Gametrak


(don EDNA, juin 2025)

Le gametrak est un contrôleur original, qui mesure les positions en 3D, breveté par In2Games et paru au début des années 2000, vendu par exemple avec des jeux comme «Real World Golf» sur PS2

1. https://en.wikipedia.org/wiki/Gametrak
2. test (2006) : https://arstechnica.com/gaming/2006/05/gametrak/

Il en existe plusieurs versions, celui est du type Version 2 Rev. 1, il n'est utilisable qu'avec une PS2 et les jeux compatibles, et pas utilisable sur un PC, contrairement à la version Rev.2. (modifiable pour PC). Il existe aussi des versions spécifiques pour PC / XBox
+ **il y a une soudure à refaire sur le circuit imprimé interne** : sur le port micro-jack qui permet un interrupteur à pied

Tentative d'utilisation

En le branchant sur un ordi, il est reconnu par `lsusb`

```
Bus 002 Device 015: ID 14b7:0982 In2Games Ltd. Game-Trak V1.3
Bus 002 Device 014: ID 0f30:001c Jess Technology Co., Ltd PS3 Guitar Controller Dongle
```

(PS3 Guitar Controller Dongle est associé aussi, sans trop savoir pourquoi : être utilisé avec guitar hero ?)

La commande `sudo dmesg |tail -n 80` renvoie

```
35467.625612] usb 2-1: new full-speed USB device number 14 using xhci_hcd
[35467.776128] usb 2-1: New USB device found, idVendor=0f30, idProduct=001c, bcdDevice= 3.12
[35467.776143] usb 2-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[35467.776148] usb 2-1: Product: Ɲb
[35467.776151] usb 2-1: Manufacturer: Ɲb
[35467.777440] hub 2-1:1.0: USB hub found
[35467.777782] hub 2-1:1.0: 3 ports detected
[35468.070265] usb 2-1.1: new full-speed USB device number 15 using xhci_hcd
[35468.176857] usb 2-1.1: New USB device found, idVendor=14b7, idProduct=0982, bcdDevice= 0.01
[35468.176872] usb 2-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[35468.176877] usb 2-1.1: Product: Game-Trak V1.3
[35468.176880] usb 2-1.1: Manufacturer: In2Games Ltd.
[35468.180028] input: In2Games Ltd. Game-Trak V1.3 as
/devices/pci0000:00/0000:00:14.0/usb2/2-1/2-1.1/2-1.1:1.0/0003:14B7:0982.000C/input/input33
[35468.180424] hid-generic 0003:14B7:0982.000C: input,hidraw2: USB HID v1.01 Joystick [In2Games Ltd. Game-Trak V1.3] on
usb-0000:00:14.0-1.1/input0
```

Mais impossible de récupérer les données HiD, les tests avec ChucK ou un script python adapté n'ont rien donné. Les formats de communication utilisés sur PS2 pour le USB-HID ne sont pas les mêmes que sur PC, il ne suffit pas d'avoir un périphérique USB pour pouvoir le brancher sur un PC ou vice versa, triste nouvelle!

evtest permet aussi de voir les events liés au périphériques d'entrée (https://manned.org/evtest)

```
sudo evtest
```

Évènements bruts HID (hidraw)

Voici un script python3 pour retrouver les valeurs brutes HID (merci les LLM), à démarrer avec `sudo python3 hidraw_reader.py –device /dev/hidraw2` à adapter en fonction de ce que qu'a relevé dmesg ci-dessus (pour les options en ligne de commande, voir commentaires du script)

hidraw_reader.py (cliquer pour afficher le code)

