以pq2的事件文本为例

- 解包.3ds游戏文件
- 汉化：
  - 提取游戏文本(.bf, .msg)与字库(.bcfnt)，
  - 字库处理
  - 游戏文本解包与打包
- 打包.3ds游戏

解包.3ds文件

https://github.com/dnasdw/3dstool
目前目录

```
D:.
│   3dstool.exe
│   ignore_3dstool.txt
└───pq2.3ds
```

1. 新建一个名为cci的文件夹，之后提取的文件都放入其中

   ```
   D:.
   │   3dstool.exe
   │   ignore_3dstool.txt
   │   pq2.3ds
   └───cci
   ```

2. 初步提取游戏文件 在终端(cmd / windows terminal/ powershell)输入以下命令。该命令将会提取所有分区到对应的cxi/cfa文件中，也导出文件头ncsdheader.bin。
   虽然命令写了0-7分区，但对pq2来说只有0 1 6 7 分区,提取过程中会提示“2345分区不存在，不创建文件”

   ```
   .\3dstool.exe -xvt01234567f cci cci\0.cxi cci\1.cfa cci\2.cfa cci\3.cfa
   cci\4.cfa cci\5.cfa cci\6.cfa cci\7.cfa .\pq2.3ds --header
   .\cci\ncsdheader.bin
   ```

   ```
   D:.
   │   3dstool.exe
   │   ignore_3dstool.txt
   │   pq2.3ds
   │
   └───cci
       │   0.cxi
   ```

```
    │   1.cfa
    │   6.cfa
    │   7.cfa
    └───ncsdheader.bin
```

3. 大部分游戏文件都位于0.cxi。对于我们来说可以保持其他的cfa文件不动，提取0.cxi的内容 在cci文件夹下创建cxi0文件夹后：

```
.\3dstool.exe -xvtf cxi cci\0.cxi --header cci\cxi0\ncchheader.bin --exh
cci\cxi0\exh.bin --logo cci\cxi0\logo.bcma.lz --plain cci\cxi0\plain.bin --
exefs cci\cxi0\exefs.bin --romfs cci\cxi0\romfs.bin --key0
```

```
D:.
│   3dstool.exe
│   ext_key.txt
│   ignore_3dstool.txt
│   pq2.3ds
│
└───cci
    │   0.cxi
    │   1.cfa
    │   6.cfa
    │   7.cfa
    │   ncsdheader.bin
    │
    └───cxi0
        │   exefs.bin
        │   exh.bin
        │   logo.bcma.lz
        │   ncchheader.bin
        │   plain.bin
        │   romfs.bin
```

4. 提取游戏文件 data.cpk。同样，我们需要的编辑的游戏文件位于romfs.bin，可以不动exefs.bin等，只提取romfs。我们需要的就是romfs文件夹下的data.cpk，接下来就可以转入游戏特定的处理了

```
.\3dstool.exe -xvtf romfs cci\cxi0\romfs.bin --romfs-dir cci\cxi0\romfs
```

```
D:.
│   3dstool.exe
│   ext_key.txt
│   ignore_3dstool.txt
│   pq2.3ds
│
└───cci
```

```
    │   0.cxi
    │   1.cfa
    │   6.cfa
    │   7.cfa
    │   ncsdheader.bin
    │
    └───cxi0
        │   exefs.bin
        │   exh.bin
        │   logo.bcma.lz
        │   ncchheader.bin
        │   plain.bin
        │   romfs.bin
        │
        └───romfs
            │   data.cpk
            │
            └───iWnn
                └───dic
                    └───JA_small
                        │   njcon.a
                        │
                        └───32
                                njexyomi.a
                                njexyomi_new.a
                                njexyomi_re.a
                                njfzk.a
                                njpsq2Memo.a
                                njpsq2Name.a
                                njtan.a
                                njubase1.a
                                njubase2.a
```

