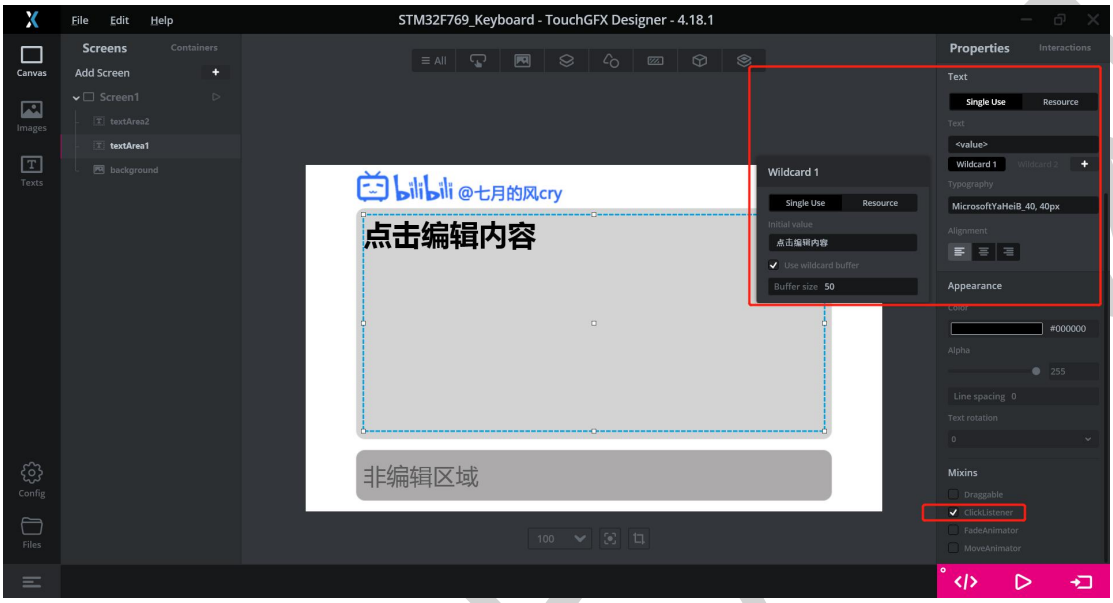
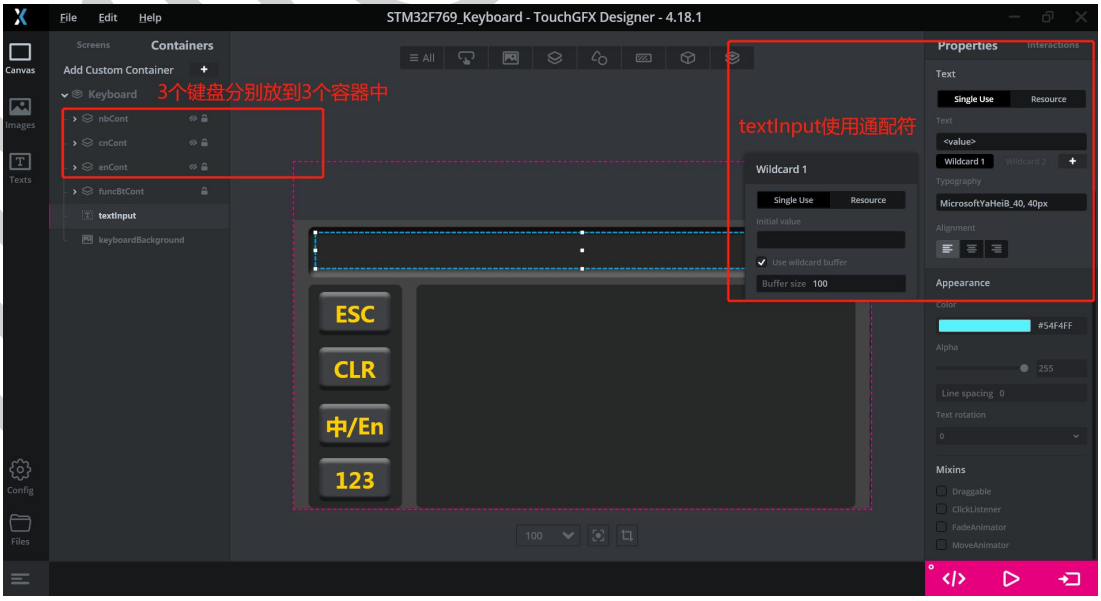
1. 在主界面放入两个 textArea 控件，textArea1 为可编辑内容，需要使能 ClickListener 监控 textArea1 的点击动作；并且要使用通配符缓存，显示修改文本显示的内容的功能。

