LaTeX 模版生成程序使用手册
LaTeX Template Generator Doc

温刚*, 王奕轩†

May 18, 2020

Contents

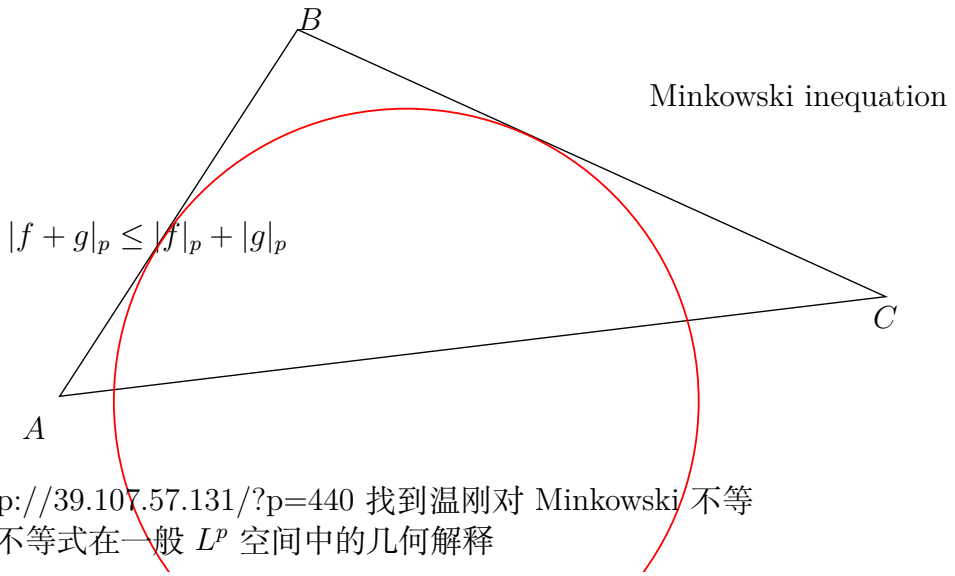
*jnwengang@pku.edu.cn
†tom-yixuan_wang@pku.edu.cn

1 需求分析

用 LaTeX 编辑数学论文和其他理科材料已经成为科研工作者的一种必备技能. 无论是 markdown 还是 word 都远不及 LaTeX 编辑数学公式来的快和实在. 然而, 用 LaTeX 插入图片和很多固定格式的组件长久以来让人不满, 而且阻碍所有人使用 LaTeX 进行编辑 (这导致很多数学论文丑陋不堪).

而对于插入的图片, 同学们往往追求方便使用 includegraph 命令, 这使图片难以调整, 而且很难将 LaTeX 的强大功能与图片联动起来. 想想 Stein 的数学书的插图吧! 尤其是 Complex analysis, 他的图片就像是用 tikz 绘出的一样——然而, 我们可以用 inkscape, 一款开源的却非常强大的矢量图绘制软件达到同样的效果, 而且还将结合 vim 与 LaTeX 弥补二者的缺点.

如果您想查看用此种方法得到的图片效果, 请参考 http://39.107.57.131/?p=204 阅览温刚的数学分析笔记插图; 或者 http://39.107.57.131/?p=509 阅览温刚的实变函数笔记插图. 将此课堂笔记与您的插图做做笔记, 看您是否能一边记着随堂笔记一边达到这样的效果?



你可以在 http://39.107.57.131/?p=440 找到温刚对 Minkowski 不等式和 Holder 不等式在一般 L^p 空间中的几何解释

Figure 1: 一起 LaTeX!

2 设计

本软件本学期由于设定了跨平台的目标, 设计过程较为曲折, 设计方向多次更改, 现按时间顺序说明设计思路及其变动.

学期初, 设定了使用快捷键插入代码片段和打开 inkscape 的小目标, 此时还没有对 gui 有什么指望.

经过第一轮尝试,Mac 端由于权限问题无法使用 keyboard 进行按键监听, 退而求其次, 温刚设计了软件主窗口的雏形. 或者说, 此时软件主窗口就已经做好了, 我们期望用户使用外部窗口实现交互, 而不是快捷键.

第二轮尝试, 王奕轩重构了 gui 代码, 完善了项目的代码架构, 实现了 windows 的快捷键操作, 并添加新的功能: 蓝图. 详细见文档内部.