汉化游戏文本

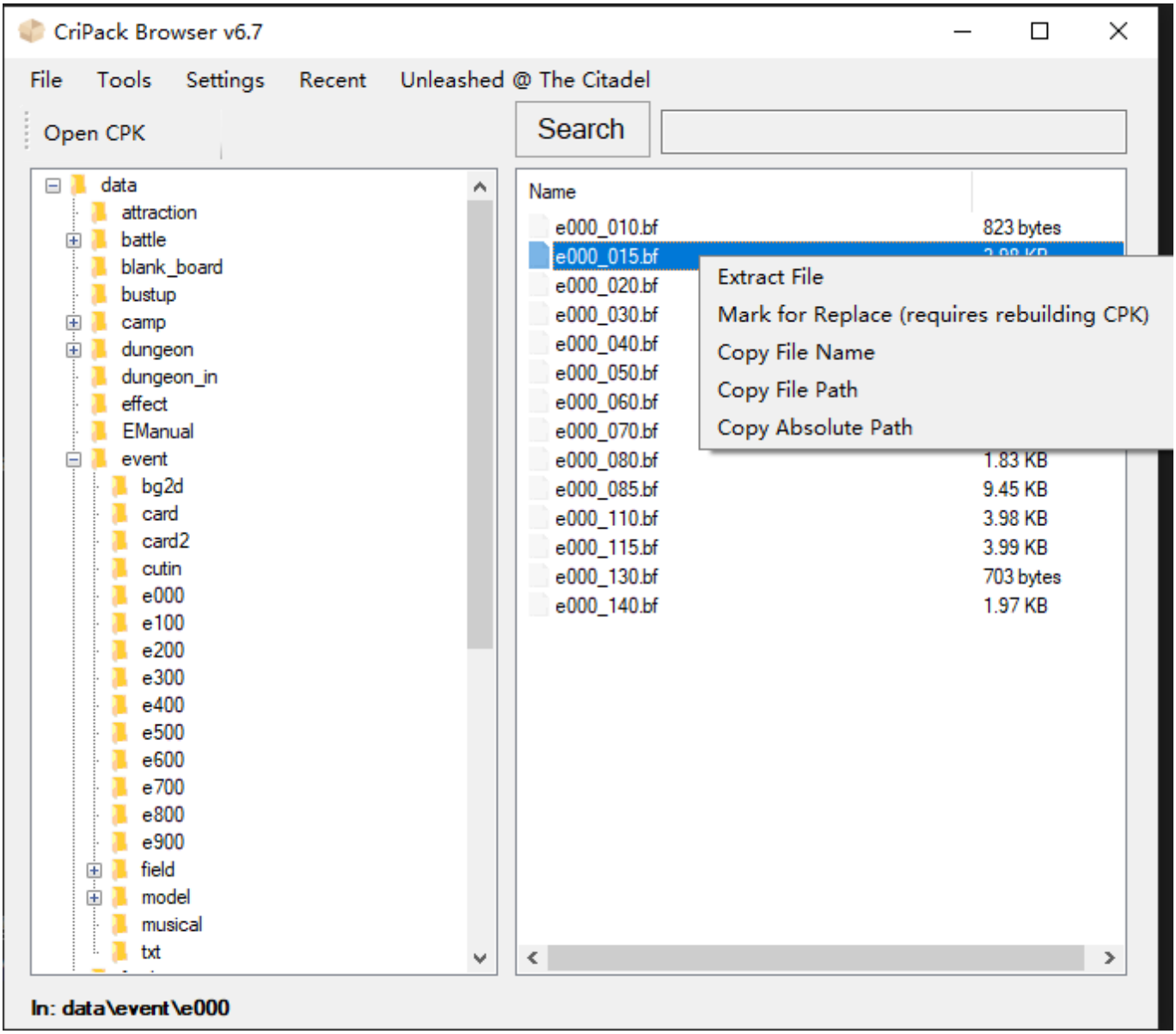
上一部分我们得到了data.cpk，这是游戏文件的打包，语音、模型、文本等等等都在里面。其中我们要改动的是： data/event/e四位数字/以.bf结尾的文件：控制游戏事件的进行，文本也在其中 font/以bcfnt结尾的文件：字库

解包cpk

有非常多的工具可以做到，这里用带图形界面的"Cpk-Browser-View-and-Extract-Files"，可以用窗口形式展示data.cpk的内容，以及替换某个文件然后重新打包cpk。但对批量操作并不是很友好，例如要汉化所有事件文本，需要逐个点击替换，仅仅事件文本就有935个。这里只是用该软件作为实例。其他的cpk处理软件可以在以下链接找到
https://amicitia.miraheze.org/wiki/CPK

以修改pq2开头主角醒来的事件，打开CPKBrowser.exe，选择“OPEN CPK”，选择前文得到的data.cpk。之后软件会列出cpk的内容 找到font/seurapro_12_12.bcfnt （另有seurapro_13_13.bcfnt，数字应该代表字号，但测试中开头事件用的是seurapro_12_12.bcfnt），右键“Extract File”。文件会被提取到与data.cpk同目录的文件夹下。

同样找到data\event\e000\e000_015.bf ，右键提取



字库bcfnt处理

https://github.com/dnasdw/3dsfont https://github.com/dnasdw/3dsfont/tree/master/bin/ctr_FontConverter

1. 提取字库中的字符集、字符列表(xllt)、字符顺序(xlor)

```
./bcfnt2charset.exe seurapro_12_12.bcfnt charset.txt
```

charset.txt 中是字库所拥有的所有文字,每行16个字符（空格看不见，但也是一个字符），内容会类似：

```
!"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~。
```