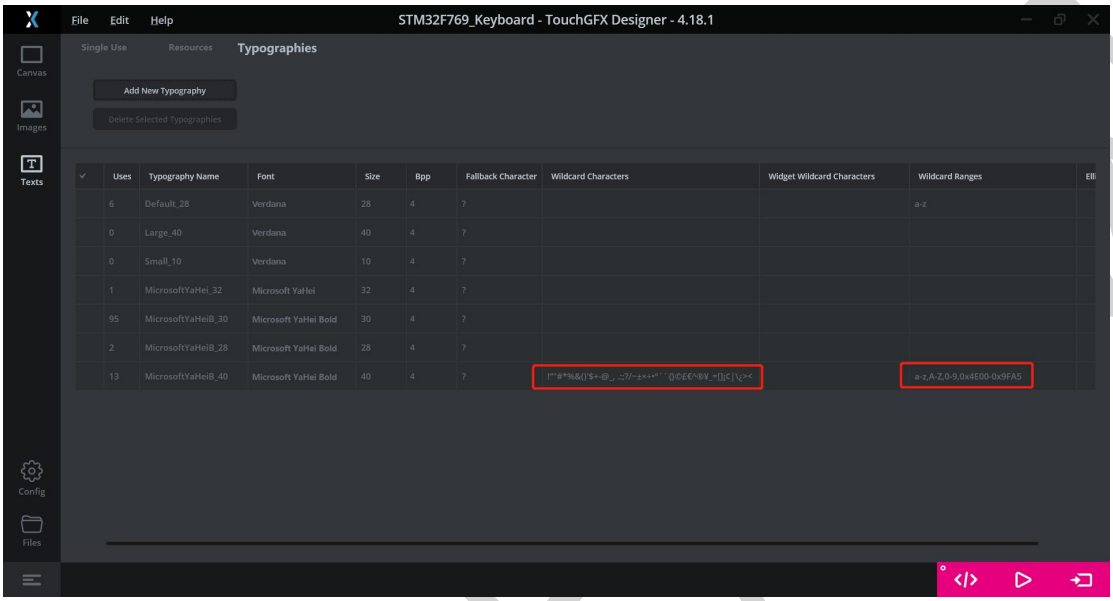


2. 键盘为用户自定义的控件，需要我们自己区实现，我这里将数字、中文、英文键盘分别放到 3 个容器中，方便单独显示；同时，键盘的输入文本 textInput 同样需要使用通配符。

