HEADTRACKING: CONTROLLING YOUR COMPUTER WITH YOUR HEAD

New Mexico

Supercomputing Challenge

Final Report

April , 2013

Team: 64
Los Alamos Middle School

Team Members:
Alex Ionkov

Teacher:
Pauline Stephens

Project Mentor:
Latchesar Ionkov

TABLE OF CONTENTS

Executive Summary

This project explores the possibilities in creating a system which allows paralyzed people to use a computer. Paralyzed people have hard time living normal life. They always need someone else, such as a caretaker. A study by the Christopher and Dana Reeve Foundation reports that 5.6 million people, representing 1.9 percent of the population, or roughly 1 in 50 Americans are paralyzed. In order to create a system such as this, research was needed such as what parts of the body are not affected by paralysis. The parts of the body that are not affected depends on the type of paralysis you have. For spinal paralysis, it was found that the tongue is not affected since it is not connected to the spine. But, in order for to use the tongue a magnetic ball implant would be required to do the testing, so that was not a choice. In order, to avoid this transplant it was decided to use the head as the base of this project; it's moves will control the mouse. Using Python, Xlib, and Pymouse as the software parts of this project this program was created. Using the Wiimote, and safety glasses with infrared LEDs as the hardware parts of this project the link from the program to the outside world was created. The mouse would, when calibrated correctly, move according to the position of the average of the two infrared LEDs hence your head position. To do a click, an idea was established that the tilting of the head would do the job nicely and a coefficient was added which decided how much you have to tilt your head for a click to occur.

Introduction

I saw a video called “Johnny Chung Lee demonstrates Wiimote hacks”. I clicked on it and saw an interesting idea: Making a Head-Tracking device using a Wiimote and sensor bar. I read more about it and the article said that you could use infrared LEDs to have the same effect as the sensor bar. I then started experimenting with the Wiimote and the program.

Problem Definition

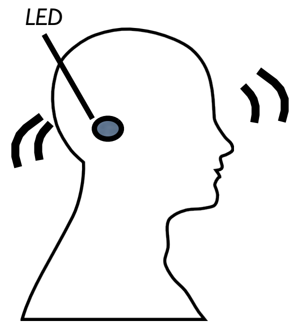
Paralysis is loss of muscle function for one or more muscles. People that are paralyzed no longer have a life of their own. They do not have as much freedom to do things as a healthy person does. I want to write a program which allows them to control a computer with their head. For some paralyzed people this program will not help as their type of paralysis is not allowing them to move their head, but for others it will help as they can move their head at least a little. This will help their lives because they can now do something on their own.

Problem Solution

The goal of this entire project was to create a head-tracking system that enables the user to control their computer with their head. The Wiimote allows the tracking of up to four infrared LEDs. I am going to attach one or more LEDs to a head-piece and use the Wiimote to track their position. The Wiimote connects to the computer over Bluetooth and sends messages to the computer. The program will receive all the messages from the Wiimote, scan all these messages and pick out the ones that concern the LEDs position. Then the program will move the mouse on the screen according to the position of the infrared LED. The environment in which this program will work in is Linux-only since it will use the Xlib library to move the mouse and the Cwiid library to communicate with the Wiimote.