hidraw_reader.py

```
#!/usr/bin/env python3
"""
hidraw_reader.py - lire les reports HID bruts depuis /dev/hidrawN et afficher hex + timestamp
Usage examples:
  # lister les hidraw disponibles
  python3 hidraw_reader.py --list
  # lire un device précis
  python3 hidraw_reader.py --device /dev/hidraw2
  # lire en spécifiant vendor:product (hex, ex: 046d:c52b)
  python3 hidraw_reader.py --vidpid 046d:c52b
  # écrire la sortie dans un fichier
  python3 hidraw_reader.py --device /dev/hidraw2 --output dump.log

Fonctions:
 - détection simple des attributs USB via /sys (idVendor,idProduct,manufacturer,product)
 - lecture non-bloquante avec select, affichage hex + ascii + timestamp
 - gestion propre des droits (tester en root si permission refusée)
"""
import os, sys, time, select, binascii

def find_hidraw_devices():
    """Retourne liste de dicts: {'path':'/dev/hidraw0','hid':'hidraw0','vid':'046d', 'pid':'c52b', 'manufacturer':...,
'product':...}"""
    res = []
    devs = sorted([d for d in os.listdir('/dev') if d.startswith('hidraw')])
    for d in devs:
        path = os.path.join('/dev', d)
        info = {'path': path, 'hid': d, 'vid': None, 'pid': None, 'manufacturer': None, 'product': None}
        # sys path e.g. /sys/class/hidraw/hidraw0/device -> may be a symlink to ../../../.../usbX/X-X:1.X/...
        try:
            sysnode = os.path.join('/sys/class/hidraw', d, 'device')
            if os.path.exists(sysnode):
                # climb up parents to find idVendor/idProduct (up to 6 levels)
                p = os.path.realpath(sysnode)
                for _ in range(6):
                    vendor_file = os.path.join(p, 'idVendor')
                    product_file = os.path.join(p, 'idProduct')
                    if os.path.exists(vendor_file) and os.path.exists(product_file):
                        with open(vendor_file,'r') as f: info['vid'] = f.read().strip()
                        with open(product_file,'r') as f: info['pid'] = f.read().strip()
                        # optional fields
                        man = os.path.join(p, 'manufacturer')
                        prod = os.path.join(p, 'product')
                        if os.path.exists(man):
                            try:
                                with open(man,'r') as f: info['manufacturer'] = f.read().strip()
                            except:
                                pass
                        if os.path.exists(prod):
                            try:
                                with open(prod,'r') as f: info['product'] = f.read().strip()
                            except:
                                pass
                        break
                    # climb up
                    parent = os.path.dirname(p)
                    if parent == p:
                        break
                    p = parent
        except Exception as e:
            pass
        res.append(info)
    return res

def hexdump_line(data, base_offset=0):
    hexpart = ' '.join(f"{b:02x}" for b in data)
    asciipart = ''.join((chr(b) if 32 <= b < 127 else '.') for b in data)
    return f"{base_offset:08x}  {hexpart:<48}  |{asciipart}|"

def print_report(ts, bts, out):
    # print timestamp and hexdump-like line(s)
    header = f"[{time.strftime('%Y-%m-%d %H:%M:%S', time.localtime(ts))}.{int((ts - int(ts))*1000):03d}] len={len(bts)}"
    print(header, file=out)
    # chunk into 16 bytes per line
    for i in range(0, len(bts), 16):
        chunk = bts[i:i+16]
        print(hexdump_line(chunk, i), file=out)
    out.flush()

def open_nonblocking(path):
    import fcntl, os, stat
```

```
    fd = os.open(path, os.O_RDONLY | os.O_NONBLOCK)
    # make file descriptor a file object for read()
    return os.fdopen(fd, 'rb', buffering=0)

def main():
    import argparse, sys, time
    parser = argparse.ArgumentParser(description="hidraw reader - affichage des reports HID bruts")
    parser.add_argument('--list', action='store_true', help='Lister les /dev/hidraw* détectés')
    parser.add_argument('--device', '-d', help='Chemin vers /dev/hidrawN (ex: /dev/hidraw2)')
    parser.add_argument('--vidpid', help='Filtrer par vendor:product hex (ex: 046d:c52b)')
    parser.add_argument('--interval', type=float, default=0.2, help='Timeout select en secondes (default 0.2)')
    parser.add_argument('--output', '-o', help='Chemin fichier pour écrire la sortie (stdout si non fourni)')
    args = parser.parse_args()

    devs = find_hidraw_devices()
    if args.list:
        if not devs:
            print("Aucun /dev/hidraw trouvé.")
            return 0
        print("Detected hidraw devices:")
        for d in devs:
            print(f" {d['path']}  vid:pid={d.get('vid') or '----'}:{d.get('pid') or '----'}  manufacturer={d.get('manufacturer') or
''} product={d.get('product') or ''}")
        return 0

    target = None
    if args.device:
        target = args.device
    elif args.vidpid:
        v,p = args.vidpid.split(':')
        for d in devs:
            if d.get('vid') and d.get('pid') and d['vid'].lower() == v.lower() and d['pid'].lower() == p.lower():
                target = d['path']
                break
        if not target:
            print(f"Pas de périphérique hidraw correspondant à {args.vidpid}", file=sys.stderr)
            return 2
    else:
        # if only one device, choose it
        if len(devs) == 1:
            target = devs[0]['path']
        else:
            print("Plusieurs périphériques hidraw détectés — précisez --device ou --vidpid, ou utilisez --list pour voir.",
file=sys.stderr)
            return 3

    out = sys.stdout
    if args.output:
        out = open(args.output, 'w', buffering=1)

    print(f"Opening {target} (non-blocking) ...", file=sys.stderr)
    try:
        f = open_nonblocking(target)
    except PermissionError:
        print("Permission refusée: lancez en root ou ajustez udev pour accéder à /dev/hidraw*", file=sys.stderr)
        return 4
    except FileNotFoundError:
        print("Fichier non trouvé:", target, file=sys.stderr)
        return 5

    print("Entering read loop. Press Ctrl-C to stop.", file=sys.stderr)

    try:
        while True:
            r, _, _ = select.select([f], [], [], args.interval)
            if not r:
                # timeout; continue to allow Ctrl-C responsiveness
                continue
            try:
                data = f.read(64)  # typical hid report length; if returns b'' skip
            except BlockingIOError:
                continue
            if not data:
                # no data read; continue
                continue
            ts = time.time()
            print_report(ts, data, out)
    except KeyboardInterrupt:
        print("\nStopped by user.", file=sys.stderr)
        return 0

if __name__ == '__main__':
    sys.exit(main())
```

