Créez des niveaux supplémentaires

1. Installez des niveaux supplémentaires

L'installation et la création de niveaux supplémentaires nécessite une version **1.7** (ou supérieure) de CoLoBoT. Si ce n'est pas le cas, vous pouvez charger un patch sur le site www.colobot.com.
Les niveaux supplémentaires sont accessibles par le bouton « Suppl. » de l'écran principal :

Tous les niveaux supplémentaires sont contenus dans le dossier `user\` placé dans le dossier principal où est installé CoLoBoT. Généralement, il s'agit de :

- `C:\Program Files\Colobot\user\`

Si le dossier `user\` n'existe pas, il suffit de le créer.
Un niveau supplémentaire peut contenir une ou plusieurs missions. Chaque niveau supplémentaire apparaît sous la forme d'une ligne dans la liste de gauche :

Un niveau supplémentaire est un dossier contenant plusieurs fichiers. Dans l'exemple ci-dessus, le niveau « Sample #1 » est contenu dans le dossier :

- `C:\Program Files\Colobot\user\sample01\`

Notez que le nom du niveau tel qu'il apparaît dans la liste de gauche n'est pas le nom du dossier, mais le nom contenu dans le fichier `sample01\scene00.txt`, sous la commande :

- `Title.E text="`**`Sample #1`**`" resume="Sample"`

Les différentes missions d'un niveau présentes dans la liste de droite correspondent aux fichiers `sample01\sceneNN.txt` :

Fichier	**Titre**
`sample01\scene01.txt`	Alerte
`sample01\scene02.txt`	Système D

Les fichiers doivent avoir des numéros successifs. Supposons par exemple que les fichiers suivants existent :

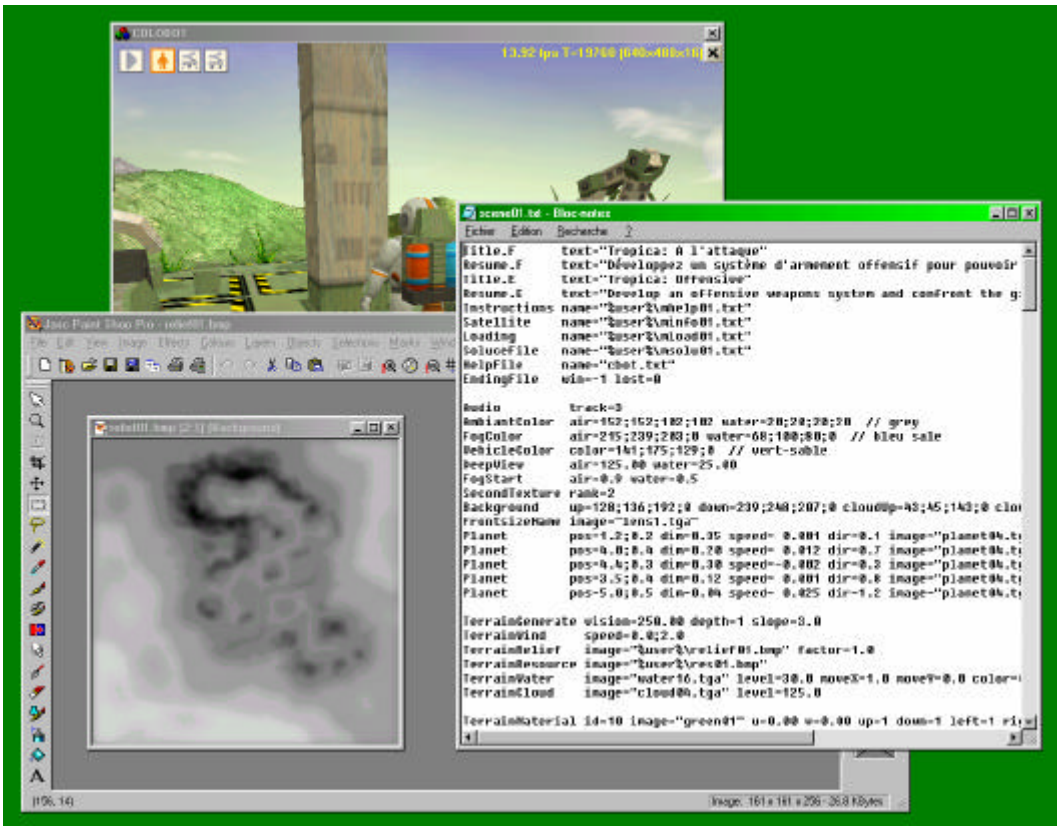
- `scene00.txt`
- `scene01.txt`
- `scene02.txt`
- `scene05.txt`

Le niveau ne contiendra dans ce cas que 2 missions, les numéros 01 et 02. Le numéro 05 n'apparaît pas dans la liste, car il manque les numéros 03 et 04.

