mini-tuto: la librairie Xlib

ou comment créer des jeux en restant dans l'esprit TI Basic, sans se casser la tête, mais avec de beaux effets graphiques...

Créateur de la librairie: Patrick Prendergast

Créateur du tuto: Persalteas pour Espace TI:Forum

Lien de téléchargement de l'archive: http://www.ticalc.org/pub/83plus/asm/libs/xlib.zip

petites infos préliminaires

-Qu'est-ce qu'une librairie ?

C'est une application, ou un programme, qui permet de rajouter des fonctions au TI Basic normal.

-Pourquoi Xlib et pas une autre librairie ?

Absolument aucune raison, les librairies se ressemblent toutes, je vous invite à voir aussi le Celtic etc...

Xlib supporte les sprites, mais pas les niveaux de gris, ni l'audio.

-Qu'ai-je besoin de savoir faire pour apprendre le Xlib ?

Je considère que vous savez déjà programmer en TI Basic pour TI 83+ ou 84+,

et que vous savez envoyer une application à votre TI.

Si ce n'est pas le cas, demandez de l'aide sur le forum.

-Comment ma TI va-t-elle comprendre quand j'éxécute une fonction Xlib, si le programme n'est pas compilé ?

Les fonctions Xlib sont toutes introduites par la commande **real(** disponible dans les menus

de la TI en faisant: **[MATH] [>][>][2]** .

I. Dessiner sur l'écran

Le premier avantage d'une librairie est de pouvoir mélanger l'écran graphique et l'écran tableur (l'écran normal pour les calculs).

-mais alors, pour effacer, je dois utiliser ClrHome, ou ClrDraw ?

une fonction existe pour effacer l'écran, **real(0** .

real(0 , UpdateLCD

-pourquoi real ?

pour que la TI comprenne que j'appelle une fonction du Xlib.

-pourquoi 0 ?

pour distinguer cette fonction des autres fonctions du Xlib. 0 correspond a l'effaçage d'écran,

les autres fonctions auront un autre numéro.

-Qu'est-ce que UpdateLCD ?

Il faut remplacer "Update LCD" par un booléen, 0 ou 1.

Si c'est 1, l'écran s'effaçera tout de suite, si c'est 0, l'écran s'effacera la prochaine fois qu'il sera concerné.

Retenez bien ce principe du Update LCD, il sert dans beaucoup des fonctions Xlib:

1=modifications appliquées tout de suite, 0=modifications la prochaine fois qu'on demande quelque chose a l'écran.

II. Afficher un sprite

Je vous ai dit tout à l'heure que Xlib supportait les sprites. Vous avez peut être déjà vu le principe en TI Basic, un sprite est une petite image déplaçable à souhait à l'écran selon des coordonnées.

Comme pour le dessin en TI Basic, il faut savoir les coordonnées de l'objet qu'on veut afficher. Les coordonnées correspondent à X (l'abscisse) et Y (l'ordonnée).

-comment enregistrer mon sprite dans la TI ?

Il vous faudra le dessiner sur l'écran graphique, et l'enregistrer dans une image (Pic).

Pour économiser de la place, vous pouvez stocker plusieurs sprites par Pic !

Il vous faut par contre faire attention: le sprite devra être positionné sur un pixel multiple de 8: du pixel 0 au pixel 7, un sprite, du pixel 8 au 15, un deuxième, etc.. ce qui nous permet de caser 12 sprites dans la largeur d'une Pic si on utilise des sprites de 8x8 pixels.

exemple de sprites dans une Pic:

- Et comment afficher un seul de ces sprites à l'écran ?

En utilisant une fonction qui sert à n'afficher qu'une partie d'une Pic, **real(1** .

real(1 , X , Y , largeur , hauteur , Pic , position X , position Y , méthode , flip , Update LCD

C'est long, hein ! On peut donc très bien se retrouver avec des instructions du type: **real(1,34,54,2,8,2,44,1,0,1** . ça fait peur.

real(1: dit à la TI qu'on utilise la fonction qui sert a afficher un bout d'image.

X et Y: Les coordonnées du coin en haut à gauche de votre sprite quand il va s'afficher sur votre écran.