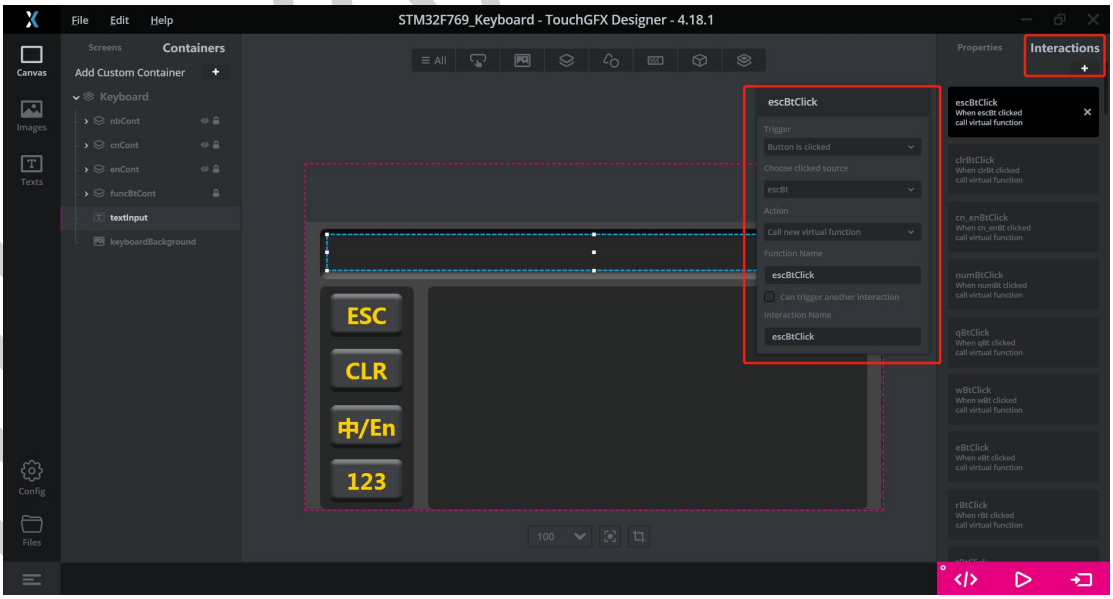
可以理解为，动态文本控件(textArea)，要想改变其显示内容，就必须要使用通配符。



3. 设置常用字符以及常用中文的 Unicode 编码范围，就是制作一个字库。只有设置了 Unicode 编码的字体，textArea 控件在使用通配符是才能正常显示中文，否则会乱码的。



4. 添加键盘按钮点击回调，up 主比较菜没能看懂官方的键盘映射，所以直接使用了按钮的动作回调。3 个类型的键盘按钮加起来还是比较多的，所以这一步比较繁琐。



5. 代码撸起来。

6. 主页 textArea1 控件使能了 ClickListener，我们要实现 textArea1 点击动作；textArea2 没有使能 ClickListener，是不会识别点击动作的。撸代码的时候可以看到他们区别，textArea1 继承了 ClickListener 并且可以调用 setClickAction 方法，而 textArea2 不会。

```
void Screen1View::setupScreen()
{
    Screen1ViewBase::setupScreen();

    KeyboardExist = 0;

    textArea1.setClickAction(textArea1ClickedCallback);

    MyKeyboard.setXY(0, 80);
    MyKeyboard.setescBtCallback(KeyboardEscBtClickedCallBack);
    MyKeyboard.seten_entBtrCallback(KeyboardEnEnterBtClickedCallBack);
    MyKeyboard.setnb_entBtrCallback(KeyboardNbEnterBtClickedCallBack);
    MyKeyboard.setcn_entBtrCallback(KeyboardCnEnterBtClickedCallBack);

    textArea1.invalidate();
}
```

