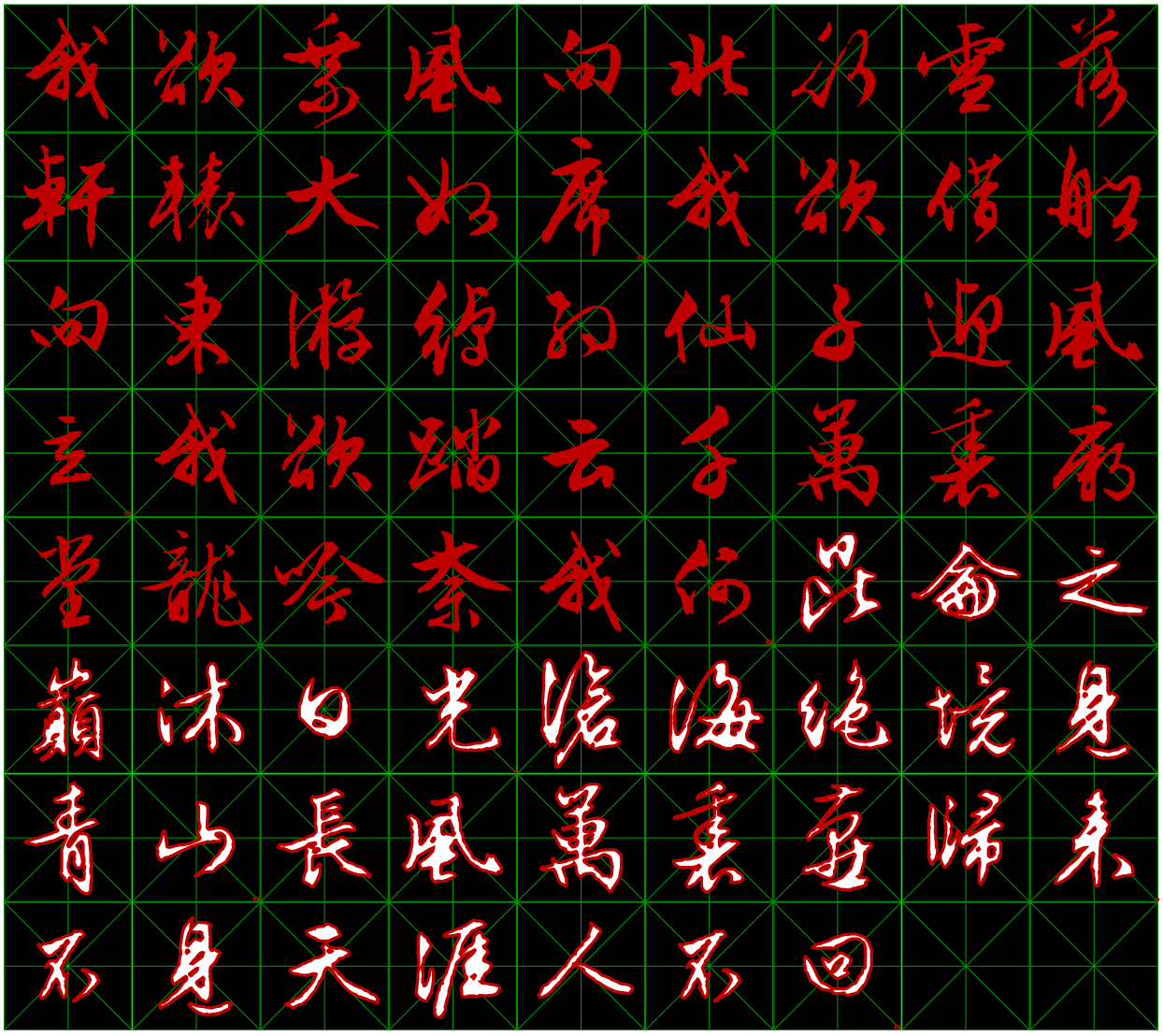
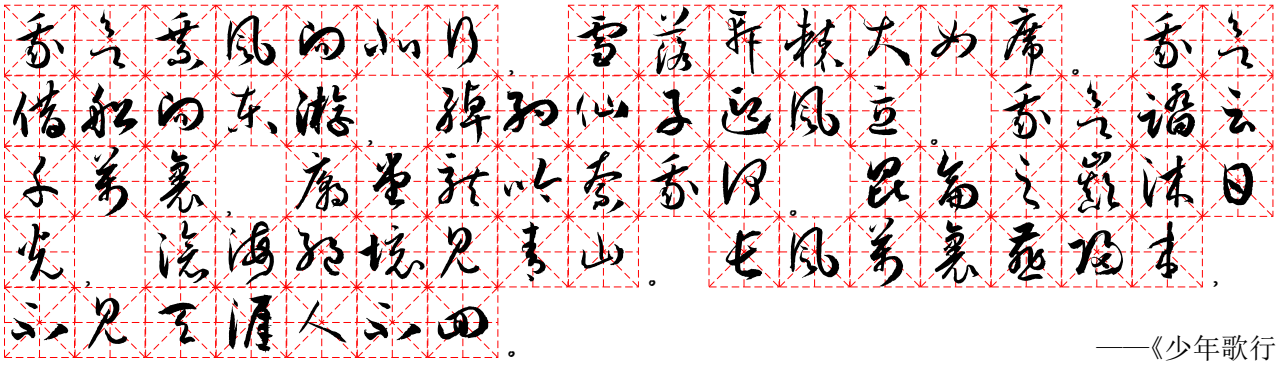
字帖宏包