L'origine sur l'écran est le coin en haut à gauche, et il n'y a pas de rapport avec la position actuelles de axes de la TI : n'essayez pas de changer la position du sprite en changeant Xmin, Xmax etc...

largeur: la largeur du sprite, **en bytes**. techniquement, ça correspond à la largeur du sprite en **pixels, divisée par 8.**

hauteur: la hauteur du sprite à afficher, en pixels.

si vous vous trompez dans la hauteur et la largeur de votre sprite,

la TI affichera un petit bout de plus/de moins de la Pic qui contient votre sprite.

Pic : Le numéro de la Pic ou vous avez enregistré votre sprite. Si vous l'avez mis dans Pic 3 , mettez 3. Si vous vous trompez de numéro, ça affichera un mauvais sprite si la Pic existe, ou rien si la Pic n'existe pas.

Position X: On a vu tout à l'heure qu'on pouvait enregistrer jusqu'à 12 sprites par Pic. Il s'agit du "numéro de colonne", entre 0 et 11 qui indique l'endroit ou vous avez aligné votre sprite dans la Pic. En gros, c'est l'abscisse du coin haut-gauche de votre sprite dans la Pic, divisé par 8.

NB: vous ne pouvez pas y mettre une valeur à virgule, seul un entier est accepté.

Position Y: il s'agit de l'ordonnée du coin haut-gauche du sprite dans la Pic, ce n'est pas forcément un multiple de 8. je ne sais pas pourquoi c'est toléré en ordonnée et pas en abscisse.

méthode: détermine comment le sprite va s'insérer parmi les pixels de l'écran, déjà plus ou moins allumés. Un pixel allumé vaut 1, un pixel éteint vaut 0, la méthode détermine le calcul entre l'état du pixel actuellement sur l'écran et sur le sprite.

si vous mettez: 0 --> Le sprite recouvre les pixels, qu'ils soient allumés ou éteints.

1 --> Les pixels sont allumés selon le test "and"

2 --> Les pixels sont allumés selon le test "or"

3 --> Les pixels sont allumés selon le test "xor"

flip: afficher le sprite normalement = 0 , effectuer une inversion droite/gauche = 1

Update LCD: pareil que tout à l'heure, détermine le moment des modifications.

Il est donc possible de créer des sprites de toutes largeur-hauteur, si la taille dépasse, ça empiète sur la Pic suivante je crois (jamais testé).

NB: Le top, c'est que **les Pic peuvent être dans la mémoire archive**, ça ne dérange pas l'affichage !

-J'ai donc mes 10 Pics pour stocker des graphismes ?

Mieux. En fait, 255 Pics sont disponibles. Nous allons voir comment les utiliser au prochain paragraphe.

III. Les Pics en Xlib

- comment créer une Pic de numéro supérieur à 9 ?

Devinez quoi, en utilisant une fonction real(!

real(9 , fonction , Pic

fonction: 0 = créer (l'équivalent de StorePic du TI Basic) , 1 = supprimer

Pic: Numéro de là Pic à créer/supprimer , entre 1 et 255.

- entre 1 et 255 ? et Pic 0, alors, comment je la modifie ?

Eh bien, vous remarquerez qu'en fait, Pic 0 , c'est Pic 10. D'ailleurs, dans les menus de la TI, elle est après Pic 9.

NB: vous remarquerez aussi si vous créez des images a numéro supérieur

*l'apparition dans le menu mémoire de Pics avec des noms anormaux, comme **Pt-On(** ou **Then**.*

C'est tout à fait normal: comme l'OS de la TI ne possède pas de nom à afficher sur cette Pic,

puisqu'elle n'est pas censée exister, il affiche n'importe quoi.

Même si votre Pic s'appelle L4 ou BDG5, ne paniquez pas, ça n'empêchera pas votre liste 4 ou votre BDG5

de fonctionner normalement.

- et si au lieu d'afficher juste un bout de la Pic, je veux l'afficher en entier ?