第三轮尝试, 温刚发现新的模块 pynput 在 mac 上运行正常, 于是实现了 Mac 上的按键监听. 此时温刚的方向主要是 inkscape 的扩展. 在此阶段, 将 inkscape 扩展到了能和软件互联互通的地步. 然而,Inkscape 的文本插入多有弊端. 基于 x11 的 inkscape 在 Mac 上甚至不能输入中文, 因此, 此时设计通过 xterm 与剪贴板交互重构 inkscape 的文本插入操作.

第四轮尝试, 王奕轩修补了 bug 并继续向蓝图和 windows 方向分化, 另外将 inkscape 已经在 Mac 实现上的功能迁移到了 windows 上. 与此同时, 王奕轩使用的 gvim 给温刚以启发, 基于 xterm 设计的重构的文本框换为了 macvim.

打包阶段,windows 可以正常打包, 对于 Mac, 温刚写了安装器 start.py.

第五轮尝试, 软件已经基本成形, 并在 github 主页发布了若干 release. 详情见下方.

3 使用说明

3.1 初级教学视频网址

http://39.107.57.131/?p=593

如果视频加载过慢, 您可以直接从网盘下载:

链接: https://pan.baidu.com/s/1iwLiezpJuhX5rDJNy2qEYA 密码: qwh6

注意到本视频所讲解的版本是 0.0.2, 如果此资源失效, 请下载最新的 release 资源.

Windows 端的初级使用与 mac 端大同小异. 但是安装过程有较大差别, 请跳过前面的安装部署过程.

3.2 快捷键声明

3.2.1 Windows

- ctrl+u: 将选定的文字格式化为插入的 LaTeX 图片代码, 原地替换; 并将此代码放入剪贴板. 另外, 自动打开 inkscape, 在工作路径下的 figures 文件夹中生成对应的图片.
- ctrl+alt+i: 这将弹出一个小型的 gvim 窗口, 您在里面输入任何公式, 退出后此公式将放入您的剪贴板. 如果您在 inkscape 中工作, 此公式将自动出现在您鼠标指定的位置.
- ctrl+j: 这将弹出一个含有完整 LaTeX 导言和代码的微型 gvim 窗口, 您在里面编辑并退出后内容将自动放入剪贴板. 当然如果您之前有配置过 vim 使其可以实现 LaTeX 预览 (例如 vim-latex-preview 插件等) , 也可以生效.

  注意, 以上弹出的 gvim 文档都是临时文件, 使用后会自动清除.

3.2.2 Mac

- cmd+u: 与 windows 的 ctrl+u 功能相同.
- cmd+0:u: 与 windows 的 ctrl+alt+i 功能相同, 只不过使用 macvim 代替 gvim.
- cmd+j: 与 windows 的 ctrl+j 功能相同. 说明同上.

Remark. Mac 端的 vim 配置可以从网盘下载:

链接: https://pan.baidu.com/s/1KygY-6Kd0-kayqT6fZ57YQ 密码: dbq9

下载之后将.vimrc 与.vim 放在主目录即可.

3.3 软件界面结构

Figure 2: 软件界面结构

先按从左到右, 从上到下的顺序依次说明各个组件的作用.

- 清空依赖区: 即清空最下方米黄色的区域的所有文字, 而这些文字显示的是您应该在 LaTeX 导言区插入的内容.
- 清空片段: 即清空上方米黄色的区域的所有文字, 而这些文字显示的是您想插入的代码片段.
- 图片名称: 您在此键入您想创建的图片名称.
- 生成片段并复制: 程序对您的图片名称进行格式化和 LaTeX 语法错误过滤, 生成可以直接使用的片段并在上方米黄色区域展示. 同时, 展示的内容也将放入您的剪贴板.
- 执行宏: 使用您在文本框中输入的图片名字生成图片.
- 经过处理的图片文件名: 展示文件名.
- 编辑已有图片: 下方的白色框中会显示您在当前工作路径下已经有的图片, 选定后点击此按钮, 将自动打开 inkscape 对其进行编辑.
- 左上方米黄色框: 片段区.
- 右方白色框: 展示当前目录下已有的图片文件.
- 左下方米黄色框: 导言区 (依赖区).
- 删除此文件: 彻底删除您选定的 svg 和附带的 pdf,pdf_tex. 毫不留情, 不会放入回收站, 所以如果没有备份就再也找不回来了.

- 更新文件列表: 如果您发现右方白色框没有展示全部的图片, 点击此按钮刷新缓存.
- 控制台窗口: 用于调试和输出日志, 操作历史和错误报告.