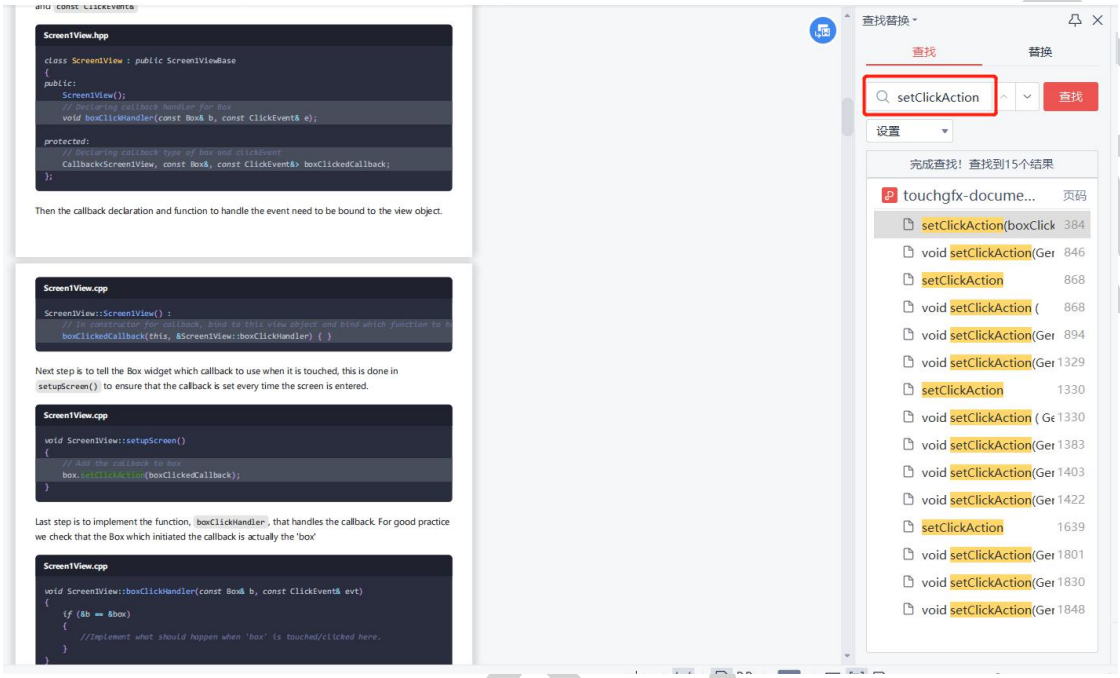
7. 实现 textArea1 点击回调：

```
Screen1View.cpp | Screen1View.hpp | Screen1ViewBase.hpp | Keyboard.cpp | Keyboard.hpp | Pinyin.cpp
Application | Screen1View
15
16      /*声明textArea1回调处理函数*/
17      void textArea1ClickHandler(const TextAreaWithOneWildcard& text, const ClickEvent& event);
18
19      void KeyboardEscBtClickHandler(Keyboard& element);
20      void KeyboardEnEnterBtClickHandler(Keyboard& element);
21      void KeyboardNbEnterBtClickHandler(Keyboard& element);
22      void KeyboardCnEnterBtClickHandler(Keyboard& element);
23
24  protected:
25
26      /*声明TextArea1点击回调*/
27      Callback<Screen1View, const TextAreaWithOneWildcard&,const ClickEvent&> textArea1ClickedCallback;
28
29      /*声明Keyboard控件及其相关按钮回调*/
30      uint8_t KeyboardExist;
31      Keyboard MyKeyboard;
32      Callback<Screen1View, Keyboard&> KeyboardEscBtClickedCallBack;
33      Callback<Screen1View, Keyboard&> KeyboardEnEnterBtClickedCallBack;
34      Callback<Screen1View, Keyboard&> KeyboardNbEnterBtClickedCallBack;
35      Callback<Screen1View, Keyboard&> KeyboardCnEnterBtClickedCallBack;
36  };
37
```