On va utiliser l'équivalent Xlib-ien de RecallPic, qui est mieux car il supporte **même les images archivées :**

real(3 , Pic , méthode , Update LCD

Vous devriez commencer à connaitre ;) :

Pic est le numéro de la Pic entre 1 et 255

La méthode correspond au mode de superposition des pixels (recouvrir, and , or , xor)

L'update LCD, comme toujours, pour le moment d'application des modifs.

Techniquement, pour les connaisseurs qui ont fait de l'Axe ou de l'ASM, le Update LCD correspond à l'affichage du Back-buffer sur l'écran.

Il y a donc une fonction qui ne fait que ça, afficher le Back-buffer: **real(6** .

IV. Scrolling de l'écran

c'est assez amusant a effectuer. Il s'agit de "déplacer l'écran par rapport à lui-même". Je ne sais pas comment expliquer, le mieux est donc de tester soi-même:

real(4 , direction , nombre de pixels , Update LCD

direction:

0= haut	3= droite	6= bas-gauche
1= bas	4= haut-gauche	7= bas-droite
2= gauche	5= haut-droite	

nombre de pixels : déplacer l'écran du nombre de pixels voulus dans la direction demandée.

Note pour ceux qui voudraient faire du scrolling ou du tile mapping avec cette fonction: vous allez galérer et pas y arriver. Il y a une fonction pour ça, **real(2** , mais comme je ne l'ai pas comprise, je ne peux pas vous expliquer... :s

V. Dessiner... avec des traits, quoi, sans les sprites...

Une fonction bien pratique pour tout ce qui est menu graphique de jeu etc...

cette fonction permet de dessiner des lignes ou des rectangles, avec la notion de transparence. (oui, oui, transparence sur une TI, avec des énormes pixels noirs ou blancs. Parfaitement).

real(12, fonction , X1 , Y1 , X2 , Y2 , Update LCD

fonction: elle détermine ce que vous voulez dessiner:

0 = ligne noire	3 = rectangle transparent noir	6 = rectangle plein noir
1 = ligne blanche	4 = rectangle transparent blanc	7 = rectangle plein blanc
2 = ligne qui inverse les pixels	5 = rectangle transparent qui inverse	8 = rectangle plein qui inverse
9 = rectangle blanc à cadre noir	10 = rectangle noir à cadre blanc	

X1 et Y1 , X2 et Y2: les coordonnées des deux points qui indiquent comment tracer. Pour les lignes, ce sont les extrémités, pour les rectangles, ce sont le coin en haut à gauche et le coin en bas à droite.

Je ne vous fait pas l'affront de vous redire ce qu'est Update LCD ! =D

VI. Fonctions système

- afficher la version du système: **real(13**

si la valeur renvoyée dans Ans est: 0 , vous avez une TI 83+

1 , vous avez une 83+ Silver Edition

2 , vous avez une 84+

3, vous avez une 84+ Silver Edition

- régler le contraste: **real(5, Fonction, valeur**

fonction: 0 = donner au contraste une valeur

1 = mettre dans Ans la valeur du contraste (ne rien mettre comme valeur)

valeur: soit la valeur du contraste que vous voulez fixer soit rien.

- désactiver l'indicateur de réflexion: vous savez le truc moche qui tourne pendant les programmes en haut à droite de votre écran...