3.4 软件菜单栏结构

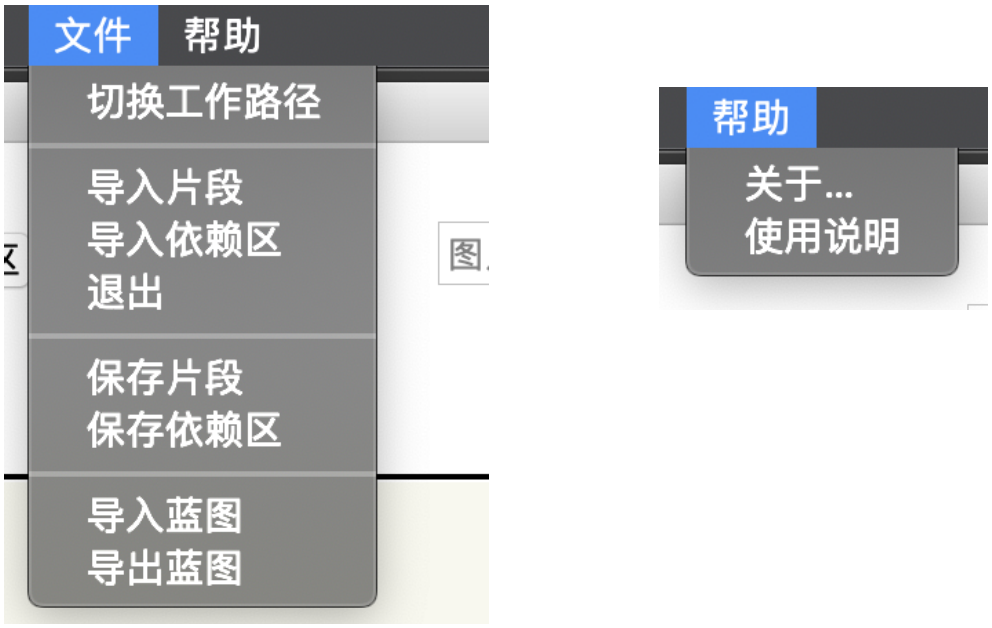


Figure 3: 菜单栏

蓝图功能尚在开发. 其实现可以类比 vscode 的 json 配置.

4 编码实现

4.1 第三方软件

Inkscape 和 gVim 都提供命令行接口。

Inkscape

```
$ inkscape "foo.svg"
# 打开SVG
$ /Applications/Inkscape.app/Contents/MacOS/inkscape "foo.svg" \
--export-file='foo.pdf' --export-latex
# (MacOS)转换为PDF_TEX.这是inkscape1.1dev版本,不稳定;现在api可能已经更改.
$ /Applications/Inkscape.app/Contents/MacOS/inkscape "foo.svg" -o \
'foo.pdf' --export-latex
#(MacOS)转换为PDF_TEX.这是inkscape1.0 (4035a4f, 2020-05-01) 版本,也是我们采用的
```

稳定版本.

```
$ inkscape "foo.svg" -o 'foo.pdf' --export-latex
# (Windows)转换为PDF_TEX
```

gVim & MacVim

```
$ /Applications/MacVim.app/Contents/MacOS/Vim -fg "bar.tex"
# (MacOS)用MacVim打开一个文件,并将线程锁定到当前PID.
$ gvim "bar.tex" # (Windows)用gVim打开一个文件
```

4.2 第三方包

appdirs 用于获取特殊的应用文件夹路径，跨平台通用。便于存取一些内部数据。

```
appdirs.user_config_dir("Project", "Author") # 设置文件夹
appdirs.user_data_dir("Project", "Author") # 数据文件夹
```

pyperclip 用于读取和写入剪贴板。

对于开启了剪贴板管理器功能的 Windows 系统，剪贴板中数据存储在栈中，复制为压栈，粘贴为获取栈顶，因此复制空字符串不能起到清空剪贴板效果。

```
pyperclip.copy("Foo") # 复制到剪贴板
pyperclip.copy('') # (MacOS)清空剪贴板
pyperclip.paste() # 获取剪贴板内容
```

requests 提供更简单的网络访问。用于下载在线安装指南,MacOS 还用于请求下载必要的安装组件和配置。

```
requests.get("https://foo.com/bar.pdf") # HTTP GET
```

keyboard, mouse, pynput 提供键盘和鼠标的监听和模拟操作。

在 Windows 上使用 keyboard 和 mouse，它们的代码比较简洁。pynput 在 Windows 上与中文输入法冲突。

```
keyboard.send("F8") # 模拟键盘按下—抬起事件
keyboard.add_hotkey('ctrl+alt+i', func) # 监听某一按键组合
mouse.click("left") # 模拟鼠标左键单击事件
```

