Enregistrement et diffusion de trames GPS pour la cartographie libre

J.-M Friedt & É. Carry
Association Projet Aurore, Besançon

Présentation disponible à `http://jmfriedt.free.fr`

August 5, 2006

Introduction: objectifs de ce projet

- Constat : toutes les données européennes de cartographie sont propriétaires (IGN en France) et ne sont pas accessibles au public
- une façon de remédier à cet état : créer des bases de données libres à des fins de cartographie
- une façon d'atteindre cet objectif est d'équiper un maximum de voyageurs de récepteurs GPS puis de mettre en commun les trajets suivis
- cependant, les récepteurs GPS capables de stocker les traces sont coûteux

$\Rightarrow$ ce projet vise à réaliser un récepteur peu coûteux dédié au stockage de traces GPS (*pas* de navigation)

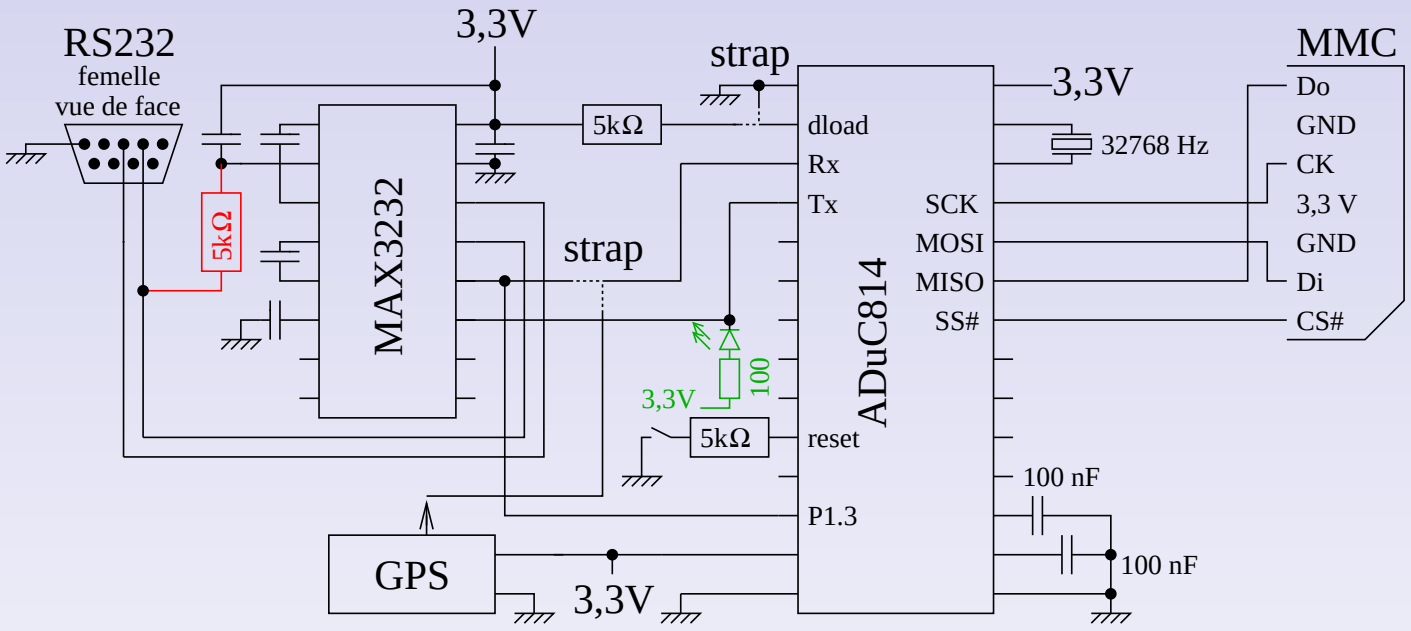
Plan de la présentation

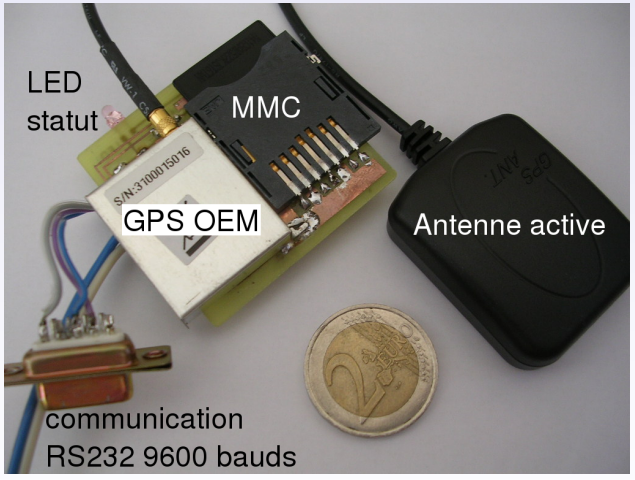
- circuit électronique : acquisition, stockage et restitution des traces
- affichage et validation des traces (Google Maps)
- affichage des traces en 3D (Google Earth)
- partage des traces (UPCT, `www.upct.org`)
- fusion des traces avec d'autres données géographiques (M_MAP pour Matlab, GRASS)
- perspective : transmission des traces en temps réel

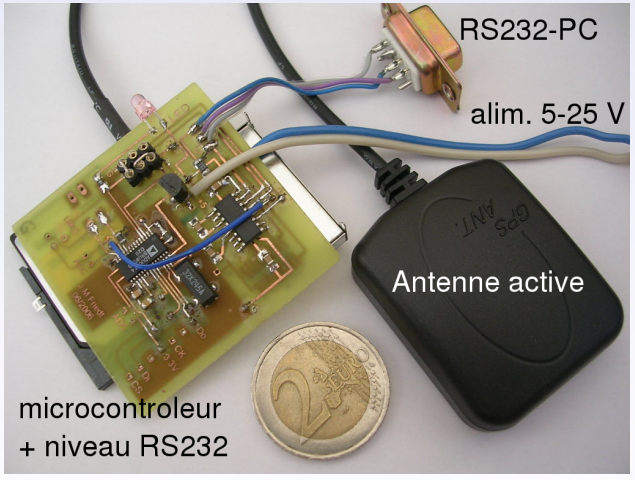
Choix technologiques

- Microcontroleur 8 bits ADuC814
  - disponible gratuitement en échantillon chez Analog Devices
  - petit (TSSOP28)
  - possède tous les périphériques nécessaires (ADC, UART, SPI)
  - cœur de 8051 : assembleur (`asxxxx`, `asl`)/compilateur (`sdcc`) libres
  - petite zone mémoire non-volatile pour le stockage de données
  - programme en mémoire flash (EEPROM)
  - programmation par le port série (bootloader) ne nécessitant pas de matériel dédié
- Stockage de masse non volatile sur MultiMediaCard : facilement disponible (électronique grand publique), simple à programmer (SPI, 4 fils), remplaçable
- Récepteurs GPS OEM chez Lextronic (`www.lextronic.fr`), $\simeq$80 euros
  - Laipac UV40 basé sur un composant Sony `www.lextronic.fr/laipac/uv40.htm`
  - Laipac PG31 `www.lextronic.fr/laipac/tf30.htm`