雾月, Longaster (longaster@163.com)

农历二〇二一年六月廿五 v1.4.0





——《少年歌行》

zitie 宏包是用纯 LaTeX3 模块编写的（即，不加载 pgf、pstricks 等绘图宏包），也不使用 xcolor 宏包，来生成网格化的字、生成网格化的背景的宏包。绘图功能完全依赖 l3draw，但应注意 l3draw 目前仍然是实验性的，其接口在后续版本可能无法使用。

LaTeX 2_ε 版本应在 2020-10-01 及以后。

目前支持 XeTeX、LuaTeX，不支持 pdfTeX。应注意由于 LuaTeX 处理中文字符和 XeTeX 不同，使用 XeTeX 和 LuaTeX 编译效果会有不同，最好使用 XeTeX 编译。

目录

第 1 节 zitie 宏包的加载和基本命令

加载宏包：\usepackage[⟨*key-val*⟩]{zitie}。⟨*key-val*⟩ 选项见第 2 节。

使用 XeTeX 将加载 xeCJK 宏包 [1]。使用 LuaTeX 将加载 ctex 宏包。

\zitienewfont
\zitieCJKfamily

```
\zitienewfont {⟨font family⟩} {⟨font name⟩} , ... }
\zitienewfont {⟨font family⟩} {⟨font name⟩} {⟨font features⟩} , ... }
\zitienewfont * {⟨font family⟩} {⟨font name⟩} {⟨font features⟩}
\zitieCJKfamily {⟨font family⟩}
\zitieCJKfamily + {⟨font family⟩}
\zitieCJKfamily - {⟨font family⟩}
```

这个命令用来加载和设置需要的字体，⟨*font family*⟩ 为在 \framezi 等命令的 font 选项中使用的名字，也可通过 \zitieCJKfamily 来在文档中使用该字体。⟨*font name*⟩ 为字体的名字，⟨*features*⟩，为 fontspec 宏包 [2]，xeCJK 宏包（XeTeX 中）或 luatexja 宏包（LuaTeX中）支持的字体特性。

\zitieCJKfamily 和 xeCJK 宏包的 \CJKfamily 作用完全相同，只不过首先判断该 ⟨*font family*⟩ 是否被 \zitienewfont 声明，若没有，则再使用 \CJKfamily 进行处理。

\framesingle
\framezi
\framerange
\framezifile

```
\framesingle [⟨key-val⟩] {⟨字⟩}
\framezi * [⟨key-val⟩] {⟨字或词句⟩}
\framerange [⟨key-val⟩] {⟨逗号分隔的16进制序列⟩}
\framezifile * [⟨key-val⟩] {⟨文件名⟩}
```

以上命令用来生成网格化的字。

⟨*key-val*⟩ 为第 2 节定义的键。

⟨字或词句⟩ 应仅包含类代码为 11 或 12 的字符，即正常情况下输入的字符，不应包含 TEX 控制序列即其它特殊字符。\framezi 带星号的版本接收一个控制序列，这个序列包含想要网格化的字。

⟨逗号分隔的 *16* 进制序列⟩ 应使用如下的形式："4E00 -> "4E27, "4E30 或 `一 -> `丧, `丰，与其它设置字符序列的宏包相似（如 xeCJK 宏包的 \xeCJKDeclareSubCJKBlock）。注意，由于网格化操作需要大量的计算，因此单个 ⟨*range*⟩ 不应太大，依据编译环境而定，一般不应超过 3200 个，否则可能无法成功编译。其它命令中也是如此。若要使用大量字符，则可以使用 \frametallrange、\framezitallfile 等 tall 型命令，或 background 模块及其 xrange、yrange 特性。

⟨文件名⟩ 为想要网格化的文字的文件名。\framezifile 带 * 的版本，可在 ⟨*key-val*⟩ 中使用 filepath 设置文件搜索路径，详见第 2 节的说明。带星号和不带星号的版本实现略有差异，速度也可能不一样。

当连续多次使用 \framezi 等命令时，为去掉后面的空白，可使用 % 注释符。

\framesingle 专门用于制作单个字的方框。

```
zitie/framesingle/before
zitie/framesingle/after
zitie/framezi/before
zitie/framezi/after
zitie/framerange/before
zitie/framerange/after
zitie/framerange/range
zitie/framezifile/before
zitie/framezifile/after
```

这三个命令还各自定义了钩子（hook），其中带有 after 为 reversed，即先添加的后执行。before 为选项设置完成后执行，after 为整个命令执行完之后执行。其中 zitie/framerange/range 为每一个列表均执行一次，包括无效的和最后一次。

可以通过 \AddToHook、\AddToHookNext、\RemoveFromHook 来添加和移除，详细请参考 source2e.pdf[3] 或 lthooks[4] 的说明文档。

```
zitieframe
zitie/zitieframe/before
zitie/zitieframe/after
zitie/zitieframe/par
```

```
\begin{zitieframe} [⟨key-val⟩]
  ...
\end{zitieframe}
\begin{zitieframe} {⟨initial material⟩} [⟨key-val⟩]
  ...
\end{zitieframe}
```

