Figure 5. Gestion des polices TRUETYPE avec inclinaison

Mais l'utilisation du LCD ne serait pas complète sans l'emploi d'un écran tactile. Bref, faisons un tour d'horizon du fonctionnement d'un tel écran avant de passer à la mise en œuvre de notre équipement.

Écran tactile

Un écran tactile est composé de 2 films qui se superposent et qui détectent une pression exercée sur leur surface.

Pour cela, placez l'écran tactile sur le LCD mais pas directement en contact avec celui-ci, car cela pourrait provoquer comme résultat une pression constante sur l'écran, ce dernier étant parfois très sensible.

Il convient alors d'utiliser un « *driver* » adéquat, tel un ADS7843, s'interfaçant au CPU via un bus 4 fils SPI. Les drivers sont le plus souvent disponibles dans les distributions Linux et uClinux (*snapgear*).

Cas concret sur uClinux

Voyons à présent de la mise en œuvre d'un tel système au travers d'une plate-forme ARM7.

L 'équipement choisi dispose de :

- CPU S3C44 à 60 MHz,
- 16Mo de SDRAM,
- 16 Mo de Flash NAND,
- ethernet 10T,
- bus I2C, bus CAN 2.0B, RS485,
- contrôleur vidéo intégré,
- LCD STN 320x240 256 couleurs
- dalle tactile.

La distribution utilisée est une Snapgear uClinux 2.4. Si votre carte n'est pas livrée avec le noyau pré-installé et configuré pour gérer le Frame Buffer de Linux, il vous faudra recompiler le noyau. Pour cela, référez-vous aux documents fournis avec votre carte de développement.

Quelle que soit la méthode utilisée, vous devrez au final disposer de la gestion du *Frame Buffer* et de l'écran tactile via un contrôleur tel le ADS7843. Dans notre cas, la gestion du *Frame Buffer* sera native dans le noyau, mais nous chargerons le driver d'écran tactile à l'aide de la commande :

```
insmod /home/ads7843.o
```

Certains kits de développement permettent ainsi d'avoir un environnement « prêt à l'emploi », vous rendant fin prêt à développer dès que la carte est démarrée et équipée du LCD et de son écran tactile,.

Application graphique

Voici l'objectif : nous cherchons à afficher des objets graphiques (appelés `WIDGET` ci-dessous) sur l'écran et connaître les interactions de l'utilisateur sur l'écran. Au final, le résultat doit être comme celui montré sur la Figure 4.

Du côté de la programmation, il serait intéressant de pouvoir programmer facilement en C une telle application et de gérer directement :

- les images GIF et BMP,
- les lignes et les pixels,
- les formes simples (cercle, rectangle, plein ou vide, ...),
- les formes complexes (polygones, nuages de points, ...),
- les chaînes de texte,
- les copies de zone d'écran,
- les couleurs des objets et les couleurs de fond.

Côté code C, l'affichage d'un bitmap se ferait très simplement part :

```
GrDrawImageToFit(MainWindow, gc,
wPosX, wPosY,wWidth, wHeight, aImageId);
```

À cette fin, une librairie graphique adaptée à l'embarqué est exigée, nécessitant une faible empreinte mémoire, capable de s'interfacer avec le Frame Buffer de Linux (*/dev/fb0*). Cette librairie existe, il s'agit de Nano-X !

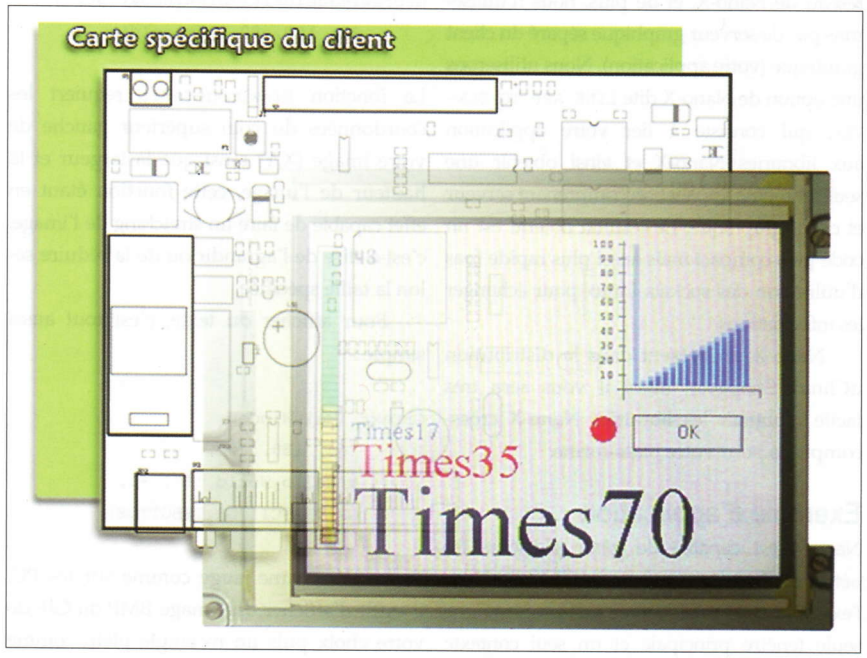

Figure 6. Plate-forme de développement XB10