Schéma de principe







Logiciel embarqué

- assembleur 8051 très bien documenté (370 lignes de code)
- deux logiciels en un : acquisition des trames et restitution
- sélection du logiciel en fonction de la présence/absence du cable RS232
- restitution de trames ASCII $\Rightarrow$ n'importe quel terminal convient (`minicom`, `cat < /dev/ttySi`, Hyperterminal ...)
- protocole d'accès à la MMC décrit auparavant [1]
- la MMC conserve en RAM le bloc en cours d'écriture (512 octets) et une fois le bloc plein, on donne l'ordre de stockage
- absence de buffer dans l'UART $\Rightarrow$ perte de quelque caractères pendant le transfert des données de la RAM tampon à la mémoire non volatile de la MMC

[1] J.-M Friedt & É. Carry, *Enregistrement de trames GPS ...*, GNU/Linux Magazine, Mars 2006, pp.80-91

Autonomie

Deux limites principales pour l'autonomie : le stockage de données et la source d'énergie

- format NMEA (ASCII) génère entre 800 et 1300 KB/h
  $\Rightarrow$ une carte 128 MB ($\simeq$20 euros) contient plus de 1 semaine de données à raison de 14 h/jour
- consommation de l'ordre de 100 mA (70% récepteur GPS, 30% MMC) soit 20 h d'autonomie sur une batterie 2000 mAh
- régulateur 3,3 V nécessite $>$3,8 V en entrée $\Rightarrow$ 4 accumulateurs NiMH ou NiCd suffisent

Les trames GPS

Motorola Oncore fournit des données au format binaire
La plupart des récepteurs actuels supportent un format standard **NMEA**

```
$GPGSA,A,3,10,21,26,29,27,08,28,,,,2.6,1.4,2.2*39
$GPRMC,112839.696,A,4754.0590,N,00158.9989,E,3.43,122.05,010806,,*06
$GPGGA,112840.696,4754.0585,N,00159.0000,E,1,07,1.4,143.8,M,47.6,M,0.0,0000
$GPGSA,A,3,10,21,26,29,27,08,28,,,,,2.6,1.4,2.2*39
$GPRMC,112840.696,A,4754.0585,N,00159.0000,E,3.31,118.92,010806,,*0E
$GPGGA,112841.696,4754.0581,N,00159.0012,E,1,07,1.4,143.8,M,47.6,M,0.0,0000
$GPGSA,A,3,10,21,26,29,27,08,28,,,,,2.6,1.4,2.2*39
$GPGSV,2,1,07,10,69,208,46,08,64,059,44,29,47,293,46,28,40,130,45*72
$GPGSV,2,2,07,267,289,46,27,35,057,42,21,11,323,39*43
$GPRMC,112841.696,A,4754.0581,N,00159.0012,E,3.33,119.86,010806,,*0E
$GPGGA,112842.695,4754.0576,N,00159.0023,E,1,07,1.4,143.9,M,47.6,M,0.0,0000
```

Extraction de la position

- on ne conserve que les informations de position $GPGGA

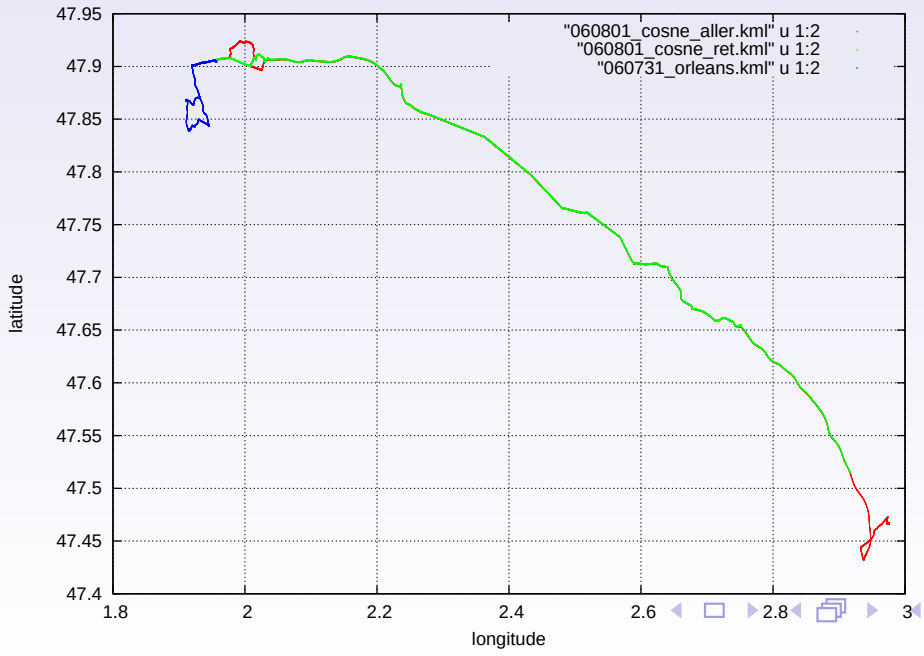
  `$GPGGA,112840.696,4754.0585,N,00159.0000,E,1,07,1.4,143.8,M,47.6,M,0.0`

- on ne garde que les champs de latitude, longitude, altitude (3, 5, 10)
- on vérifie la forme de ces champs (4+4, 5+4 chiffres)
- ```
  grep -a GGA ${nom} | cut -d, -f3,5,10 | sed 's/,/ /g' | grep -v "[A-Z]" |
  grep "^[0-9][0-9][0-9][0-9]\.[0-9][0-9][0-9][0-9]\ [0-9][0-9][0-9][0-9][0-9]\.[0-9][0-9][0-9][0-9]\ ...."
  | grep -v $'\xff' | grep "^4"
  ```
- on obtient des données sous forme degrés.minutes_fractions

  ```
  4753.9537 00159.2386 144.8
  4753.9532 00159.2397 144.4
  4753.9525 00159.2407 144.2
  ```

- utilisation de Matlab/Octave pour convertir en degrés.décimales :

