Aide pour l'utilisation des différents outils

Nicolas Lebas
UMR INRA - INA-PG
Environnement et Grandes Cultures

30 septembre 2005

Table des matières

1 Description du format .can

Le format `.can`[1], est un format relativement libre qui permet de définir une liste de polygones dans un espace 3D. Il est également possible d'ajouter de nouveaux champs sans difficulté (Canestra utilise un champs pour définir un label permettant de déterminer les propriétés optiques de chaque polygone de la scène).

c1	c2	c1'	c2'	c3	c4	.	.	.	.	.	.	.	c_n
p	2	100001001002	8	3	32.6818	28.27	3.4611	32.42	28.01	3.5307	32.66	26.96	3.14

• Champ 1 : type de figure géométrique (p = polygone, seul figure interprété par canestra actuellement).
• Champ 2 : nombre de champs qui vont être déclarés avant de donner le nombre de cotés du polygone (2 champs sont spécifié ici).
Remarque : un champs est obligatoire pour Canestra placé en position *c1'* : qui correspond au label du polygone).

⋆ Champ 1' (1^{er} champs additionnel) : label permettant de décrire les propriétés optiques du polygone :

1	.....	00001	.....	001	.....	002
espèce optique	.....	n^o de plante	.....	type d'organe (0 = tige, sinon feuille)	.....	n^o de triangle

⋆ Champ 2' (2^{eme} champs additionnel) : autre champs personnalisé.

• Champ 3 : nombre de sommets du polygone (3 dans l'exemple car Canestra gère des triangles).
• Champ 4 ... n : coordonnées des sommets.

2 Description du format .opt

Le format `.opt`[2], permet de décrire les propriétés optiques d'une scène 3D, sous l'hypothèses de surfaces Lambertiennes (c'est-à-dire isotropes). Les propriétés optiques du sol sont définis uniquement par une valeur de réflectance (la transmitance étant nulle). Une liste d'espèces optiques est ensuite définie contenant chacune les propriétés optiques données sous la forme d'une réflectance pour les tiges, et d'une réflectance et d'une transmitance pour chacune des faces des feuilles. Les tiges sont considérées comme étant opaques, c'est pourquoi seul une valeur de réflectance est donnée (la transmitance $\tau = 0$).

```
1 : #format e : tige,  feuille sup,  feuille inf
2 : # nbre d'especes
3 : n 1
4 : #1 ESPECES
5 : s d 0.5
6 : # espece 1
7 : e d 0.4 d 0.00001 0.000005 d 0.00001 0.000005
8 : e d 0.4 d 0.02501 0.00025 d 0.00401 0.054
```

Le `'e'` signifie que l'on décrit une nouvelle espèce optique. Le `'d'` quant à lui signifie 'diffuse' ce qui signifie que l'on va déclarer une propriété diffuse qui prend une valeur de réflectance ρ, et dans le cas d'une texture transparente une valeur de transmitance τ.
Dans les fichiers `.opt` le nombre donné après le premier `d` correspond à la réflectance de tige (R_s), les deux nombres donnés après le second `d` concernent les faces supérieures des feuilles : réflectance de feuille (R_f) et transmitance de feuille (T_f), les deux nombres donnés après le troisième `d` correspondent aux mêmes paramètres que précédemment mais pour la face inférieure des feuilles.
• Ligne 3 : nombre d'espèces optiques différentes.
• Ligne 5 : propriétés optiques du sol
• Ligne 7 : propriétés optiques de l'espèce pour une longueur d'onde ou une bande spectrale (ex. : NIR, PAR).

`e`	`d`	`0.4`	`d`	`0.00001`	`0.000005`	`d`	`0.00001`	`0.000005`
nouvelle espèce	diffuse	Rs	diffuse	R_f(face sup)	T_f(face sup)	diffuse	R_f(face inf)	T_f(face inf)

[1] La documentation officielle du format *can* est présentée en annexe
[2] La documentation officielle du format *opt* est présentée en annexe

3 Canestra

Canestra est une application permettant d'estimer l'énergie absorbée (E_{abs}), l'énergie incidente (E_{inc}) et l'énergie émise/réfléchie (B_i) d'une scène 3D. Différentes méthodes de calcul peuvent être utilisées pour simler les échanges radiatifs : la projection (ou ombre portée), la radiosité classique et, s'il est associé avec *S2v* et *Mc-Sail*, la radiosité mixte.

