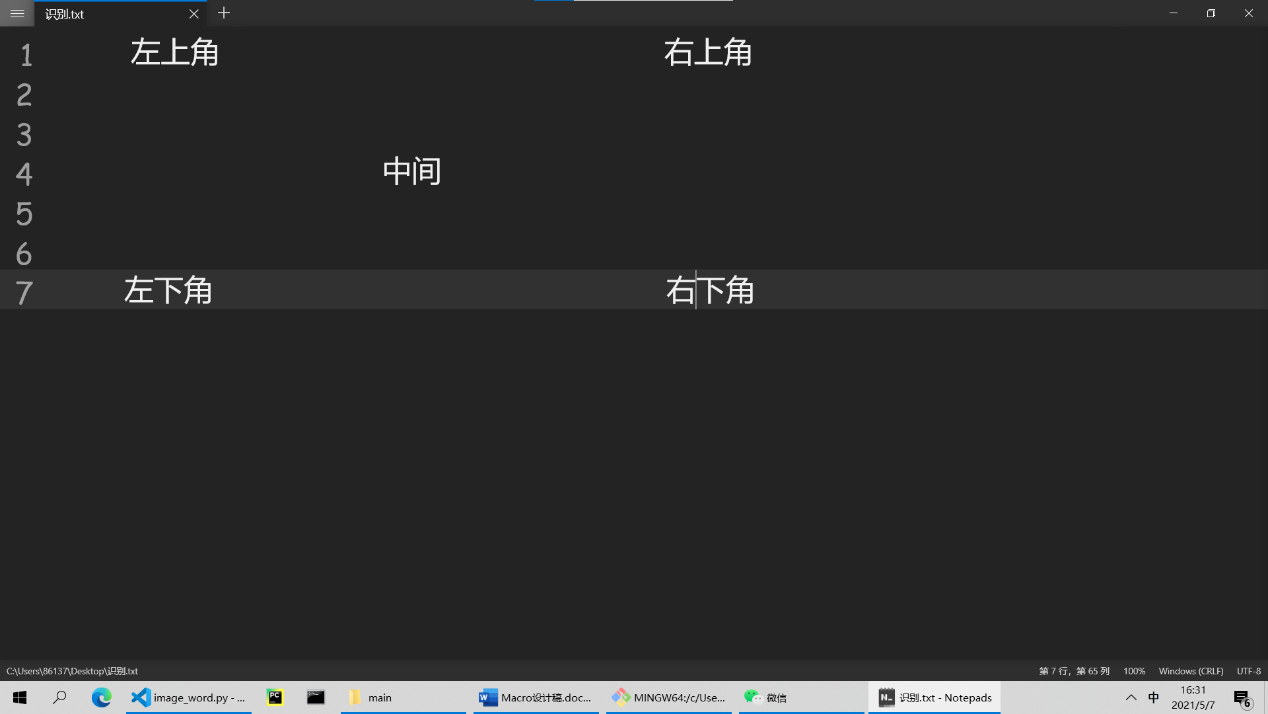
识别的图片为：****

图 5-12 传入屏幕截图

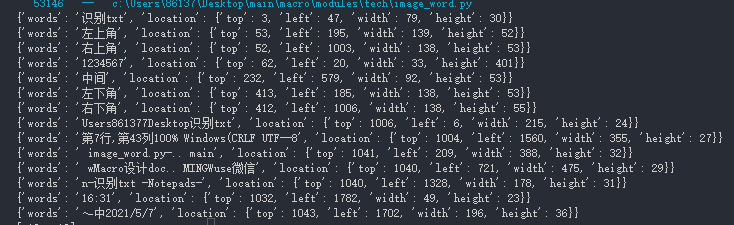
****

图 5-13 屏幕文字的识别结果

在识别小图片的文字时，使用外部库pytesseract的OCR技术。该OCR方法可以无限次使用，识别小图片的精度与百度OCR技术的精度接近。

* 1. **语音识别**

语言识别利用百度的语音识别技术，可以比较精准的识别用户说的中文，简单的英语与阿拉伯数字。对于Macro中的重要语法关键字，我们利用百度提供的训练接口，对模型进行训练，确保它相比于一般词汇，能被更精准地识别。