Results

Figure 1



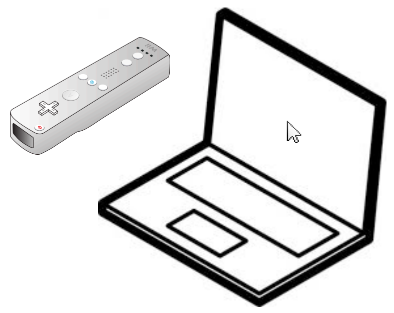
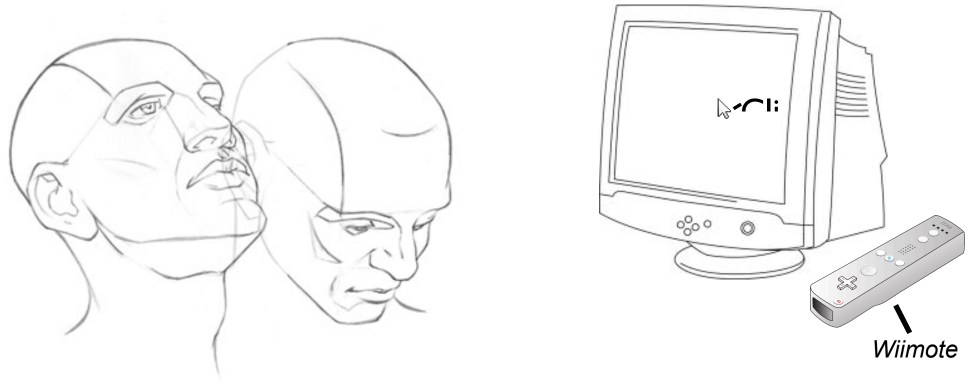


Figure 1 shows the concept of moving your head in order to move the mouse.

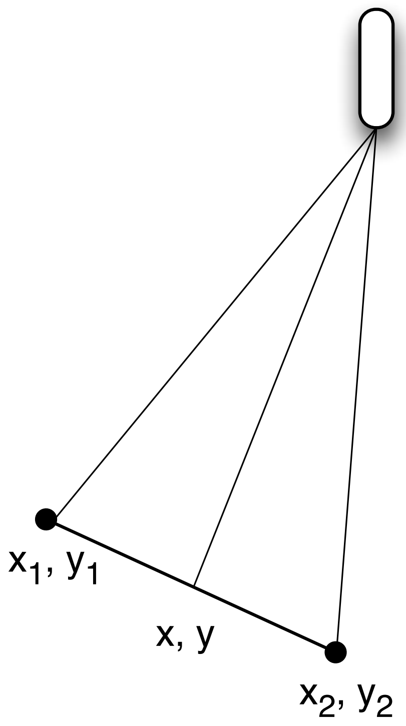
Figure 2



Turn your head to click, return your head to previous position to release the click.

Figure 2 shows the concept of tilting your head in order to make the mouse click.

Figure 3



$$x = \frac{x_1 + x_2}{2}$$

$$y = \frac{y_1 + y_2}{2}$$

$$dx = x_1 - x_2$$

$$dy = y_1 - y_2$$

$$dist = \sqrt{dx^2 + dy^2}$$

Figure 3 shows the formulas used for calculating the positions of each infrared LED and finding how close the LEDs are to the Wiimote though that wasn't used.

