Polytech' SI 3 - Master ISI - ONIHM octobre - novembre 2006

Application Qt de visualisation de flux vidéo issu d'une Webcam.

Nous allons modifier l'application du TD précédent afin de remplacer l'image fixe par les images acquises par une Wecam. Nous allons procéder en trois étapes :

- Acquisition et visualisation d'une image en niveaux de gris.
- Acquisition et visualisation d'une image en couleurs.
- Acquisition et visualisation des images au rythme de la Webcam.

Avant toute chose, il nous faut commencer par parler des drivers de Webcam.

1 Driver de la Webcam sous Linux.

Il n'existe pas de driver universel pour l'ensemble des webcams existant sur le marché. Selon les constructeurs ou plutôt les familles de puces dans les webcams, différents drivers ont été développés.

1.1 Historique

Les modèles de Webcam dont vous disposez ont des chipsets de la même famille, dite "Philips". Jusqu'en Septembre 2004, existaient des drivers binaires et une API documentée sur le site :

`http://www.smcc.demon.nl/webcam/api.html`

Depuis, des disputes entre partisants de l'Open Source et du "faut qu'ça marche" se sont battus suffisamment pour que le mainteneur du site cité précédemment décide de tout arrêter.

Depuis Octobre 2004, d'autres développeurs ont développé un nouveau driver pour les noyaux Linux 2.6.*. C'est donc ce driver que nous allons utiliser.

`http://www.seismo.ethz.ch/linux/webcam.html`
`http://www.saillard.org/pwc/`

1.2 Installation et chargement du module `pwc`

Il suffit de suivre les instructions d'installation :

`http://www.saillard.org:8080/pwc/INSTALL.fr`

1. télécharger le fichier `http ://www.saillard.org/pwc/pwc-10.0.12-rc1.tar.bz2` (ou le dernier fichier disponible sur le site)
2. décompresser le fichier par la commande : `tar xjf pwc-10.0.12-rc1.tar.bz2`
3. compiler le module (si on dispose des sources du kernel actuels) `cd pwc-10.0.12-rc1 ; make`
4. Si une version précédente du module a déjà été installée, lancer la commande suivante pour effacer le module : `find /lib/modules/‘uname -r‘/ -name "pwc*.ko*" ; rm <nom du fichier.ko>` et décharger l'ancien module : `rmmod pwc`
5. Copier le nouveau modules sur l'ancien : `make install`
6. Décharger l'ancien module et charger le nouveau : `modprobe pwc`

Attention, selon les noyaux, le `make install` n'est pas toujours à jour et peut installer le fichier pwc.ko dans un mauvais répertoire !

Premier test. Lancez : `xawtv -device /dev/video0` ou bien tout autre utilitaire permettant de lire un flux webcam (logiciel de video-conférence, ...).

Ouverture : La première étape consiste à ouvrir le pseudo fichier ou *device* correspondant à la webcam : `int cam_fd = open("/dev/video0", O_RDONLY) ;`

Différentes commandes `ioctl` vont ensuite permettre de consulter et modifier des paramètres de ce périphérique.

1.3 Test de la présence et compatibilité du driver et de la webcam :

```
struct video_capability vcap;
struct pwc_probe probe;
if (ioctl(cam_fd, VIDIOCGCAP, &vcap) < 0)
        return ;
if (sscanf(vcap.name, "Philips %d webcam", &type) == 1) {
        IsPhilips = TRUE;
        printf("ok for philips\n");
}
else    {
        /* No match yet; try the PROBE */
        if (ioctl(cam_fd, VIDIOCPWCPROBE, &probe) == 0)  {
           if (!strcmp(vcap.name, probe.name)) {
               IsPhilips = TRUE;
               type = probe.type;
               printf("%s detected ; type = %d \n ",
                probe.name, probe.type);
             }
         }
     }
```

1.4 Taille de l'image :

```
struct pwc_imagesize image_size;
/* Image size */
ioctl(cam_fd, VIDIOCPWCGREALSIZE, &image_size);
printf("width = %d  and height = %d \n",
        image_size.width, image_size.height);
width = image_size.width;
height = image_size.height;
if (ioctl(cam_fd, VIDIOCGMBUF, &VMBuf) == 0) {
        size = width*height;
        vid_io_buffer_size = VMBuf.size;
        vid_io_buffer = (unsigned char *)
                malloc(size*sizeof(unsigned char));
}
```

1.5 Codage des couleurs :

Différentes palettes sont disponibles. C'est généralement la palette `YUV420P` qui est utilisée par les webcams. C'est d'ailleurs celle que vous utiliserez. Cette palette précise que les valeurs

des luminances (Y) sont données les unes à la suite des autres, chacune codée sur 1 octet. Si vous ne lisez donc que les `width*height` premiers octets, vous obtiendrez une image en niveaux de gris. Essayez donc !