可以在 zitieframe 环境中使用分段（显式的或隐式的，特殊命令仍然不可使用）。

⟨*initial material*⟩ 为在选项和钩子执行完毕之后要执行的内容。这里的内容不会被网格化。

zitie/zitieframe/par 钩子为环境中的每个段落分段后要执行的命令。

\frametallrange
\framezitallfile
zitie/frametallrange/before
zitie/frametallrange/after
zitie/frametallrange/range
zitie/framezitallfile/before
zitie/framezitallfile/after

```
\frametallrange [⟨key-val⟩] {⟨逗号分隔的16进制序列⟩}
\framezitallfile * [⟨key-val⟩] {⟨文件名⟩}
```

一般情况下 zitie 宏包的方框化命令只能容纳不超过 3200 词。但 \frametallrange 和 \framezitallfile 却摆脱了这种限制，即使是 2 万词也不在话下。不过在这种情况下，编译是十分慢的。若要保证速度，可以使用 background 模块及其 xrange、yrange 特性。

TEXhackers note: tall 类型命令隐式地在固定词数后进行分段，这个词数由 tallheight 选项来控制，见第 2 节。

为了执行速度，tallheight 需要手动控制。

第 2 节　选项

以下选项在方框化命令的宏中是局部设置的。

basechar　basechar = ⟨*CJK char*⟩　初始值 = 好
fontsize　fontsize = ⟨*fontsize command*⟩　初始值 = \normalsize
zihao　zihao = ⟨字号⟩

basechar 设置当 resize 为 base 类时的基字符，这个用来计算缩放比例，基字符不同时，即使给定相同的缩放比例，其实际缩放比例也可能不同。

fontsize 设置计算基字符时的字大小，默认值也为 \normalsize。zihao 设置计算缩放比例时的字号大小，必须加载 ctexsize 或 ctex 宏包才可使用。

position　position = ⟨...⟩
position*　position*= {⟨*replace*⟩}
anchor　anchor = ⟨*east*|*southeast*|*south*|*southwest*|*west*|*northwest*|*north*|*northeast*|*center*|...⟩　初始值 = southwest

anchor 设置当前点与方框化盒子的哪个锚点重合。初始值为 southwest，即当前点与盒子的西南角点重合：\framezi[anchor=southwest]{好}，\framezi[anchor=center]{好}。

position 键用于引用 position 处理器定义的值，position 处理器接收当前方框化的盒子作为其参数。使用 xcoffins 宏包，position 处理器还可以对已经构造好的盒子进行任意变换，如旋转、重设大小等，只是最后不要忘记使用 \TypesetCoffin 命令将其输出。该盒子为一个 coffin。关于 coffin 详见 xcoffins[6] 宏包的说明。关于“处理器”，详细介绍请看第 5 节。

position 处理器预先定义了 anchor-east、anchor-south …等规则，实际上，anchor=east 就是 position=anchor-east 的简写，anchor 的其它可选值也是如此。因此，只要 position 的规则具有 anchor-⟨*anchor value*⟩ 形式，就可以使用 anchor=⟨*anchor value*⟩ 来引用它。

还可使用 \zitienewrule 定义新的标点符号处理方式。详见 \zitienewrule 命令的说明和第 5 节。

position* 键将 position 处理器定义为其值，接收当前方框化的盒子作为其参数。

使用 xcoffins 宏包，则设置 position*={\TypesetCoffin#1[l,b](0pt,0pt)} 就相当于设置了 anchor=southwest 或 position=anchor-southwest。

punctuation　punctuation = ⟨*ignore*|*leave*|*onlast*|*scale*|...⟩　初始值 = ignore
punctuation*　punctuation*= {⟨*replace*⟩}

zitie 宏包对字符类型进行区分，对 CJK 字符和标点符号采用不同的处理方式。该选项设置标点符号采用何种处理方式。初始情况下为 ignore，即忽略该字符。

leave 选项把该标点符号原样输出，不对其进行任何处理，字体和其它属性和正文相同。

onlast 选项把标点符号放到 0 宽度的盒子中，并且忽略它的深度和高度，看起来就像是在最后一个字的方框中。

scale 选项把标点符号按最后一个字的缩放倍数进行缩放，再将其进行输出。

选项 punctuation* 定义处理标点符号的方式为其值，可接收一个参数。如设置 punctuation*=\fbox{#1}，则将标点符号放入 \fbox 内。

若 punctuation 并不存在，则将其值视为已定义的宏。如，假设 @gobble 值并不存在，若使用 punctuation=@gobble，则标点符号处理命令为 \@gobble，这是 LaTeX2ε 定义的命令，其作用与 punctuation=ignore 相同。

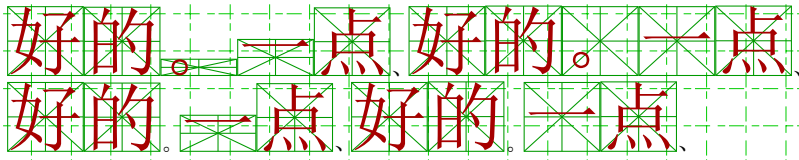
还可使用 \zitienewrule 定义新的标点符号处理方式。详见 \zitienewrule 命令的说明和第 5 节。

