

摇篮文的书写规则

在游戏的 StreamingAssets 文件夹下有一个 localization 文件夹存放了游戏中所有文本的各语言译文。其中名称为 “_”（单个下划线）的子文件夹是日语，里面有一个名为 __tx_whole.txt 的文本文件（其他文件夹下的同内容文件无效）记录了诺艾儿习得新魔法时屏幕上闪过的所有横排摇篮文，（截至 0.26c）如下：

```
/* -正在初始化 [魔法名称? ]- */
-Initializing [&1]-
/* 正在为ID是.....的PCD注册 */
Registering for:
    Personal casting device
Device ID:
    H4NIW4-TON3
/* -个人信息- */
-Personal Info-
/* 姓名、性别、种族、年龄 */
Name: Noel Cornehl
Race: Female/Elf
Tosi: 105
/* 所属、年级 */
Belongs: Bermit Public Univ.
Class: Cadet 3
/* -注意- */
-Notice-
/* 因为败给了一只魔族，她失去了.....她的第二个设备是..... */
She have Lost the PCD[ST3LL4-M4GC]
due to defeat to a monster.
Her 2nd device: [H4NIW4-TON3]
/* -解码中- */
-Decoding Data-
/* -正在检查合格性- */
-Checking conformity-
/* 魔力容量、精神清洁度、同步性、回路脆性、身体等级*/
Mana Capacity: &1 [CLEAR]
Mental Clearance: 304% [CLEAR]
Synchronicity: 405% [CLEAR]
Circuit Fragility: 201% [CLEAR]
Physical Grades: A [CLEAR]
/* -认证进程- */
```

```

-Authentication process-
/* -正在检查一致性- */
-Checking consistency-
/* 开发ID、魔力消耗、咏唱时长、效率、公平性? */
Dev ID: &1
Mana Consume: &2
Cast Time: &3
Efficiency: &4
Fairness: &5
/* -插槽信息- */
-Socket Info-
/* 四个哈希值? */
1. SNTHNTFN
2. U41AK010
3. KA5GHALT
4. 1RTI4NL3
/* 关于为什么这次的演出这么夸张, 有点在乎这件事 */
nande konnnani oogesana
ensyutu nanoka
kininaru tte hanasi
/* 这个世界的魔法是兽人们管理生产的 */
kono sekaino mahou ha
zyuuzin tachi ga kanri
seisan site imasu
/* 愿市民能够安心使用已经确立了安全性的魔法 */
anzen sei ga kakuritsu
sareta mahou wo simin ga
ansin site tsukaeru you ni
/* 出于安全考虑每个魔法认证设备只能使用一次 */
mahou media ha security no
tsugou de ikkai sika
tukaenai youni natteimasu
/* 学生在校习得魔法时必须有和人数相等的登录设备 */
gakkou de seito ga mahou wo
oboeru toki ha ninzuu bunn no
touroku media ga hitsuyou
/* 使用魔法时请小心不要对着路人施放哦 */
mahou wo tsukau tokiha
hito ni mukete utanai youni

```

ki wo tsuke te tsukatte ne

上面的几处百分比看上去很奇怪，这是因为游戏程序中有一个将非负整数转换为摇篮文的函数。该函数的程序代码有一些错误，笔者在理解其意图后修复的结果如下：

```
public static string nel_num(int t, bool zero_empty = false) { // t 为非负整数
    if (t == 0) return !zero_empty ? "." : ""; // 单独的零是句点 "."
    if (t < 10) return t.ToString(); // 一到九就是字符 "1" 到 "9"
    if (t < 100) // 十、百、千、万在内存中分别是 "0", "!", "#", "$"
        return (t < 20 ? "" : (t / 10).ToString()) + "0" + NEL.nel_num(t % 10, true);
    if (t < 1000) // 最终结果像极了整数的日语写法，但最多只支持八位数。
        return (t < 200 ? "" : (t / 100).ToString()) + "!" + NEL.nel_num(t % 100, true);
    if (t < 10000) // 和汉语的区别是所有"零"以及不在个位或万位的"一"都要省略。
        return (t < 2000 ? "" : (t / 1000).ToString()) + "#" + NEL.nel_num(t % 1000, true);
    return NEL.nel_num(t / 10000, true) + "$" + NEL.nel_num(t % 10000, true);
} // 示例：11111111 -> #!01$#!01, 23456789 -> 2#3!405$6#7!809
```

修改前述的 txt 文件，即可在游戏中（按 F7 键热刷新）看到 49 个 ASCII 字符¹，包括 13 个数字（一到十、百千万）、26 个拉丁字母（不区分大小写）、9 个符号以及空格（全角宽度）。