Ensuite, sont codées les chrominances U et V à raison d'une colonne sur deux et d'une ligne sur deux. Toutes les valeurs U sont données avant de passer aux valeurs V. Ajoutez donc ces valeurs pour obtenir une image couleur !

```
/* get the palette */
ioctl(cam_fd, VIDIOCGPICT, &video_pic);
printf("palette = %d\nVIDEO_PALETTE_YUV420P = %d\n",
     video_pic.palette,VIDEO_PALETTE_YUV420P);
/* set image compression algorithm to none */
compr_ratio = 0;
ioctl(cam_fd,VIDIOCPWCSCQUAL, &compr_ratio);
printf("compression ratio = %d\n",compr_ratio);
```

A titre de rappel, on convertit des valeurs YUV en RGB à l'aide des formules suivantes :

$$y = 0.299R + 0.587G + 0.114B$$
$$(u - 128)/0.493 + y = B$$
$$(v - 128)/0.877 + y = R$$

1.6 Fréquence d'acquisition :

```
struct video_window vwin;
struct video_mbuf VMBuf;

memset(&VMBuf, 0, sizeof(VMBuf));
/* See if device has mmap(); */
VMBuf.size = 0;
VMBuf.frames = 0;
vid_io_buffer_size = 0;
ioctl(cam_fd, VIDIOCGWIN, &vwin);
if (vwin.flags & PWC_FPS_FRMASK) {
     int new_frame_rate=15;
     printf("Camera has framerate setting; current framerate: %d fps\n",
        (vwin.flags & PWC_FPS_FRMASK) >> PWC_FPS_SHIFT);
   /* Set new framerate */
     vwin.flags &= ~PWC_FPS_FRMASK;
     vwin.flags |= (new_frame_rate << PWC_FPS_SHIFT);
     vwin.width=640;
     vwin.height=480;
     ioctl(cam_fd, VIDIOCSWIN, &vwin);
}
```

2 Réalisations

2.1 Classe `WebcamWidget`

Crééz une classe `WebcamWidget` dérivant de la classe `QWidget`. Réécrivez la méthode `paintEvent` afin de redessiner l'image courante dans un `QPainter` à l'aide de la méthode `drawPixmap`.

Remplacez la classe `ImageCanvas` par la classe `WebcamWidget` dans votre classe `ImageViewer`.

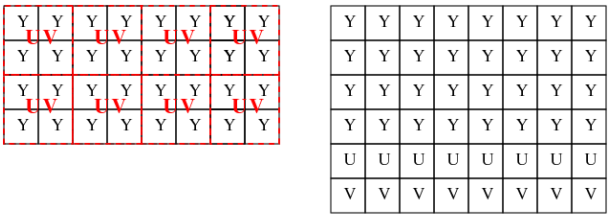
2.2 Acquisition d'une image en niveaux de gris

Lors de la création d'une instance de `WebcamWidget`, initialisez la communication sur le port USB et capturez une image (à stocker dans une variable de classe, instance de `QImage`). En partant d'une taille figée (par exemple 640x480), récupérez les 640x480 octets, chaque octet représentant un niveau de gris.

Ajoutez un bouton poussoir "capture" à votre interface. Lorsque ce bouton est activé, une nouvelle image sera acquise.

2.3 Acquisition et visualisation d'une image en couleurs.

Modifiez le code précédent afin de récupérer également les composantes u et v de la couleur. La palette de couleurs utilisée est de type "4 :2 :0 planar"ce qui signifie que les valeurs y (composante de luminance) sont toutes données en premier temps. Viennent ensuite les composantes de chrominances u et v sachant qu'une valeur de u et de v est partagée entre 4 pixels voisins (le pixel, son voisin de droite, son voisin d'en dessous ainsi que le voisin diagonal bas et droite).



2.4 Acquisition et visualisation des images au rythme de la Webcam.

Ajoutez un nouveau bouton permettant de lancer les acquisitions au rythme de la Webcam (*framerate*) ou de repasser au mode précédent (acquisition d'images fixes).

2.5 Modification des paramètres de la Webcam.

Ajoutez dans votre interface la possibilité de modifier les paramètres de la webcam tels que la résolution ou le *framerate*.

2.6 Deux webcams

Modifiez votre application afin de permettre la capture vidéo depuis deux webcams. Pouvez-vous utiliser les capacités maximales de vos webcams ? Pourquoi ?

Aide : Regarder le débit maximal sur une liaison USB. Comparer avec le débit maximal d'une webcam (taille de l'image et *framerate*). Regarder le nombre de controlleur USB disponibles sur l'ordinateur. Conclure.