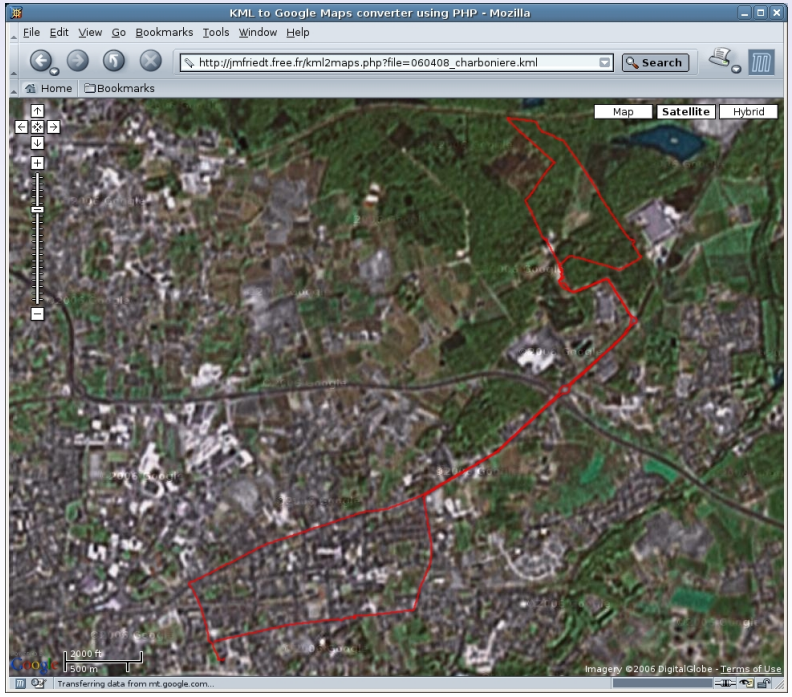
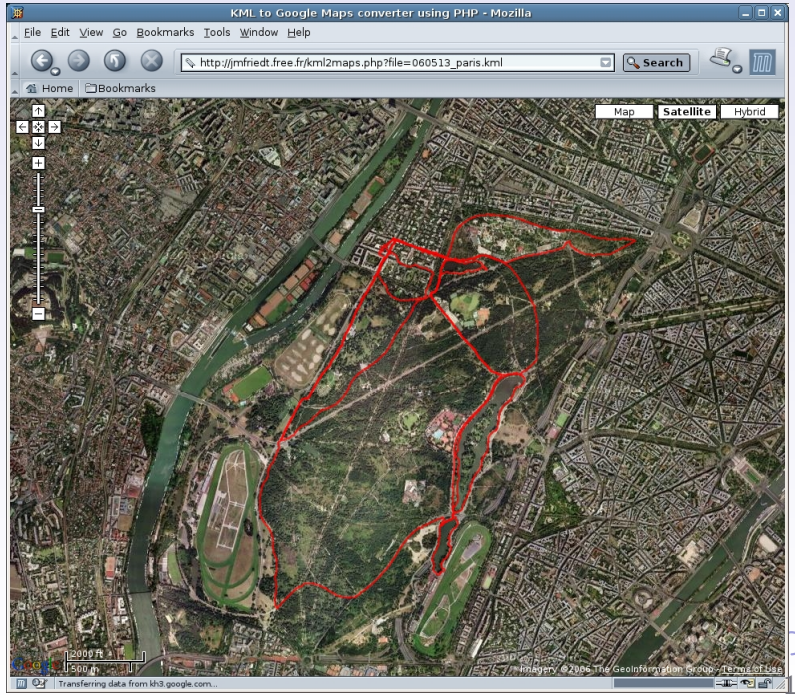
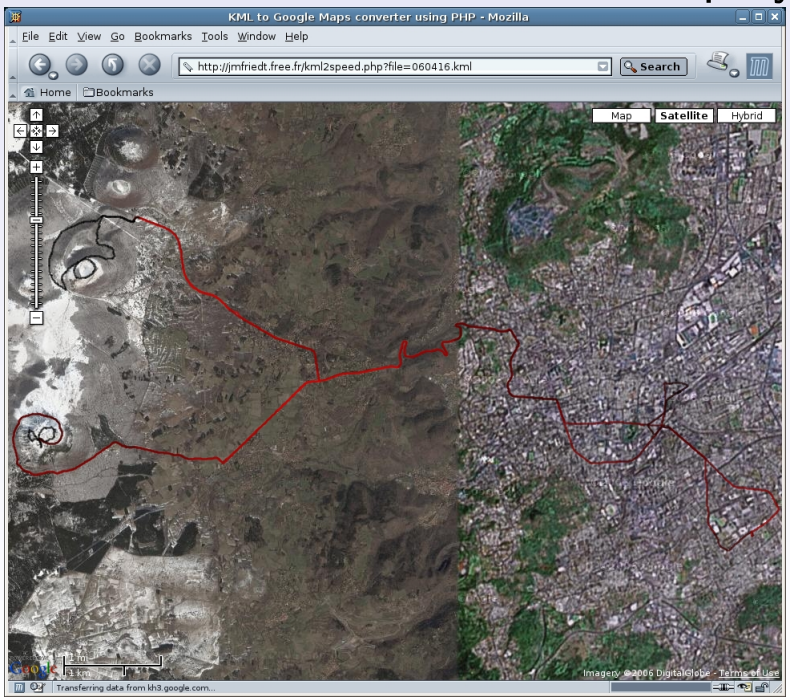
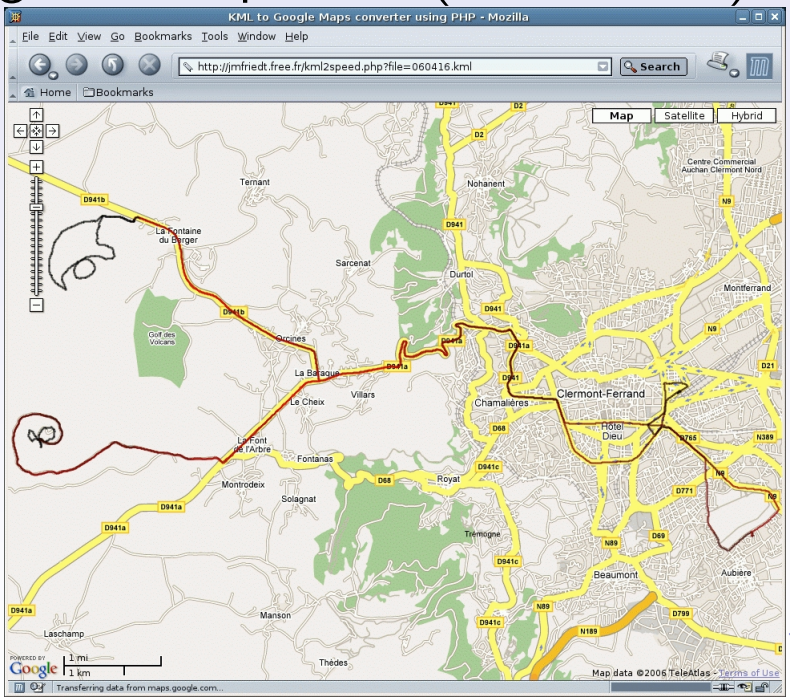
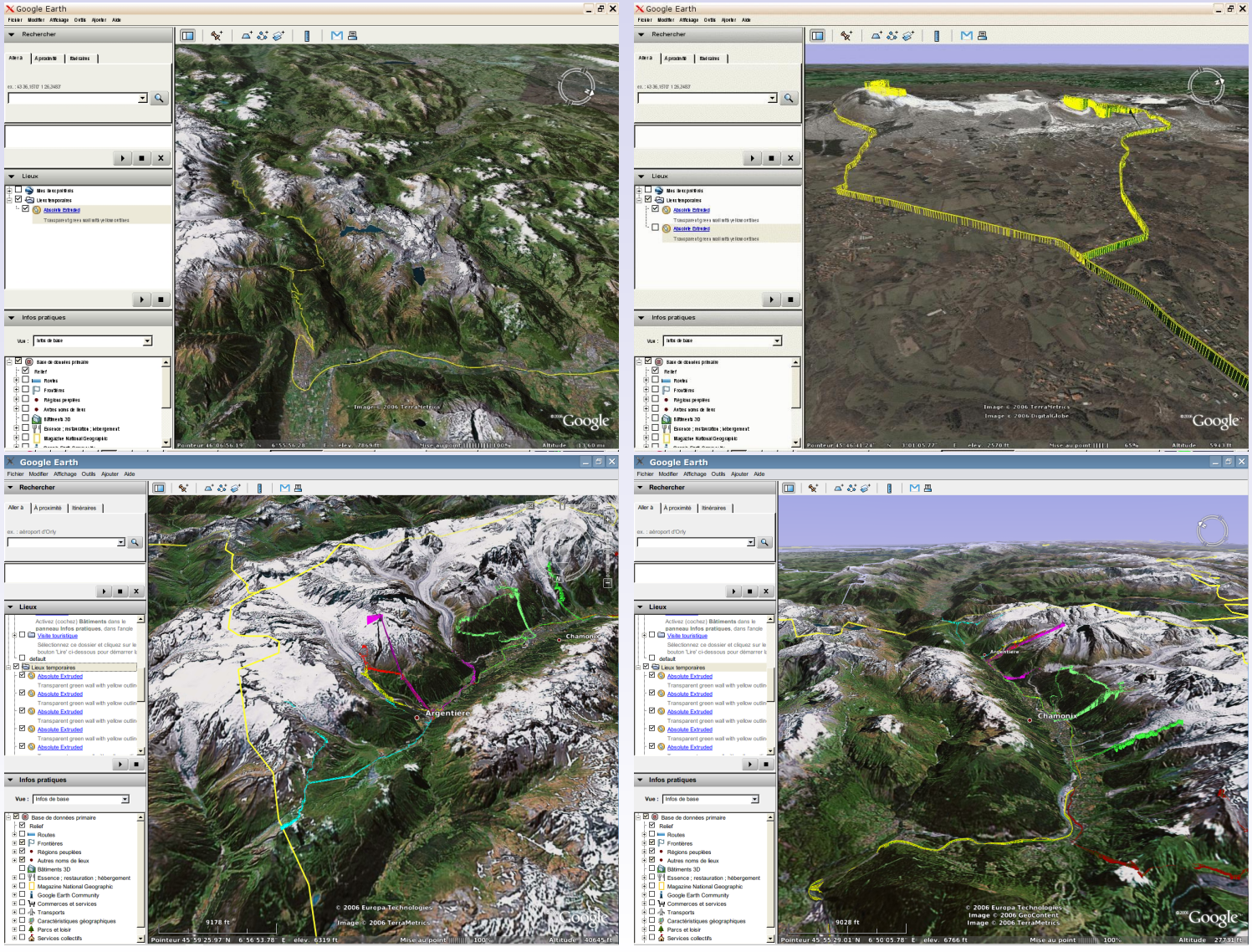
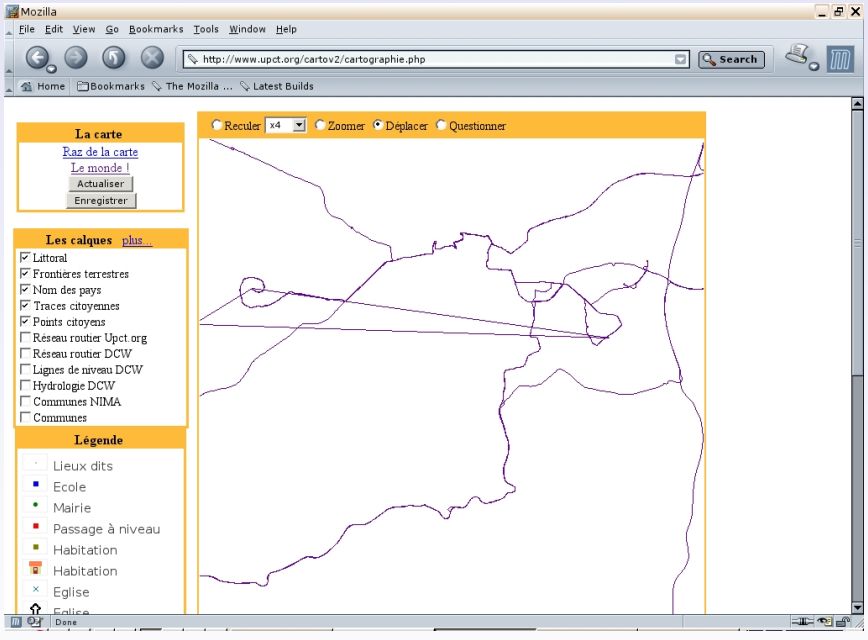
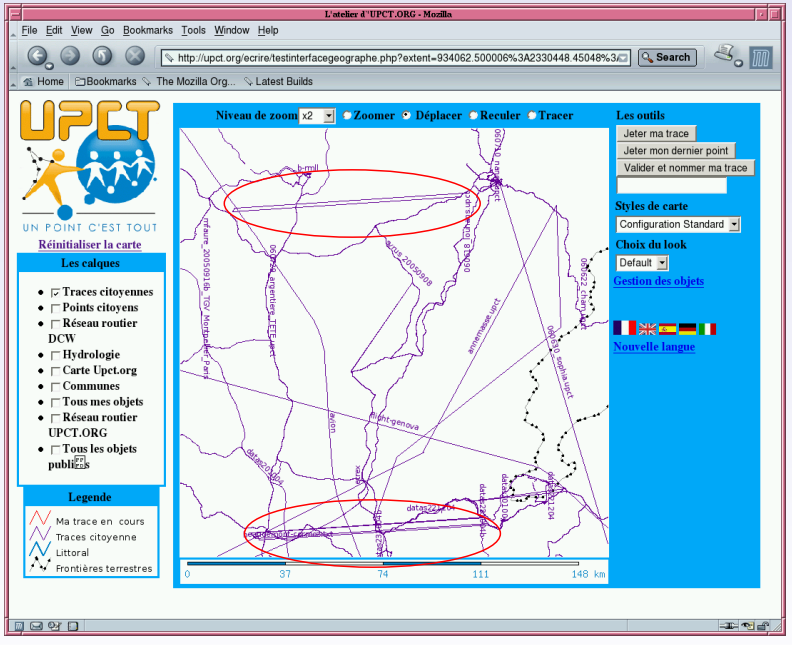
  ```
  function sortie=deg2dec(entree)
  sortieX=floor(entree(:,1)/100);
  sortieX=sortieX+((entree(:,1)/100)-sortieX)/60*100;
  sortieY=floor(entree(:,2)/100);
  sortieY=sortieY+((entree(:,2)/100)-sortieY)/60*100;
  sortie=[sortieY sortieX entree(:,3)];
  ```