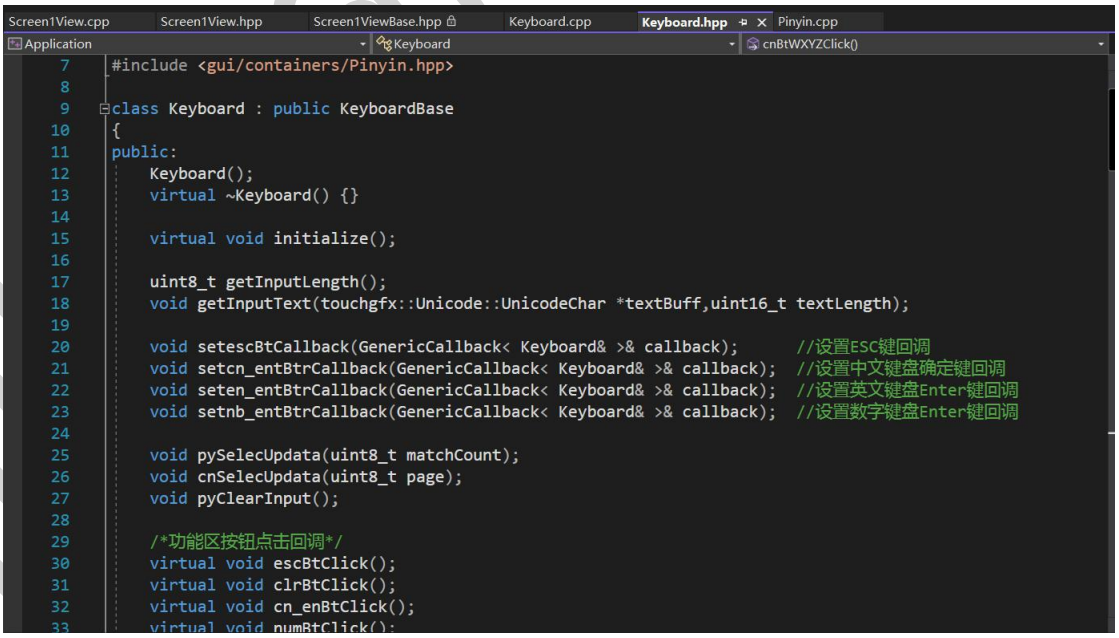
```
Screen1View.cpp | Screen1View.hpp | Screen1ViewBase.hpp | Keyboard.cpp | Keyboard.hpp | Pinyin.cpp
Application | Screen1View | setupScreen()
117
118 /*********************************************************************************
119  Name         :textArea1ClickHandler()
120  Description  :textArea1点击回调处理函数
121  Input        :text:文本控件  event:点击事件
122  OutPut       :无
123  Return       :无
124  Note         :无
125  *********************************************************************************/
126 void Screen1View::textArea1ClickHandler(const TextAreaWithOneWildcard& text, const ClickEvent& event)
127 {
128     if (&text == &textArea1)
129     {
130         if (event.getType() == ClickEvent::PRESSED)
131         {
132             if (KeyboardExist == 0)
133             {
134                 KeyboardExist = 1;
135
136                 add(MyKeyboard);
137                 MyKeyboard.initialize();
138                 MyKeyboard.invalidate();
139             }
140         }
141     }
142 }
143
```