实际上，onlast 和 scale 就是由 \zitienewrule 定义的，这个键的处理器接收一个标点符号作为其参数。具体的定义第 5 节。

TEXhackers note: 按照 xeCJK 宏包的字符分类，zitie 宏包将字符类别为 FullLeft、FullRight、HalfLeft、HalfRight 视为标点符号。将 CJK 视为 CJK 字符。

于是可以通过修改标点符号的字符类别为 CJK 来使用 CJK 的处理规则，但是要注意，一般标点符号的宽高与常用字符的宽高并不一致。某些字符也不是全高的，这时，可用后文介绍的 holdbasecharheight 键来保证至少有 basechar 字符的高度。

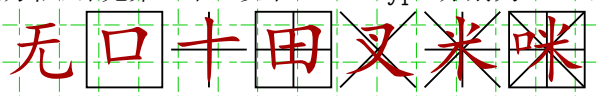
```
\xeCJKDeclareCharClass{CJK}{`。}
\framezi[resize=real,punctuation=leave,holdbasecharheight=false]{好的。一点、}
\framezi[resize=real,punctuation=leave,holdbasecharheight]{好的。一点、}
\xeCJKResetPunctClass
\framezi[resize=real,punctuation=leave,holdbasecharheight=false]{好的。一点、}
\framezi[resize=real,punctuation=leave,holdbasecharheight]{好的。一点、}
```



由于 LuaTEX-ja 的实现问题，目前这一方法在 LuaTEX 中效果并不好。

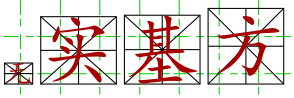
frametype　frametype = ⟨*none*|口|十|田|×|米|咪|...⟩　初始值 = none
resize　resize = ⟨*none*|*real*|*base*|*square*|...⟩　初始值 = none

frametype 设置方框样式。可用值的效果正如选项值：口–仅方框，十–仅中间的横线和竖线，田–常见的田字格，×–斜的两条线，米–十字格再加上斜的两条线，咪–常见的米字格。还可自定义方框，详见第 6 节。以下 frametype 分别为 none，口，十，田，×、米，咪。



resize 设置缩放方式。real，使用字符实际宽高缩放，base，使用 basechar 设置的字符的宽高缩放，square，使得字符的宽高相等再进行缩放。还可自定义缩放方式，详见第 6 节。

以下为宽度设置为 1cm，resize 分别为 none，real，base，square 时的缩放情况。



xscale　xscale = ⟨*scale ratio*⟩　初始值 = 1
yscale　yscale = ⟨*scale ratio*⟩　初始值 = 1
scale　scale = ⟨*scale ratio*⟩
width　width = ⟨*dim*⟩
height　height = ⟨*dim*⟩
holdbasecharheight = **true**|false　初始值 = false
holdbasecharwidth = **true**|false　初始值 = false
holdbasechar = **true**|false

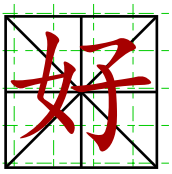
设置缩放比例和盒子宽高。

宽高具有更高的优先级，即若比例和宽高都设置了，则使用宽高来计算。宽高为二者都为 0cm 视为未设置，二者有一大于 0cm，视为设置了宽高。

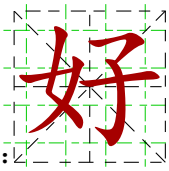
此处的宽高和最终的宽高可能略有差异，最终的宽高保存在 \zitiewidth 和 \zitieheight dim 寄存器中。

holdbasecharheight 设置为 true 时，将保证单个盒子至少有 basechar 的高度，holdbasecharwidth 设置为 true 时，将设置盒子宽度为 basechar 的宽度。holdbasechar 将同时设置二者的值。

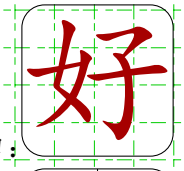
linewidth　linewidth = ⟨*dim*⟩　初始值 = 0.4pt
dashpattern　dashpattern = { ⟨*dim1*⟩, ⟨*dim2*⟩, ... }
framearc　framearc = ⟨*dim*⟩
framearc*　framearc* = {⟨*dim1*⟩} {⟨*dim2*⟩}



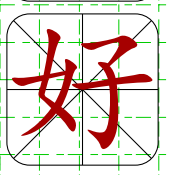
linewidth=1pt:



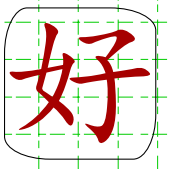
dashpattern={1mm,2mm,3mm,4mm}:



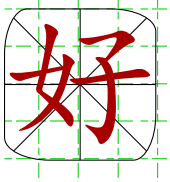
framearc=3mm,frametype=口:



framearc=3mm,frametype=咪:



framearc*={3mm}{6mm},frametype=口:



framearc*={3mm}{6mm},frametype=咪:

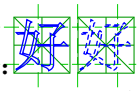
framecolor	framecolor = ⟨*color expr*⟩	初始值 = black
framecolor*	framecolor* = {⟨*models*⟩} {⟨*values*⟩}	
charcolor	charcolor = ⟨*color expr*⟩	初始值 = black
charcolor*	charcolor* = {⟨*models*⟩} {⟨*values*⟩}	
fillcolor	fillcolor = ⟨*color expr*⟩	
fillcolor*	fillcolor* = {⟨*models*⟩} {⟨*values*⟩}	

