DC\_Land / Manuel du programmeur

©2004 NOUGUIER Olivier

# Table des matières

D C	C $land$
1.1	$^{\prime\prime}$ La librairie DC_Land
1.2	$^{2}$ La vidéo (2d.h) $^{-}$
	1.2.1 $v_init()$
	$1.2.2$ $v\_clear()$
	1.2.3 $v_pixel()$
	1.2.4 $v\_circle()$
	$1.2.5$ $v\_drawstring()$
	$1.2.6  v\_drawstring\_ctr()  \dots  \dots  \dots  \dots  \dots  \dots$
	1.2.7 v_fade()
	1.2.8 $v\_line()$
	$1.2.9  v\_box()$
	1.2.10 $getPixel()$
	1.2.11 $fillToBorder()$ ? ? ?
	1.2.12 v_clr()
1.3	Accès $maple$
	1.3.1 $detect\_maple()$
	$1.3.2$ $keyboard\_exists()$ ? ? ?
	$1.3.3  vmu\_exists()???$
	1.3.4 pad_exists()
1.4	Afficher un menu
	1.4.1 add_option()
	1.4.2 display_menu ()
1.5	5 Le $pad$
	1.5.1 get_button ()
1.6	$\delta$ Jouer un son
	1.6.1 $s3m_play()$ ? ? ?
1.7	$egin{array}{ll} egin{array}{ll} Les \ chaînes \ de \ caract\`{e}res \end{array} \end{array} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \end{array}$
	1.7.1 char *strrev (char *s)
	1.7.2 char *itoa (int val)
1.8	·
	1.8.1 draw_tilegrid ()

	1.8.2	dr	$aw_{\_}$	tile	0	? ?	?																										10
	1.8.3	ad	dTi	le()																													10
	1.8.4	cre	eate	_ <i>til</i>	egr	id	)																										11
	1.8.5	$cl\epsilon$	ear_	tile	0																												1
	1.8.6	TI	LE	_ <i>ID</i>	aa	ddT	$\Gamma ile$	e (	(st	ri	$\iota ct$	t t	ile	gr	id	_ t	*	til	eC	ri	d,	ir	it :	xP	os	, i	nt	yI	Pa	sj	?	? ?	11
	1.8.7	pu	tTi	le ()	? ?	? ?																											11
	1.8.8	m	ove'	Tile	0																												11
1.9	utils.																																12
	1.9.1	D0	C_ 8	sleep	0																												12
1.10	vmu ?	? ?																															12
	1.10.1	vn	$u_{\perp}$	icon	<i>1_ i</i>	nit	()																										12
1.11	Remer	cier	nen	ts .																													12
1.12	A prop	00s			_							_	_	_		_					_	_						_				_	12

# 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.

KallistiOS ©2000-2002 DAN POTTER. Autres portions © leurs auteurs individuels; s'il vous plaît contacter les auteurs pour information afin d'utiliser et/ou distribuer leur code.

DC\_Land GREEN GIANT. Autres portions © leurs auteurs individuels; s'il vous plaît contacter les auteurs pour information afin d'utiliser et/ou distribuer leur code.

La redistribution et l'usage sous forme de source ou de binaire, avec ou sans modification, sont permises tant que les conditions suivantes sont réunies :

- 1. Les redistributions du code source doivent respecter le copyright ci-dessous, cette liste de conditions et le disclaimer suivant.
- 2. Les redistributions sous forme binaire doivent reproduire le copyright ci-dessous, cette liste de conditions et le disclaimer suivant dans la documentation et/ou autres supports distribués avec la distribution.
- 3. Dans aucune mesure le nom de Cryptic Allusion ni les noms de ses collaborateurs ne doivent être utilisé endosser ou promouvoir les produits dérivés de ce logiciel sans permissions spécifiques écrites.

CE LOGICIEL EST FOURNI PAR LES AUTEURS ET LES COLLABORTEURS "COMME TEL" ET NE DONNE AUCUNES GARANTIES. DANS AUCUN CAS LES AUTEURS OU LES COLLABORATEURS NE SONT TENUS RESPONSABLES DES DOMMAGES DIRECTS, INDIRECTS, PERTE D'UTILISATION, DE DONNÉES, OU DE BÉNÉFICES; OU INTERRUPTION D'AFFAIRES) CAUSES PAR L'UTILISATION DE CE PROGRAMME.

#### 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<sup>1</sup>.

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).

<sup>&</sup>lt;sup>1</sup> La manipulation est identique sous Linux et Windows.

#### 1.2.2 v clear()

Prototype void v\_clear(int r, int g, int b)

Paramètres  $\mathbf{r} = \text{rouge}, \, \mathbf{g} = \text{vert}, \, \mathbf{b} = \text{bleu}$ 

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()

 ${\bf Prototype} \ {\rm void} \ v\_{\rm fade}()$ 

Paramétres Aucun

**Description** Permet de créer un effet de fondu.

#### 1.2.8 v line()

 $\mathbf{Prototype} \ \mathrm{void} \ \mathrm{v\_line}(\mathrm{int} \ \mathbf{Ax}, \ \mathrm{int} \ \mathbf{Ay}, \ \mathrm{int} \ \mathbf{Bx}, \ \mathrm{int} \ \mathbf{By}, \ \mathrm{int} \ \mathbf{r}, \ \mathrm{int} \ \mathbf{g}, \ \mathrm{int} \ \mathbf{b})$ 

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.

#### $1.2.9 \quad \text{v} \quad \text{box}()$

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 coordonnées 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
В	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.

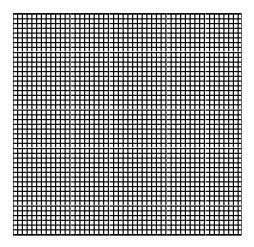
#### 1.7.2 char \*itoa (int val)

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  $\mathbf{x}$ , int  $\mathbf{y}$ )

 ${f Paramètres}\,$  le tilegrid/  ${f x},{f y}$  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	${ m tile UP}$
Bas	${ m tile DOWN}$
Gauche	${ m tile LEFT}$
Droite	${ m tile RIGHT}$

#### 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.