Input v2.0

Creators: Gavin Ovsak, Aaron Krolik, Ying Chen

Source: https://github.com/aaronkrolik/CS308FinalProject_Inputs.git

Table of Contents

- Summary
- Implementation
- Keyboard Input Actions
- Mouse Input Actions
- Overriding Setting Resources
- AlertObject Methods:
- PositionObject Methods:
- RollObject Methods:
- KeyName List (Case sensitive):

Summary

Thank you for downloading the Input v2.0 package. Input version 2.0 is a package of java classes designed to be added into your code as a package which a game genre team can implement in order to allow the user to interact with their game with an expansive range of input devices. As of v.2.0, the goal was to provide additional compound behavior notifications to the game designer which would be convenient and useful. One of the biggest changes for the users of this API is the new layout of the Mapping Resource file which is detailed below. While there is a new recommended design, the old resource file layout is still supported as is all of the methods provided in v1.1 of Input. Additional devices will be implemented in future versions while maintaining backward compatibility and minimal code modification on the game designer's part to implement additional features. In the next version, the ability to permanently change / create a new resource mapping properties file based on current modified mappings will be implemented.

Implementation

To use the Input package, your game will have to instantiate an Input object for a JComponent which a user will be focused on when interacting with your game. The core of the implementation is the Input class. To initialize the Input class you call new Input(String resourcePath, JComponent component).

When building applications that use Input, simply do 3 things: annotate classes that have listening methods with the @InputClassTarget annotation, annotate specific methods with the @InputMethodTarget() and include a resource file (using our provided format) with the input mappings.

To add an instance of an object which has methods that you want Input to call, simply call from anywhere Input.addListenerTo(instanceofobject). And thats it! if you want to remove the listener, call from anywhere Input.removeListener(instanceofobject) and that object will no longer be called from input.

For reference, here are the public methods that can be called on an Input object:

- **Input.overrideSettings(String resourcePath)**
  This is to override your default input setting. And the string resourcePath is the relative location to the settings resource file.

- **Input.overrideMapping(String gameBehavior, String inputBehavior)**
  Allow for dynamic remapping of inputs at runtime. String parameter gameBehavior is analogous (and formatted the same way) as the resource file key, String parameter inputBehavior is analogous to (and formatted the same way as) the resource file key. ****Warning**** This method will override your input mappings! Be sure that all inputBehaviors that you want mapped to a gameBehavior are included. Also make sure that you want to reassign each included inputBehavior. If you get into trouble, call Input.restoreMappingDefualts() to return to the origional mappings.

- **Input.addListenerTo(Object in)**
  Passed an object instance, addListenerTo() allows Input.java to call properly annotated methods when prompted by the corresponding input event.

- **Input.removeListener(Object in)**
  Passed an object instance, removeListener removes that instance from Input.java's cache of instances, and methods will no longer be invoked on that instance.

- **Input.restoreMappingDefaults()**
  Removes any dynamic mappings during current session and will revert input mappings to those specified by the resource file.

- **Input.replaceMappingResourcePath(String path)**
  Input mappings will be reset to reflect the resource file at the end of path. Useful for switching between two game modes (ex. battle and exploring mode each different input mappings)

An example program is included in the Input v2.0 package. Highlighted below is the instantiation of Input in the example along with the custom resource file. The recommended form of a resource file is shown in Game1MappingsMode1_en_US.properties, but the legacy format as shown in LegacyGame1Mappings_en_US.properties is still supported. In the new format, a pipe can be used to indicate multiple input actions mapping to a single game behavior.

From Game1.java:

```
@InputClassTarget
public class Game1 {
...
     @InputMethodTarget(name="continue")
     public void goPastPopup() {
          popup = false;
     }

     @InputMethodTarget(name = "setMode1")
     public void setMode1(AlertObject x){
input1.replaceMappingResourcePath("examples/Game1MappingsMode1");
     }

     @InputMethodTarget(name = "setMode2")
     public void setMode2(AlertObject x){
input1.replaceMappingResourcePath("examples/Game1MappingsMode2");
     }

     ...
}
```

