Camera APP 的互斥实现

孙延宾

5月10日, 2017年

目 录

I	数据结构	1
II	操作流程	2

第1部分 数据结构

互斥相关的数据结构只有一个:ListPreference,不过并非所有属性都与互斥相关,与互斥相关的只有红色的 mValue、mEnable 以及蓝色的三个 list (list 和 map)。

ListPreference String mTitle; CharSequence[] mEntries; int mTitleIcon; int mListIcons[]; String mKey; String[] mValue={old, mutex}; CharSequence[] mDefaultValues; CharSequence[] mEntryValues; boolean mSave; boolean mLoaded; boolean mEnable; ArrayList<String> beauty pro auto mModeLevelList; HashMap<String, String> mFunctionLevelList; key-1 mode-5 HashMap<String, String> key-2 mode-1 mFunctionPriorityList; key-1 mode-1key-3 mode-1 SharedPreferences key-1 mode-2 mSharedPreferences; key-4 mode-1 key-1 mode-3 OnPreferenceSettingListener key-5 mode-1 mListener; mode-4 key-2 mode-5 key-2

- mValue 是一个 length 为 2 的数组,第一个元素表示其本来的值,第二个元素表示它跟别的功能冲突而被禁用时的临时值(一般为 off)
- mEnable 表示互斥的结果,互斥的结果只有两个,要么打开、要么被关闭(有特例)

• 蓝色的三个 list 表示与该 preference 冲突的功能点,注意功能点是与模式相关的, auto模式下的闪光灯跟 professional 模式下的闪光灯是不一样的。其中的两个 hashmap 其实是表达一种"多对多"的关系:一个模式对应多个功能、一个功能也可以对应多个模式。

第 II 部分 操作流程

数据结构明确了,那 ListPreference 里的数据是如何修改的呢?答案就是 MutualHandle,任何一个 preference 的打开、关闭,以及与其冲突的功能点设置都是在 MutualHandle 中操作的。该类主要做了两件事情:

- 1. 为每一个可能存在冲突的功能点定义一个处理函数,注意这里的功能点与模式无关,它与上面定义的"功能点"不同。
- 2. 设置互斥的入口:doMutualHandle(manager, key),即有 preference 变动就要到互斥里面来走一圈。

下面是一个代码实例,

```
/*
1
    * xxxMutual:
2
        当 xxx 功能(如 flash)因为冲突(或取消冲突)而 disable(或 enable)时,
        调用该函数来处理 xxx 的 preference 以及处理与该功能存在冲突的其他功能。
    * enable: 打开还是关闭
    * mutualState: 当该功能被关闭时要设置的值,一般为 off,表示因为冲突而被关闭
    * levelValue: 模式级别还是功能级别,
         1. 模式级别表示在某个模式下该功能 enable/disable
         2. 功能级别表示该功能与某个功能冲突, 注意功能是与模式相关的
10
    * modeValue: 相关的模式
    * lastMutualKey: 功能点,如 flash,与 modeValue 组成具体的功能点
    * beAssociate: 处理循环冲突。如 A、B 互相冲突, 水火不容, A 变动会引起 B 变动,
13
         但是 B 变动是就不要再引起 A 变动了,否则就死循环了。
14
15
  private void xxxMutual(SettingManager manager, boolean enable, String mutualState,
16
      int levelValue, String modeValue, String lastMutualKey, boolean beAssociate) {
      // 1. 拿到 pref 对象 , 因为要修改的就是它
18
      ListPreference pref = ((ListPreference)
19
                 SettingProvider.getListPreference(SettingFuncConstant.KEY_XXX));
20
      if (pref == null)
21
          return;
      // 2. 把 key 添加到变动列表里 , 告知 UI 哪些 preference 有变动
      manager.addChangedKey(pref.getKey());
      // 3. 记录与 pref 冲突的功能点 ,并 enable/disable pref
25
      pref.setPriority(enable, levelValue, modeValue, lastMutualKey);
26
      // 4. 如果 pref 是被 disable 了,那 disable 之后要怎么变化呢,
27
           一般是将其值设置为"\circ f f",或者 \circ,如倒计时,等等
28
      if (!pref.isEnable()) {
          pref.setValue(mutualState, ListPreference.UPDATE_VALUE_MUTEX);
      // 5. 避免循环冲突
32
      if (!beAssociate)
33
          return;
34
      // 6. 最后,哪些功能点会与"我"冲突呢,你们该变得也赶紧变吧
      if ("on".equals(pref.getValue())) {
          supLightMutual(manager, false, PRIORITY_LEVEL_FUNCTION, modeValue,
37
                       SettingFuncConstant.KEY_LIVE_SHOOT);
38
      } else {
39
          supLightMutual(manager, true, PRIORITY_LEVEL_FUNCTION, modeValue,
                       SettingFuncConstant.KEY_LIVE_SHOOT);
      }
42
43
```