上面代码在点击 textArea1 后在主界面弹出 Keyboard 控件。

8. 如果这些接口不知道怎么用，没关系，查看官网文档说明，up 主也不会用~~~~~~：

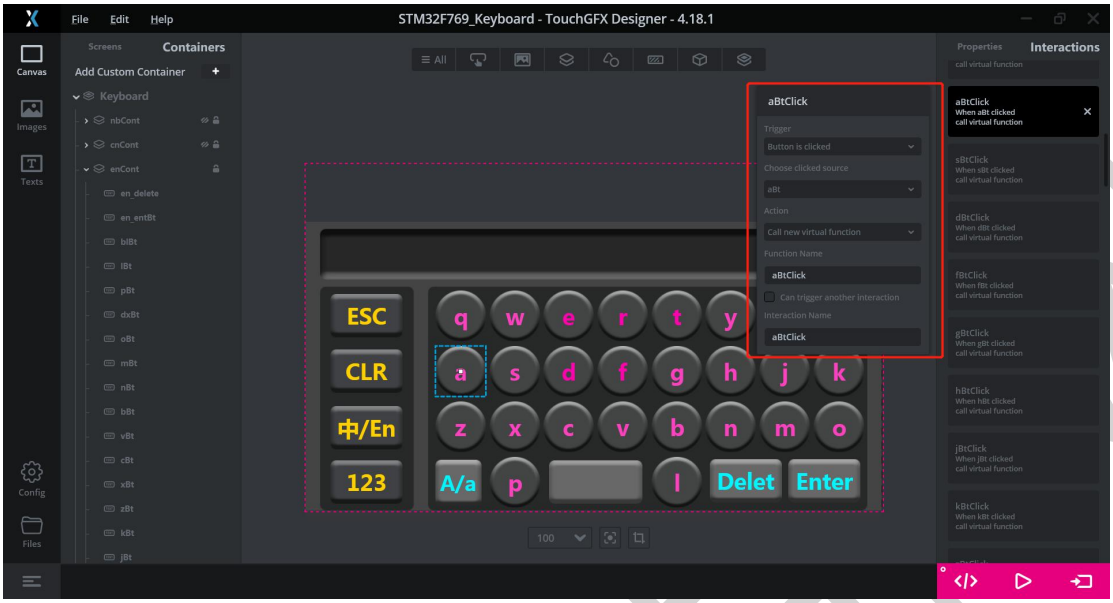


9. 键盘功能的实现：嗯，这里用了太多的回调，屏幕装不下。



如何获取键盘的输入的文本呢？

前面说了，要在 touchGFX 键盘按钮里添加回调，拿英文键盘 a 按钮来说：



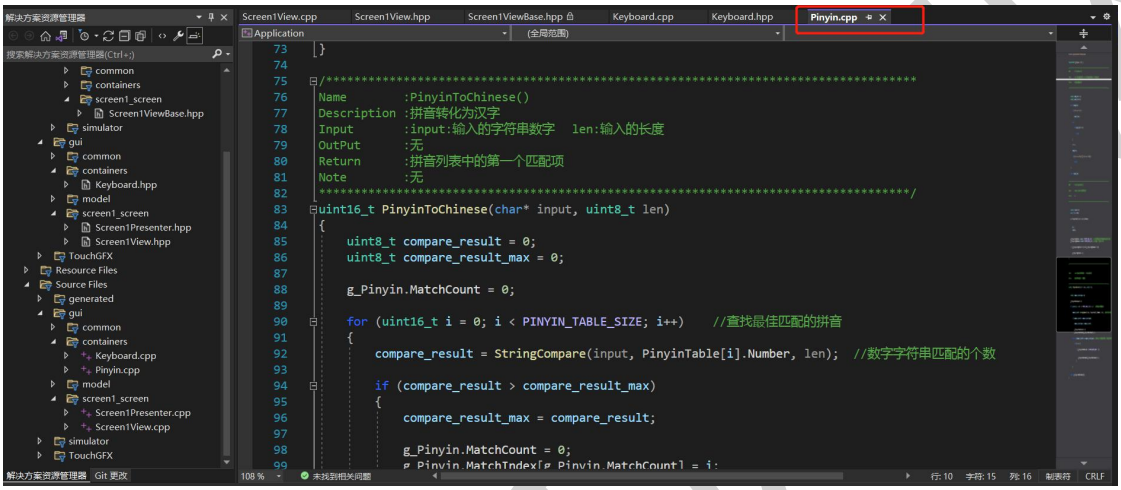
声名 a 按钮回调函数为虚函数，然后实现回调函数相应的动作：

```
Screen1View.cpp | Screen1View.hpp | Screen1ViewBase.hpp | Keyboard.cpp | Keyboard.hpp | Pinyin.cpp
Application | Keyboard | cnBtWXYZClick()
34
35      /*英文键盘按钮点击回调*/
36      virtual void en_entBtClick();
37      virtual void en_delBtClick();
38      virtual void dxBtClick();
39      virtual void aBtClick();
40      virtual void bBtClick();
41      virtual void cBtClick();
42      virtual void dBtClick();
43      virtual void eBtClick();
44      virtual void fBtClick();
45      virtual void gBtClick();
46      virtual void hBtClick();
47      virtual void iBtClick();
48      virtual void jBtClick();
49      virtual void kBtClick();
50      virtual void lBtClick();
51      virtual void mBtClick();
52      virtual void nBtClick();
53      virtual void oBtClick();
54      virtual void pBtClick();
```

```
346 /*********************************************************************************
347  *a按钮点击回调
348  *********************************************************************************/
349 void Keyboard::aBtClick()
350 {
351     if (textInputCount >= TEXTINPUT_SIZE)
352     {
353         return;
354     }
355
356     if (letter == LOWER_LETTER)
357     {
358         textInputBuffer[textInputCount++] = 'a';
359     }
360     else
361     {
362         textInputBuffer[textInputCount++] = 'A';
363     }
364
365     textInput.invalidate();
366 }
```

