System Practicum: CS 307

(Mini- Project)

MouseGes- Gestures on Mouse Trackpad

Abhigyan Khaund (B16082)
Rohit Kaushal (B16028)
Vishnu Priya Jindal (B16041)
Anubhav Kumar (B16013)

Indian Institute of Technology Mandi

E-mail: : b16082@students.iitmandi.ac.in , b16028@students.iitmandi.ac.in
b16041@students.iitmandi.ac.in b16013@students.iitmandi.ac.in

May 2019

Revision History

Version	Date	Author(s)	Description
v1.0	05/21/19	Abhigyan, Rohit, Vishnu Priya, Anubhav	Initial version

Table of Contents

1 Problem Statement

Implement a linux kernel module for recognising trackpad gestures and opens a user-defined application corresponding to the gesture. Also create a utility script for easy defining of applications to be launched.

2 Specifications and Usage

Start recording gesture – 3 continuous left click in < 1sec anywhere

Stop recording gesture – Any click anywhere

Gesture recording– After recording start, do gesture using tracpad/mouse **anywhere** on the screen.

Gestures -

- Horizontal Line
- Vertical Line
- Right Diagonal
- Left Diagonal
- V Shape

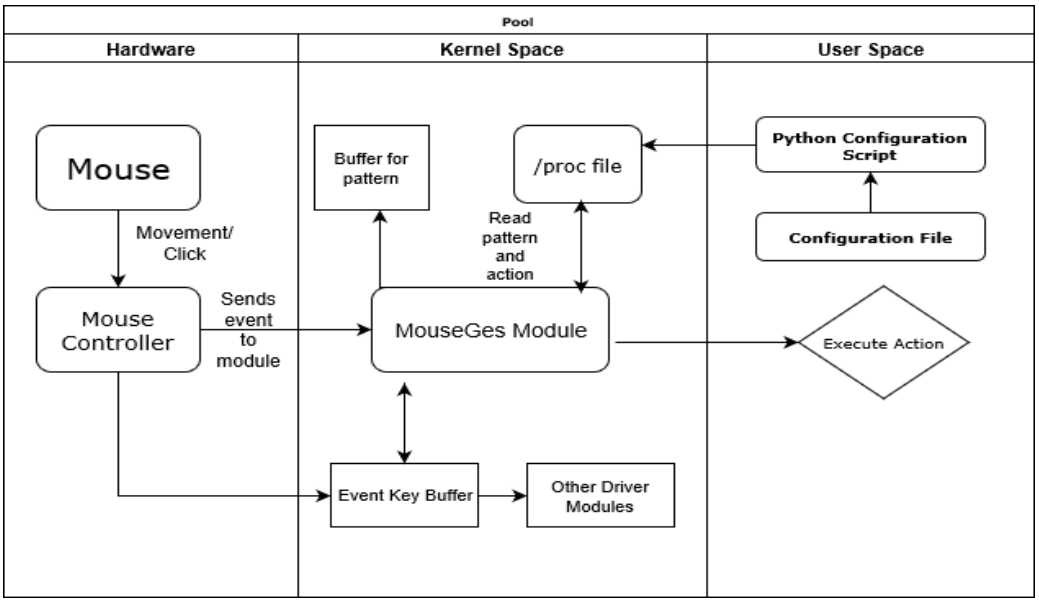
Configuration file – Applications to be lauched corresponding to gesture

Python script config.py - Update applications in kernel proc file

3 System Design

3.1 Architecture

The kernel module has an event driven function which invokes on mouse movement and click logs. /proc filesystem is used to communicate between user space and kernel space. The gestures file in /proc/ is used to identify the the corresponding action with pattern which then invokes a userspace api from the module to complete the action.



3.2 Data Structures

```
static struct action
{
     char execute[100];
     int len;
}action_list[5];
```

action _list is a structure array which stores the application to be executed corresponding to a gesture. len stores the length of the execute string.

A structure is used for this keeping in mind a future scope where we store more information regarding the action like path and environemnt variables.

```
static char* envp[] = {
     "SHELL=/bin/bash",
     "HOME=/home/user-name",
     "PATH=/sbin:/bin:/usr/sbin:/usr/bin",
     "DISPLAY=:0",
      NULL
};
```

envp sets the environment for launching applications after gesture detection.