keyboard 和 mouse 不兼容 MacOS，因此使用 pynput。pynput 可以正确处理 MacOS 的授予权限操作。pynput 的实现类似于进程操作。

```
with pynput.keyboard.GlobalHotKeys({'<cmd>+u': fun1, '<cmd>+8': fun2}) as hotkey:
    hotkey.join()
# 绑定多个按键组合
def for_canonical(f):
    return lambda k: f(listener.canonical(k))
```

```
listener = pynput.keyboard.Listener(
            on_press=for_canonical(hotkey.press),
            on_release=for_canonical(hotkey.release),
        )
listener.start()
# 监听hotkey按下—抬起动作

with pynput.keyboard.Controller.pressed(pynput.keyboard.Key.cmd):
    pynput.keyboard.Controller.press('c') # 模拟按下一个按键
    pynput.keyboard.Controller.release('c') # 模拟抬起一个按键
# 模拟按下按键1、按下按键2、抬起按键2、抬起按键1

pynput.mouse.press(pynput.mouse.Button.left) # 模拟按下一个鼠标键
pynput.mouse.release(pynput.mouse.Button.left) # 模拟抬起一个鼠标键
```

4.3 程序结构

- start.py（MacOS 启动器）
- main.py（主程序）
- globe.py（全局变量）
- gui.py（tkinter GUI）
- widget.py（tkinter 自定义组件）
- workspace.py（工作路径处理）
- util.py（杂项，字符串处理函数）
- edit_scroll_process.py（获取路径下的全部 svg）
- blueprint.py（蓝图）
- inkscape_control.py（inkscape 控制）
- paste.py（快捷键监听：自动复制代码片段和 vim 输入框功能）

4.3.1 全局变量

全局变量是通过 globe 模块中 Globe 类的属性来实现的。

- ui 是一个利用字典构建的树结构。Tkinter 组织界面元素的方式不便于动态修改（`mainloop()` 后会丢失对组件的引用，而且难以再找到），而 ui 可以提供可读性更好引用，避免在初始化 GUI 时使用大量的回调函数。使用方式：`ui['button']['edit']` 会返回功能是“编辑”的按钮的引用。
- workspace 是一个记录工作路径和子文件夹（如 figures 文件夹）的字典。
- blueprint 是当前正在使用的蓝图。

- system 是表示操作系统的常量。

4.3.2 工作路径

MacOS 中需要切换 Python 工作路径才能正常读取文件。在 workspace 模块中提供了设置工作路径 `cwd`、生成子文件夹（如 figures 文件夹）`sub` 以及历史工作路径的处理相关的函数。每次启动程序时，会自动恢复上次设定的工作路径。初始设置的工作路径是程序所在的文件夹。

4.3.3 字符串处理函数

util.StrUtils 类提供了一些字符串处理方法，包括检查合法性 `rmOperand`（去除 LaTeX 控制字符）、转换为标题格式 `caption`、转换为标签格式 `label`，转换为文件名格式 `fileName`。

4.3.4 和 Inkscape 的协调

在 inkscape_control 模块中实现。

在加载此模块时，将一个空白的 SVG 模板复制到用户应用配置文件夹中（由 appsdir 寻找路径；方便 MacOS 上的读取）。

需要用到 Inkscape 进行绘图时，首先判断是新建 SVG 还是编辑 SVG。如果是新建，则复制 SVG 模板到工作路径下的 figures 子文件夹，并将指定的字符串转换为文件名格式用来命名，再打开。如果是编辑，直接打开。

打开 SVG 的函数使用到 Python 自带的多进程模块，在另一个进程（称作 I 进程）中打开，主进程的 GUI 依旧可以响应。由于 MacOS 上的进程线程机制，这里不能使用线程。在 I 进程中，首先启动一个子进程，并用 `wait` 方法等待其结束，在其中打开 inkscape 的 GUI 界面，可以绘图。当关闭这个 GUI 界面时，I 进程继续执行，启动另一个子进程，将 SVG 文件转换为 PDF 和 PDF_TEX 文件，用于插入 LaTeX 中。之后，I 进程结束，完成整个和 inkscape 协调的流程。