2. 接着生成字符列表xllt，该文件表示字库应该拥有哪些字符，这个在后续使用字数比较多的字体反过来生成字库图片的时候，可以起到筛选的作用。

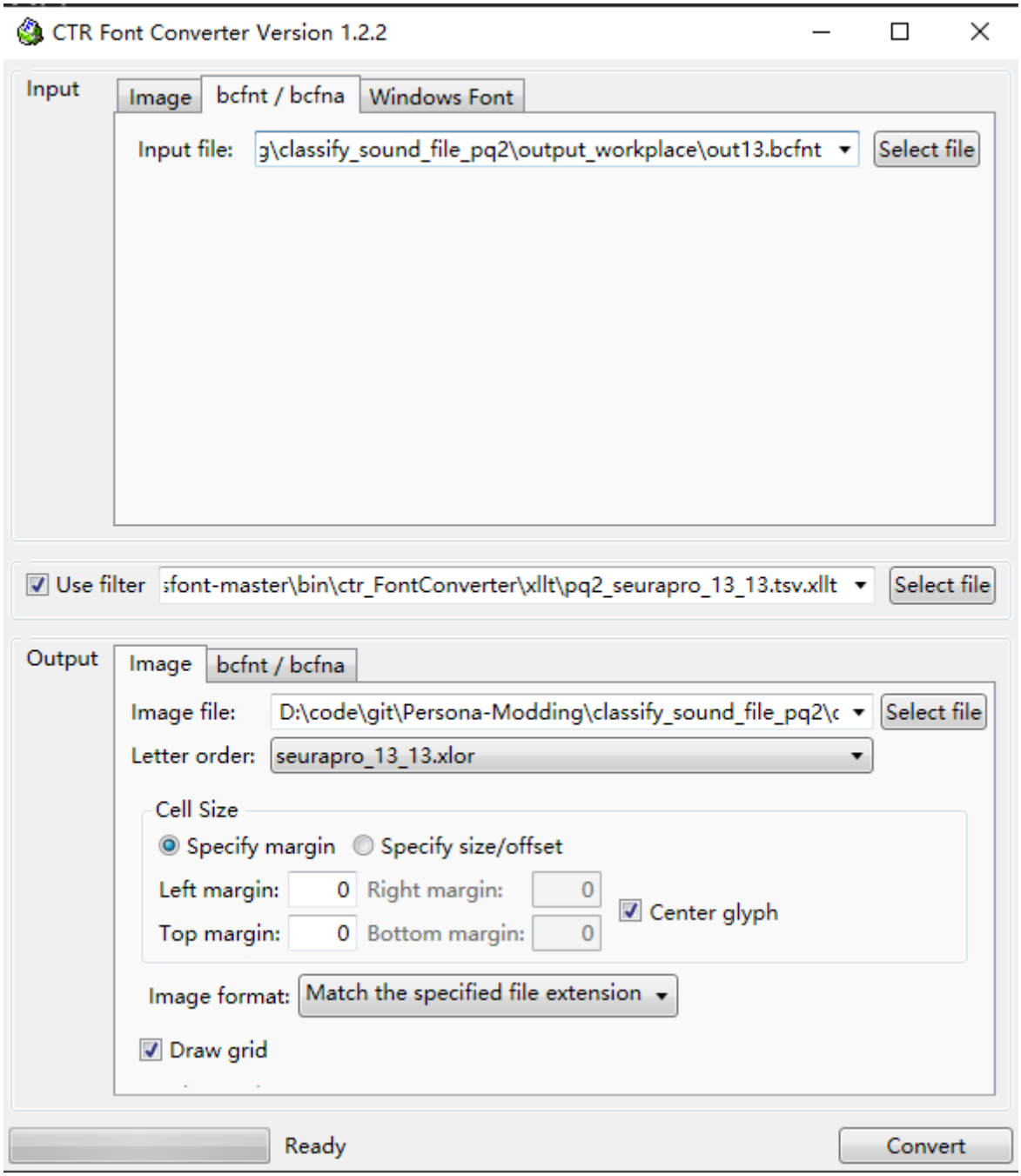
```
./charset2xllt.exe charset.txt font.xllt
```

3. 接着生成字符列表xlor，该文件表示字库中字符的顺序，表示字库图片对应的字的顺序

```
./charset2xlor.exe charset.txt font.xlor
```

4. 虽然不是必要流程，但可以从字库中导出字库图片，感受一下到底每个字是如何对应字符的图片
把上一步得到的font.xllt复制一份到ctr_FontConverter.exe所在文件夹的xllt文件夹，font.xlor复制到xlor文件夹

打开ctr_FontConverter.exe



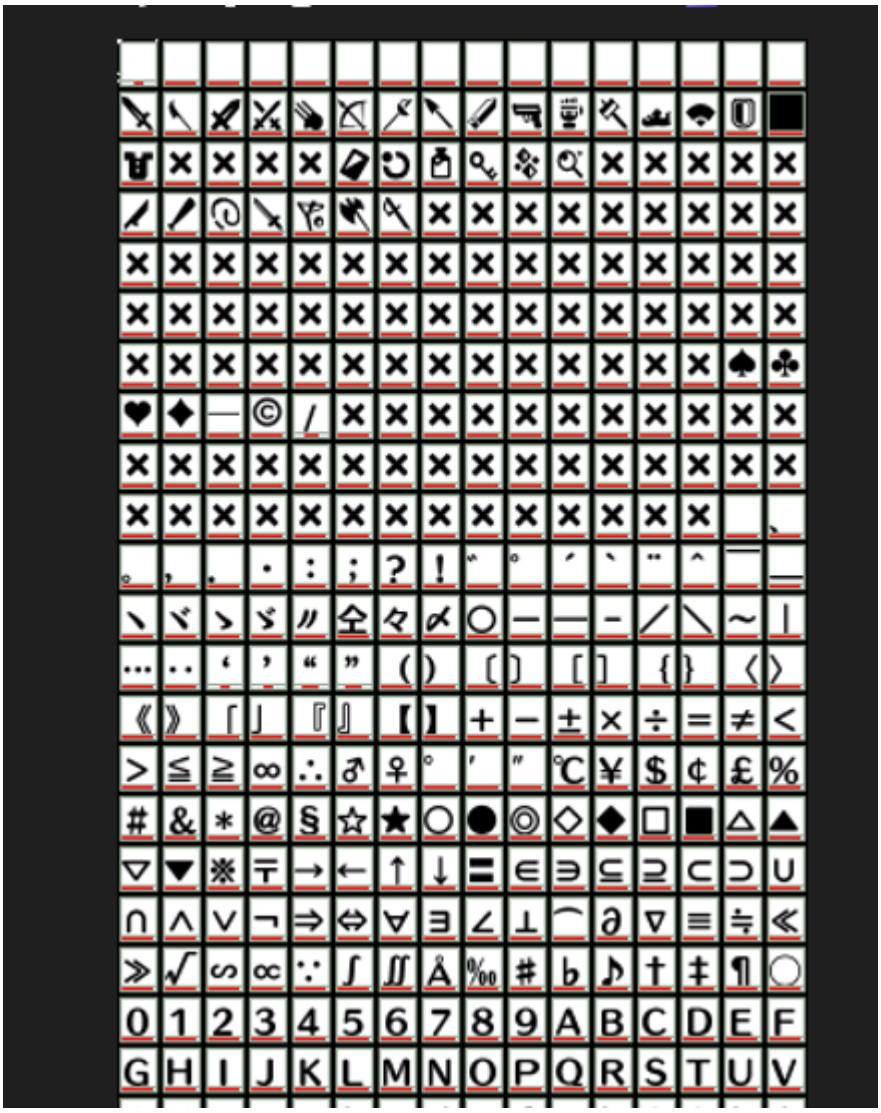
input 部分：
切换到bcfnt/bcfna标签，点击select file选择seurapro_12_12.bcfnt

