Input v1.1

Creators: Gavin Ovsak, Aaron Krolik, Ying Chen

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

Table of Contents

Summary

Thank you for downloading the Input v1.1 package. Input version 1.1 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 v1.1, the goal was to allow for basic keyboard and mouse behaviors to be mapped to commands in a game. 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.

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. This class follows the singleton pattern with a special constructor to pass in information critical for initialization. To initialize the Input class you call new Input(String resourcePath, JComponent component).

When building applications that use our input manger, 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 to our manager, 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.

An example program is included in the Input v1.1 package. Highlighted below is the instantiation of Input in the example along with the custom resource file.

From Game1.java:

```
@InputClassTarget
public class Game1 {
...
     @InputMethodTarget(name="jump")
     public void jumpInput(AlertObject alObj) {
          if (you.getTimeSinceJump(time) > 1 && you.getBottom() >
              448) {
                you.jump(time);
          }
     }

     @InputMethodTarget(name="cheat")
     public void cheat(AlertObject alObj) {
          you.setCheating(true);
     }

     @InputMethodTarget(name="anticheat")
     public void anticheat(AlertObject alObj) {
                you.setAntiCheating(true);
     }

     @InputMethodTarget(name="stopcheat")
     public void stopcheat(AlertObject alObj) {
          you.setCheating(false);
     }

     @InputMethodTarget(name="stopanticheat")
     public void stopanticheat(AlertObject alObj) {
          you.setAntiCheating(false);
     }

     @InputMethodTarget(name="continue")
     public void goPastPopup(PositionObject posObj) {
          popup = false;
     }

     @InputMethodTarget(name="test")
     public void movementCoordTest(PositionObject posObj) {
        System.out.println(posObj.getX() + ", " + posObj.getY());
    }

    ...
}
```

From Resource_en_US.properties:

```
Keyboard_Spacebar_KeyDown = jump
Keyboard_G_KeyDown = cheat
Keyboard_G_KeyUp = stopcheat
Keyboard_F_KeyDown = anticheat
Keyboard_F_KeyUp = stopanticheat
Mouse_Left_Click = continue
Mouse_Move = test
```

The following list of input actions can be mapped to a custom game behavior:

Keyboard Behaviors

Action Name	**Object Provided**
Keyboard_A_KeyDown	AlertObject
Keyboard_A_KeyUp	AlertObject
Keyboard_B_KeyDown	AlertObject
Keyboard_B_KeyUp	AlertObject
... for all keyboard letters A-Z and # 0-9	AlertObject
Keyboard_Spacebar_KeyDown	AlertObject
Keyboard_Spacebar_KeyUp	AlertObject
Keyboard_Up_KeyDown	AlertObject
Keyboard_Up_KeyUp	AlertObject
Keyboard_Down_KeyDown	AlertObject
Keyboard_Down_KeyUp	AlertObject
Keyboard_Left_KeyDown	AlertObject

Keyboard_Left_KeyUp	AlertObject
Keyboard_Right_KeyDown	AlertObject
Keyboard_Right_KeyUp	AlertObject

Mouse Behaviors

Action Name	**Object Provided**
Mouse_Right_Down	PositionObject
Mouse_Right_Click	PositionObject
Mouse_Right_Up	PositionObject
Mouse_Left_Down	PositionObject
Mouse_Left_Up	PositionObject
Mouse_Left_Up	PositionObject
Mouse_Move	PositionObject
Mouse_Drag	PositionObject

AlertObject Methods:

getTime() returns double

PositionObject Methods:

getX() returns double
getY() returns double
getPoint2D() returns Point2D