  peut aussi se faire en `awk`
  ```

Extraction de la position (2)

- sous Octave :
  `load 060801_cosne_ret.dat`
  `s=deg2dec(X060801_cosne_aller); save -ascii 060801_cosne_ret.kml s`
- on obtient des données sous forme degrés.décimales
  `1.9873 47.8992 144.8000`
  `1.9873 47.8992 144.4000`
  `1.9873 47.8992 144.2000`
- tracé sous Gnuplot : `plot "060801_cosne_aller.kml" u 2:1`



Affichage et validation des traces (Google Maps)

- Google Maps [2] fournit une API pour charger des cartes depuis leur serveur sur sa page web (javascript)
- aucun logiciel à installer en local
- vue en 2D, photographies aériennes ou carte des routes
- nécessité de s'enregistrer pour obtenir une clé d'accès associée à son site web : `http://www.google.com/apis/maps/signup.html`
  $\Rightarrow$ obtention d'un exemple de base dont nous allons nous inspirer pour ajouter nos traces

[2] R. Gibson & S. Erle, *Google Maps Hacks – Tips and tools for geographic searching and remixing*, O'Reilly (2006)

Premier exemple de base

- la clé est associée au site `http://friedtj.free.fr/`

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Google Maps JavaScript API Example</title>
    <script src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAAQ5
iKlBqfwApuWu7aIzUiPRTELJePVJ4bAZX18VT1uNz_yt5gwxTQjL4ngbxeSIY1ZypSj6avKm_2Ig"
      type="text/javascript"></script>
    <script type="text/javascript">

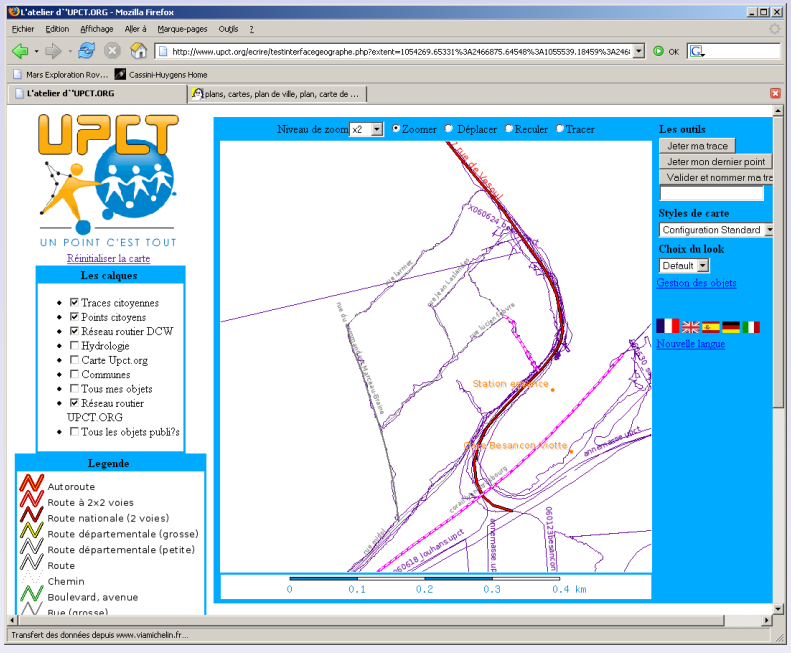
    //<![CDATA[

    function load() {
      if (GBrowserIsCompatible()) {
        var map = new GMap2(document.getElementById("map"));
        map.setCenter(new GLatLng(37.4419, -122.1419), 13);
      }
    }

    //]]>
    </script>
  </head>
  <body onload="load()" onunload="GUnload()">
    <div id="map" style="width: 500px; height: 300px"></div>
  </body>
</html>
```

