**第三章：输入接口**

**所谓“控制感”**

这篇文章主要介绍使用输入接口对图元进行控制。

[游戏当中的输入分类]

1. 主动检索式，在Serial中检测按键的状态，作出不同的反应，这个适用于控制要求及时、快速的情况，例如控制游戏角色等。
2. 被动驱动，类似于AWT的事件回调，用于控制要求频度不高，但往往是单次调用的过程，例如游戏菜单的控制。

二者谁也无法替代谁，使用回调控制角色，会有很大的滞后感；而使用主动检索驱动UI菜单，则会敏感度过高，稍微一按就飞出好几项之后了。

[主动检索式]

例子：

@Override

public void Serial(int arg0) {

if(UES.getKeyStatus(KeyEvent.VK\_LEFT)){

rect.setDx(rect.getDx() - 3);

}

if(UES.getKeyStatus(KeyEvent.VK\_RIGHT)){

rect.setDx(rect.getDx() + 3);

}

if(UES.getKeyStatus(KeyEvent.VK\_UP)){

rect.setDy(rect.getDy() - 3);

}

if(UES.getKeyStatus(KeyEvent.VK\_DOWN)){

rect.setDy(rect.getDy() + 3);

}

}

将第二章动态绘制中的Serial改写如上，运行的结果就是可以使用键盘的上下左右控制绿色矩形移动。

API介绍：

UES:boolean getKeyStatus(int keyCode);

keyCode是键盘按键对应的ascii码，但是我们不需要去记忆整数码，可以使用KeyEvent.VK\_键盘字母 的格式进行引用。

该方法返回此时键盘的按键状态，若为按下，则返回true，否则为false;

除去键盘，鼠标也有类似的API：

UES.getMouseButtonStatus(arg0)，它可以获得当前鼠标的按钮状态，1,2,3在本人的设备上分别对应左键、中键（滚轮按下）、右键；

UES.getCrtMouseX()，UES.getCrtMouseY()则返回当前鼠标位置。

=====================================================

[被动驱动]

被动驱动的实现就没有上面简单了。

以键盘为例：

[interface geivcore.KeyFactor]

KeyFactor接口：所有响应键盘回调的类需要实现KeyFactor接口，它包括了三个方法：

public void setKeyListener(KeyListener keyListener);

public void removeKeyListener();

public void keyFactor(int keyCode,boolean ispressed);

如果需要响应键盘的类没有继承其他类，则可选择继承geivcore.DefaultFactor类，进行默认的键盘响应实现。

[interface geivcore. KeyListener]

KeyListener接口：所有处理回调的逻辑实现自该接口，该接口只有一个方法：

public void doKeyBord(KeyFactor whom,int keyCode,boolean ispressed);

whom是响应的KeyFactor对象，keyCode是按键对应的ascii码，ispressed标识着是按下还是抬起。

在实现了上两个接口后，使用UES.pushKeyBoardIO(KeyFactor factor)方法，将KeyFactor压入键盘IO的栈顶，我们的键盘指令会始终传递给栈顶的KeyFactor，这是为了适应传统的UI系统。

\*注意，虽然KeyFactor、KeyListener与UES键盘栈系统被保留到高层，允许开发者自己去实现，但这一过程确实有些复杂，因此在开发工具集中留有Component类作为UI组件工具，也有AutoList类用于从XML文件中构架菜单结构，在工具类的介绍章节里会详细介绍。

[总结]

介绍了主动监测与被动回调两种输入方案，其中被动驱动的实现较为复杂，因此建议尽量使用工具类中的现成组件。