Figure 4

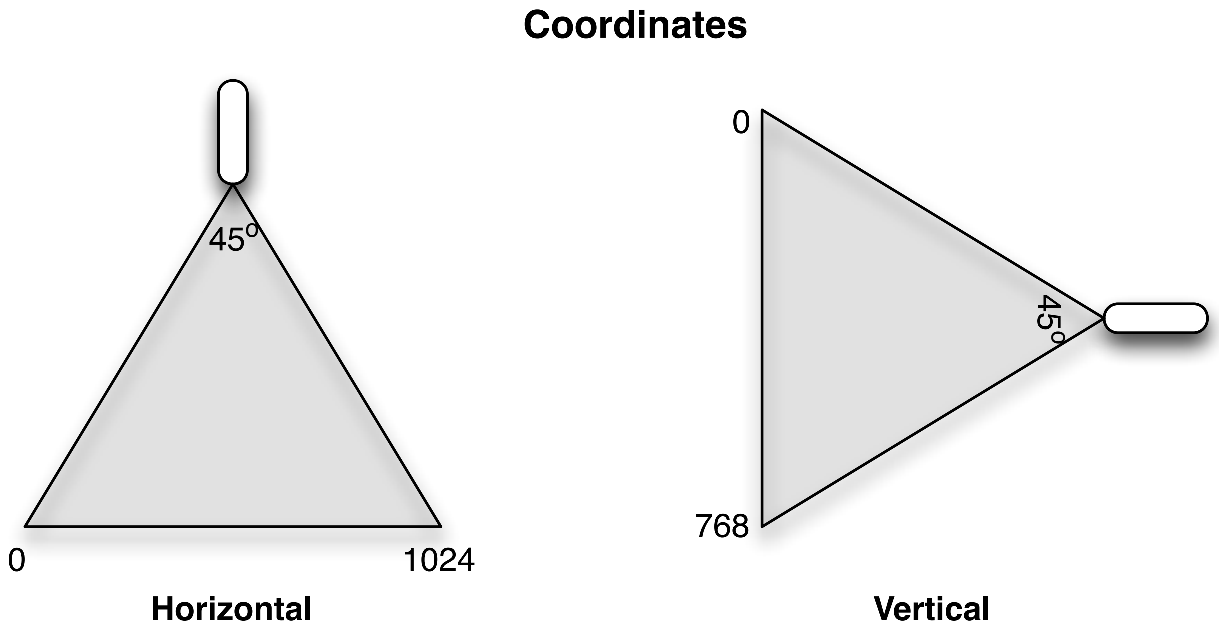


Figure 4 shows an important factor of this project; the range of the wiimote. The range of the wiimote (at a 45˚ angle) horizontally is 1024 pixels and vertically it is 768 pixels

Figure 5

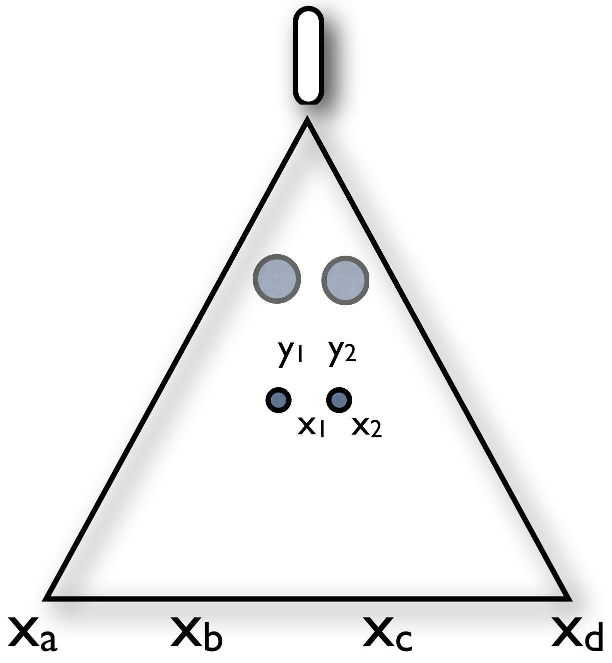
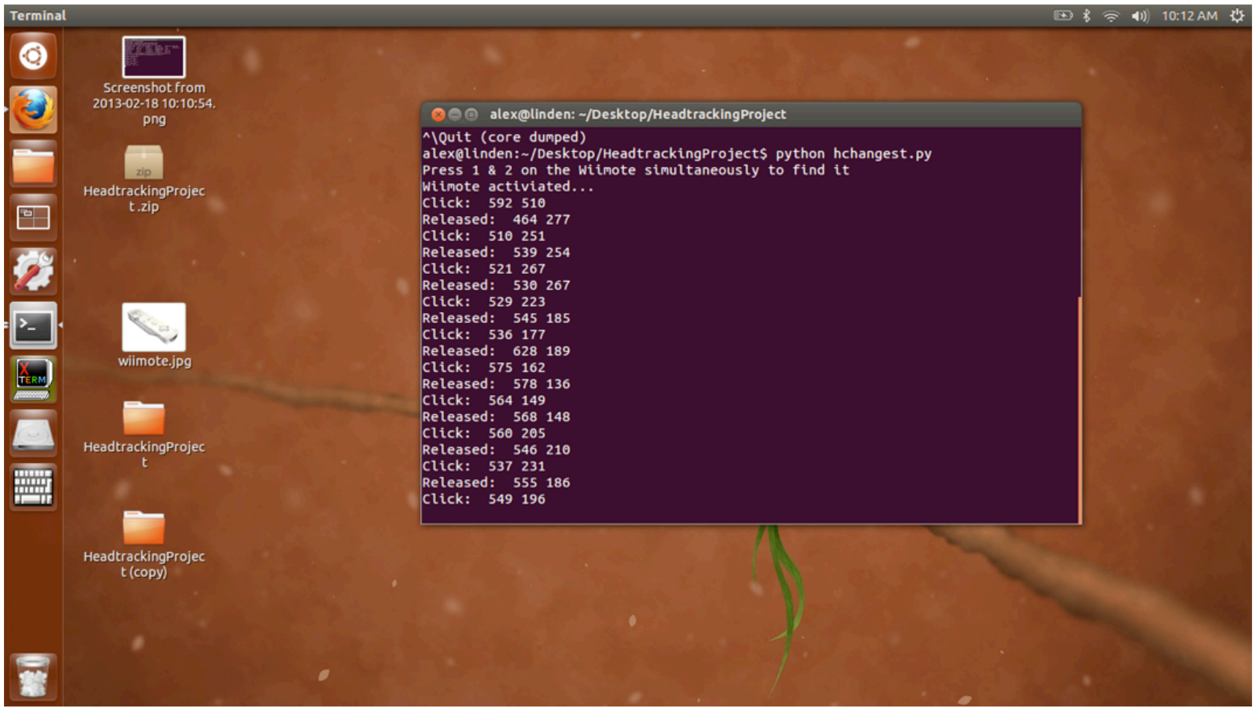