- cet exemple montre comment positionner une carte par ses coordonnées du centre et son facteur d'agrandissement

Exemple complété

- on veut ajouter un polygone avec notre trace :

```
var points = new Array();
points.push(new GPoint(5.04382,45.6604383333333));
map.addOverlay(new GPolyline(points, "#FF0000",2,.75));
```

- maximum $\simeq$1000 points/polygone, donc périodiquement afficher et réinitialiser : `points=[];`
- on obtient donc (par sed) une page du genre

```
var points = new Array();
points.push(new GPoint(5.04382,45.6604383333333));
points.push(new GPoint(5.04424666666667,45.6602583333333));
points.push(new GPoint(5.04468,45.6601933333333));
...
points.push(new GPoint(5.124995,45.6376066666667));
points.push(new GPoint(5.12546333333333,45.6375033333333));
map.addOverlay(new GPolyline(points, "#FF0000",2,.75));
points=[];
points.push(new GPoint(5.125925,45.63738));

...
```

- problèmes : trop de décimales et trop de séquences `points.push(new GPoint` $\Rightarrow$ page trop grosse (1,1 MB pour 5 heures de trajet)

Exemple optimisé

- remplacer la vue par défaut (routes) par une vue aérienne :
  `map.setMapType(G_SATELLITE_TYPE);`
- utilisation des fonctions

```
function p(lon,lat) {points.push(new GPoint(lon,lat));}
function o()
    {map.addOverlay(new GPolyline(points,"#FF0000",2,.75));points=[];}
```

- couper les décimales inutiles (1 degré de latitude$\simeq$111 km donc 5 décimales sont suffisantes)





Exemple amélioré

- générer la page web au vol au lieu de la générer à la main :PHP

```
<?php
$handle = fopen($file, "r") or exit("Unable to open file!");
do {
     do {$buffer = fgets($handle, 4096);} while ((strpos($buffer,'GPGGA')===false) && (!feof($handle)));
     list($rien1,$heure,$lat,$N,$lon,$E,$pasalt)=explode(",",$buffer);
     if ((preg_match("/[0-9]{4}\.[0-9]{4}/i",$lat)) && (preg_match("/[0-9]{4}\.[0-9]{4}/i",$lon))) {
          $lati=floor($lat/100);$loni=floor($lon/100);
          $lati=$lati+floor( (($lat/100)-$lati)/60*100*1000000)/1000000;
          $loni=$loni+floor( (($lon/100)-$loni)/60*100*1000000)/1000000;
          print "p($loni,$lati);";
      } while (!feof($handle));
 printf("map.addOverlay(new GPolyline(points, \"#FF0000\", 2, .75));"); fclose($handle);
?>
```

- encoder dans la couleur du polygone une quantité (ici la vitesse)





Affichage des traces en 3D (Google Earth)

- Google Earth (à l'origine Keyhole) est disponible en natif depuis mi-juin pour Linux
- ajoute la troisième dimension (élévation) à Google Maps
- partage des informations dans Google Earth Community (`bbs.keyhole.com`)
- accepte des données au formal KML (HTML)
- décrit à `http://www.keyhole.com/kml/kml_tut.html` : ajout de polygones et d'images bitmaps au globe (remise à jour `http://earth.google.com/kml/kml_21tutorial.html`)

Le format KML

- un entête constant :

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Placemark>
  <description>Trajet Besancon-Chamonix, 23-06-2006</description>
  <name>Absolute Extruded</name>
```

- définition de point de vue et de la direction de visée initiale

```
<LookAt>
    <longitude>6.02194166666667</longitude>
    <latitude>47.24624</latitude>
    <range>4451.842204068102</range>
    <tilt>44.61038665812578</tilt>
    <heading>-125.7518698668815</heading>
</LookAt>
```

- options de tracé du polygone