real(7 , valeur

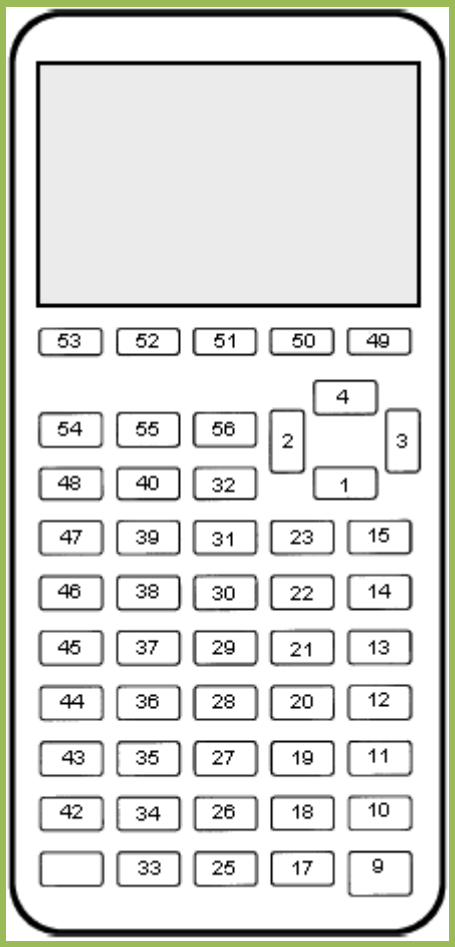
valeur: Off = 0 On = 1

- connaitre la valeur de la RAM libre: exécutez **real(14** , et Ans prendra la valeur de la RAM libre.

VII. Le getkey

Comme en TI Basic, cela sert à détecter si des touches sont pressées.

Getkey, en Xlib, se dit **real(8** . Quand on exécute l'instruction real(8 , Ans prend une valeur. Si Ans=0 , c'est qu'aucune touche n'est pressée, si Ans=nombre, cette carte peut vous aider à savoir quelle touche est pressée:



Attention, ce sont bien **les mêmes qu'en Axe**, ils sont **différents du TI Basic**.

Selon le créateur, Xlib peut supporter la pression simultanée de plusieurs touches.

Personnellement, je ne l'ai jamais réussi, Ans ne renvoie jamais de liste, si vous y arrivez, prévenez moi !

-Mais alors, quel avantage a utiliser le real(8 par rapport au Getkey ?

Il y a un chose qui est pratique avec real(8 : La valeur est réactualisée en permanence.

Le getkey affiche la valeur de la dernière touche pressée.

Si vous ne relâchez pas la touche pendant quelques secondes, le getkey revient à zéro, pas le real(8.

C'est donc pratique pour faire des jeux ou on doit avancer en permanence: appuyer en continu sur la touche suffit,

pas besoin de faire clic-clic-clic-clic-clic-clic-clic-clic...

Deuxième chose: si, tout en gardant la première touche pressée (pour avancer par exemple),

vous appuyez sur une autre pendant que la première est appuyée (pour sortir votre épée par exemple),

le Xlib va détecter cette seconde pression, contrairement au getkey qui reste sur 0.

Vous avez du mal à vous en rendre compte ? Faisons ensemble ce petit programme:

prgmTEST

```
:Repeat Ø          //ou While 1, comme vous préférez, c'est du TI Basic, pour faire une boucle infinie
:real(8
:ClrHome
:Output(1,1, Ans
:Output(2,1,getkey
:End
```

quand vous lancez ce programme, vous allez voir s'afficher en haut à gauche de votre écran les nombres donnés par le real(8 et en dessous ceux donnés par le getkey.

Pressez une ou plusieurs touches, vous verrez les différences.

VIII. Les options de texte.

Comment rendre un programme super beau facilement ? Avec la mise en forme du texte.
Il existe une fonction qui permet de rendre beau votre texte, en autorisant les minuscules, et/ou en inversant le texte, c'est à dire qu'au lieu d'écrire en noir sur blanc, la TI va écrire en blanc sur noir.

real(13, fonction , numéro du caractère

fonction: 0 = revenir à la normale 2 = minuscules On 4 = afficher un caractère spécial

1 = inverser le texte 3 = minuscules Off

numéro du caractère: à utiliser seulement si vous avez mis la fonction 4.

Vous avez surement remarqué que la TI utilise des caractères, des symboles, que l'on ne peut pas avoir via les menus. Cette fonction permet d'afficher sur le tableur un caractère en entrant son numéro.

Dans un programme, une fonction real(13,4,x n'est pas utile, puisqu'elle interrompt votre programme pour afficher un caractère sur l'écran, et qu'on ne peut rien faire avec.

-Comment savoir quel numéro qui correspond à quel caractère ?

Avec un programme:

prgmCHARACT

```
:Input "numero ? ",A
:real(13,4,A
```

ou encore, avec cette capture que j'ai faite pour vous dans mon immense bonté:

< uploader ici charXlib.gif >

Bien, bien, que nous reste-t-il donc a voir ?

Héhé, le plus intéressant, la copie de programme et l'archivage.