* 1. **键盘与鼠标动作的捕捉**

利用win32的钩子函数，布置到软件中，可以实时监听鼠标和键盘的事件。通过这个功能，我们实现了录屏操作，让用户可以不用书写代码，直接绘制脚本，大大降低用户的学习成本。此外，允许用户调整监听鼠标事件的频率，让录屏操作更加贴合用户的需求。

* 1. **GUI**

GUI提供主窗口与子窗口。

主窗口主要进行代码的编辑，参数的设置与脚本文件管理。在利用Qt Designer进行整体界面布局之后，利用QSS在一些细节上进行优化，比如对于按钮的状态，在按钮的闲置悬浮，点击时显示的图标有所不同。告别了Qt原版僵硬的点击体验，给用户更好的使用感。

子窗口用于显示鼠标所在的屏幕位置，方便用户更好的进行脚本的编辑。



图 5-14 子窗口

除此以外，我们设立两种主题样式——日间模式和夜间模式供用户选择。

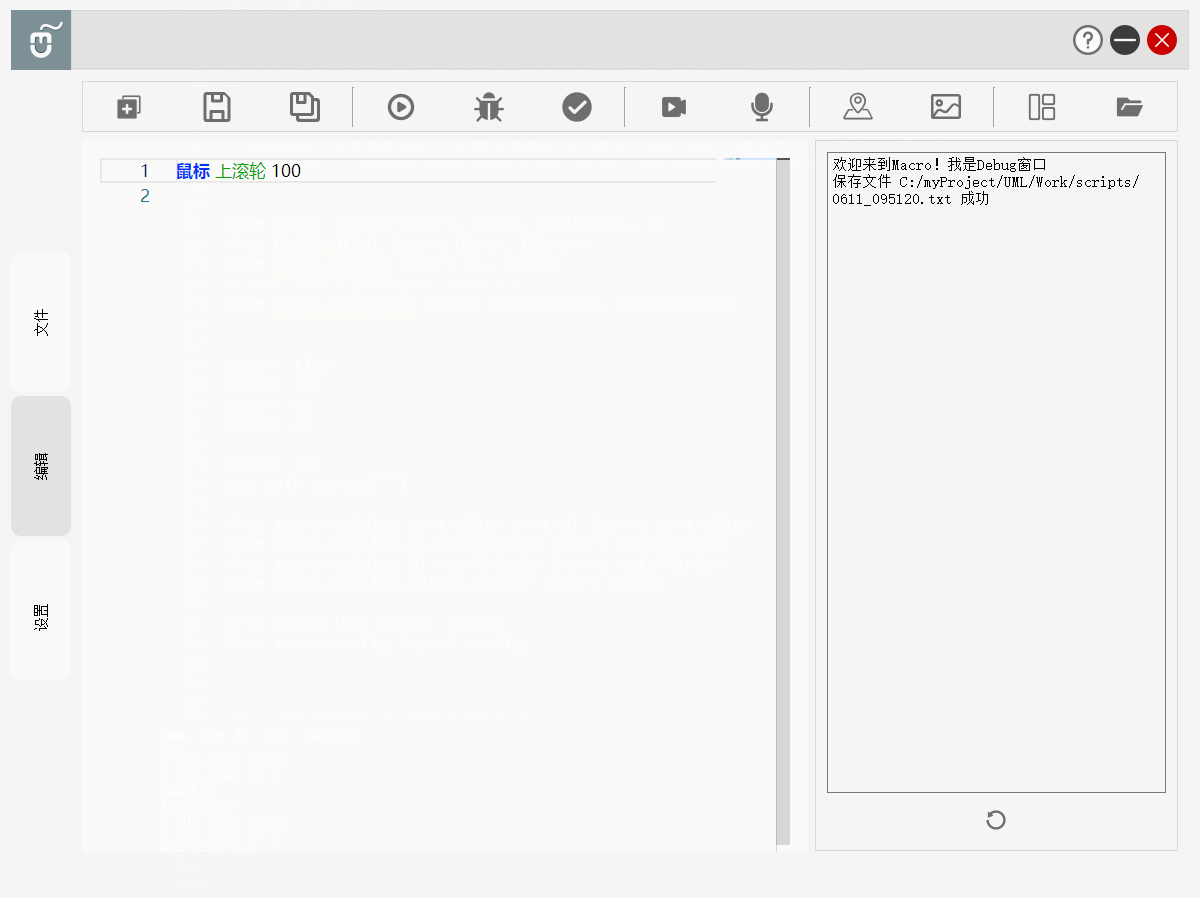


图 5-15 日间模式

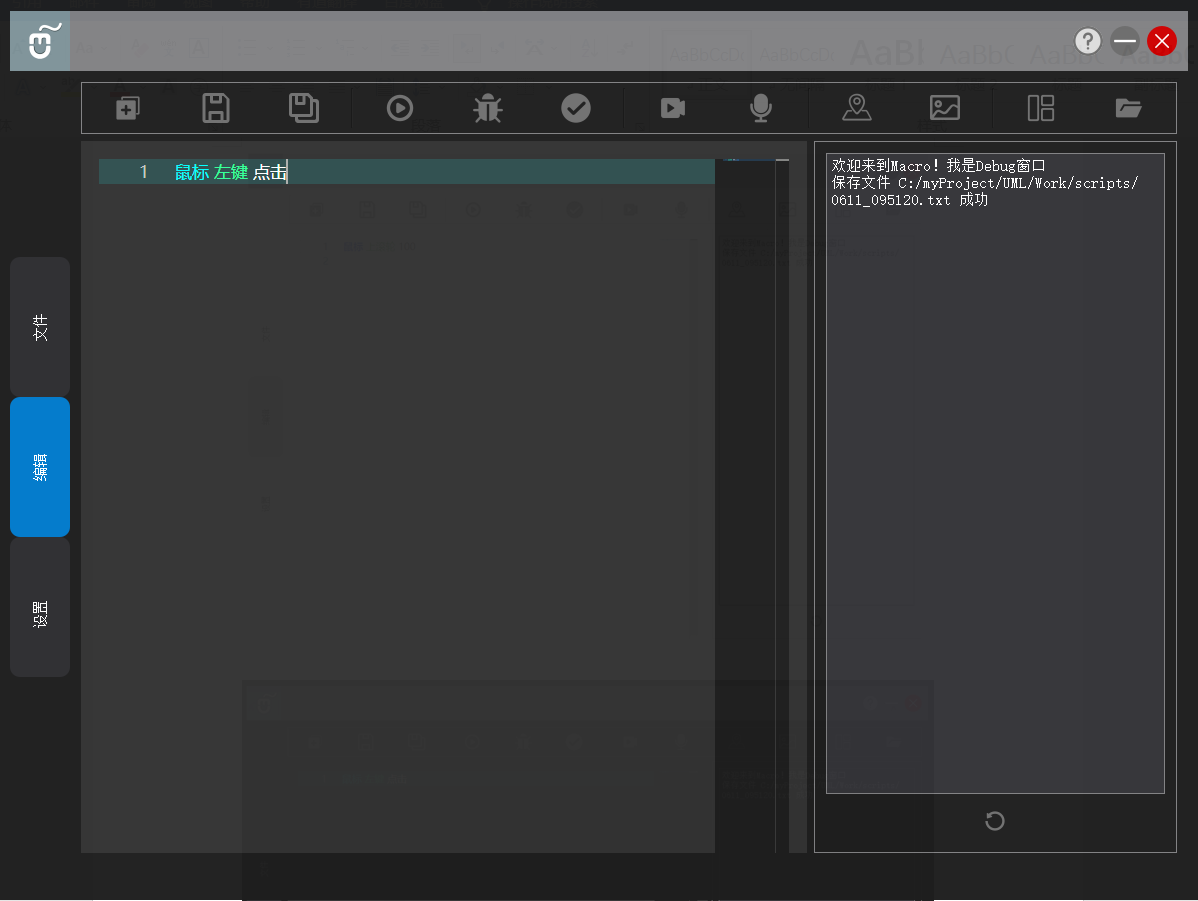


图 5-16 夜间模式

**5.6 编辑器**

基于Monaco Editor，Macro为用户提供了拥有自定义语言的文本编辑器。之所以选择Monaco Editor，是因为它是VS Code的浏览器版本，性能十分强大。为了让用户与脚本的交互体验更佳，编辑器自定义了Macro专属的语言，提供了类似于vs的编辑页面。编辑器目前实现的功能包括关键词高亮、代码自动补全、语法联想等，后面计划实现语法检测、基于输入的动态补全等功能。以下为编辑器的展示：

**1.基础页面**

编辑页面实现了行号显示、当前行标示、小地图、滑动条等功能。