4.3.5 自动复制代码片段

在 paste 模块中实现。

先注册一个键盘快捷键的监听器（使用 keyboard 或 pynput），在捕捉到键盘事件后，调用 `on_activate` 函数。此时用户已经选中了一个字符串，模拟按下 Ctrl/Cmd+C，将这个字符串复制到剪贴板。复制到剪贴板的模拟键盘按下操作不是在同一个进程内执行的，因此需要设定一个短暂的延时以确保在复制到剪贴板的操作已经结束后，下面的程序才会继续执行。之后读取剪贴板中的内容，并用当前蓝图的方法获取代码片段，写入剪贴板，并用一段延时保证以上操作正常结束。这时，模拟按下 Ctrl/Cmd+V 将剪贴板的内容粘贴到选中位置，实现了将选中的字符串自动扩展成对应代码片段的操作。在一段延时后，在 MacOS 上，通过复制空字符串实现剪贴板清空的操作。最后调用和 Inkscape 协调的函数，可以实现快速创建/编辑 SVG。

4.3.6 Vim 输入窗口

在 paste 模块中实现。

同样地，先注册快捷键的监听器，回调函数是 `open_vim` 函数。利用自带的 tempflie 模块创建一个临时 TEX 文件，并写入用于标注 LaTeX 公式的"$$"。之后在 `open_editor` 函数中用 gVim 编辑这个文件。编辑结束后，读取文件内容并写入到剪贴板，模拟按下鼠标左键使焦点回到 Inkscape 界面，再模拟按下文本框工具的快捷键（MacOS 上是 t，Windows 上是 F8，再此之前，MacOS 还需要模拟按下 Esc 使 Inkscape 退出其他工具），接着模拟按下鼠标左键创建文本框，再按下 Ctrl/Cmd+V 将剪贴板的内容粘贴到文本框中，实现在 inkscape 外部用 vim 辅助输入文本。最后，清除临时 TEX 文件。

4.3.7 TinyTex

在 paste 模块中实现。

tiny_tex 的实现是基于 Vim 输入窗口。会利用 tempfile 生成一个完整的临时 TEX 文件，写入预制的导言区内容，之后在 Vim 输入窗口中打开它，关闭后，文件内容会写入到剪贴板，之后销毁临时文件。

5 Windows 测试部署

Windows 上使用 PyInstaller 进行了打包，而 MacOS 则需要使用 start.py 做启动器。Windows 上的测试部署过程如下。

5.1 安装第三方软件

1. 安装 Inkscape，**仅兼容 1.0+ 版本**。典型安装即可。
2. 安装 Vim，其中图形化界面 gVim 是必需的。典型安装即可。

在北大网盘上提供了安装包链接。

64 位 Inkscape 1.0：

https://disk.pku.edu.cn:443/link/9B21ED1C95639B9424186456A21A48AA，有效期限：2020-10-01 23:59

32 位 Inkscape 1.0:

https://disk.pku.edu.cn:443/link/477334A3B1C8518E9497DF4A69A2FF77，有效期限：2020-10-01 23:59

通用 gVim:

https://disk.pku.edu.cn:443/link/81E4EDF2F9230A544650F928C0296F45，有效期限：2020-10-01 23:59

5.2 设置环境变量

环境变量设置方法请参看：百度经验 - 添加环境变量：

https://jingyan.baidu.com/article/47a29f24610740c0142399ea.html

找到 Inkscape 和 Vim 的安装路径（在/ProgramFiles 或/ProgramFiles(x86) 文件夹下），并：

1. 将 Inkscape 安装路径文件夹中**/bin** 文件夹的路径添加到用户环境变量的 **PATH** 中

   路径示例：C:\Program Files\Inkscape\bin

2. 将 Vim 安装路径文件夹添加到用户环境变量的 **PATH** 中

   路径示例：C:\Program Files(x86)\Vim

环境变量生效可能需要重启程序或者**重启计算机**。

5.3 运行程序

主程序 main.exe，无需安装。为了便于查看调试信息，这个版本中保留了命令行窗口。

6 Mac 测试部署

见在线安装指南:

http://39.107.57.131/?p=568

7 蓝图

“蓝图” 指的是软件中使用的模板系统。这一系统增加了软件的可扩展性。

蓝图是整个程序操作流程的抽象。一段 LaTeX 代码中可能用到一些第三方宏包中的指令或自定义的指令，因此**代码片段**和**导言区中第三方宏包及自定义指令的声明**应该是一体化的。同时，对于一个代码片段，其中一部分内容是预制的，其中一部分是在使用过程中随用随填的**变量**，这些变量需要经过一定的**处理**才能呈现在**最终插入 TEX 文件的代码**中。我们也希望能以这些变量为参数**执行一段程序**，比如编辑同名的 SVG。基于以上几点，我们设计了 “蓝图”。由于其复杂性较高，目前暂未实现全部的功能设想，但预留了相关接口。