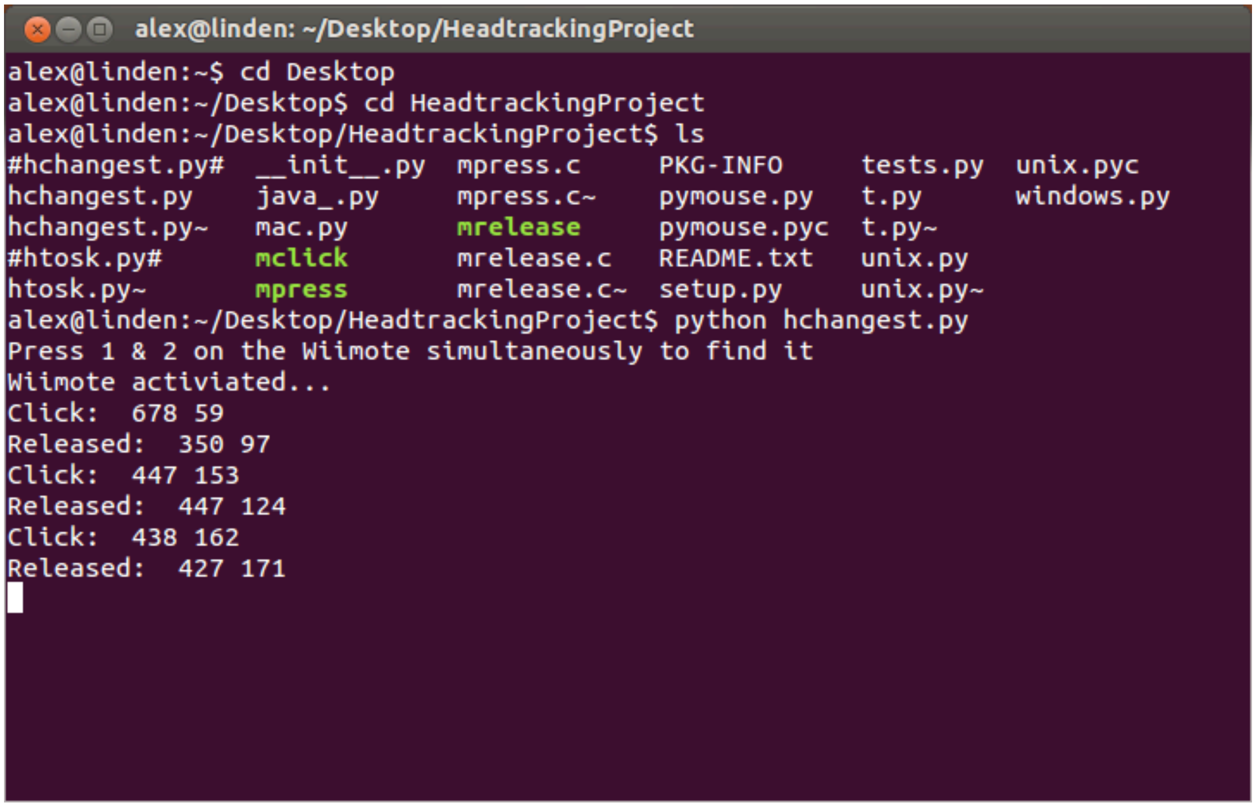
Figure 5 shows how the wiimote “sees” the infrared LEDs. The farther away, the smaller the LEDs; the closer, the bigger the LEDs.