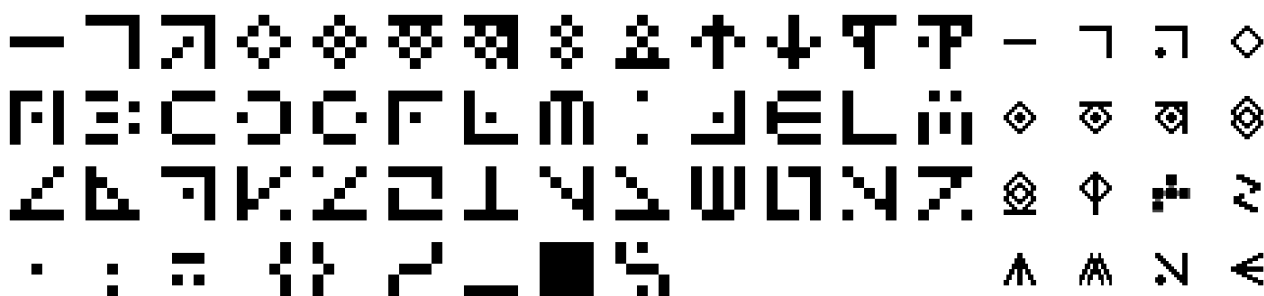
从上述摇篮文中的最后六段话可以看出，日语的罗马字也可以用摇篮文转写，但是拼写规则很不统一。比如同时存在 toshi（とし，年龄）、ensyutu（えんしゅつ，演出）、zyuuzin tachi（じゅうじんたち，兽人们）、kakuritsu（かくりつ，确立）、simin ga ansin site（市民が安心して）、tsugou（つごう，出于）、ninzuu（にんずう，人数）。特别是つかい（使用）一词的变形中 tu 和 tsu 混用，非常混乱。

另一方面，日语用摇篮文转写以后需要用空格分词，は、へ、を会无视读音一律转写为 ha, he, wo，同时还会混入 media 和 security 这样的英文单词，后果是像 site、take、made 这样的词会看不出是哪种语言。








将 StreamingAssets 文件夹下的 Pxl 子文件夹中的 _icons.pxls.dat 文件用 AssetStudio 等第三方软件解包后可以得到一个 _icons.pxls 文件（多余的后缀名如 bytes 要删去），再用游戏作者自研的像素画工具 PixelLiner 打开，在左下角找到名为 nel_character_2 的姿势（posture, pose），就能看到上述 48 个字符（空格除外）的字形²，此时再按下 Ctrl+Shift+P（macOS 系统为 Command+Shift+P）打开保存对话框，将最左边的列数由 4 改为 13，并选中调色板右边的“背景色”勾选框（如果需要透明背景则不选中），最后点击最下面的 OK 就能导出一张 91×28 的图片（下图左侧），如果觉得太小则可在刚才的对话框中将“放大”一栏的 1 改为更大整数。

¹只有 0.20 和 0.21 中“圣光爆发”的习得动画可以反复观看，然而这两个版本中 J 和 K 弄反了。

²如果因为透明背景的色块也有纯白像素而看不清楚，则可按下 Ctrl+3 切换背景色。

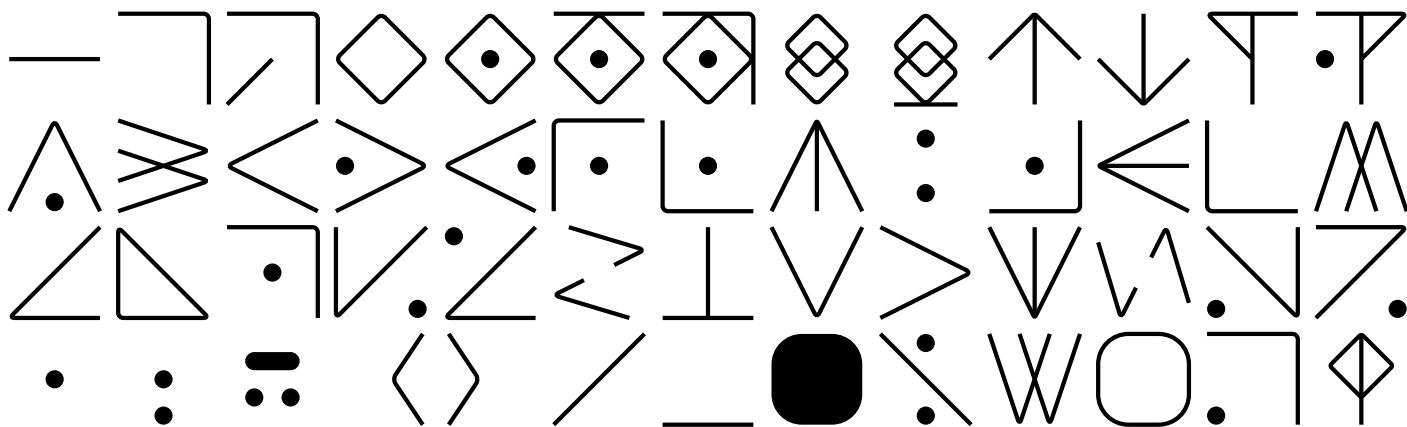


另外还有一个名为 `nel_character` 的姿势，里面虽然只有 16 个图形，但是比起 `nel_character_2` 的 5×5 点阵要更加精细，可同样导出（这次列数建议保持默认的 4）一张 112×112 的图片（上图右侧）。其中后 7 个字符被用于游戏中的大宝箱，它们是：