Use filter：
选择之前放进去xllt文件夹的font.xllt（实际上这里应该用不到，试试也无所谓）

output部分：
切换到Image标签，Image file栏点击select file选择好输出图片的位置。

Letter order 选择之前放进去的seurapro_12_12.xlor（这个名字可能跟你原本的font.xlor）不一样，因为是在文件内部的。其他的保持默认也没关系。
点击conver，会得到看起来像这样的图片：
（这里截图是操作seurapro13_13时截取的，换成seurapro12_12就好）

接下来先告一段落，进入游戏文本提取



游戏文本提取

我们先前得到的e000_015.bf是a社特有的文件打包格式，可以使用atlus script tools中的AtlusScriptCompiler来解包:
https://github.com/tge-was-taken/Atlus-Script-Tools
AtlusScriptCompiler.exe 和 e000_015.bf的路径要注意替换成自己的具体路径

```
AtlusScriptCompiler.exe .\e000_015.bf --Decompile -Library pq2 -Encoding SJ
```

运行完毕后应该会得到3个文件：

- e000_015.bf.flow 事件流程控制脚本
- e000_015.bf.msg 事件文本（含文本格式化符号）
- e000_015.bf.msg.h 事件文本头文件

我们只需要关注e000_015.bf.msg就好了，打开之后看起来是这样：

```
......
[msg MSG_002 [モルガナ]]
[f 0 5 65278][f 2 1][f 3 1 1 0 0 3][f 6 1 3 0 0 1]お目覚めか。[n]授業、とっくに終わったぜ？
```

```
[n][f 1 3 65535][f 1 1][e]
......
```

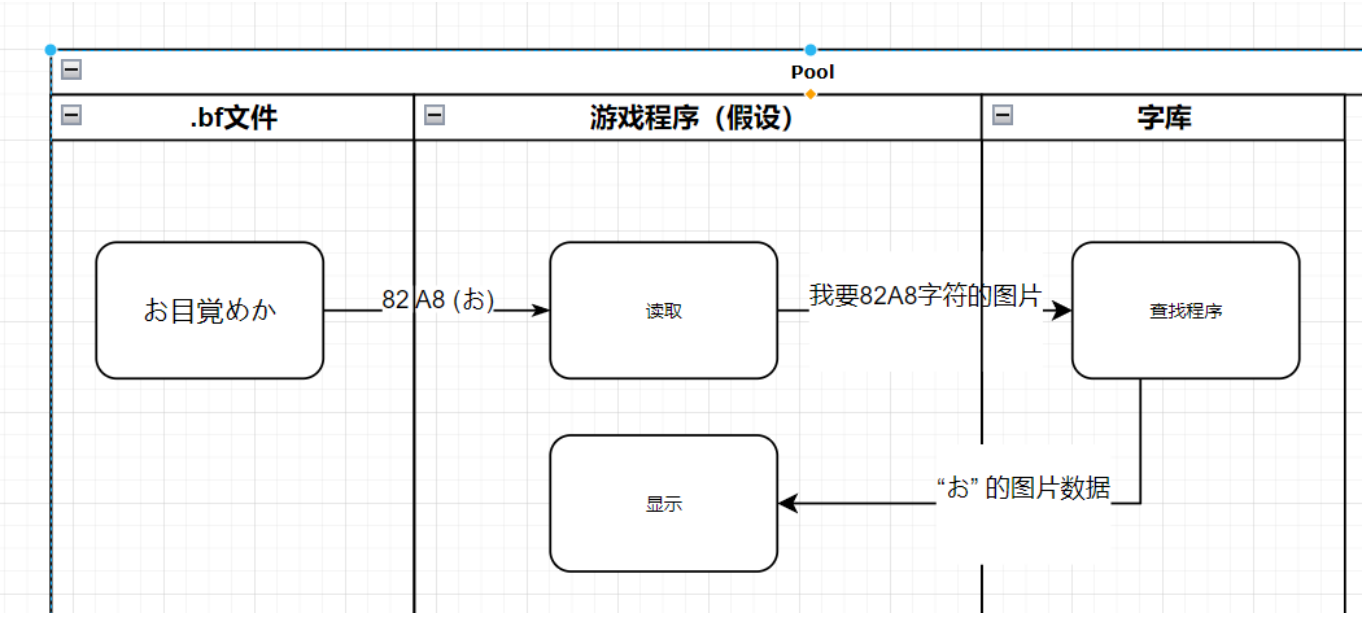
中括号括住的表示一些控制，例如[f 3 1 1 0 0 3]表示调用事件内编号为3的语音，[n]表示换行。基本上除了[n]之外可以都不改动。