Program Shots

The program is not particularly graphic so it does not have much to see but you can at least watch Terminal run it.


Terminal
10:12 AM
Screenshot from 2013-02-18 10:10:54.png
HeadtrackingProject.zip
wiimote.jpg
HeadtrackingProject
HeadtrackingProject (copy)
alex@linden: ~/Desktop/HeadtrackingProject

```
^\Quit (core dumped)
alex@linden:~/Desktop/HeadtrackingProject$ python hchangest.py
Press 1 & 2 on the Wiimote simultaneously to find it
Wiimote activiated...
Click:  592 510
Released:  464 277
Click:  510 251
Released:  539 254
Click:  521 267
Released:  530 267
Click:  529 223
Released:  545 185
Click:  536 177
Released:  628 189
Click:  575 162
Released:  578 136
Click:  564 149
Released:  568 148
Click:  560 205
Released:  546 210
Click:  537 231
Released:  555 186
Click:  549 196
```



```
alex@linden:~$ cd Desktop
alex@linden:~/Desktop$ cd HeadtrackingProject
alex@linden:~/Desktop/HeadtrackingProject$ ls
#hchangest.py#  __init__.py  mpress.c     PKG-INFO    tests.py  unix.pyc
hchangest.py    java_.py     mpress.c~    pymouse.py  t.py      windows.py
hchangest.py~   mac.py       mrelease     pymouse.pyc t.py~
#htosk.py#      mclick       mrelease.c   README.txt  unix.py
htosk.py~       mpress       mrelease.c~  setup.py    unix.py~
alex@linden:~/Desktop/HeadtrackingProject$ python hchangest.py
Press 1 & 2 on the Wiimote simultaneously to find it
Wiimote activiated...
Click:  678 59
Released:  350 97
Click:  447 153
Released:  447 124
Click:  438 162
Released:  427 171
```

As you can see the program asks you to press the 1 & 2 buttons on the wiimote in order to get a signal that bluetooth can read and recognize the wiimote. It then prints “Wiimote activated” which shows that a connection has been established from the computer to the wiimote over bluetooth. After that, it prints the coordinates of the click and then the release of the click. Notice the change of x and y for each click and it’s pair release, this was implemented so you can drag windows, or click and hold on a application to quit it, etc...

There are certain limitations of my program including: the range of the Wiimote, the program only runs on Linux because of CWiid, a certain library that is linux-only, and not all paralyzed people are able to use my program.