10. 中文键盘的实现，这里用了是一个很经典的中文输入发，前辈们称之为 T9 拼音输入法。**感谢前辈们为开源做出的努力，我们现在才能在单片机上跑中文输入法。**

中文输入的代码都房子啊这个文件里了，代码就不一一讲解了，这里只是针对 GFX 的使用特性对输入法做了优化。



这里每个中文按钮对于的有一个数字，T9 就是有 9 个中文按键，对于的数字从 1~9。没个按键按下在 cnNumber 里增加一个相应的数字字符，用于和拼音列表里对应的数字字符串逐一作比较，直至找出最佳匹配项。

拼音列表了有数字字符串、拼音字符串以及指向对应的汉字列表。

```
const PinyinDef PinyinTable[] =
{
{"2","a",(wchar_t*)PY_mb_a},
{"3","e",(wchar_t*)PY_mb_e},
{"6","o",(wchar_t*)PY_mb_o},
{"24","ai",(wchar_t*)PY_mb_ai},
{"26","an",(wchar_t*)PY_mb_an},
{"26","ao",(wchar_t*)PY_mb_ao},
{"22","ba",(wchar_t*)PY_mb_ba},
{"24","bi",(wchar_t*)PY_mb_bi},
{"26","bo",(wchar_t*)PY_mb_bo},
{"28","bu",(wchar_t*)PY_mb_bu},
{"22","ca",(wchar_t*)PY_mb_ca},
{"23","ce",(wchar_t*)PY_mb_ce},
{"24","ci",(wchar_t*)PY_mb_ci},
{"28","cu",(wchar_t*)PY_mb_cu},
{"32","da",(wchar_t*)PY_mb_da},
{"33","de",(wchar_t*)PY_mb_de},
{"34","di",(wchar_t*)PY_mb_di},
{"38","du",(wchar_t*)PY_mb_du},
{"36","en",(wchar_t*)PY_mb_en},
```

中文键盘每键入一个数字字符，都会进行一次匹配，同时根据最佳匹配项显示出对于的拼音字符串以及汉字。

```
/*****************************************************************************
*拼音按钮4点击回调
*****************************************************************************/
void Keyboard::cnBtGHIClick()
{
    if (cnNumberCount >= CN_NUMBER_SIZE)
    {
        return;
    }

    cnNumber[cnNumberCount++] = '4';
    g_Pinyin.TableIndex = PinyinToChinese(cnNumber, cnNumberCount);
    PinyinPageUpdata(g_Pinyin.TableIndex);

    g_Pinyin.PageIndex = 0;
    pySelecUpdata(g_Pinyin.MatchCount);
    cnSelecUpdata(g_Pinyin.PageIndex);
}
```

中文键盘的布局如下：

- 1.匹配到的拼音列表；
- 2.第一项拼音对于的汉字；
- 3.输入法键盘；

pySelecUpdata()就是刷新匹配到的拼音列表显示；

cnSelecUpdata()刷新最佳匹配的拼音列表里的汉字。