```
<visibility>1</visibility>
  <open>0</open>
  <Style>
     <LineStyle> <color>ff00ffff</color> </LineStyle>
     <PolyStyle> <color>7f00ff00</color> </PolyStyle>
  </Style>
  <LineString>
  <extrude>1</extrude> <tessellate>1</tessellate>
  <altitudeMode>absolute</altitudeMode>
```

- une suite de points de la forme {longitude,latitude,altitude}

```
<coordinates>
6.02194166666667,47.24624,100
6.02194333333333,47.2462416666667,100
6.02194333333333,47.2462433333333,100
...
</coordinates> </LineString> </Placemark> </kml>
```

Résultat des tracés dans Google Earth



Partage des traces (UPCT, www.upct.org)

- Objectif : création d'une base de données pour la carthographie libre
- Moyens : les participants placent leur trace dans la base de données. Lorsque suffisamment de traces définissent une route, elle est vectorisée manuellement





Format gpsman

- Théoriquement il semble possible d'importer du NMEA dans gpsman pour le transformer en format natif, mais nous voulons filtrer nos traces
- un entête

```
% Written by GPSManager 18-Mar-2006 15:06:57 (CET)
% Edit at your own risk!

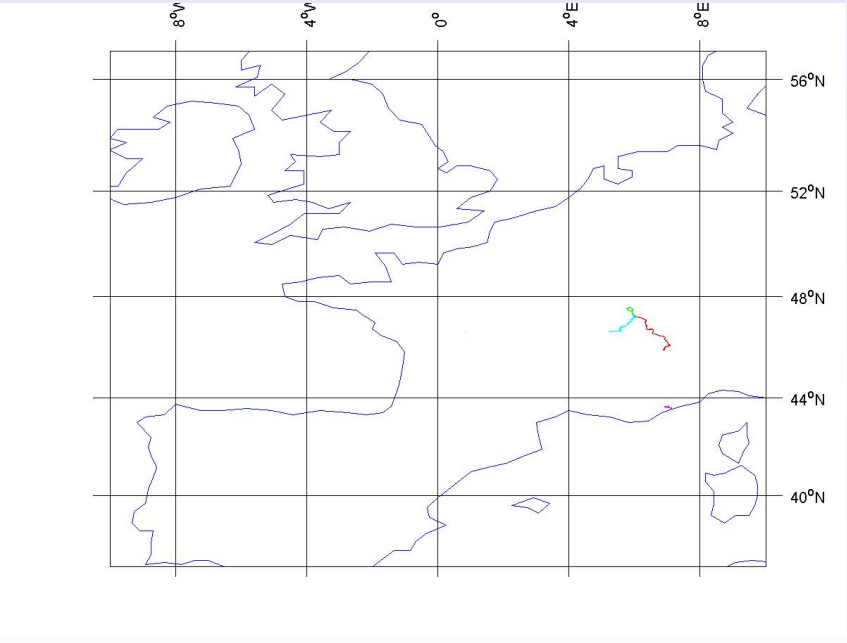
!Format: DMS 1 WGS 84
!Creation: n

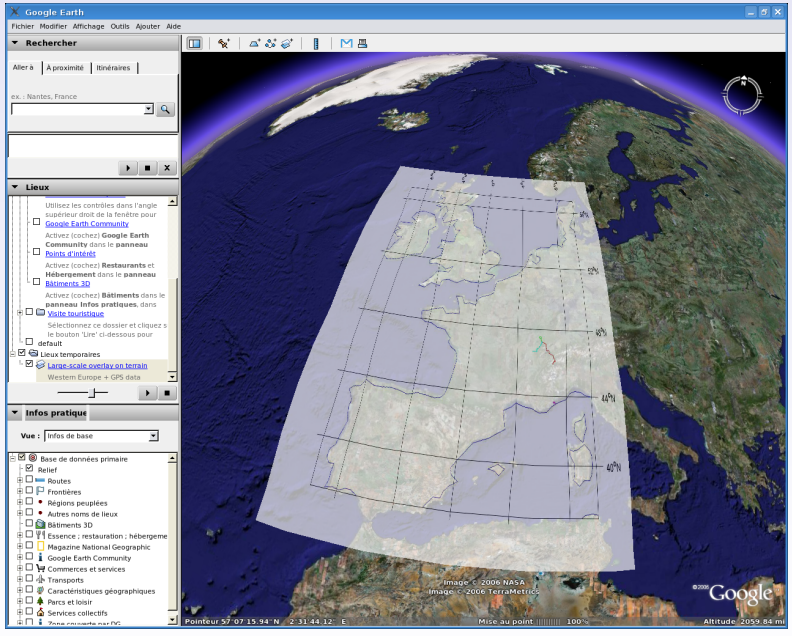
!T:     ACTIVE LOG      width=2 colour=#8b0000  GD310:display=~|Z
```

- le point de départ de la trace (première coordonnée GPS pertinente)

```
        01-Jan-2006 00:00:00    N47 54 15.51    E1 57 21.14     0
```

- la définition de la trace `!TS:`
- les points sous la forme {degrés minute seconde.fraction}

```
        01-Jan-2006 00:00:00    N47 54 15.47    E1 57 21.15     0
        01-Jan-2006 00:00:00    N47 54 15.44    E1 57 21.16     0
        01-Jan-2006 00:00:00    N47 54 15.39    E1 57 21.19     0
...
```

- le passage de degrés,fraction à ce format se fait par Octave :

```
function sortie=deg2dec(entree)
sortieX=floor(entree(:,1)/100);
sortieX=sortieX+((entree(:,1)/100)-sortieX)/60*100;
sortieY=floor(entree(:,2)/100);
sortieY=sortieY+((entree(:,2)/100)-sortieY)/60*100;
sortie=[sortieY sortieX entree(:,3)];
```

Résultat UPCT



- l'interface de vectorisation de UPCT est uniquement disponible *via* le web
- interface conçue pour une faible bande passante, supporte la déconnexion, rapide et souple d'emploi *mais* ne permet pas de travailler déconnecté
- UPCT est particulièrement orienté sur la France. D'autres projets se focalisent sur d'autres pays européens : Openstreetmap (interface de vectorisation autonome, plus lourde – Java)

Fusion des traces avec d'autres données géographiques (M_MAP pour Matlab)

- Matlab est probablement *la* référence en traitement du signal
- cependant, le problème de la projection des points devient non-triviale [3] dès lors que la sphère ne s'apparente plus localement à un plan
- la toolbox gratuite `M_MAP` (Matlab uniquement) (`http://www.eos.ubc.ca/~rich/map.html`) s'occupe de la conversion d'un mode de projection à un autre
- ```
  path(path,'c:\matlab\m_map');
  m_proj('Miller Cylindrical','longitudes',[-10 10],'latitudes',[37 57]);
  m_coast; axis square;              % trace du fond de carte
  load 060622_cham.dat;              % fichier long,lat,alti issu du grep
  s=deg2dec(X060622_cham);           % conversion en format decimal
  u=diff(s(:,1)); v=diff(s(:,2));    % vecteur vitesse
  lat=s(1:length(s)-1,2); lon=s(1:length(s)-1,1);
  hold on;m_quiver(lon,lat,u,v,'r'); % iterer pour chaque nouvelle trace
  ```

[3] J.P. Snyder, *Flattening the Earth – Two thousand years of map projections*, The University of Chicago Press (1993), pp.178-183
  ```