汉化到底要做什么？

经过前面的操作，我们已经获得了字库和具体的文本，那么到底要怎么汉化呢？
直接把文本改成中文是不行的，先来讲讲编码

编码

一套对应关系。以拼音为例，利用 'b' 'i' 'a' 'n' ，可以找到'变' '边' '编' 等等等一系列的字符，加上音调可以进一步筛选。 那游戏是怎么找到字的呢？pq2使用一种名为shift jis的字符编码，"お目覚めか"在.bf文件内存储的时候，实际上存储的是"82 A8 96 DA 8A 6F 82 DF 82 A9"(十六进制表示)， 82 A8 表示お。然后向字库询问82 A8对应的图片数据，字库程序返回对应的数据用于显示:



虽然也幻想过：要是bcfnt能够支持自定义编码，那整个流程来说可以加不少字，但这个坑实在是有点深。
还是使用 游戏光辉物语汉化项目 分享的一种办法： https://github.com/hoothin/RadiantHistoriaHans

在上面的流程图中，如果我们构建bcfnt字库的时候，让"お"的图片数据实际上是某个中文字符例如"哦"，再利用这个"お - 哦"的对应关系。我们可以建立一套新的对应关系。当我们希望文本用到"哦"字，在打包的时候将其替换为"お"。上面的显示流程中，读取.bf以及查询字库不变，只是返回的图片数据被我们替换了。

构建字库

shift jis汉字的范围：

> https://baike.baidu.com/item/Shift_JIS/2206614
> 第一字节 使用0x81-0x9F、0xE0-0xEF (共47个)
> 第二字节 使用0x40-0x7E、0x80-0xFC (共188个)

也就是总的字数 47*188*16个，最好在翻译完成后，统计用到了哪些字，再把那些字安排到字库中，没必要放置太多字符，使得字库文件过大，可能会导致游戏崩溃。并且原本pq2使用的字库汉字容量在 3200 左右，还是比较大的了。至于要如何安排就自行决定了。 这里我们简化，只想办法显示出一行中文文本。

```
目标：e999_015.bf.msg的第十行：
[f 0 5 65278][f 2 1][f 3 1 1 0 0 3][f 6 1 3 0 0 1]お目覚めか。[n]授業、とっくに終わったぜ？
[n][f 1 3 65535][f 1 1][e]
```

机翻后：我们（目前）需要的字有：“你 醒 已 经 下 课 了”。

1. 决定替换哪些文本
   在charset.txt的第48行中，我们对文字进行如下替换：第一行为原本的文字，第二行为替换后对应中文汉字。假设你用某种自动化方式得到了这个所有 日文-中文字 对应关系的文件，这里我们命名为 charset_jp_zhsc_map.txt

   ```
   (这里失误了，其实不需要两个"了")
   唖　娃　阿　哀　愛　挨　姶　逢　葵　茜　穐　悪
   你　醒　了　已　经　下　课　了　吗　电　影　院
   ```

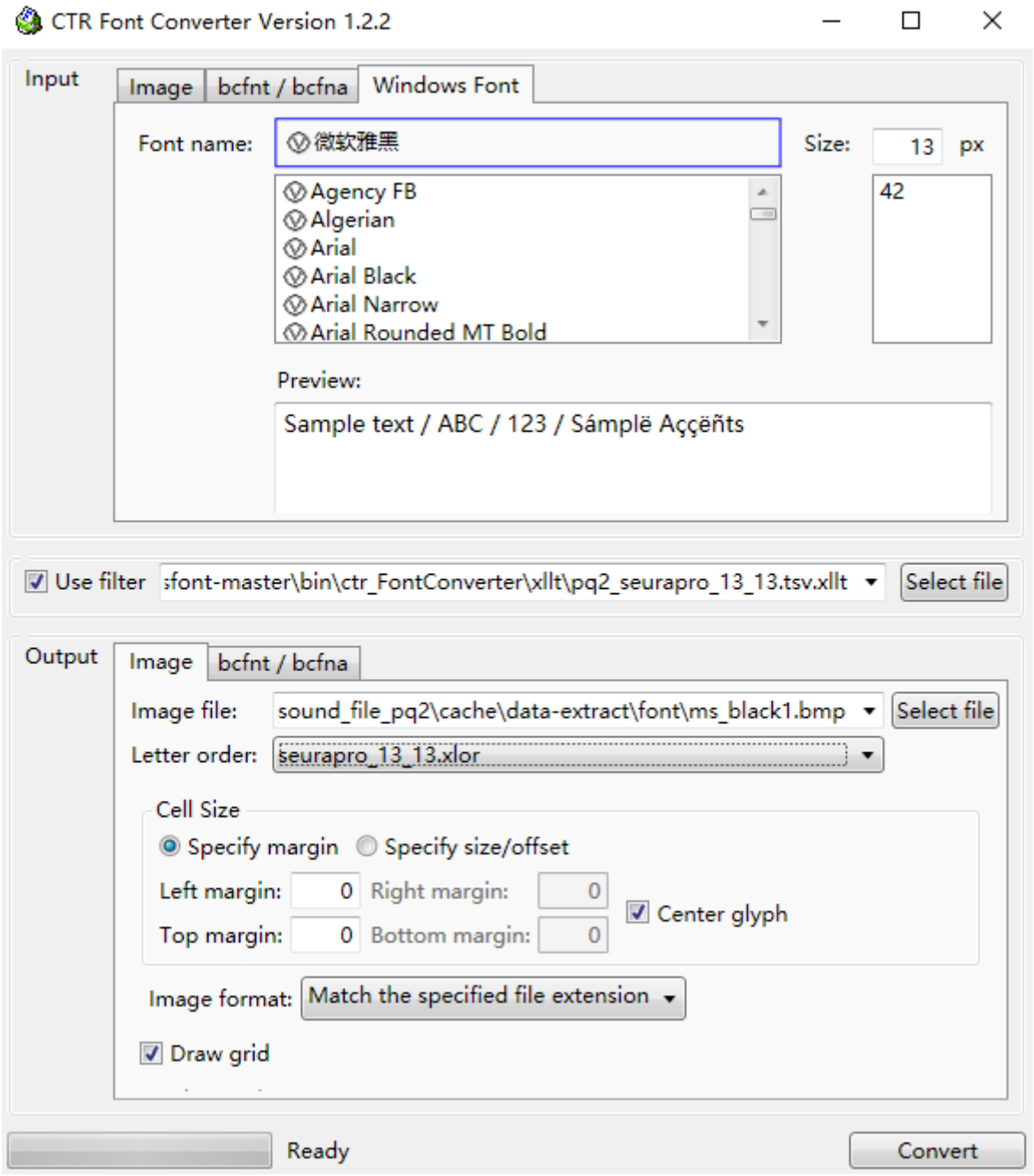
2. 生成中文汉字的字库图片
   跟之前导出xllt xlor操作类似，但新生成的文件要跟原本的区分开来，这里我们命名加上-zhsc后缀