Exemple d'utilisation : `canestrad -M Test.can -8 Test.8 -l sky.light -p par.opt -e par.env -r 0.2`.
Voici les différentes options possibles :

```
-M filename    File describing the scene
-m shm_key     Shared memory containing the scene
-s Ns          Append a soil to the scene (Ns is a treshold for the number of triangles)
-p filename    File describing the optical properties
-l filename    File describing the light sources
-8 filename    Infinite periodic canopy
-e filename    Mean fluxes data (computed by SAIL)
-r Rsph        Radius of the surrounding sphere
-d Dsph        Diameter of the surrounding sphere
-f filename    Simulate and store the matrix in filemane
-w filename    Read the matrix file to simulate an other radiative case, without to compute form factors
-t dirname     Name of the directory where the FF file is stored
-F             Print the form factors matrix
-R nb          Resolution of the projection disk [52]
-S nb          Number of simulations [1]
-i nb          Number of iteration of the CG solver [100]
-a threshold   Threshold of the CG solver [1e6]
-1             Compute only the direct lightning
-L nb          Resolution of the light screen [1536]
-A             Generate  energy vector (Eabs.dat, Einc.dat)
-g             Generate the geometry file (geom.dat)
-B             Test the effect of the choice of inner triangles (bias?)
-T             Estimate the maximum required memory
-v nb          The level of verbose
-C filename    File describing the virtual sensors
-h             This help message
```

Liste des paramètres OBLIGATOIRES :
- La maquette doit être passée en argument soit sous la forme d'un fichier `.can` : `-M <file_maq.can>`, soit à l'aide d'un segment de mémoire partagée : `-m`.
- L'environnement lumineux de la scène, correspondant à la description des différentes sources lumineuses[3], sera fournie à l'aide de l'argument `-l` : `-l <fichier.light>`.

Liste des autres principaux paramètres :
- Si l'argument `-8 <fichier.8>` est fourni, *canestra* génèrera un champ de taille infini. Dans le cas contraire (si aucun fichier `.8` n"est spécifié), soit l'option `-1` est présente, dans ce cas la méthode par projection sera utilisée, soit un diamètre de shpère $<$ à 1 doit être fourni en argument pour que *canestra* utilise la radiosité classique. Il faut noter que l'algorithme utilisé dans le cas de la radiosité classique est de complexité N^2 (contrairement aux autres), ce qui aura pour effet d'augment er le temps de calcul, voir même dans certains cas de saturer la mémoire.
Le fichier `.8` renseigne sur les bords du rectangle engobant la scène à dupliquer "infiniment" (inférieur à la taille de la scène pour éviter les effets de bords) correspondant généralement au pas de semis et à l'inter-rang. L'algorithme de *canestra* utilisé pour dupliquer les scènes ne gére pas les parties dépassant de la boîte englobante[4] (c'est l'outil

[3] voir l'annexe sur la description des *.opt*
[4] voir le chapitre sur Periodise

periodise qui va permettre de traiter ce cas de figure).
- Comme nous l'avons déjà évoqué, *canestra* prend en compte les rediffusions de lumière (en utilisant la radiosité classique ou mixte) sauf dans le cas de la projection simple. Pour pouvoir utiliser la méthode de projection, l'argument `-1` doit être fourni à *canestra*.
- Le diamètre de la sphère passé en argument `-r Ds` est exprimé dans la même unité métrique que le fichier d'entrée (maquette `.can`).
- Les propriétés optiques sont fournies sous la forme de fichiers `.opt`[5], à l'aide de l'argument `-p` : `-p <fichier.opt>`. Les fichiers `.opt` contiennent les propriétés optiques de chaque type d'organe pour une longueur d'onde ou une bande spectrale donnée. Ce sont les paramètres de Lambertien qui sont fournis ici : `Rs`, `Rf` et `Tf` (respectivement Reflectance de tige, Reflectance de feuille et Transmitance de feuille).
- Le paramètre `-e` est nécessaire pour utiliser l'algorithme de radiosité mixte. Il permet de fournir à canestra le fichier contenant les contributions lointaines calculées par *Mc-Sail* : `-e <file.env>`.
- Le paramètre `-s` permet d'ajouter des triangles reconstituant un sol, si la scene fournie n'en possède pas (il est également possible de fournir à la suite de l'argument `-s` un entier correspondant au nombre maximum de triangles qui constitueront le sol).
- L'option `-C` permet de définir des capteurs virtuels placés dans la scène. Le fichier permettant de décrire les différents capteurs est un fichier `.can` un peu particulier : la première ligne doit être constituée du symbole # suivi par le nombre de capteurs `nb` présents dans la scène (se seront les `nb` lignes suivantes non commentées qui seront lues), ex : `#3` si l'on a trois capteurs. Un capteur est défini par un unique triangle, et dans le champ réservé à l'identifiant on place le numéro du capteur. Exemple ci-dessous pour un capteur $n^0 5$:

```
p 1 5  3   4 4 18   6  4 18   6  6 18
```

Les résultats de *canestra* se retrouvent écrits dans les différents fichiers `.dat`, la première lettre du nom du fichier correspondant au nom de la mesure. Ils contiennent chacun une liste de valeur correspondant à l'estimation de la mesure pour chacun des triangles. Les triangles sont listés dans le même ordre que dans le fichier d'entrée.
ATTENTION : si la surface d'un triangle est inférieure à 10^{-8}, alors le triangle ne sera pas pris en compte par *canestra* !