From Game1MappingsMode1_en_US.properties:

```
jump = Keyboard_Spacebar_Down
cheat = Keyboard_G_Down
stopcheat = Keyboard_G_Up
anticheat = Keyboard_F_Down
stopanticheat = Keyboard_F_Up
continue = Mouse_Left_Down|Mouse_Right_Down
test = Mouse_Move
restore = Keyboard_T_Down
setMode1 = Keyboard_1_Down
setMode2 = Keyboard_2_Down
scrolltest = Mouse_Wheel_Down
```

From LegacyGame1Mappings_en_US.properties: (Deprecated Format)

```
Keyboard_Spacebar_Down = jump
Keyboard_G_Down = cheat
Keyboard_G_Up = stopcheat
Keyboard_F_Down = anticheat
Keyboard_F_Up = stopanticheat
Mouse_Left_Down = continue
```

```
Mouse_Right_Down = continue
Mouse_Move = test
Keyboard_T_Down = restore
Keyboard_1_Down = setMode1
Keyboard_2_Down = setMode2
```

Our API supports various formats of inputs from different devices. Here is brief introductions to all these input formats.

Keyboard Input Actions

Note:
<KeyNameList> can be replaced multiple key names in series in any order. See KeyName list for KeyNames. Ex: “SpacebarAB” or “ShiftA” or “AShift”
<KeyName> can be replaced with one KeyName. See KeyName list for KeyNames. Ex: “Spacebar” or “Shift” or “A”

- **Keyboard_<KeyNameList>_Down**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal is sent once the key or keys annotated in the keyNameList are pressed at the same time. If the KeyNameList has only one key, for example, “Spacebar” or “A”, it means only this specific key is pressed. If the KeyNameList has multiple keys, for example “AB” or “CDF”, it means these keys are considered to be pressed at the same time.

- **Keyboard_<KeyName>_LongPress**
  Types of Acceptable Parameters: No Parameters, AlertObject
  Our API will send this signal once a key is pressed and held for some time. This signal will be continuously sent until you released the key. In other word, you will receive this signal periodically until you release the key.

- **Keyboard_Any_Down**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal is sent after you press any key.

- **Keyboard_Any_Up**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal is sent after you release any key.

- **Keyboard_<KeyName>_Up**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal is sent after you release a key. Our API will only send this signal once.

- **Keyboad_<KeyName>_ShortClick**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal will be sent once you press a key and release it within the threshold time. Basically, you will need to execute this signal after you release a key. You can set the threshold time here. See Overriding Setting Resources.

- **Keyboard_<KeyName>_LongClick**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal will be sent once you press a key and release it after a threshold time.
  In other word, our API will calculate the interval between the time you pressed and the time you released. If the interval is greater than the threshold time, API will send this signal. Basically, you will need to execute this signal after you release a key. You can set the threshold time here. See Overriding Setting Resources.

- **Keyboard_<KeyName>_DoubleClick**
  Types of Acceptable Parameters: No Parameters, AlertObject
  This signal will be sent once you double click a key. It will be invoked if you press a key and press it again within a threshold time. Our API will calculate the interval between the two presses. If the interval is smaller than the threshold, API will send this signal.You can set the threshold time here. See Overriding Setting Resources.

- **Keyboard_<KeyName>_KeyUp and Keyboard_A_Click (Deprecated)**
  Types of Acceptable Parameters: No Parameters, AlertObject
  Renamed to Keyboard_<KeyName>_Up. Legacy support from v1.1.

- **Keyboard_<KeyName>_KeyDown (Deprecated)**
  Types of Acceptable Parameters: No Parameters, AlertObject
  Renamed to Keyboard_<KeyName>_Down. Legacy support from v1.1.

Mouse Input Actions

Note:
<MouseButton> can be replaced with either “Right”, “Center”, or “Left”.

(Left can be replaced with Right or Center):

- **Mouse_Move**

Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
This signal is sent once you move the mouse without click it. The signal is sent continuously until you stop moving the mouse.

- **Mouse_Drag**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  This signal is sent once you click the mouse and drag it. The signal is sent continuously until you stop dragging or clicking the mouse.

- **Mouse_<MouseButton>_Down**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  This signal is sent once you click the mouse. The signal will be sent only once even if you keep pressing the button.

- **Mouse_<MouseButton>_Up**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  This signal is sent once you release the button on mouse after click it. The signal will be sent only once.