Conclusion

This project was successful to some extent and it did result in a program which allows a person to move the mouse by moving a infrared LED. Also it works for paralyzed people which was the entire point and basis of this project. This program has the basic needs to use a computer: mouse movement, and left clicking.

Acknowledgements

- Thanks go to Johnny Chung Lee for the idea of creating a head-tracking system using a Wiimote.
- Thanks go to Latchesar Ionkov for helping at the sticky parts of this project.
- Thanks go to Pauline Stephens for making this project possible.

Citations

- "Head Tracking for Desktop VR Displays using the WiiRemote - YouTube." YouTube, Web. 3 Jan. 2013. http://www.youtube.com/watch?v=Jd3-eiid-Uw.

  - Hejn, Kevin, and Jens Rosenkvist. "http://image.diku.dk/projects/media/rosenkvist.hejn.08.pdf." Headtracking using a wiimote. N.p., 28 Mar. 2008. Web. 2 Jan. 2013. image.diku.dk/projects/media/rosenkvist.hejn.08.pdf.

  - "Virtual Reality Using Wii Remote Head Tracking | NintendoFuse." NintendoFuse Everything Nintendo. N.p., n.d. Web. 3 Jan. 2013. http://www.nintendofuse.com/2007/12/21

  - Gotliv, Shaul, and Samuel Sayag. "http://gip.cs.technion.ac.il/files/Projects/2009/Head_Tracking_for_GIPview_VR_Displays_using_the_Wii_Remote_07072010.pdf." Head Tracking for GIPview VR Displays using the Wii Remote. Yaron Honen, n.d. Web. 2 Jan. 2013. gip.cs.technion.ac.il/files/Projects/2009/Head_Tracking_for_GIPview_VR_Displays_using_the_Wii_Remote_07072010.pdf.

  - Cunningham, Collin. "MAKE | Wiimote headtracking FPS demo." MAKE | DIY projects, how-tos, and inspiration from geeks, makers, and hackers. N.p., n.d. Web. 3 Jan. 2013. http://blog.makezine.com/2008/03/26/wiimote-headtracking-fps/.

- RABIN, RONI CARYN. "Study Raises Estimate of Paralyzed Americans - NYTimes.com." The New York Times - Breaking News, World News & Multimedia. N.p., n.d. Web. 30 Mar. 2013. <http://www.nytimes.com/2009/04/21/health/21para.html?_r=1&>.

Appendix 1: Python Code

```
#!/usr/bin/env python
###
# Copyright (c) 2013 Alex Ionkov
#
# Permission to use, copy, modify, and distribute this software for
any
# purpose with or without fee is hereby granted, provided that the
above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WAR-
RANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE
FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAM-
AGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN
AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
#
###

import sys
import cwiid, time import thread import os

from Xlib import X, display
from pymouse import PyMouse
from math import *
from numpy import * #Import matrix libraries

print "Press 1 & 2 on the Wiimote simultaneously to find it"
# To find remote, type the following in a Shell: # hcitool scan
# Then enter the address in the following command wiimote =
cwiid.Wiimote()
m = PyMouse()

x0 = 0
y0 = 0
x1 = 0
y1 = 0 x= 0
y= 0 dist = 0 camera = 0 xc = 0

yc = 0 distc = 0
```

```
cwiid.LED1_ON
wiimote.enable(cwiid.FLAG_MESG_IFC)
wiimote.rpt_mode = cwiid.RPT_IR | cwiid.RPT_BTN
print "Wiimote activiated..." #print "wiimote", wiimote

def wiiproc():
global x0, y0, x1, y1, x, y, dist, xc, yc, distc, pressed pressed =
False

while True:
messages = wiimote.get_mesg() # Get Wiimote messages for mesg in mes-
sages: # Loop through Wiimote Messages

def main(): global camera

if mesg[0] == cwiid.MESG_BTN: if mesg[1] == 8:

print "Calibrate: ", x, y, dist xc = x
yc = y
distc = dist

elif mesg[0] == cwiid.MESG_IR: # If message is IR data s = mesg[1][0]

if s:

x0 = s['pos'][0] y0 = s['pos'][1]

s = mesg[1][1] if s:

x1 = s['pos'][0] y1 = s['pos'][1]

        dx = x0 - x1
        dy = y0 - y1
        dist = math.sqrt(dx*dx + dy*dy)
        x = (x0 + x1) / 2
        y = (y0 + y1) / 2
        m.move(1024 - x, 768 -  y)
if abs(y0 - y1) > 100 and pressed == False: pressed = True

os.system("./mpress");

print "Click: ", 1024 - x, 768 - y elif abs(y0 - y1) < 100 and pressed
== True:

pressed = False os.system("./mrelease");
print "Released: ", 1024 - x, 768 - y

    wiimote.enable(cwiid.FLAG_MESG_IFC)
    wiimote.rpt_mode = cwiid.RPT_IR | cwiid.RPT_BTN
    wiiproc()
if __name__ == "__main__": wiiproc()
```