IX. Les programmes temporaires

Là, c'est ce qu'il y a de plus compliqué, mais rassurez vous c'est toujours aussi simple:

Il existe dans votre TI seize programmes dits **temporaires,** qui sont utilisés par divers langages de progra ou par le TI-OS lors de copie de programmes, ou mieux, lors de l'**exécution de programmes archivés**.

- je ne vois pas ces programmes dans la liste, comment savoir leur nom pour les utiliser ?

Si vous ne voyez pas leur nom, c'est bon signe, ça veut dire que les programmateurs ont bien travaillé ;)

En effet, ces programmes doivent être supprimés quand il ne servent plus, car comme tout programme,

Ils **pèsent leur pesant d'octets** ;) , et ce serait dommage de gaspiller de la mémoire.

Donc, en temps normal, ils sont supprimés, et on ne les voit pas.

Leur noms: **prgmXTEMP0xx** ou xx est un nombre en 00 et 15.

- OK. Et alors, comment exécuter un programme archivé ?

En copiant ce programme archivé dans un programme temporaire, puis en exécutant

ce programme temporaire.

*Hé oui, il vous **faut assez de RAM libre** pour l'équivalent de votre programme !*

Nous allons utiliser la fonction real(10 :

"*NOMDUPROGRAMME*":real(10 , fonction , numéro du programme temporaire

"nomduprogramme": avant d'exécuter real(10 , il faut que Ans soit égal au nom du programme à copier (une chaîne).

fonction: copier = 0 supprimer = 1 supprimer tous les temporaires = 2

numéro du programme temporaire: entre 0 et 15 , ce qui donnera des programmes entre prgmXTEMP000 et prgmXTEMP015.

<u>petite démonstration concrète:</u>

je veux exécuter le programme JEUX qui pèse 2000 octets et qui est archivé.

1. Je m'assure d'avoir assez de RAM libre.
2. Je supprime XTEMP001, au cas où. (j'ai oublié de vous dire que si le programme temporaire existe déjà avant la copie, il n'est pas remplacé, ça marche pas.)
3. Je copie JEUX qui est archivé dans le programme XTEMP001 (qui se crée toujours dans la RAM)

4. j'exécute le programme XTEMP001

5. Je supprime le programme XTEMP001, pour ne pas gâcher de la mémoire. (note: si vous ne le supprimez pas, et bien vous le verrez dans votre liste de programmes ;))

```
:real(14                    // teste la RAM libre
:If Ans>2ØØØ                // vérifie qu'il y a la place dans la RAM
:Then
:real(1Ø,1,1                // supprime XTEMP001 au cas où il existerait
:"JEUX"                     // met le nom du programme à copier dans Ans
:real(1Ø,Ø,1                // copie prgmJEUX dans prgmXTEMP001
:prgmXTEMPØØ1               // éxécute le programme XTEMP001
:real(1Ø,1,1                //suprimme XTEMP001
:End                        //termine la condition If du début.
```

D'autres questions ? =P

-Est-ce possible avec des programmes en assembleur ?

Oui, sauf que pour exécuter prgmXTEMP001, il faudra écrire Asm(prgmXTEMP001.

Il vous faut par contre faire attention à ce que votre programme ASM n'utilise pas lui le programme XTEMP001 !!

Parce que là, vous aurez un joli freezing-RAM cleared.

Si c'est le cas, eh bien utilisez XTEMP002, ou un autre des 16 !

X. Un petit récapitulatif ?

-oui, volontiers :)

Fonction	rôle
real(0	effacer l'écran général
real(1	dessiner un sprite
real(2	tilemapping
real(3	rappeler une Pic entière sur l'écran
real(4	scrolling de l'écran
real(5	contraste
real(6	Update LCD (afficher le back-buffer)

real(7	indicateur de réflexion On-Off
real(8	getkey, mais en mieux
real(9	créer une Pic à haut numéro
real(10	programmes temporaires
real(11	version du système
real(12	dessin (lignes, rectangles)
real(13	options de texte
real(14	RAM libre.

Bref. J'espère que ça vous a été utile, et que vous programmerez bien ! ;)

Bonne continuation sur Espace TI:Forum !

Persalteas.