命名的颜色仅支持 LaTeX3 定义的 black, white, red, green, blue, cyan, magenta 和 yellow。颜色模型和表达式也应使用 LaTeX3 支持的模型和表达式，详见 interface3.pdf[5] 文档。

若要去掉 fillcolor，应将其设置为空，即 fillcolor={}，而不是将其设置为白色。

charstroke
charstrokespecial
\zitiestrokechars

charstroke = ⟨*none*|*solid*|*dashed*|*whitesolid*|*whitedashed*|*thicksolid*|*thickdashed*|*thickwhitesolid*|*thickwhitedashed*|*invisible*|...⟩　初始值 = none
charstrokespecial = ⟨*pdf literal*⟩
\zitiestrokechars {⟨*pdf literal*⟩} {⟨*typeset material*⟩}

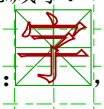
设置字符的外框，类似微软 Word 中字符的轮廓特效： 

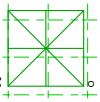
初始值 none 什么也不做。solid 设置外轮廓为 0.25bp 的实线，dashed 设置外轮廓为 0.25bp 的虚线。这两个不会进行填充操作，即背景色是什么颜色，则字显示的就是背景，字轮廓的颜色为 charcolor 的颜色。whitesolid 和 whitedashed 在 solid 和 dashed 的基础上将字填充为白色。带 thick 的值将线宽设为 0.15bp，其它的与不带 thick 的一致。invisible 将字符设置为不可见，但不影响背景和网格的显示，隐藏的字仍然可被复制。

charstrokespecial 将 ⟨*pdf literal*⟩ 为用在 \special{pdf:code ...} 中 pdf 的指令，其中，charstroke=solid 相当于 charstrokespecial={1 Tr 0.25 w [] 0 d 1 J}，charstroke=dashed 相当于 charstrokespecial={1 Tr 0.25 w [0.8] 0 d 1 J}，charstroke=whitesolid 相当于 charstrokespecial={2 Tr 0.25 w [] 0 d 1 J 1 1 1 rg}，charstroke=whitedashed 相当于 charstrokespecial={2 Tr 0.25 w [0.8] 0 d 1 J 1 1 1 rg}。

⟨*pdf literal*⟩ 可用的算符参考 pdf reference[8]。

于是可将字符渲染模式设置为 3 来隐藏字：

\framezi[charstroke=solid]{字}:  ，

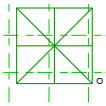
\framezi[charstrokespecial={3 Tr}]{字}:  。

还可使用 \zitienewrule 来定义新的 charstroke 处理规则。这个键的处理器接收当前的 ⟨*CJK char*⟩ 作为其参数。具体的定义见 \zitienewrule 的说明和第 5 节。

\zitiestrokechars 使用 ⟨*pdf literal*⟩ 处理 ⟨*typeset material*⟩。

于是上述隐藏字的效果可用 \zitienewrule 和 \zitiestrokechars 来定义一个 invisible 规则：

```
\zitienewrule{charstroke}{invisible}{\zitiestrokechars{3 Tr}{#1}}
\framezi[width=1cm,charstroke=invisible]{字}
```

 。这与上面的效果是相同的。实际上，zitie 就是这样实现的。

font
savefontname
fallback
fontbackfont
fontbackfont+

font = ⟨*font family*⟩　初始值 = 宋体
savefontname = **true**|false　初始值 = **false**
fallback = **true**|false　初始值 = **false**
fallbackfont = {⟨*fallback font list*⟩}
fallbackfont+= {⟨*fallback font list*⟩}

font 设置盒子中使用的字体，若将其设置为空，即 font={} 或 font，则将使用正文字体。

savefontname 设置是否保存字体名字，若设置为真，则将 font 键指定的字体名字保存在 \zitiefontname 中，以便在后文使用，如在标点符号的处理中或页眉页脚中。

fallbackfont 设置当当前字体中不存在此字形时，要使用哪些字体，字体必须被 \zitienewfont 或 \newCJKfontfamily 声明。fallbackfont+ 将 {⟨*fallback font list*⟩} 局部的添加到之前的 ⟨*fallback font list*⟩ 中。

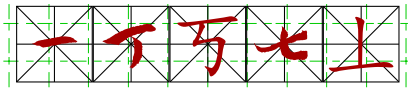
该特性仅可在 XeTeX 中使用。

注意，fallbackfont 特性和 xeCJK 的 FallBack 并不冲突，当 fallback font list 中的所有字体没有对应的字形时，才会使用 FallBack 定义的字体。这一特性可通过设置 fallback=false 来关闭。

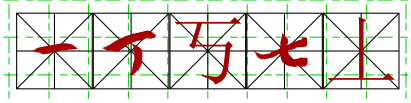
如在 FZSunGuoTingCaoShuFU（方正孙过庭草书）字体中没有“亏上”，虽然在在导言区使用 \zitienewfont{ {孙过庭草繁}{FZSunGuoTingCaoShuFU}{FallBack=SimSun} } 设置其 FallBack 字体为 SimSun（宋体），但若设置 fallbackfont={楷体},fallback=true 则会