   ```
   ./charset2xllt.exe charset.txt font-zhsc.xllt
   ./charset2xlor.exe charset.txt font-zhsc.xlor
   ```

   新生成的xllt和xlor文件也移动到ctr_FontConverter.exe的xllt,xlor文件夹。为了避免之后命名区分不了，你可以先把原本的font.xllt等移走。这里我们整理一下目前用到的文件:

   ```
   charset.txt 已经替换了中文进去的字符集
   font-zhsc.xllt 已经替换了中文的xllt
   font-zhsc.xlor 已经替换了中文的xlor
   font.xllt 原本日文的xllt
   font.xlor 原本日文的xlor
   ```

3. 生成中文的字符集图片 打开ctr_FontConverter.exe,



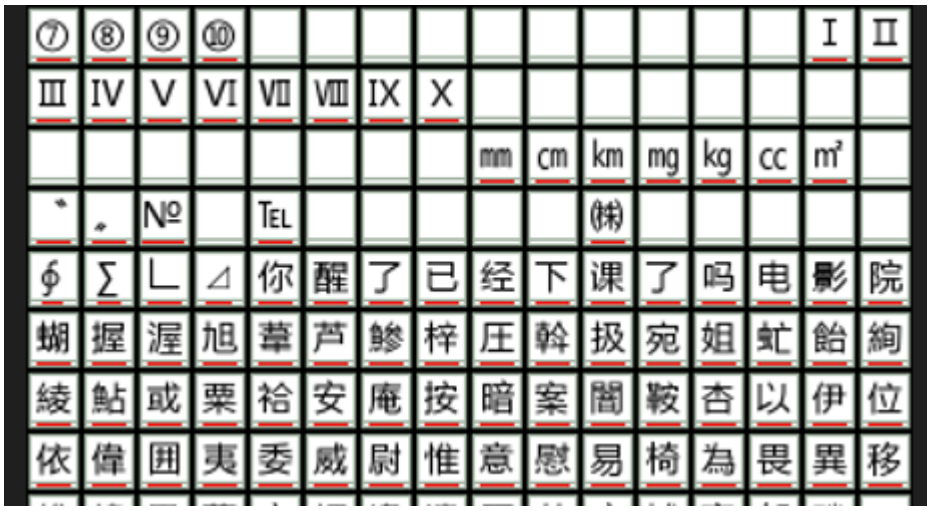
（这里截图是操作seurapro13_13时截取的，换成seurapro12_12就好）

input部分：
切到windows font标签，这里以微软雅黑为例，字号的话似乎并不是跟 seurapro_12_12 的文件名对应，也许是游戏里计量单位不同，姑且打上13。

Use Filter部分：
选择font-zhsc.xllt，免得多余的字符生成到字符图片中。（但其实我感觉xlor文件已经可以处理完，也许是没自定义到那种程度）

output部分：
切到image标签，设置跟之前一样，你想输出的图片路径等等。如果这里Letter order你发现你的选项里有两个一样的 seurapro_12_12.xlor ，可以去xlor文件夹移走日文的font.xlor

完成之后点击conver，能够得到生成的实际上是中文字符的图片：
这一步或者之后的步骤可能会提示某些字符无法编码，就看你的字是不是被你选择的字体所支持了。



4. 生成实际上是中文图片的字库文件bcfnt
   先把xlor、xllt文件夹里的font-zhsc.xllt、font-zhsc.xlor移走，换回font.xllt、font.xllt 打开ctr_FontConverter.exe
   input部分：
   切到Image标签，Image file选择上一步生成的中文的字符集图片。Letter order选择seurapro_12_12.xlor。其他设置Color format可以不动（其实我也不懂到底是干什么的）