3.3 Procedures and Algorithms

```
Procedure should_start(click, value) {
     if not value : return;
     checkClicks();
     if(trigger[0] !=-1 && trigger[1] !=-1 && trigger[2] !=-1):
          trigger = {0,0,0}
          gesture_on = 0
          determinePattern();
          count_x = 0,count_y = 0;
          return;
     end if
     if click == BTN_LEFT
          if trigger[0] is -1 :
               trigger[0] = 1;
          endif
          else if trigger[1] is -1:
               trigger[1] = 1;
          endif
          else if trigger[2] is -1 :
               trigger[2] = 1;
               gesture_on = 1;
          endif
          else :
               trigger = {-1, -1, -1};
               gesture_on = 0;
```

```
            endelse
      end if
      else :
            if gesture_on is 1:
                  trigger = {-1, -1, -1};
                  gesture_on = 0;
                  determinePattern();
            endif
            else :
                  trigger = {-1, -1, -1};
            endelse
      endelse
end procedure
```

This procedure determines whether to start recording of trackpad movements and when to stop recording it. It calls another procedure determinePattern when recording is complete.

```
static struct timespec t1,t2;
procedure checkClicks() :
      getnstimeofday(&t1);
      if trigger[0] is -1:
            do nothing as first click
      endif
      else
            if(t1.tv_sec - t2.tv_sec >1 and gesture_on is 0) :
                  trigger = {-1,-1,-1};
            endif
      endelse
      t2 = t1;
end procedure
```

This procedure determines whether two continuous clicks are within a time difference of 1 sec or not. If it isn't it update trigger to default values. In case gesture is recording, it doesn't update trigger values as time difference can be more for clicks.

```
procedure determinePattern() :
      min_x = get_min(position_x), max_x = get_max(position_x);
      min_y = get_min(position_y), max_y = get_max(position_y);
      if(max_x - min_x <=10000 && abs(y_end – position_y[0])>3000):
            call_usermodehelper(command for vertical line gesture);
      endif
      else if(max_y - min_y <=10000):
            call_usermodehelper(command for horizontal line gesture);
      endif
```

```
      else if(abs(y_end - position_y[0])<3000) :
            call_usermodehelper(command for v shapegesture);
      endif
      else if(position_x[0] < position_x[1] ):
            call_usermodehelper(command for right diangonal gesture);
      endif
      else :
            call_usermodehelper(command for left diagonal gesture);
      endelse
end procedure
```

This procedure determines the gesture based on the coordinate axis positions and does an upcall to user-space to create process corresponding to the executable using `call_usermodehelper`.

3.4 Interfaces

In the kernel module, the module directly interacts with the device as a device driver to get information from the OS. Proc filesystem */proc/mouseges_proc_file* is used to communicate between user-space and kernel space regarding actions of pattern transfer to kernel memory. When a user writes to this file, the module updates the *action_list.*

3.5 External Data

Files: *configuration*i is used to store the application executable strings corresponding to the gesures onthe hard disk. When the user runs *config.py* and on module startup, the contents of this file are read and copied into the kernel buffer through the proc filesystem */proc/mouseges_proc_file*

4 Performance and Testing

The module driver and action executables are stored in kernel memory itself as compared to a normal application where to access the device movements, we would have to first do system calls which would require few system calls to read from the memory where the mouse movements and clicks. Hence since the device module and patterns are stored in kernel memory there is no need of any system calls from user space, thus increasing overall performance in comparision to an user space application.

Further, all the trackpad events that occur when not recording are not processed by the module and directly passed on to the other device drivers . So addition of this driver doesn't cause any extra load on the processor unless it is recording.

For testing, we have launched different types of applications like gedit, firefox, calculator, terminal, calender, browser, nautilus, custom scripts etc.

5 Conclusion

The project has achieved its target of building a linux kernel module that takes trackpad movements as input and determines a set of predefined gestures from the movements. It then launches an application that the user has set in correspondence that gesture as new process in user space. The module is not limited to five types of gestures. The speciality of the module is that it can work in any window and any part of the screen.

6 Future Scope

- Support for more custom user-defined gestures
- Display a trail of gesture movement after recording has started
- Notification that gesture recognised and application is loading