在楷体中查找是否有该字形，然后才在宋体中查找：

```
\framezi[fallbackfont={楷体},width=1cm,font=孙过庭草繁,fallback,holdbasecharheight]{一丁丐七上}
```



```
\framezi[width=1cm,font=孙过庭草繁,fallback,holdbasecharheight]{一丁丐七上}
```



filepath　filepath = { ⟨*filepath1*⟩, ⟨*filepath2*⟩, ... }

filepath+

filepath 为 \framezifile* 添加文件搜索路径。其值为逗号分隔的列表。filepath+ 将新的搜索路径局部地添加到原有的路径上。

repeat　repeat = ⟨*integer*⟩　初始值 = 1

break　break = ⟨*default*|*allowbreak*|...⟩　初始值 = allowbreak

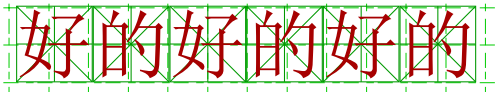
tolerance　tolerance = ⟨*dim*⟩　初始值 = 1em

zitie/repeat

设置 \framezi 等命令中的字符重复次数。注意是将整体重复 ⟨*integer*⟩ 次，而不是将单个字符重复 ⟨*integer*⟩ 次，再进行下一个字符的处理。

zitie/repeat 钩子，只在两两间执行，执行次数为 ⟨*integer*⟩ − 1。这个钩子的内容在执行完毕后不会被清除，因此，若不自行清除，则在不同的方框化命令中都会执行（如果 repeat > 1）。可将 \RemoveFromHook{zitie/repeat}[⟨*label*⟩] 放入 zitie/.../after 钩子来清除。

```
\framezi[width=1cm,repeat=3]{好的}
```



break 设置单个字符后的断行方法。默认为 allowbreak，即将 \allowbreak 插入到构造好的盒子后。tolerance 设置断行的允许值，影响 break 值为 default 时的断行效果，tolerance 大，则可能造成 Overfull。

break 和 tolerance 仍是实验性的。

tallheight　tallheight = ⟨*integer*⟩　初始值 = 1000

tallheight 选项控制 tall 型命令隐式地在 ⟨*integer*⟩ 个词后进行分段。因此，为了保证排版美观，tallheight 最好为每行能容纳词数的倍数。而且这个值也受单个最大能处理的词数的限制，即不能超过 3200。

validateglyph　validateglyph = **true**|false　初始值 = **false**

这个选项设置为真时，将先判断使用的字体是否包含该字形，若不包含，则忽略该字符。判断字形时不会考虑 FallBack 字体。

目前这个选项仅在 \framesingle 命令中有效。

\zitiesetup　\zitiesetup {⟨*key-val*⟩}

\zitiecolorlet　\zitiecolorlet {⟨*name*⟩} [⟨*model*⟩] {⟨*value*⟩}

\zitiesetup 为 zitie 宏包的设置命令，这个命令设置的值在当前组中有效。

\zitiecolorlet 定义新的颜色。注意，若加载 xcolor 宏包，使用 \colorlet 命令定义的颜色无法在 zitie 宏包的选项中使用，必须使用 \zitiecolorlet 来定义颜色。

\zitienewprocessorrule
\zitienewrule
\zitieuseprocessorrule

\zitienewprocessorrule [⟨*arg nums*⟩] {⟨*processor*⟩} {⟨*rule*⟩} {⟨*replace*⟩}
\zitieuseprocessorrule [⟨*arg nums*⟩] {⟨*processor*⟩} {⟨*rule*⟩} ⟨*args*⟩

\zitienewprocessorrule 为 zitie 宏包的处理器 ⟨*processor*⟩ 定义新的处理规则。将忽略 ⟨*option key*⟩ 和 ⟨*rule*⟩ 两端的空格。\zitienewrule 与 \zitienewprocessorrule 的作用相同。

\zitieuseprocessorrule 使用 ⟨*processor*⟩ 的 ⟨*rule*⟩ 规则，⟨*args*⟩ 为 ⟨*processor*⟩ 处理器的参数。

关于处理器和规则，见第 5 节。

⟨*arg nums*⟩ 为键预先定义的参数数目。若未设置 ⟨*arg nums*⟩，则将其设置为 1。

\zitiebasechar
\zitiebasecharwidth
\zitiebasecharheight
\zitiewidth
\zitieheight
\zitieboxwd
\zitieboxht
\zitieboxdp
\zitiefontname
\zitiexscaleratio
\zitieyscaleratio

\zitiebasechar、\zitiebasecharwidth、\zitiebasecharheight 为局部设置的。其余的命令为全局设置的接口。

zitieframecolor
zitiecharcolor
zitiefillcolor

这些是全局保存的颜色名，可在 xcolor 宏包的 \color 命令和 LaTeX3 \color_select:n 命令中使用。

第 3 节　background

在宏包加载时使用 enable-background 以启用该特性。

\zitiebackground

\zitiebackground [⟨*background key-val*⟩]

使用背景。

background 可用的键值：（设置是全局的）

true
on
false
off
next

true 和 on 作用相同，表示设置网格背景。false 和 off 作用相同，表示取消网格背景。next 表示下一次设置背景，即当前页设置背景。这个选项也将取消之前设置的背景。

colboxes
rowboxes
framewidth
frameheight
boxwidth
boxheight
onpaper
ontext

colboxes = ⟨*integer*⟩　初始值 = 1
rowboxes = ⟨*integer*⟩　初始值 = 1
framewidth = ⟨*dim*⟩
frameheight = ⟨*dim*⟩
boxwidth = ⟨*dim*⟩
boxheight = ⟨*dim*⟩
onpaper
ontext