   Use Filter部分：
   选择font.xllt，免得多余的字符生成到字符图片中。

   output部分：
   切到bcfnt/bcfna标签，Output file选择你要输出的bcfnt文件的路径。Character栏里的Character encoding选择shift_jis，其他保持不动也可以（进游戏如果发现字符偏移之类的可以在这调整或者生成图片的时候调整）。

   convert完成之后我们就得到了一份能够按日文搜索得到实际上是中文汉字显示数据的字库，这里我们命名为：seurapro_12_12_zhsc.bcfnt

打包.bf

前文提到字库是"按日文去搜索得到中文"的，通过 charset_jp_zhsc_map.txt，我们可以知道某个中文字对应的日文。 当翻译人员把游戏文本比如e000_015.bf.msg翻译完之后，文件也许是：

```
(e000_015.bf.msg的第10行)
[msg MSG_002 [モルガナ]]
[f 0 5 65278][f 2 1][f 3 1 1 0 0 3][f 6 1 3 0 0 1]你醒了已经下课了吗电影院[n][f 1 3
65535][f 1 1][e]
```

应该有某个扫描程序，能够扫描所有msg文件并把中文字符替换为对应的日文字符，比如替换后的e000_015.bf.msg：

```
[msg MSG_002 [モルガナ]]
[f 0 5 65278][f 2 1][f 3 1 1 0 0 3][f 6 1 3 0 0 1]唖娃阿哀愛挨姶逢葵茜穐悪[n][f 1 3
65535][f 1 1][e]
```

然后使用AtlusScriptCompiler打包回成bf，这里-Encoding SJ 其实就是指定编码为shift jis（实际上源码用的是Windows cp932，有更多字符）。同样注意将文件路径换成自己的。

```
AtlusScriptCompiler.exe e000_015.bf.flow e000_015.bf.msg e000_015.bf.msg.h -
Compile -OutFormat V2 -Library pq2 -Encoding SJ
```

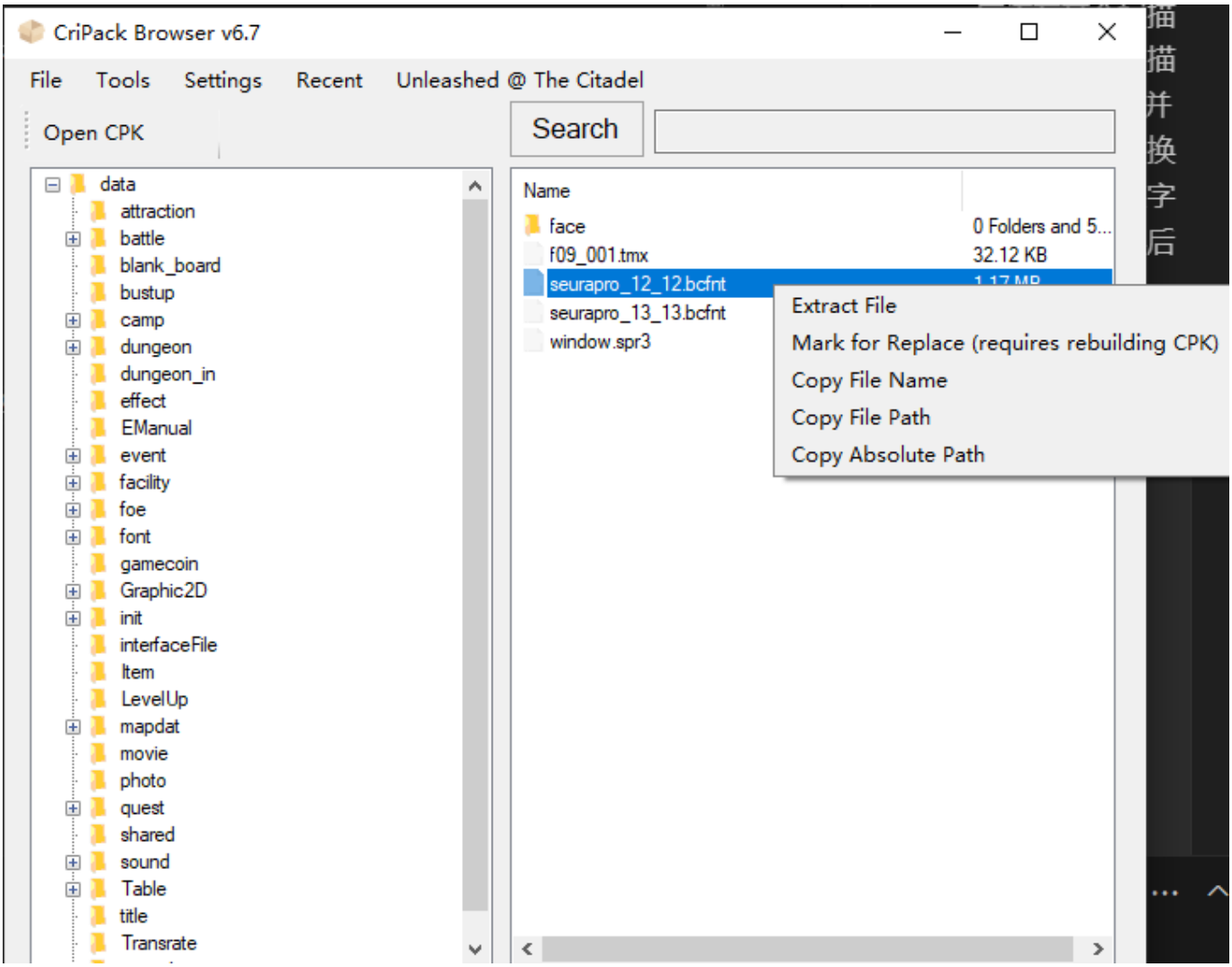
运行结束之后应该会得到e000_015.bf.flow.bf 文件，下一步该把中文字库以及汉化了的文本打包回游戏文件了。