2. Environnement de travail

Lorsque vous créez des niveaux supplémentaires, il faut souvent activer d'autres programmes que CoLoBoT, avec Alt+Tab. Ceci est beaucoup plus agréable lorsque CoLoBoT fonctionne en mode fenêtré. Pour cela, cliquez sur « Options », puis sur l'onglet « Device » :

Enlevez la coche « Full screen » puis cliquez « Apply changes ».
Notez qu'en mode fenêtré, CoLoBoT fonctionne en 640x480, et qu'il n'est pas possible de changer ces dimensions.



3. Créez des niveaux supplémentaires

Le fichier `sample01.zip` contient un exemple de deux missions complètes. Ce fichier doit être dézippé dans le dossier `user\` placé dans le dossier principal où est installé CoLoBoT. Généralement, il s'agit de :

- `C:\Program Files\Colobot\`**`user`**`\sample01\`

Si le dossier `user\` n'existe pas, il suffit de le créer.

Les copies d'écran ci-dessus montrent les deux missions contenues dans `sample01.zip`.

Ce manuel part de cet exemple concret et explique le contenu des différents fichiers. Les 24 fichiers contenus dans `sample01\` sont les suivants :

Fichier	Contenu	
`mhelp01.txt`	Instructions de la mission #1	Textes affichés dans le ***Sat*Com**
`mhelp02.txt`	Instructions de la mission #2	
`minfo01.txt`	Rapport du satellite de la mission #1	
`minfo02.txt`	Rapport du satellite de la mission #2	
`mload01.txt`	Explications des programmes envoyés par Houston pour la mission #1	
`mload02.txt`	Explications des programmes envoyés par Houston pour la mission #2	
`msolu01.txt`	Solution de la mission #1	
`msolu02.txt`	Solution de la mission #2	
`sant01.txt`	Script chargé initialement dans certaines fourmis	Scripts
`skill.txt`	Script chargé initialement dans certains robots Shooter et OrgaShooter	
`scene00.txt`	Titre de l'ensemble du niveau	Scènes
`scene01.txt`	Description de la mission #1	
`scene02.txt`	Description de la mission #2	
`back01.bmp`	Etoiles pour l'arrière-plan de la mission #1	Images
`cloud02.bmp`	Nuages pour le ciel de la mission #2	
`lens02.bmp`	Rayons de soleil de la mission #2	
`nevada1.bmp`	Sol de la mission #2 (sable-roche1)	
`nevada2.bmp`	Sol de la mission #2 (roche1-roche2)	
`relief01.bmp`	Relief du terrain de la mission #1	
`relief02.bmp`	Relief du terrain de la mission #2	
`res01.bmp`	Ressources du sous-sol de la mission #1	
`res02.bmp`	Ressources du sous-sol de la mission #2	
`terra01.bmp`	Sol de la mission #1	
`water01.bmp`	Eau de la mission #1	

Tous les fichiers de texte (`.txt`) peuvent être modifiés ou créés avec un éditeur de texte tel que le bloc-notes de Windows. Les fichiers images (`.bmp`) peuvent être modifiés avec un éditeur graphique tel que PaintShop, PhotoShop ou PhotoPaint.

Le fichier `user\sample01\scene01.txt` décrit la première mission :

Commandes	Fichiers utilisés
`Title.E text="Alert"`	
`Resume.E text="Your base is under ..."`	
`Instructions name="%user%\mhelp01.txt"`	Instructions pour le ***Sat*Com**
`Satellite name="%user%\minfo01.txt"`	Rapport du satellite pour le ***Sat*Com**
`Loading name="%user%\mload01.txt"`	Programmes envoyés par Houston pour le ***Sat*Com**
`SoluceFile name="%user%\msolu01.txt"`	Solution pour le ***Sat*Com**
`HelpFile name="cbot.txt"`	Aide officielle à la programmation pour le ***Sat*Com**
`...`	
`Background image="%user%\back01.bmp"`	Etoiles affichées en arrière-plan
`FrontsizeName image="lens1.tga"`	Rayons de soleil affichés en avant-plan
`...`	
`TerrainRelief image="%user%\relief01.bmp"`	Relief du terrain
`TerrainResource image="%user%\res01.bmp"`	Ressources du sous-sol
`TerrainWater image="%user%\water01.bmp" ...`	Texture de l'eau
`TerrainInitTextures image="%user%\terra.bmp" dx=1 dy=1 table=1`	Texture du sol
`...`	
`CreateObject pos=12.50;-112.50 dir=1.0 type=WingedOrgaShooter power=0.2 script1="%user%\skill.txt"`	Script chargé initialement dans le robot
`...`	