Appendix 2: C Source

Click Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <unistd.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>

void mousePress(int button)
{
    Display *display = XOpenDisplay(NULL);

    XEvent event;

    if(display == NULL)
    {
           fprintf(stderr, "Errore nell'apertura del Display !!!\n");
           exit(EXIT_FAILURE);
    }

    memset(&event, 0x00, sizeof(event));

    event.type = ButtonPress;
    event.xbutton.button = button;
    event.xbutton.same_screen = True;

    XQueryPointer(display, RootWindow(display, DefaultScreen(dis-
play)), &event.xbutton.root, &event.xbutton.window,
&event.xbutton.x_root, &event.xbutton.y_root, &event.xbutton.x,
&event.xbutton.y, &event.xbutton.state);

    event.xbutton.subwindow = event.xbutton.window;

    while(event.xbutton.subwindow)
    {
           event.xbutton.window = event.xbutton.subwindow;

           XQueryPointer(display, event.xbutton.window,
&event.xbutton.root, &event.xbutton.subwindow, &event.xbutton.x_root,
&event.xbutton.y_root, &event.xbutton.x, &event.xbutton.y,
&event.xbutton.state);
    }
    if(XSendEvent(display, PointerWindow, True, 0xfff, &event) == 0)
fprintf(stderr, "Errore nell'invio dell'evento !!!\n");

    XFlush(display);

    XCloseDisplay(display);
}

int main()
{
    mousePress(Button1);
}
```

Release click Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <unistd.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>

void mouseRelease(int button)
{
    Display *display = XOpenDisplay(NULL);

    XEvent event;

    if(display == NULL)
    {
           fprintf(stderr, "Errore nell'apertura del Display !!!\n");
           exit(EXIT_FAILURE);
    }

    memset(&event, 0x00, sizeof(event));

    event.type = ButtonRelease;
    event.xbutton.button = button;
    event.xbutton.same_screen = True;

    XQueryPointer(display, RootWindow(display, DefaultScreen(dis-
play)), &event.xbutton.root, &event.xbutton.window,
&event.xbutton.x_root, &event.xbutton.y_root, &event.xbutton.x,
&event.xbutton.y, &event.xbutton.state);
```

```
    event.xbutton.subwindow = event.xbutton.window;

    while(event.xbutton.subwindow)
    {
           event.xbutton.window = event.xbutton.subwindow;

           XQueryPointer(display, event.xbutton.window,
&event.xbutton.root, &event.xbutton.subwindow, &event.xbutton.x_root,
&event.xbutton.y_root, &event.xbutton.x, &event.xbutton.y,
&event.xbutton.state);
    }

    if(XSendEvent(display, PointerWindow, True, 0xfff, &event) == 0)
fprintf(stderr, "Errore nell'invio dell'evento !!!\n");

    XFlush(display);

    XCloseDisplay(display);
}

int main()
{
    mouseRelease(Button1);
}
```