设置背景格子的大小。

boxwidth 和 boxheight 具有更高的优先级，若设置了，则背景每个格子的宽高为所设置的值，否则，使用 framewidth、frameheight 及 colboxes、rowboxes 来计算大小。

onpaper 设置 frameheight=\paperheight, framewidth=\paperwidth，ontext 设置 frameheigh=\textheight, framewidth=\textwidth。

frametype　frametype = ⟨none|口|十|田|×|米|咪|二|||⟩　初始值 = none
linewidth　linewidth = ⟨dim⟩　初始值 = 0.4pt
framecolor　framecolor = ⟨color expr⟩　初始值 = black
framecolor*　framecolor = {⟨models⟩} {⟨values⟩}
fillcolor　fillcolor = ⟨color expr⟩
fillcolor*　fillcolor* = {⟨models⟩} {⟨values⟩}
dashpattern　dashpattern = { ⟨dim1⟩, ⟨dim2⟩, ... }

和 \framezi 等命令的选项作用相同，但二者互不影响。

其中 frametype 的值 二 为画外侧方框和横线，frametype 的值 || 为画外侧方框和竖线。

颜色分别保存在 zitiebackgroundframecolor，zitiebackgroundfillcolor 颜色名中。可在 \color 中直接使用，也可在 \color_select:n 中使用。

xrange　xrange = { ⟨left⟩ , ⟨right⟩ }　初始值 = {0cm, \paperwidth}
yrange　yrange = { ⟨top⟩ , ⟨bottom⟩ }　初始值 = {0cm, \paperheight}

设置网格显示的范围。⟨left⟩、⟨right⟩ 为距页面左端的距离。⟨top⟩、⟨bottom⟩ 为距页面顶端的距离。

也可以使用 \zitiesetup{background={⟨background key-val⟩}} 来设置键。

本文档的设置为：

```
\zitiebackground[
  on, linewidth=0.2pt, dashpattern={1mm,.5mm,2mm,1mm},
  frametype=田, framecolor=green!80!black,
  colboxes=20, rowboxes=30,
]
```

第 4 节　zhlipsum, 中文乱数假文

zitie 宏包完全兼容 zhlipsum 宏包 [7]。使用 \newzhlipsum 命令定义的假文也可使用。

在加载宏包时使用 enable-zhlipsum 以启用该特性。

\framezhlipsum　\framezhlipsum [⟨key-val⟩] {⟨paragraph list⟩} [⟨name⟩]
zitie/framezhlipsum/before
zitie/framezhlipsum/after
zitie/framezhlipsum/paragraph

⟨key-val⟩ 为 \framezi 第 2 节定义的选项，⟨paragraph list⟩ 为段落列表，支持如下形式：6-8,-3,9。⟨name⟩ 为假文的名字。可用值为 simp、trad、nanshanjing、xiangyu、zhufu、aspirin，详细说明参考 zhlipsum 宏包的说明文档。

定义了 zitie/framezhlipsum/before、zitie/framezhlipsum/after、zitie/framezhlipsum/paragraph 三个钩子。其中 zitie/framezhlipsum/paragraph 为假文间要执行的代码，如有 5 段，则在这 5 个段落中间格执行一次，即执行 4 次。其它两个钩子的用法和前述相似。

第 5 节　处理器和规则

处理器实际上就是一个宏，这个宏可以使用指定的规则来处理其参数。

规则可以接收给定的参数，这些参数数目及传入顺序是由对应的处理器指定的。例如 zitie 就定义了 position、punctuation、charstroke 等处理器。它们使用指定的规则对给定参数进行处理。

处理器可以使用 \zitienewprocessorrule 来定义新的规则。zitie 宏包为 position、punctuation、charstroke 等处理器预先定义了一些规则：

使用

```
\zitie_new_process_rule:nnn { position } { anchor-center }
  { \coffin_typeset:Nnnnn #1 { hc } { vc } { 0pt } { 0pt } }
\zitie_new_process_rule:nnn { position } { anchor-east }
  { \coffin_typeset:Nnnnn #1 { r } { vc } { 0pt } { 0pt } }
\zitie_new_process_rule:nnn { position } { anchor-southeast }
  { \coffin_typeset:Nnnnn #1 { r } { b } { 0pt } { 0pt } }
...
```

定义了 position 处理规则:anchor-center、anchor-east 等,这样就可使用 position=anchor-center 来引用它们。

使用

```
\zitienewrule{punctuation}{onlast}
  {\penalty10000 \smash{\makebox[0pt]{%
    \color{zitiecharcolor}\zitieCJKfamily{\zitiefontname}#1}}}
\zitienewrule{punctuation}{scale}{
  \hbox_set:Nn \l_tmpa_box
    { \color_select:n { zitiecharcolor } \zitieCJKfamily{\zitiefontname} #1 }
  \box_scale:Nnn \l_tmpa_box \zitiexscaleratio \zitieyscaleratio
  \box_use_drop:N \l_tmpa_box
}
```

定义了两个 punctuation 处理规则:onlast 和 scale,这样就可以使用 punctuation=onlast 和 punctuation=scale 来对标点符号进行处理。