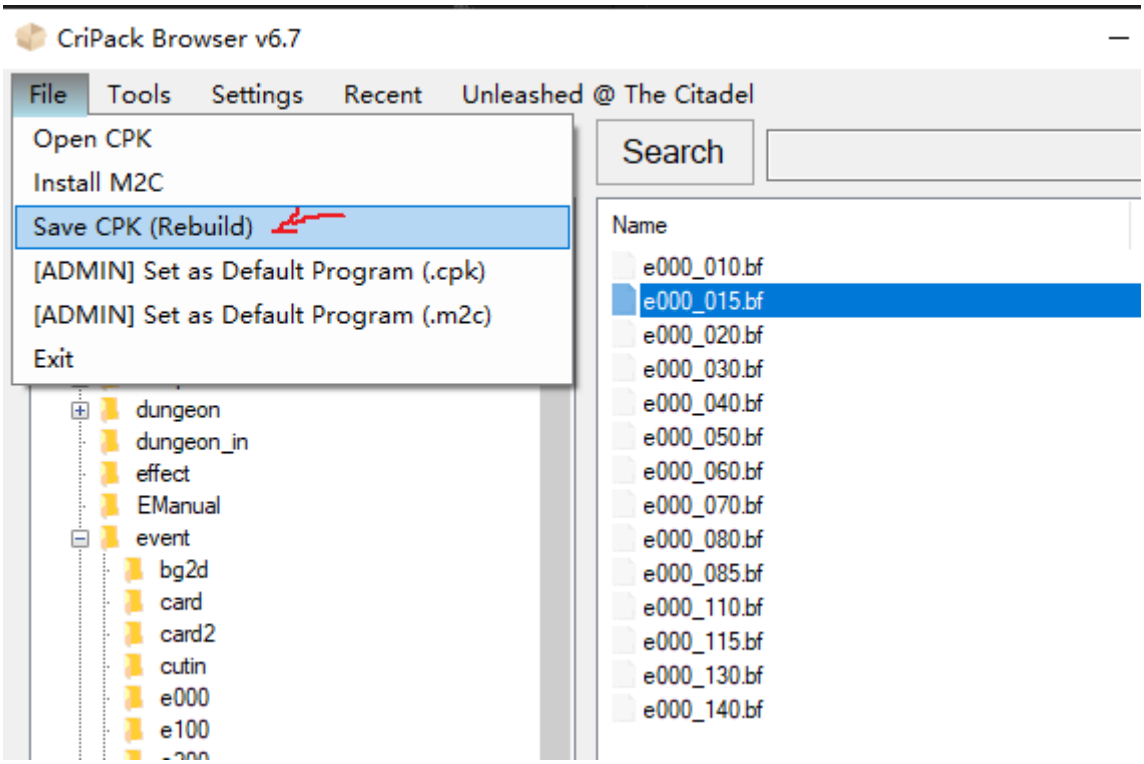
打包data.cpk

在前面的步骤中，我们得到了显示数据为中文的seurapro_12_12_zhsc.bcfnt 和 “汉化”了的游戏文本 e000_015.bf.flow.bf

打开CPKBrowser.exe，（最好先备份一下data.cpk）打开data.cpk，跟解包data.cpk的时候类似。找到data\font\seurapro_12_12.bcfnt，右键选择Mark for replace...。会弹窗让你选择文件，就选择seurapro_12_12_zhsc.bcfnt。同样找到data\event\e000\e000_015.bf，右键替换。

替换完之后，选择顶部栏的FIle-Save CPK，软件会将替换保存到CPK（不会新建一个data.cpk，所以注意备份）





打包.3ds(.cxi)游戏

将打包完成的data.cpk替换掉第一部分中解包时的文件夹:
3dstool/cci/cxi0/romfs/data.cpk

接下来就是把解包过程逆过来：

1. 打包romfs

```
.\3dstool.exe -cvtf romfs cci\cxi0\romfs.bin --romfs-dir cci\cxi0\romfs
```

2. 打包 0.cxi，其实到这一一步得到的cxi就可以用citra加载了。 --not-encrypt 表示不加密

```
.\3dstool.exe -cvtf cxi cci\0.cxi --header cci\cxi0\ncchheader.bin --exh
cci\cxi0\exh.bin --logo cci\cxi0\logo.bcma.lz --plain cci\cxi0\plain.bin --
exefs cci\cxi0\exefs.bin --romfs cci\cxi0\romfs.bin --not-encrypt
```

3. 打包 3ds

```
.\3dstool.exe -cvt0167f cci cci\0.cxi cci\1.cfa cci\6.cfa cci\7.cfa
repack.3ds --header cci\ncsdheader.bin
```

最终，我们会得到一个repack.3ds文件。

链接

3ds游戏解包操作教程
http://www.k73.com/3ds/120261.html

3ds游戏汉化教程篇之打包操作
http://www.k73.com/3ds/120205.html

3ds游戏汉化教程之文本字库操作篇
http://www.k73.com/3ds/120266.html

3DS光辉物语汉化笔记 https://github.com/hoothin/RadiantHistoriaHans

3dstool
https://github.com/dnasdw/3dstool

ctr_FontConverter
https://github.com/dnasdw/3dsfont/tree/master/bin

3dsfont
https://github.com/dnasdw/3dsfont

AtlusScriptCompiler
https://github.com/tge-was-taken/Atlus-Script-Tools

Amicitia Wiki (女神异闻录系列mod wiki)
https://amicitia.miraheze.org/wiki/CPK