7.1 变量的处理

一个代码模板好比是一个填空题，大部分的代码都是预设好的，**但是里面有几处是可变的，可以由用户自己填写内容**。比如插入一个 SVG 图片的代码片段，\end{figure} 就属于预设好的内容，无需改动，而\label{fig:your-fig-label} 中的 your-fig-label 就是可变的，可自定义的，**可以随使用指定**。我们的片段中还有几处这样的空位，比如\incfig{your-fig-file}，\caption{your-caption}，your-fig-title 和 your-caption 都是可以自己指定的。

但这三处内容（your-fig-label，your-fig-file，your-caption）具有相关性，它们的值可能是同一个量的不同变化形式。比如我指定 SVG 图片的名称叫做叫做 `test? fig`，图片的标

签应该写成 `test?-fig`，而文件名中不许出现问号所以要变成 `test-fig`，而标题的首字母要大写，应该写成 `Test? fig`。这三个量都是同一个量的变体。我们把每一个这样的变体叫做 **factor**，把原来的那个量称作 **variable**，从一个 variable 到一个 factor 需要一种转换操作。

目前的模板里只有一个 variable，就是 `name`，我们图片的名字。但假如说我们在不同的地方想要设置不同的浮动体位置，那么\begin{figure}[**ht**] 里的 **ht** 也应该变成一个 variable，因此**预留了多个 variable 的可能**。目前的 GUI 只能使用一个 variable，但目前的代码实现可以兼容多个 variable 的处理。

蓝图对象的 variable 属性是一个元组（因为集合不能转换成 json，所以只好用元组），用来表明在这个对象的代码模板 snippet 里需要用到哪几个 variable，元组中每个元素的值就是这个 variable 的名字。目前有且只能有一个 variable，所以默认的蓝图中这个属性中只有一个元素，其值为 `('name',)`

蓝图对象的 factor 属性是一个字典，用来表明在这个对象的代码模板 snippet 中使用到的、**需要用 variable 替换的**这些空位。字典的键表示这个 factor 的名称，而字典的值是一个元组，第 0 位表示转换成这个 factor，**需要什么转换操作**，而第 1 位则是说明这个 factor 是**由哪个 variable 转换而来**的。

```
{
    'caption': ('$caption', 'name'),
    'fileName': ('$fileName', 'name'),
    'label': ('$label', 'name')
}
```

其中第一个键值对表示 `caption` 这个 factor 是：由 `name` 这个 variable，经过一个叫做 `caption` 的内置操作转化而来的。如果从 variable 到 factor 不需要任何格式转换，第 0 位设为空字符串即可。

7.2 代码片段的处理

snippet 就是一个**代码的模板**，像这样：

```
\begin{figure}[ht]
    \centering
    \incfig{ {{fileName}} }
    \caption{ {{caption}} }
    \label{fig:{{label}} }
\end{figure}
```

这里用 `{{`和`}}` 包裹的部分就是上面提到的 factor，**需要填充的位置**。因此，蓝图对象的 snippet 属性是**符合 latex 语法的字符串**，同时包含**用于字符串替换（填充 factor）**的部分。经过字符串替换（填充 factor）的过程得到的结果称作 **fragment**，也就是如下样式的内容：

```
\begin{figure}[ht]
    \centering
    \incfig{my-fig}
```

```
    \caption{My? fig}
    \label{fig:my?-fig}
\end{figure}
```

这一个 fragment 就是**可以直接插入 tex 文件**的代码片段。因为 fragment 是根据 variable、factor、snippet 直接生成的，所以没有对应的属性。

dependency 是 snippet 中使用到的第三方宏包的声明（\usepackage）和自定义指令的定义（\newcommand），是**加入 tex 导言区**的部分。dependency 是依附于 snippet 存在的。加载一个蓝图时，dependency 的内容会显示在下方的文本框内。

7.3 第三方程序的处理

当我们根据变量 variable 生成了一段 tex 代码以后，可能还希望能以这些变量为参数**执行一段第三方程序**，比如根据我之前的输入，创建一个 svg 图片并在 inkscape 里打开……这一系列操作叫做 **macro**。macro 属性的值就是你要执行的函数名称，默认是 `$inkscape`，将传入的参数视为 SVG 的名称，用 inkscape 打开。如果不需要执行任何 macro，将这一属性设成空字符串即可。