Projection du résultat dans Google Earth

Sous réserve d'avoir choisi la bonne projection (cylindrique), l'image résultante se projette en transparence (`color`) sur le globe de Google Earth (balise `Icon` du KML) :

```
<color> a9ffffff </color>
  <Icon> <href>http://mmyotte.free.fr/france_mmpap2.jpg</href> </Icon>
  <LatLonBox id="khLatLonBox751">
    <north>59.57</north> <south>34.333</south>
    <east>12.70</east> <west>-13.7666</west>
    <rotation>0</rotation>
  </LatLonBox>
```





Fusion des traces avec d'autres données géographiques (GRASS)

- GRASS (*Geographic Resources Analysis Support System*) est l'outil de référence lorsqu'une donnée référencée spatialement est considérée
- proche de la philosophie unix : de nombreux petits modules simples s'assemblent pour donner un logiciel puissant [4]
- exemple qui nous concerne ici : afficher des traces de randonnées en montagne dans le relief associé
- au lancement de GRASS, définition de la zone géographique sur laquelle on va travailler (latitude/longitude/mode de projection : WGS84)

[4] S. Erle, R. Gibson & J. Walsh, *Mapping Hacks – Tips and tools for electronic cartography*, O'Reilly (2005)

GRASS : importer les traces

- premier point : importer un tableau de données au format ASCII dans GRASS : on part du fichier 060729_argentiere contenant

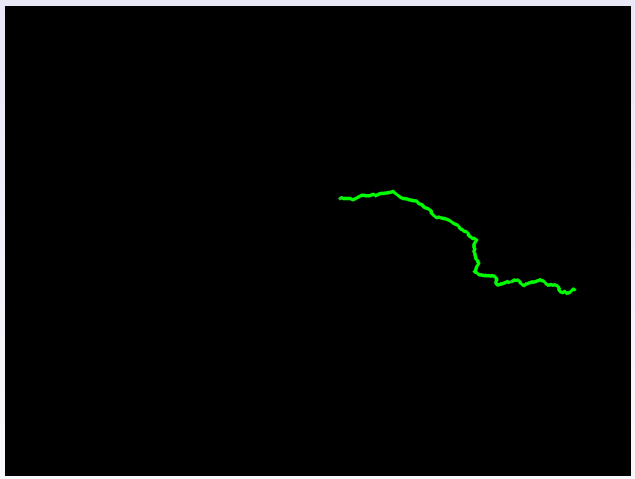
```
6.92555166666667 45.98091 1220.4
6.92552833333333 45.9808883333333 1212.8
...
```

qu'on importe par `v.in.ascii input=060729_argentiere.dat output=waypoint format=point fs=space`

⇒ GRASS nous confirme la lecture du bon nombre de points :

```
Maximum number of columns:  3
Number of points:  34311.
```

Le vecteur résultant est placé dans la variable `waypoint`



- il nous faut définir la sortie sur laquelle tracer (sortie graphique) :

```
d.mon start=x0
```

- tracer la courbe : `d.vect map=waypoint icon=basic/diamond size=8 color=green fcolor=red`

Importation données topographiques

- Les données topographiques ne sont pas libre en Europe : on va les chercher aux États-Unis (GLOBE : *Global Land One-kilometer Base Elevation*) : `http://www.ngdc.noaa.gov/mgg/topo/globeget.html`
- un tutoriel explique comment importer les données dans GRASS et commnent géoréférencer les informations d'altitude : `http://www.ngdc.noaa.gov/mgg/topo/report/s11/s11Gvi.html`
- fichier nécessaire au positionnement des points : `ftp://ftp.ngdc.noaa.gov/GLOBE_DEM/data/elev/grass/grasstar`
- nous récupérons la base de donnée des élévations autour des Alpes : fichier `g10g` à `http://www.ngdc.noaa.gov/mgg/topo/gltiles.html`.

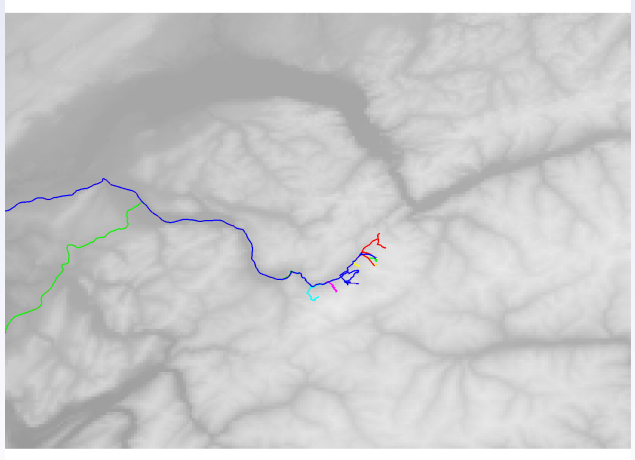
Fusion de notre trace avec les données de topographie

- cette fois au lancement de GRASS on ne définit par une région vide mais sélectionnons la zone couverte par notre base de données : `Location: grass` et `Mapsets: user`
- la base de données des élévations est chargée par `g.region rast=g10g.ux` après avoir placé `g10g.ux` dans le répertoire `grass/user/cell` tel que décrit dans le tutoriel, et translatée pour avoir une altitude correcte

  `r.mapcalc 'g10g=if(g10g.ux-32768,g10g.ux-65536,g10g.ux,g10g.ux)'`

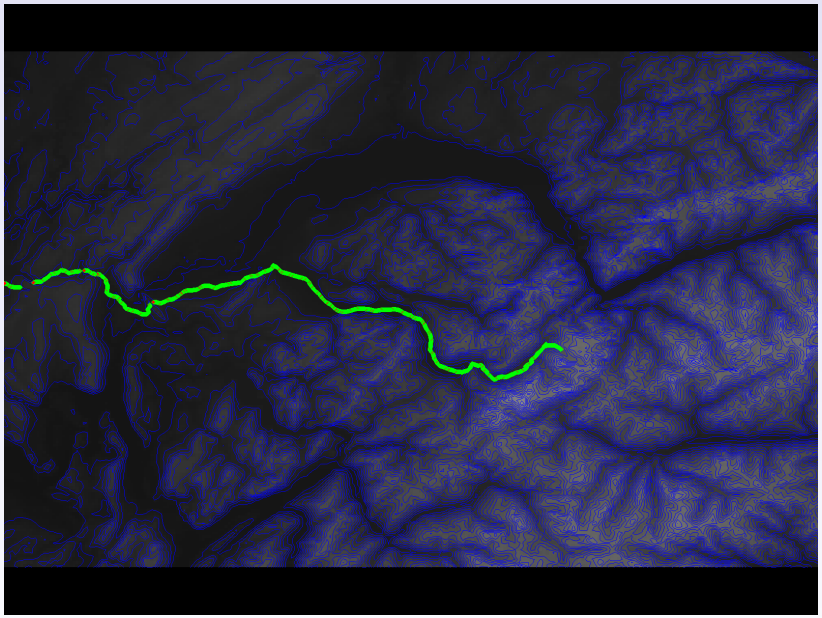
  Le résultat est la variable g10g
- le tableau résultant est affiché par `d.rast g10g` auquel on assigne une palette de dégradés de gris par