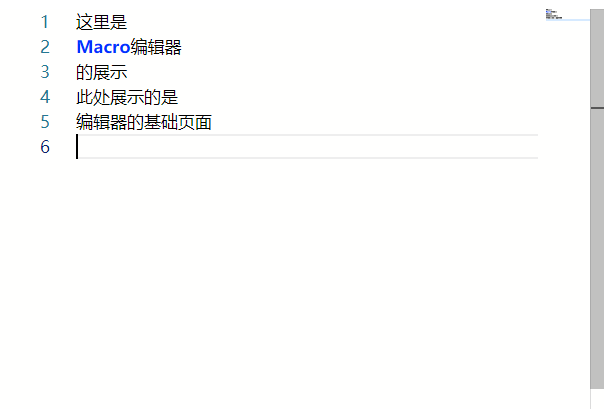


图 5-17 基础页面

**2.自定义语言**

根据Macro自定义的语法对关键词进行分类，根据关键词类别提供高亮功能。

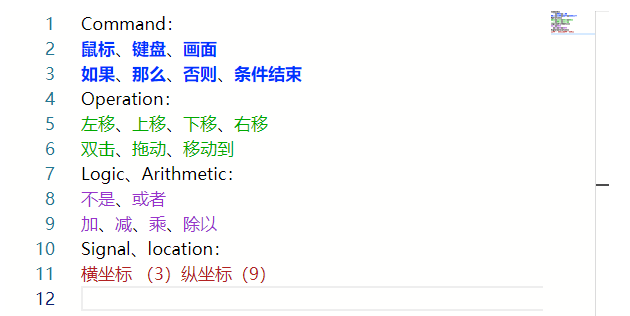


图 5-18 关键词

1. **代码自动补全、语法联想：**

编辑器自动识别用户的输入，匹配满足条件的代码，提示用户并进行自动补全。

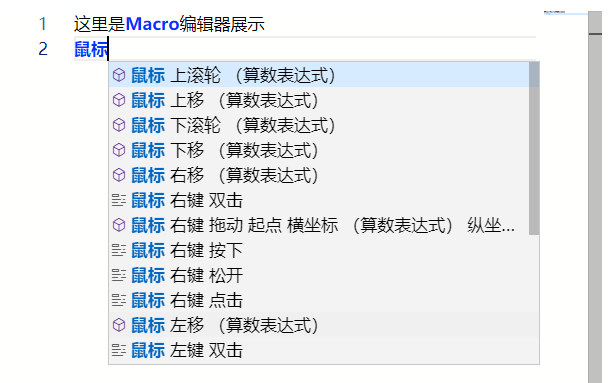


图 5-19 自动补全

内置函数提示用户需填写的文本框，代码自动补全、语法联想的实现，大大提高脚本编写效率。

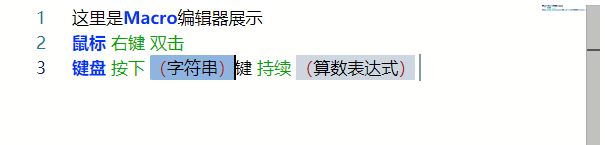


图 5-20 语法联想

除此以外，也提供搜索，替换，前往等十余种常用快捷键的功能。