Renseignons-nous

En recherchant un peu plus sur les gametrak, je note qu'il y a plusieurs versions

1. Version 1 : autre facteur de forme, facile à reconnaître

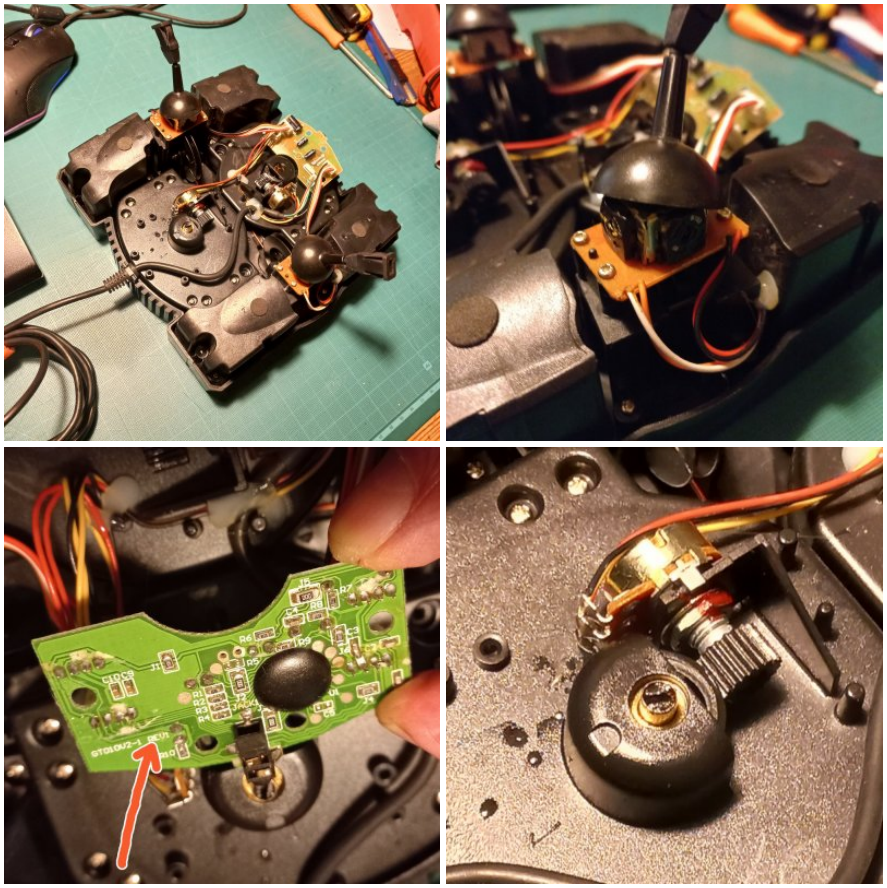
2. Version 2 Rev 1.x : USB-HID compatible PS2 uniquement (*)
3. Version 2 Rev 2.x, version PS2 fournie avec les jeux : USB-HID compatible PC (avec une légère modification à faire sur le circuit imprimé)
4. Version 2 Rev 2.x, version PC / XBox : USB-HID compatible PC sans modification.