- **Mouse_<MouseButton>_ShortClick**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  This signal is sent once you press a button and release it within a threshold time. The API will calculate the interval between you press and release the same button. If the interval is smaller than the threshold, API will send the signal. You can set the threshold time here. See Overriding Setting Resources.

- **Mouse_<MouseButton>_DragLeft**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you clicked the left button of mouse and drag it towards direction left. This signal will be sent only once until you release the button.

- **Mouse_<MouseButton>_DragRight**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you clicked the left button of mouse and drag it towards direction right. This signal will be sent only once until you release the button.

- **Mouse_<MouseButton>_DragUp**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you clicked the left button of mouse and drag it upwards. This signal will be sent only once until you release the button.

- **Mouse_<MouseButton>_DragDown**

Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
Our API sends this signal when you clicked the left button of mouse and drag it downwards. This signal will be sent only once until you release the button.

- **Mouse_MoveOut**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you move the mouse out of the screen. The signal will be sent only once.

- **Mouse_MoveIn**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you move the mouse into the screen from the outside. The signal will be sent only once.

- **Mouse_DragOut**
  Types of Acceptable Parameters: No Parameters, AlertObject, PositionObject
  Our API sends this signal when you clicked any button of the mouse and drag it outside the screen. The signal will be sent only once.

- **Mouse_Wheel_Up**
  Types of Acceptable Parameters: No Parameters, AlertObject, RollObject
  Our API sends this signal when you roll the mouse wheel away from you. The signal will be sent only once for each roll.

- **Mouse_Wheel_Down**
  Types of Acceptable Parameters: No Parameters, AlertObject, RollObject
  Our API sends this signal when you roll the mouse wheel towards you. The signal will be sent only once for each roll.

Overriding Setting Resources

We provide default settings for inputs, most of which are time threshold. The filePath of the default settings file is ***input/DefaultSettings.properties.***

In the current version of API, the contents of the default settings are like this. All these times are in milliseconds.

```
ShortClickTimeThreshold = 100
LongClickTimeThreshold = 400
DoubleClickTimeThreshold = 200
DoubleClickDistanceThreshold = 10
```

- **ShortClickTimeThreshold = 100**
  This defines the threshold time for short click. In other words, if the interval between a click and a release is smaller than this threshold, our API will send a signal indicating this is a short click.

- **LongClickTimeThreshold = 400**
  This defines the threshold time for long click. In other words, if the interval between a click and a release is greater than this threshold, our API will send a signal indicating this is a long click.

- **DoubleClickTimeThreshold = 200**
  DoubleClickDistanceThreshold = 10
  These two define the threshold for double click. If the interval between two clicks is smaller than the time threshold, and the distance of two click points is smaller than the distance threshold, our API will send a signal indicating this is a double click.

Users can override the default settings by creating another setting file. For example, we can set up a file like the following:
ShortClickTimeThreshold = 200
LongClickTimeThreshold = 400

In this case, the time threshold for short click and long click has modified, while the other thresholds keep unchanged. You can validate the override setting file by calling the method in the input class.

void overrideSettings(String resourcePath), where the input is the path of the new setting file.

AlertObject Methods:

getTime() returns double

PositionObject Methods:

getRelativeX() returns double ranging from 0 to 1. Defaults to 0
getRelativeY() returns double ranging from 0 to 1. Defaults to 0
getRelativeZ() returns double ranging from 0 to 1. Defaults to 0
getPoint2D() returns Point2D.Double of X and Y points

getX() returns double. Defaults to 0 (**Deprecated**)
getY() returns double. Defaults to 0 (**Deprecated**)
getZ() returns double. Defaults to 0 (**Deprecated**)

RollObject Methods:

getUnitsRotated() returns integer

Units value is positive or negative. Positive means the user scrolls the wheel towards him, which consider as "Mouse_Wheel_Down". Negative means the user scrolls the wheel away from him, which consider as "Mouse_Wheel_Up".

Rounds divides units would be different static value which depend on different mouses due to the mechanism.

KeyName List (Case sensitive):

0-9, A-Z (capital only), Left, Right, Up, Down, Shift, Spacebar, Enter, Delete, Control, Tab, Add, Minus, F1-F12