Remarque : il existe une version de canestra "Hard Drive" permettant de stocker les calculs de facteurs de forme dans des fichiers sur le disque afin de ne pas les recalculer lors d'une nouvelle estimation du rayonnement intercepté sur la même scène (pour une longueur d'onde différente par exemple). Ce sont les arguments `-f` et `-w` qui permettent respectivement d'écrire sur le disque et de lire la matrice des facteurs de forme (l'option `-d` permet de choisir le répertoire dans lequel se trouve le fichier à écrire ou à lire s'il n'est pas dans le répertoire courant).
C'est cette version qui est actuellemnt utilisée.

Exemple d'utilisation (écriture) :
```
canestrahd -M /tmp/virtualis.can -8 Tout.8 -l sky2.light -p nir.opt -e nir.env -s -r 0.2 -t tmp -f caribu.
```
Exemple d'utilisation (lecture) :
```
canestrahd -M /tmp/virtualis.can -8 Tout.8 -l sky2.light -p nir.opt -e nir.env -s -r 0.2 -t tmp -w caribu.
```

Fichiers générés par *canestra* :
`Bf.dat`, `Mcoef.dat` et `Mvec.dat` sont les fichiers de stockage des matrices et vecteurs servant au calcul de la radiosité.
`B0.dat` est le fichier réfléchi à l'ordre 1, `B.dat` est celui pour tous les ordre de rediffusion.

Pour les `Eabs.vec`, `Einc.vec` et `Etri.vec` il faut avoir mis l'option `-A` dans la ligne de commande !
`Einc` contient l'éclairement par unité de surface.
`Eabs` contient l'énergie absorbée par triangle (et non par unité de surface).
`Etri` est une synthèse des 2 avec des informations sur le triangle, à savoir n lignes constituées ainsi :
label_du_triangle aire_du_triangle énergie_absorbée_par_unité_de_surface éclairement_face_inf éclairement_face_sup

Les fichiers `B_nir.dat`, ...etc sont génerés par `caribu.csh`, a priori c'est une copie du B.dat pour la longueur d'onde donnée.

[5] voir chapitre sur le format *.opt*

4 Mc-sail

Multi Couches SAIL est un programme qui calcule les profils verticaux de flux hemispheriques E+ et E- necessaires à la radiosité mixte pour calculer les `Bfar`. Il calcule les contributions lointaines (sous la forme de fichier `.env`). *Mc-Sail* nécessite pour fonctionner que les fichiers générés par *S2v* se trouve dans le répertoire courant.

Exemple d'utilisation : `mcsail sky.light`.

Argument OBLIGATOIRE :
- Le fichier météo correspondant à l'environnement lumineux (dans le même format que pour *canestra*).

REMARQUE : les fichiers générés par s2v doivent être présents dans le répertoire où *mc-sail* est exécuté pour pouvoir l'utiliser.

5 S2v

S2v calcule les parametres de structure (LAI, LIDF) de SAIL a partir d'une liste de triangles. Il stratifie la scène en `nz` couches.

Exemple d'utilisation : `s2v plante.can 3 18 Tout.8 test nir par`.

Les différentes options OBLIGATOIRES :

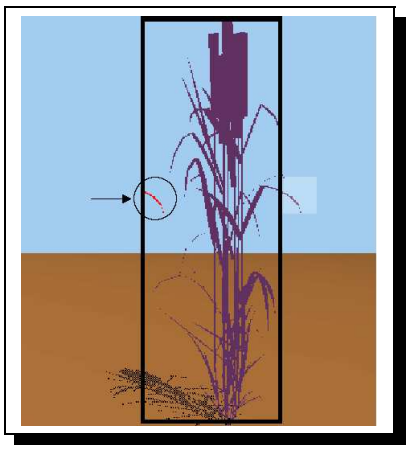
```
s2v [[shm_id | file.can] nz Dz file.8   [file_1.opt ... file_n.opt]]]
```

- Le 1^{er} argument est la maquette 3D (sous la forme d'un fichier `.can` ou sous la forme d'un segment de mémoire partagée).
- Le 2^{eme} argument est le nombre de couches : `nz` du couvert.
- Le 3^{eme} argument est la hauteur du couvert `Dz` (donc a quelque chose près le max des `z` des triangles de la scene `.can`).
- Le 4^{eme} argument est le fichier `.8` contenant le motif de la scène à dupliquer.
- Enfin un ou plusieurs fichiers contenant les propriétés optiques des éléments de la scène sont ensuite fournis en argument.