7.4 蓝图的代码实现

蓝图类 Blueprint 有三个方法：get*factor*，*get*fragment，do_macro。

这三个方法的参数完全一致，是一个解包的字典，每一个键就是 **variable 的名称**，对应的值就是 **variable 的值**。由于在目前未实现多 variable，因此这个字典**只能包含一个键值对**。使用字典是为了向后兼容。

get_factor 方法返回所有的 factor，是一个字典，键是 factor 名，值是对应 factor 的值，即经过处理的变量 variable。

get_fragment 方法返回 fragment 字符串，一段**可以直接插入 tex 文件的代码片段**。

do_macro 方法没有返回值，在里面调用 macro 函数，**执行第三方程序**。

7.5 内部函数和外部函数

为了方便导入导出，蓝图中存储的是**函数名**，而非**函数的引用**。blueprint 模块中的 FacMethod 和 Macro 两个类分别实现了从**函数名字符串**获取**变量处理方法**（将 variable 转换成 factor）和**第三方程序控制函数**（macro）的功能。

程序内置了一些**处理变量的方法**(util.StrUtils)以及**控制 inkscape 的函数**(在 inkscape_control 模块下的各个函数)，使用这些内置方法/函数时，相应的方法/函数名前需要加'$' 符号。比如：`$caption` 表示转换为标题的内置字符串处理方法，`$inkscape` 表示控制 inkscape 的内置函数。

所有不加 "$" 的函数名字符串会被视作外部函数。调用外部函数/方法，即**加载第三方 Python 脚本**的功能涉及到更复杂的接口设计，暂未实现，但已经预留了相关的接口。

7.6 蓝图的导入和导出

蓝图**可以自定义**。在“文件”命令菜单中，可以导入和导出当前使用的蓝图。

但是 JSON 的可读性比较差，直接编辑是不方便的。由于蓝图的接口比较复杂，暂时未实现在 GUI 中编辑蓝图。建议使用随附的蓝图：

default	默认加载的蓝图，仅允许浮动体在原位或顶端
bottom	仅允许浮动体在原位或底端
norestriction	仅允许浮动体在原位或顶端，并解除浮动体数目限制
float	仅允许浮动体在顶端、底端或单独成页。LaTeX 的默认值。
captionfirst	仅允许浮动体在原位或顶端，标题在图片之前。

8 贡献

8.1 王奕轩负责的部分

- 重构代码（调整模块及函数编排）
- workspace 模块
- 导出 pdf_tex 和 inkscape 进程
- paste 模块在 Windows 上的实现
- blueprint 模块

8.2 温刚负责的部分

- Gui 的初步类和各个组件实现
- paste 模块
- 激活 inkscape 和 vim
- 扩展 inkscape 方向的功能
- 构建 MacOS 软件安装和启动器 start.py

9 评价与总结

9.1 跨平台开发

跨平台开发是困难重重的。MacOS 和 Windows 处理线程、文件读写的方式存在差异，第三方软件和第三方 Python 包对不同平台的支持也不同，而 Python 自带的包跨平台适配优良。

跨平台代码的理想状态是在代码中不出现任何 `platform.system()` 的分支结构，那样的代码可读性很差。除了处理键盘事件部分，我们基本做到了这一点。

9.2 GUI

Tkinter 生成图形界面较为方便，但它在界面元素的组织上较差。为此，我们使用了树结构存储元素的引用，可以减少回调函数数量和传递的参数数量，动态调整 GUI 更加快捷。

9.3 剪贴板的利用

利用剪贴板可以较好地完成不同程序之间的配合，基于此，自动复制功能可以在两个平台的任意文本编辑器中实现。

9.4 全局变量的引用

原先采用的是函数式全局变量在模块之间的相互引用, 这是没有用一个字典模块 globe.py 来存储简便稳定的.

9.5 跨平台打包

几乎是不可能完成的任务, 在 MacOS 上还有各种奇奇怪怪的错误, 时间紧迫就使用了自写启动器的方法.

10 鸣谢

已经毕业的唐仁于学长给予了 Mac 上程序开发的经验;

Pierre Glaser 在 github 上的项目给了我们初始的动力, 他将 Gilles Castel 在

https://github.com/gillescastel/inkscape-figures 的 ubuntu 系统的项目迁移到了 MacOS 上 (见相应的 pull request, 现在已经 merge) , 他慷慨的参与了我们第一轮的 debug 并给予了部分启示.