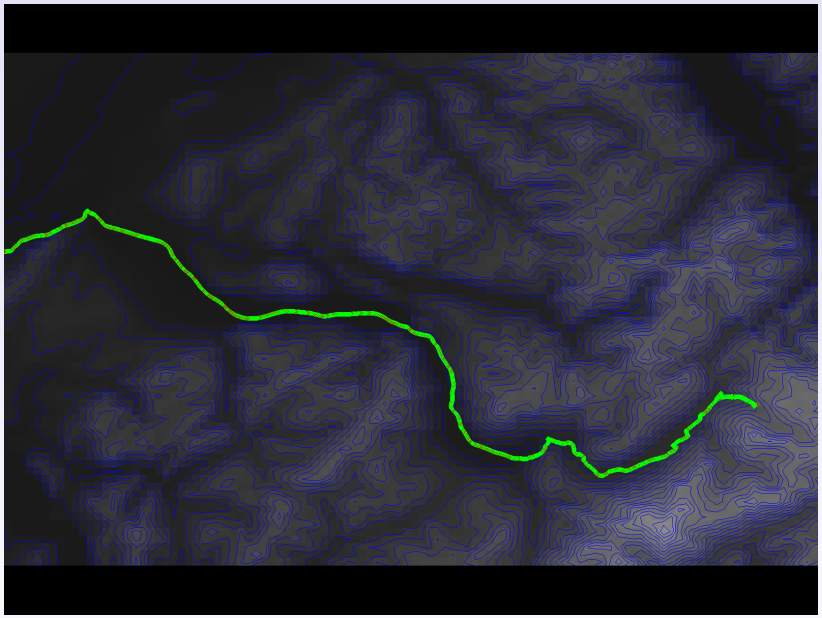
  `r.colors map=g10g color=grey.log`

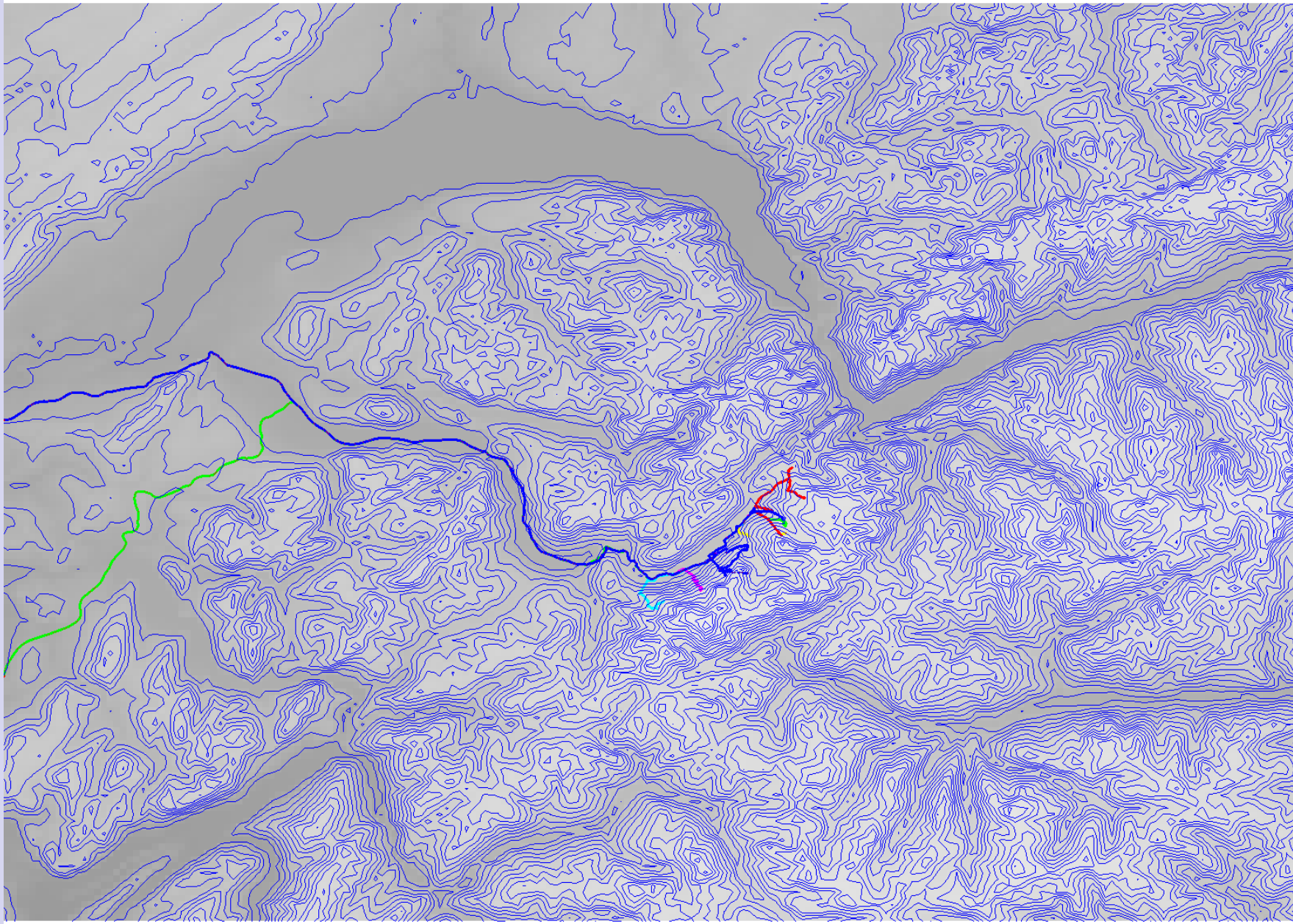


Fusion de notre trace avec les données de topographie (2)

Pour un dégradé linéaire de gris, `r.colors map=g10g color=grey`
GRASS fournit les outils de conversion d'une base de données d'altitude en lignes de niveau : `r.contour in=g10g out=cont200 min=0 step=200`







Conclusion

- Réalisation d'un circuit de coût, d'encombrement et de consommation réduits pour l'acquisition de trames GPS
- extraction et filtrage des informations pertinentes des trames NMEA
- fusion des trames avec des bases de données géographiques existantes : Google Maps, Google Earth, UPCT
- traitement des informations dans des logiciels spécialisés (Matlab, GRASS)

Perspectives :

- diffusion de ce circuit afin de promouvoir la cartographie libre, baisse des coûts par une production plus importante (objectif :100 euros incluant la source d'énergie)
- transfert des données en temps réel par GPRS (téléphonie mobile)