DC_Land / Manuel du programmeur

Table des matières

Chapitre 1

DC_land

Avertissements

"Sega" et "Dreamcast" sont des marques déposées de Sega Enterprises, Ltd. Ce document n'a aucun lien avec Sega ou ses membres, travaillant sur leur matériel.

Toutes les autres marques citées sont des marques déposées par leurs propriétaires respectifs.

Avant-tout

Nous partons du principe que vous possédez quelques notions de programmation C et que votre environnement de développement est correctement configuré.

Pour cela, vous devez donc posséder:

- un compilateur capable de compiler pour processeur SH-4 (celui de la Dreamcast)
- la librairie KOS qui elle aussi doit être installée et correctement configurée (une lecture de sa doc ça peut aider)
- la librairie DC_LAND et ses exemples

Le dossier **dc_land** doit être copié dans **/kos-1.x.x/include/** et le fichier **libdc_land.a** dans **/kos-1.x.x/lib/**. Si tout ceci est ainsi configuré sur votre ordinateur vous ne devriez pas rencontrer trop de difficultés pour utiliser DC_LAND[1].

Les fonctions comportant trois points d'interrogation ??? ne sont pas vraiment utilisables où n'ont pas été défini par manque de documentation.

1.1 La librairie DC_Land

DC_Land a été écrit par GREEN GIANT afin de faciliter la programmation de la console SEGA DREAMCAST. Cette librairie est en grande partie basée sur KOS développée et maintenue par DAN POTTER.

DC_LAND possède peu de fonctions car elle a été abandonné, mais elle est un bon moyen de s'initier à la programmation sur DREAMCAST.

1.2 La vidéo (2d.h)

Dc_Land ne gère que la 2D et fournit quelques fonctions d'affichage. Ces fonctions sont décrites ci-dessous.

1.2.1 v_init()

Prototype void v_init()

Paramètres Aucun

Description Initialise l'écran (640x480).

[1] La manipulation est identique sous Linux et Windows.

1.2.2 v_clear()

Prototype void v_clear(int **r**, int **g**, int **b**)
Paramètres **r** = rouge, **g** = vert, **b** = bleu
Description Remplit l'écran avec la valeur définit par int **r**, int **g** et int **b**.

1.2.3 v_pixel()

Prototype void v_pixel(int **x**, int **y**, int **r**, int **g**, int **b**)
Paramètres **x**,**y** les coordonnées du point, **r**, **g**, **b** la couleur
Description Affiche un pixel.

1.2.4 v_circle()

Prototype void v_circle(int **x**, int **y**, int **radius**, int **r**, int **g**, int **b**)
Paramétres **x**,**y** les coordonnées du centre du cercle/ **radius** le rayon/ **r**, **g**, **b** la couleur
Description Trace un cercle.

1.2.5 v_drawstring()

Prototype void v_drawstring(int **x**, int **y**, char ***s**)
Paramétres **x**,**y** les coordonnées de la chaîne/ ***s** la chaîne de caractères
Description Permet d'afficher une chaîne de caratères.

1.2.6 v_drawstring_ctr()

Prototype void v_drawstring_ctr(int **x**, int **y**, char ***text**)
Paramétres **x**,**y** les coordonnées du texte/ ***text** le texte
Description Ressemble à la fonction précédente mais affiche un texte.

1.2.7 v_fade()

Prototype void v_fade()
Paramétres Aucun
Description Permet de créer un effet de fondu.

1.2.8 v_line()

Prototype void v_line(int **Ax**, int **Ay**, int **Bx**, int **By**, int **r**, int **g**, int **b)**
Paramétres **Ax**,**Ay** coordonnées du premier point/ **Bx**,**By** coordonnées du second point/
r, **g**, **b** la couleur
Description Trace une ligne à l'écran.

1.2.9 v_box()

Prototype void v_box(int **x**, int **y**, int **sx**, int **sy**, int **r**, int **g**, int **b**)

Paramétres **x**,**y** coordonnées du point supérieur/ **Bx**,**By** coordonnées du point inférieur/ **r**, **g**, **b** la couleur

Description Dessine un rectangle.