11 我们的项目主页

https://github.com/pkumath/datastructure

欢迎提出 issue 和 pull request.

或者致邮联系

12 其他问题

如果您使用了老版的 inkscape, 例如 0.92 版本 on mac, 请放心, 新版本的 inkscape(我们的软件强烈建议您使用 1.0 版本而不是 1.1) 不再基于 X11 设计, 和老版的 inkscape 完美共存.

如果您想继续使用 X11 来控制 inkscape, 建议您换成 xterm 而不是 macvim. 在 MacOS 上, 您只需要把/Applications/project 文件夹中的 paste.py 文件做相关修改即可. 这里提供一份支持中文显示的 xterm 配置:

```
XTerm*preeditType: Root
XTerm*cursorColor: OliveDrab1
XTerm*background: black
XTerm*foreground: AntiqueWhite1
XTerm*font: -misc-fixed-medium-r-normal-*-18-120-100-100-c-90-iso10646-1
XTerm*wideFont: -misc-fixed-medium-r-normal-*-18-120-100-100-c-180-iso10646-1
xterm*faceNameDoublesize: -misc-fixed-medium-r-normal-*-18-120-100-100-c-180-iso10646-1
XTerm*scrollBar:true
XTerm*rightScrollBar:true
XTerm*inputMethod:fcitx
xterm*VT100.Translations: #override \
                Ctrl Shift <Key>V:    insert-selection(CLIPBOARD) \n\
                Ctrl Shift <Key>C:    copy-selection(CLIPBOARD)
XTerm*SaveLines: 4096
```

如果缺少字体, 请自行下载必要字体.mac 端将能正常显示中文. 如果您的中文无法正常显示, 考虑修改配置如上. 修改方法是在主目录下创建.Xresources 并在其中输入以上内容, 保存之后在命令行输入

```
xrdb ~/.Xresources
```

然后重启 X11. 另外,X11 的偏好设置需要如下:

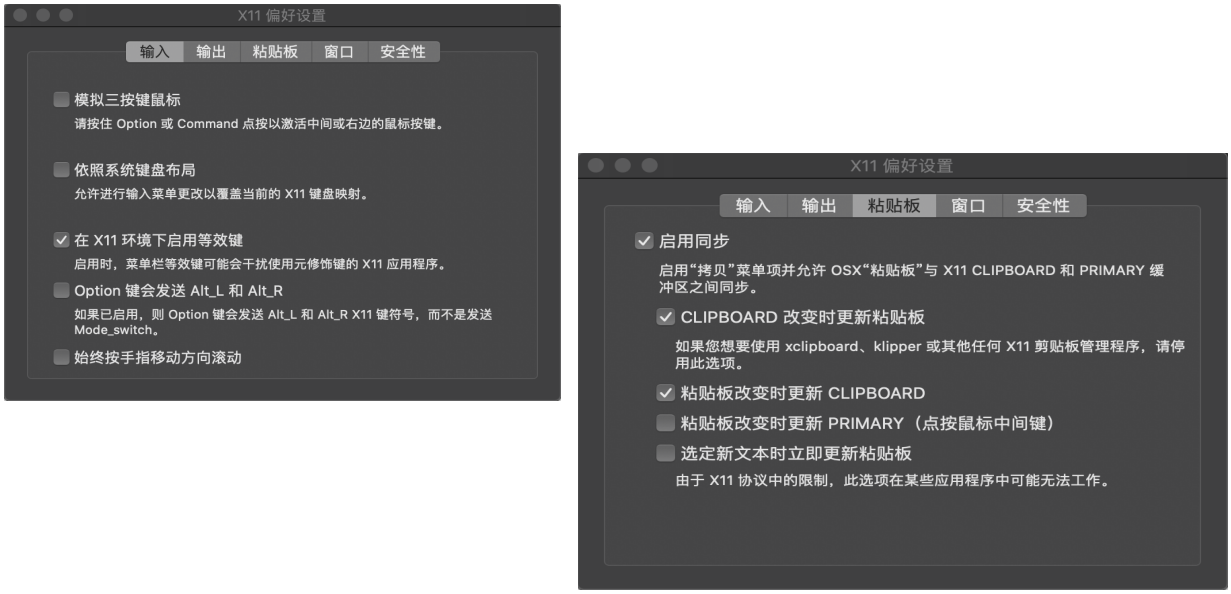


Figure 4: 烦人的 X11