

Adding support for a mouse in libratbag

...

Gaming mice

Per-game settings

Stored in on-board memory

Grouped in profiles

Resolution

Button mapping

LED colors and effects

Libratbag

By Peter Hutterer and Benjamin Tissoires

Unified interface to support all devices

Supports common features across many HW vendors and models

Integration into the desktop environments

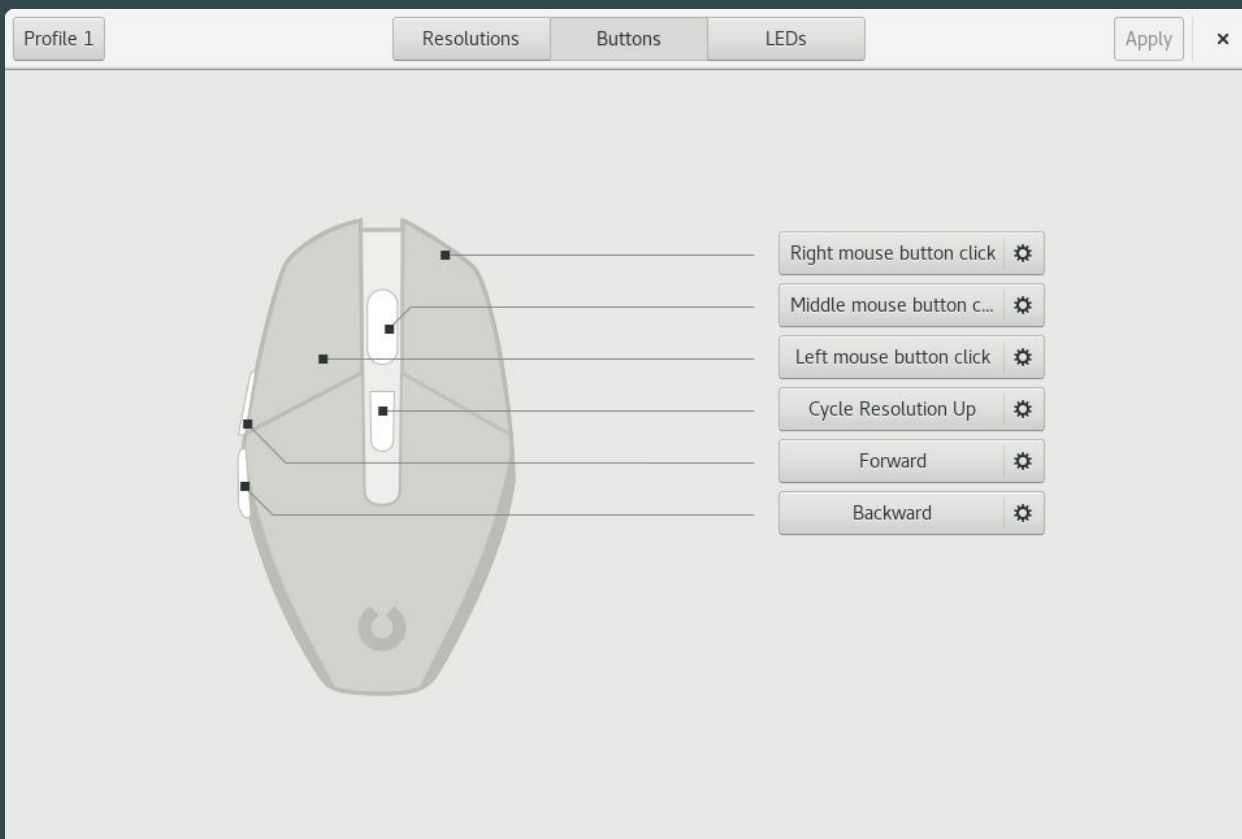
Architecture

User interface clients

Ratbagd daemon

Libratbag lib

Driver backends



Architecture

User interface clients

Ratbagd daemon

Libratbag lib

Driver backends

Architecture

User interface clients

Ratbagd daemon

Libratbag lib

Driver backends ←

Discovering devices

Matches connected devices to a list of vendor and product IDs

Probes the matching devices

Sets the capabilities of the device

Loads the current on-board configuration to libratbag

Changing settings

Clients alters the settings state in libratbag

When done, it asks libratbag to commit the changes to the device

Drivers use protocol specific calls to transfer the altered settings

Matching devices

.device files in data/devices

[Device]

Name=SteelSeries Kinzu V2

DeviceMatch=usb:1038:1378

Driver=steelseries

Svg=steelseries-kinzu-v2.svg

Matching devices

[Driver/steelseries]

DeviceVersion=1

Buttons=3

Leds=0

DpiList=400;800;1600;3200

Data structure

Device

Profiles

Resolutions

Buttons

LEDs

Capabilities

RATBAG_DEVICE_CAP_QUERY_CONFIGURATION,
RATBAG_DEVICE_CAP_RESOLUTION,
RATBAG_DEVICE_CAP_SWITCHABLE_RESOLUTION,
RATBAG_DEVICE_CAP_PROFILE,
RATBAG_DEVICE_CAP_SWITCHABLE_PROFILE,
RATBAG_DEVICE_CAP_DISABLE_PROFILE,
RATBAG_DEVICE_CAP_DEFAULT_PROFILE,
RATBAG_DEVICE_CAP_BUTTON,
RATBAG_DEVICE_CAP_BUTTON_KEY,
RATBAG_DEVICE_CAP_BUTTON_MACROS,
RATBAG_DEVICE_CAP_LED,

Reverse engineering setup

Run windows in a VM

Install vendor configurator

Redirect the device to the VM

Wireshark

Monitor USB bus

Analyze -> Decode as... -> USBHID

Reverse engineering the protocol

Change a setting with the vendor software

Inspect the content of the USB packet

Repeat

Some mice use checksums

Endianness

DEMO

Writing the driver

Create a driver in src/

Register the driver

Add a .device file

Implement these function pointers

probe

commit

set_active_profile

remove

Probe

Basic sanity checks if the driver is relevant for the device

Initialize data structures in libratbag

Query all configuration from device

Set relevant capabilities

Commit

Push all changes to the device

Loop over profiles, resolutions, buttons, and LEDs

Can use dirty bit to check what to push

Set_active_profile

Change the active profile if possible

Remove

Free allocated memory

Helper functions

Converters for endianness

Tables of HID keys

Piper

Jente Hidskes worked on it as part of GSoC

Communicates with the daemon over DBus

Set up the GUI layout based on the SVG provided by libratbag

SVG

Layout of the device

Buttons

LEDs

Leader lines

Use light colors

Follow rules in `data/gnome/README.md`

SVG

3 layers: Device, Buttons, LEDs

Take a picture, add as base layer. Add transparent layer on top and draw on that

Button objects are named e.g. “button0”, “button1”, etc

Led objects are named e.g. “led0”, “led1”, etc

SVG leader lines

Small square at edge named e.g. “button1-leader”

A line connects to the button/LED

If the leader line goes to the right use “text-align:start”.

On the left use “text-align:end”

Inkscape demo