1.2.10 getPixel ()

Prototype uint32 getPixel(int **x**, int **y**)

Paramétres **x**,**y** coordonnées du point

Description Définit si un pixel est affiché à l'écran.

1.2.11 fillToBorder() ? ? ?

Prototype void fillToBorder(int **x**, int **y**, int **border**, int **colour**)

Paramétres

Description Cette fonction n'est pas correctement implémentée (bug).

1.2.12 v_clr()

Prototype void v_clr(int xpos, int ypos, int nx, int ny, int r, int g, int b)

Paramétres **xpos,ypos** coordonnnées supérieures/ **nx**,**ny** coordonnées inférieures/ **r**, **g** ,**b** la couleur

Description Remplit une partie de l'écran. S'utilise comme v_box et affiche un rectangle plein.

1.3 Accès maple

Le maple est un bus développer pour/par SEGA afin de gérer les périphériques (clavier, pad, souris, ...).

1.3.1 detect_maple()

Prototype void detect_maple()

Paramètres Aucun

Description Détecte si un périphérique d'entrée est connecté à la DREAMCAST.

1.3.2 keyboard_exists() ? ? ?

Prototype uint8 keyboard_exists()

Paramètres Aucun

Description Permet de vérifier si un clavier est branché. (Cette fonction n'est pas prise en compte)

1.3.3 vmu_exists() ? ? ?

Prototype uint8 vmu_exists()

Paramètres Aucun

Description Permet de vérifier si un vmu est branché. (Cette fonction n'est pas prise en compte)

1.3.4 pad_exists()

Prototype uint8 pad_exists()

Paramètres Aucun

Description Permet de vérifier si un pad est branché.

1.4 Afficher un menu

Dc_Land fournit une fonction intéressante qui permet de créer un menu. Le bouton A sert à valider la sélection et le bouton start permet de sortir du menu.

1.4.1 add_option ()

Prototype int add_option (struct a_menu ***menu**, char *caption)

Paramètres Pointe sur une structure ***menu**/ ***caption** nom du choix

Description Permet de créer la liste de choix composant le menu.

1.4.2 display_menu ()

Prototype int display_menu (struct a_menu *menu, int x, int y, int box_flag)

Paramètres **x**,**y** la position du menu à l'écran/ renvoie le numéro du choix éffectué 0 étant le premier

Description Affiche le menu créer grâce à add_option().

1.5 Le pad

1.5.1 get_button ()

La fonction get_button() permet de définir quel bouton a été appuyé. Elle ne prend pas en compte les gachettes et le stick. Lorsque vous appuyez sur une touche, get_button() renvoie le bitmask correspondant.

Bouton	Bitmask
X	btnX
Y	btnY
A	btnA
B	btnB
Start	btnSTART
Haut	dpadUP
Bas	dpadDOWN
Gauche	dpadLEFT
Droite	dpadRIGHT

Prototype int get_button ()

Paramètres aucun

Description Renvoie le bitmask du bouton qui a été appuyé.

1.6 Jouer un son

1.6.1 s3m_play() ? ? ?

Prototype void s3m_play(char *s3m)

Paramètres Lit un fichier dans "/rd/*.s3m" (romdisk)

Description Permet de jouer un son au format s3m (format Amiga).

1.7 Les chaînes de caractères

Ces deux fonctions permettent de récupérer soit une chaîne de caractéres soit une valeur. Par exemple pour savoir dans un menu quel choix a été sélectionné.

1.7.1 char *strrev (char *s)

Récupère une chaîne de caractères.