(C'est probablement une seule soudure qui fait la différence entre les version V2 Rev2 PS2 et PC/XBox)

(*) La version 2 Rev 1 peut être réutilisée en refabriquant l'électronique à partir d'une carte arduino ou teensy, mais ça nécessiterait pas mal de patience et aurait aussi un coût...

Hardware

Avant de découvrir tout ça, je l'ai démonté pour vérifier que tout ça se passe bien au niveau des capteurs (et lire la version du pcb), à l'intérieur on trouve 2 doubles potentiometres pour XY, type joystick, assez classiques et 2 potentiometres qui mesurent l'élongation en Z avec un système de poulies. C'est quand même bien ingénieux tout ça!
En testant les potentiomètres tout fonctionne d'un point de vue mécanique et électrique



Tout au bout de la flèche, sur le PCB, on peut lire qu'il s'agit d'une version Rev.1

Comment reconnaître la version compatible ?





Elle est (semble t'il) vendue dans une boîte de couleur orange (à droite, ci-dessus), c'est la version vendue avec real world golf 2007. On peut chercher par exemple “gametrak + jeu real world golf 2007 ps2”. En oct. 2025, je trouve cette version sur leboncoin pour 20e port compris, à voir si cette hypothèse se confirme!

→ confirmé : la version ci-dessous en boîte orange correspond bien au modèle PS2 qui peut être adapté en périphérique pour ordinateur avec une petite soudure. Ci-dessous : ce que j'ai reçu. Il doit exister des

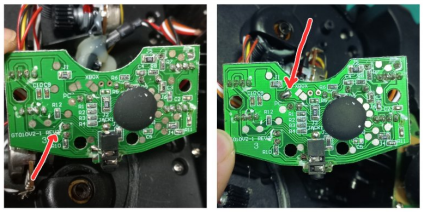


Modification pour utiliser un Gametrak en périphérique PC

D'après alex at x37v.



Passer un tournevis à travers les 6 pastilles de mousse sous le gametrak, dévisser pour ouvrir le boîtier.
Dévisser le circuit imprimé indiqué sur la photo ci-dessus.



Il s'agit bien d'une version Rev.2 (photo de gauche). Sur l'envers du circuit imprimé, souder les 2 pastilles correspondant au label «PC» (photo de droite)

On peut tester que le gametrak est bien reconnu avec https://samiare.github.io/Controller.js/ ou https://hardwaretester.com/gamepad

Suite

Tests avec comme point de départ musical ces scripts du CCRMA pour Chuck : https://ccrma.stanford.edu/courses/220a-fall-2018/homework/5/

Ressources

Le site de gametrak en 2003 : https://web.archive.org/web/20040206082043/http://www.game-trak.com/blackwind.htm

Différents hacks de gametrak pour en faire des contrôleurs musicaux

- https://gareth.prof/2014/10/23/gametrak-game-controller-project/
- https://www.davidgoldberg.net/gesticularcontroller
- https://maxkrien.com/gametrak-instrument-controller
- https://janoc.rd-h.com/archives/129 : fabrication d'un circuit USB sur mesure!
- https://github.com/anoujeremia/gametrak refabrication d'un circuit sur base teensy pour en faire un contrôleur midi

In2Games a par la suite développé quelques jeux pour PS2 avec des contrôleurs de mouvements originaux, sous le nom de gamme «realplay»:

- realplay puzzlesphere (avec une sphère sans fil)
- test : https://www.gamespot.com/articles/realplay-puzzlesphere-hands-on/1100-6180149/
- realplay pool, realplay golf, realplay racing, voir https://gamefaqs.gamespot.com/games/company/76569-in2games

Autres développements logiciels open-source

- https://github.com/casiez/libgametrak

Article extrait de : http://www.lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://www.lesporteslogiques.net/wiki/materiel/gametrak/start**
Article mis à jour: **2025/12/08 13:50**