使用

```
\zitienewrule {charstroke} {solid}
  { \zitiestrokechars { 1 Tr 0.25 w [] 0 d 1 J } {#1} }
\zitienewrule {charstroke} {dashed}
  { \zitiestrokechars { 1 Tr 0.25 w [0.8] 0 d 1 J } {#1} }
\zitienewrule {charstroke} {whitesolid}
  { \zitiestrokechars { 2 Tr 0.25 w [] 0 d 1 J 1 1 1 rg } {#1} }
\zitienewrule {charstroke} {whitedashed}
  { \zitiestrokechars { 2 Tr 0.25 w [0.8] 0 d 1 J 1 1 1 rg } {#1} }
\zitienewrule {charstroke} {thicksolid}
  { \zitiestrokechars { 1 Tr 0.15 w [] 0 d 1 J } {#1} }
\zitienewrule {charstroke} {invisible} { \zitiestrokechars {3 Tr} {#1} }
...
```

定义了 charstroke 处理规则,这样就可以使用 charstroke=solid, ... 来处理字符。

规则的定义中还可以使用 \zitieuseprocessorrule 来引用已经定义好的规则。

第 6 节　编程接口

```
\zitie_new_frame_construct:nn
\zitie_frame_type:n
\zitie_frame_type_c:n
\zitie_frame_type_set_eq:nn
\zitie_new_resize_method:nn
\zitie_resize_method_set_eq:nn
\zitie_new_font:n
\zitie_new_font:nnn
```

创建新的构造器、resize 方法和字体。

```
\zitie_new_process_rule:nnn
\zitie_new_process_rule:nnnn
\zitie_processor:n
\zitie_processor_c:n
\zitie_processor:nnn
\zitie_processor_c:nnn
```

创建和使用处理器。

```
\zitie_color_set:nn
\zitie_color_set:nnn
```

颜色选择。

\zitie_cjk_glyph_use:nN
\zitie_token_class_dispatch:Nnnnn
\zitie_token_class_dispatch_o:Nnnnn
\zitie_token_class_dispatch_f:Nnnnn

字形选择和字符类别判断。仅在 XeTeX 中有效。

\zitie_token_if_punctuation:NTF
\zitie_token_if_punctuation_o:NTF
\zitie_token_if_punctuation_f:NTF

是否为标点符号。仅在 XeTeX 中有效。

\zitie_single_construct:N
\zitie_single_construct:nN
\zitie_single_validate_glyph_construct:N
\zitie_single_validate_glyph_construct:nN

构造单个字的方框。

\zitie_background_new_frame_construct:nn
\zitie_background_frame_type:n
\zitie_background_frame_type_c:n
\zitie_background_frame_type_set_eq:nn

创建和使用新的 background frame 类型。

第 7 节 TODO

- 实现更快的、消耗资源更少的版本。
- 修改断行算法。
- 优化对 LuaTeX 的支持。
- 优化背景模块。
- 增加间距功能。
- 更多网格效果。
- 更多缩放和变换效果。
- 对字符类别进行更多的分类处理。
- 支持透明度，（需要 l3opacity）。
- 等。

参考文献

[1] CTEX.ORG, *xeCJK 宏包*

[2] WILL ROBERTSON, With contributions by Khaled Hosny, Philipp Gesang, Joseph Wright, and others. *The fontspec package*

[3] Johannes Braams, David Carlisle, Alan Jeffrey, Leslie Lamport, Frank Mittelbach, Chris Rowley, Rainer Schöpf, *The LaTeX 2ε Sources*

[4] Frank Mittelbach, *The lthooks package*

[5] The LaTeX Project, *The LaTeX3 Interfaces*

[6] The LaTeX Project, *The xcoffins package*

[7] 曾祥东, *zhlipsum: 中文乱数假文（Lorem ipsum）*

[8] Adobe® Systems Incorporated, *Adobe® Portable Document Format*

版本历史

v1.2 (2021/09/23)
General: charstroke 新增 invisible 值。 6
不再加载 ctexsize 宏包。不再设置宋体为主字体 2
将 \zitie_color_select:.. 改为 \zitie_color_set:..。 10
新增 fontsize 选项，不再使用 zihao 选项作为默认字号选项。 3
新增 holdbasecharwidth、holdbasecharheight、holdbasechar 选项。 5
v1.3 (2021/09/25)
General: 开始支持 LuaTEX。 2
新增 \...set_eq:nn。 10
新增 position、position*、anchor 键。 3
新增 repeat 功能。 7
新增处理器和规则一节。 9
新增带 thick 前缀的值。 6
v1.3.2 (2021/09/26)
General: 单个 range 可达 3200 字。 2
新增 break、tolerance 功能。 7
v1.4.0 (2021/10/01)
General: 修改了方框的行为，使得当使用设置 framearc 时，arc 外的线将隐藏。 5
新增 \framesingle 命令。 2
新增 \zitie_single_validate_glyph_construct:N、\zitie_single_validate_glyph_construct:nN。 . . . 11
新增 tallheight 选项。 7
新增 validateglyph 选项。 8

代码索引

意大利体的数字表示描述对应索引项的页码；带下划线的数字表示定义对应索引项的代码行号；罗马字体的数字表示使用对应索引项的代码行号。