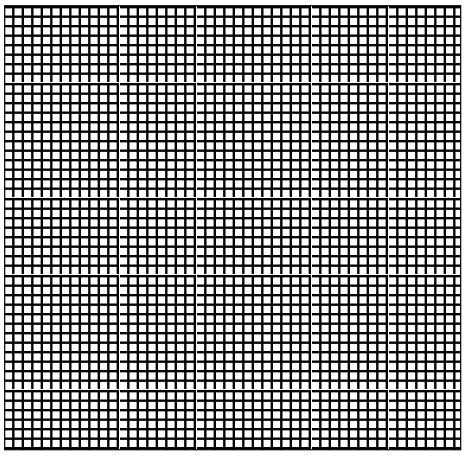
1.7.2 char *itoa (int val)

Récupère une valuer.

1.8 les tiles

La Dreamcast a une architecture lui permettant d'afficher rapidement des tiles (tesselles) grâce à sa puce graphique NEC PVR2.

Une tesselle est la plus petite partie qui compose une mosaïque. Donc le tile est la plus petite partie qui compose une image à l'écran. Un carré sur notre grille (tilegrid).



1.8.1 draw_tilegrid ()

Prototype void draw_tilegrid (struct tilegrid_t *tileGrid)

Paramètres

Description Affiche un tilegrid c'est-à-dire une grille formée par les tiles mis côte à côte.

1.8.2 draw_tile () ? ? ?

Prototype int draw_tile (struct tilegrid_t *tileGrid, TILE_ID id)

Paramètres

Description Dessine un tile.

1.8.3 addTile()

Prototype void addTile(struct tilegrid_t *tileGrid,int **x**, int **y**)

Paramètres le tilegrid/ **x**,**y** sont les coordonnées du tile sur la grille

Description Dupplique un tile.

1.8.4 create_tilegrid()

Prototype void create_tilegrid(int x, int y, int s1, int s2, int sx, int sy)

Paramètres **x**,**y** coordonnées du point supérieur/ **s1**,**s2** dimension de la grille/ **sx**,**sy** coordonnées du point inférieur)

Description Crée une grille permettant d'afficher les tiles.

1.8.5 clear_tile ()

Prototype int clear_tile (struct tilegrid_t *tileGrid, **TILE_ID id**)

Paramètres **tileGrid**, **TILE_ID id**

Description Efface un tile.

1.8.6 TILE_ID addTile (struct tilegrid_t *tileGrid, int xPos, int yPos) ? ? ?

Prototype

Paramètres

Description

1.8.7 putTile () ? ? ?

Prototype int putTile (struct tilegrid_t *tileGrid, TILE_ID id, int xPos, int yPos)

Paramètres

Description

1.8.8 moveTile ()

Prototype int moveTile (struct tilegrid_t ***tileGrid**, TILE_ID id, int **direction**)

Paramètres ***tileGrid** le tile/ **direction** la direction (voir le tableau plus bas)

Description Permet de déplacer un tile sur l'écran suivant une direction définie.

Direction(croix directionnelle)	Fonction de déplacement
Haut	tileUP
Bas	tileDOWN
Gauche	tileLEFT
Droite	tileRIGHT

1.9 utils

1.9.1 DC_sleep()

Prototype void DC_sleep(int m)

Paramètres m exprime le temps en milliseconde

Description Gèle la console pendant m millisecondes.

1.10 vmu ? ? ?

Dc_land permet d'afficher son propre icône sur le vmu (visual memory unit).

1.10.1 vmu_icon_init()

Prototype void vmu_icon_init(const char *vmu_icon)

Paramètres const char *vmu_icon contient le fichier.

Description Permet d'afficher un dessin sur le vmu.

1.11 Remerciements

Je tiens premièrement à remercié JMD pour le temps qu'il a bien voulu me consacrer afin de m'expliquer le fonctionnement et les subtilités de Kos et Green Giant qui a écrit Dc_Land.

1.12 A propos ...

Ce document a été écrit par Olivier NOUGUIER (olivier.nouguier@wanadoo.fr) avec LyX 1.3.3 sous GNU/Linux Mandrake 10 Community se basant sur la documentation Kos de Dan POTTER, les exemples se trouvant dans les dossier **dcland_examples** et le livre *X window* de Albert JANSSENS.