ATTENTION : les différents noms des fichiers `.opt` doivent être fournis sans l'extension, ex. :
`s2v plante.can 3 18 Tout.8 test nir par` (pour les fichiers `nir.opt` et `par.opt`).

6 Periodise

Periodise permet de dupliquer un motif d'une scène 3D de façon périodique. Il prend en compte les parties de la scène qui "débordent" de la boîte englobant le motif à dupliquer : elles sont réinjectées dans le motif de l'autre coté (comme si la scène était périodique).



Exemple d'utilisation : `periodise -m plante.can -8 Tout.8 -o /tmp/virtualis.can`.

`usage : periodise [8<infty> o<out> m<in> p<opt>]`

Les différentes options OBLIGATOIRES :
- La maquette de la scène : `-m <scene.can>`.
- Le fichier définissant la boîte englobant le motif à dupliquer : `-8 <bounding_box.8>`.

Les autres options utilisables :
- Un fichier de propriétés optiques peut être transmis en argument : `-p file.opt`.
- La sortie peut être rediriée dans un fichier : `-o out`.

7 Gensky

Gensky permet de créer un ciel discrétisé sous la forme d'un ensemble de sources lumineuses, au format `.light`, à partir de données météo.

Exemple d'utilisation : `gensky` (répondre aux différentes questions).

8 Can2vgx

Can2vgx permet de convertir un fichier `.can` (avec un seul champ en identifiant) en `.vgx` (entrée de *VegeStar*).

Exemple d'utilisation : `can2vgx <inputFile.can> <outputFile.vgx>`.

REMARQUE : Le fichier `.can` ne doit pas comporter de lignes vides a la fin !

9 Graphtal

Graphtal est un simulateur basé sur les L-Systems. Il permet également de faire de l'interprétation géométrique. Un visualiseur statique (X11) permet d'avoir accès aux sorties graphiques du modèle.

Exemple de lancement : `graphtal -d no leaf.lsys`.

```
usage: graphtal [options] [filename]
Where options include:
        -O outfile      (Set output file name.)
        -R xres yres    (Render at given resolution.)
        -E x y z        (Set eyepoint vector.)
        -L x y z        (Set lookat vector.)
        -U x y z        (Set up vector.)
        -Dname          (Define name as 1 (cpp option).)
        -Dname=def      (Define name as "def" (cpp option).)
        -f angle        (Set field of view.)
        -d drivername
           no           (No turtle interpretation.)
           example      (Example driver.)
           can   (Can driver.)
           bbox         (Calculate bounding box and view parameter.)
           rayshade     (Rayshade driver.)
           x11simple    (Simple line drawing driver for X11.)
           x11wire      (Wire frame driver for X11.)
           flat         (Simple z-buffering.)
        -c              (Toggle cone spheres generation.)
        -s              (Show defined hulls.)
        -v              (Verbose output.)
        -q              (Run quietly.)
        -l              (Print l-system definition.)
        -p              (Print resulting module string.)
        -e              (Erase module after turtle interpretation.)
        -h              (Print this message.)
```

Les différentes options OBLIGATOIRES :
- Le script `.lsys` contenant la description du modèle doit être fourni en dernier.

Autres arguments :
- L'option `-d` permet de choisir le type de sortie souhaité (par défaut *graphtal* affiche les sorties sur son visualiseur). Si Cette option est suivie de `can` alors la scène s'affichera sur la sortie standard sous la forme d'un fichier au format `.can`, mais d'autres types de sorties sont possibles.
- L'option `-p` permet d'afficher la chaîne L-System obtenue après application des règles.

Input files for canestrad

M. Chelle (1998)

A.1 Canopy File (.can) Description

A CAN file contains the geometric description of a vegetation canopy. The CAN format allows to associate double precision data to a geometric description for each primitive.

This ASCII file contains one line per primitive. Each line is made like this :

T(type of primitive)	nb_id (number of data)	data 1	...	data nb_id	Geometric description
char	int	double	...	double	char string

The type of primitive is : polygon (p), polygon with vertex normals (n), disc (d), cylinder (y), cone (c), sphere (s) and a special type ($\#$), which allows comment lines in the file.