-  道具。可能是代表法杖的单独记号，也可能就是“十”。
-  强化插件。这也是唯一不是纯色的符号，或许不应该算作摇篮文。
-  技能 (Skill)。
-  生命 (Health) 上限。
-  魔力 (Mana) 上限。
-  埴轮人偶的护符³。该宝箱在 0.23 被移除并在 0.26 改为“替罪猫”道具宝箱。
-  金币 (Kane)。这个单词是日语 **おかね** 的后两个字。

`nel_character_2` 中的 48 个字符被称为摇篮文的像素体，第一行是一到九、十百千万，中间两行是 A 到 Z，最后一行依次是 “.: - □ / _ , %”。可以看出，容易被混淆的有 FGJP 和 QRYZ 这两组。此外和 W 上下对称的居然不是 M 而是 H，需要特别注意。

除了像素体，摇篮文也有印刷体。因为在游戏中并不完整，因此下图（右下角是前述的疑似法杖符号）作为玩家社区推断的结果仅供参考。



对于印刷体和像素体区别的说明如下：

- 数字八和九基于 `nel_character`，诺艾儿的课本上有该字符。
- 十、百、千、万、三（变体和 P 较难区分）基于像素体，其余数字基于 `nel_character`。
- ACDEILNORW 基于素材中的 Alice In Cradle/Wonderland.
- 字母 HKMS 以及法杖符号（第四行末尾）基于 `nel_character`。
- 字母 FGJP 基于像素体，字母 B（游戏中未出现）参考像素体由 M 旋转得到。

³直译是 Haniwa Statue Talisman，并没有字母 Y。

- 字母 Q 和 X（游戏中未出现）参考像素体由 R 和 S 分别绕对角线⁴ 翻转得到。
- 字母 T（游戏中多次出现）基于像素体。
- 字母 U 出现在诺艾儿点餐的菜单上，因此调整为轴对称图形。
- 字母 V（游戏中未出现）参考 U 做了类似的调整，以使这两个字符更适配手写的发力方向⁵。
- 字母 Y 和 Z 在游戏中出现太少（army, grazia），参考像素体由 R 旋转得到⁶。
- 第四行的前八个字符基本上都是像素体把线条拉直后的结果。
- 百分号右边是 W 和 O 的变体，女洗手间的门牌上有这两个字符。
- 法杖符号左边是基于 `nel_character` 的数字“三”的变体，牧场奶牛佩戴的牌子上有该字符。
- 最后，印刷体可以手写（大黑块除外），手写时边角处的圆点可适当向中心偏移以增加辨识度。

为了方便地在 \LaTeX 中输入摇篮文，我自定义了名为 `cradle` 的宏包，使用它后可以直接用命令 `\pixelXX` 或 `\cradleXX` 输入某个摇篮文字符的像素体或印刷体，其中 XX 为字符的 ASCII 码（两位十进制数）。印刷体的特殊字符命令分别是 `\doubleU(W)` `\squareO(O)` `\varthree(.7)` `\cane(Φ)`，本宏包不支持强化插件符号。此外有四个调节字号的命令，它们是（花括号内为默认值，可以修改）：

```
\renewcommand\pixelcm{.72}      % 像素体总宽度(cm)含两侧空白
\renewcommand\pixelratio{1.2}    % 像素体总宽度除以非空白宽度，必须大于1
\renewcommand\cradlecm{.72}     % 印刷体总宽度(cm)含两侧空白
\renewcommand\cradleratio{1.2}  % 印刷体总宽度除以非空白宽度，必须大于1
```

为了进一步的使用方便，可以用以下 JavaScript 代码在浏览器中直接将一个 ASCII 字符串转换成上述命令序列：

```
var f = function (a, isPixel = false) {
  return Array.from(a).map(function (c) {
    if ('(<{' .includes(c)) c = '['; if ('>}' .includes(c)) c = ']'; // 括号一律视为方的
    c = c.codePointAt(0);
    if (97 <= c && c <= 122) c -= 32; // 小写一律视为大写
    if (!(44 <= c && c <= 58 || 65 <= c <= 91 || [33, 35, 36, 37, 93, 95].includes(c)))
      c = 32; // 不存在于 48 个中的字符一律视为空格
    return (isPixel ? '\\pixel' : '\\cradle') + c;
  }).join('');
} // 使用方法：console.log(f('Yin Shua')) 或 console.log(f('Xiang Su', true))
```

例句如下：

```
console.log(f('Ixia and Alma are two female elves'))
: i : A A< > ALMA A< < LV D F< MAL< < L> < Z
console.log(f('Noel Cornehl is their best friend', true))
Z B C L C B Z Z C M L : Q L M C : Z E : C Q L F Z : C Z O
```

⁴也有玩家认为 X 应和 S 互为左右镜像。

⁵现实中的 U 和 V 在几百年前也确实是混用的。

⁶也有玩家认为 Y 和 Z 应该是分别在 U 和 V 的基础上加圆点。