The current version of our programs uses only polygonal primitives (p). The geometric description of a polygone is a char string :

nbv (number of vertices)	1_x (x-coordinate of the 1st vertex)	1_y	1_z	...	nbv_x	nbv_y	nbv_z
int	float	float	float	...	float	float	float

Each line contains also a list of nb_id integer data. To use our programs the first data may be construct like this :

$$data\,1 \;=\; e * 1e11 \;+\; p * 1e6 \;+\; f * 1e3 \;+\; t$$

where e is the optical specie index (if e is equal to 0, the primitive belongs to the soil), p is the plant index, f is the leaf index (if f is equal to 0, the primitive belongs to a stem, else it belongs to the leaf number f), t is the polygon index.

For example, if $data1 = 20073005025$, we know that this primitive is the 25th primitive of the leaf number 5 of the 73th plant of the optical specie number 2 !

This $data\,1$ is very important for our programs, because it allows to associate optical properties to geometric primitive. It allows also to compute cumulated statistics by leaf, plant, specie,....

To sum up, this is a example of a CAN file, which describes a pyramid :

```
# pyramid.can
p   1   100001001000   3    0.0   0.0   0.0    5.0   8.66    0.0   10.0    0.0    0.0
p   1   130001001000   3    0.0   0.0   0.0   10.0    0.0    0.0    5.0   2.88   8.16
p   1   160001001000   3    0.0   0.0   0.0    5.0   2.88   8.16    5.0   8.66    0.0
p   1   190001001000   3   10.0   0.0   0.0    5.0   8.66    0.0    5.0   2.88   8.16
```

A.2 Optical Properties File (.opt)

This current version of *canestrad* deals only with lambertian or diffuse objects. A diffuse property (d) is described by a real coefficient : the reflectance (ρ), and in the case of transparency, by an other one : the transmittance (τ).

This properties are described in the .opt file by optical species and by kind of diffuser : opaque (soil or stem) or transparent (leaf).

Each line beginning with "#" is a comment line. The line beginning with "n" gives the number of species described in .opt file :

```
n 4
```

The line beginning with "s" gives soil properties *eg*

```
s d 0.35
```

The lines beginning with "e" describe optical properties by specie *eg*

e $\underbrace{d\ 0.4}_{\text{stem}}$ $\underbrace{\underbrace{d\ 0.4\ 0.45}_{\text{up side}}\ \underbrace{d\ 0.3\ 0.3}_{\text{down side}}}_{\text{leaf}}$

For a given specie *ie* a "e"-line, the first optical property describes stem property, the second the up-side of a leaf and the third the down-side of a leaf. "up-side" and "down-side" are determined in function of the orientation of the normal of a primitive *ie* "up-side" is the face of the primitive that "looks" in the direction of the normal. The optical specie of a Lsytem module i To sum up, this is an example of an .opt file for 2 optical species :

```
# Example of .opt FILE
# number of species
n  2
# Soil
s  d 0.35
# Specie 1
e  d 0.2   d 0.2 0.2     d 0.2 0.2
# Specie 2
e  d 0.05  d 0.045 0.05  d 0.045 0.05
```

A.3 Light sources file

Light sources, available in *canestrad*, are described as a set of directional collimated sources. A collimated source is described geometrically and radiatively. Geometry requires only a vector given by cartesian coordinates. A source is radiatively characterized by the the irradiance of a horizontal surface located just above the canopy due to the source.

The .light file is made of one line by collimated source *ie* by direction of the sky hemisphere. Each line contains the irradiance followed by the vector of direction of light *eg* $\underbrace{650.0}_{\text{PAR}}$ $\underbrace{0\ 0\ \text{-}1}_{\text{vector}}$ for the sun located at the zenith in the PAR spectral band.

Spécificités techniques des utilitaires

Nom	Langage	O.S.	Version	Description
Canestra	C++	Win, MacOs, Linux	1.0 (1998)	estimation du rayonnement intercepté
Mc-Sail	C++	Win, MacOs, Linux	1.9 (1998)	estimations des contributions lointaines
S2v	C++	Win, MacOs, Linux	1.0 (1998)	calcule les paramètres de structure (LAI, LIDF)
Periodise	C++	Win, MacOs, Linux	2.0 (1998)	génère une scène périodique
Gensky	Python	Win, MacOs, Linux	0.1 (1998)	génère un ciel discrétisé
Can2vgx	Python	Win, MacOs, Linux	0.1 (2005)	convertit un `.can` en `.vgx`
Graphtal	C++	Win, MacOs, Linux	2.0 (1992)	simulateur basé sur les L-Systems