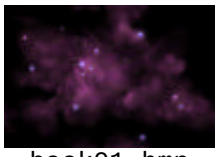
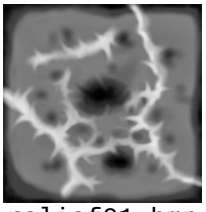
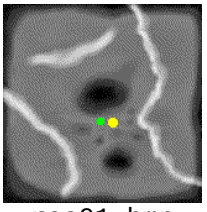
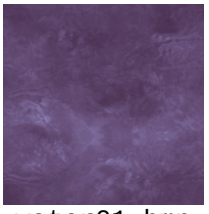
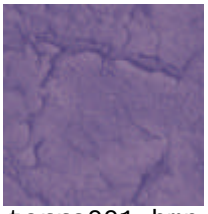
Certaines commandes contenues dans le fichier `user\sample01\scene01.txt` font référence à d'autres fichiers. Par exemple, la commande :

- `Instructions name="%user%\mhelp01.txt"`

indique quel est le fichier qui contient les instructions affichées par le ***Sat*Com**, pour expliquer ce qu'il faut faire dans cette mission.

Lorsqu'un nom de fichier est précédé de `%user%\`, cela signifie qu'il est dans le dossier spécifique au niveau supplémentaire (donc `user\sample01\` dans les exemples donnés ici).

Les images contenues dans le dossier `user\sample01\` utilisées pour la première mission sont :

 `back01.bmp`	 `relief01.bmp`	 `res01.bmp`	 `water01.bmp`	 `terra001.bmp`
512x347 pixels	161x161 pixels	161x161 pixels	256x256 pixels	256x256 pixels
16 millions	256 niveaux gris	256 couleurs	16 millions	16 millions

3.1. Les fichiers généraux

Tous les fichiers spécifiques au niveau doivent être dans le même dossier `user\sample01\`, mais il est possible de faire référence à certains fichiers généraux de CoLoBoT, placés dans d'autres dossiers. Par exemple, le fichier qui explique comment programmer est toujours le même. Il est inutile de le dupliquer dans chaque niveau supplémentaire. La commande :

- `HelpFile name="cbot.txt"`

indique qu'il faut le chercher dans le dossier général, à savoir :

- `C:\Program Files\Colobot\help\cbot.txt`

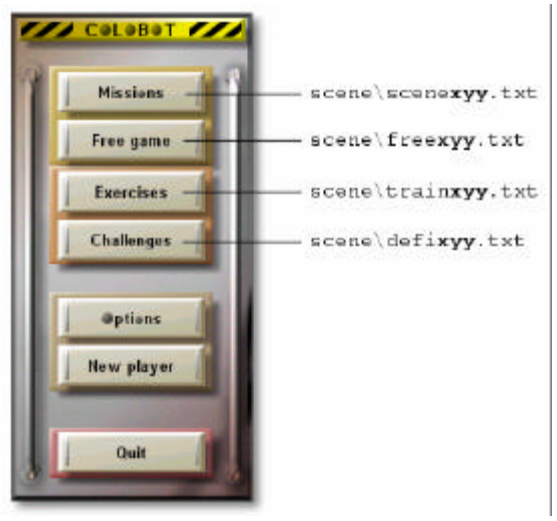
Normalement, un nom de fichier doit être précédé de `%user%\` pour indiquer qu'il est dans le dossier spécifique au niveau supplémentaire. Sans cette indication, il est dans le dossier général de CoLoBoT. Les différents dossiers généraux sont :

Dossier	Contenu
`scene\`	Description des exercices et des missions.
`help\`	Instructions affichées dans le ***Sat*Com**.
`diagram\`	Images affichées dans le ***Sat*Com**.
`textures\`	Images pour le relief du terrain et les ressources en sous-sol.
`script\`	Programmes CBOT chargés initialement dans les robots.

Vous pouvez faire référence à des fichiers dans ces dossiers, ou les copier dans votre dossier `user\sample01\` afin de servir de modèle à vos créations. **Mais ne modifiez jamais un fichier dans un dossier général**. En effet, votre niveau supplémentaire ne fonctionnerait pas une fois installé chez quelqu'un d'autre !

Tous les fichiers généraux des exercices et des missions sont dans le sous-dossier `scene\` :

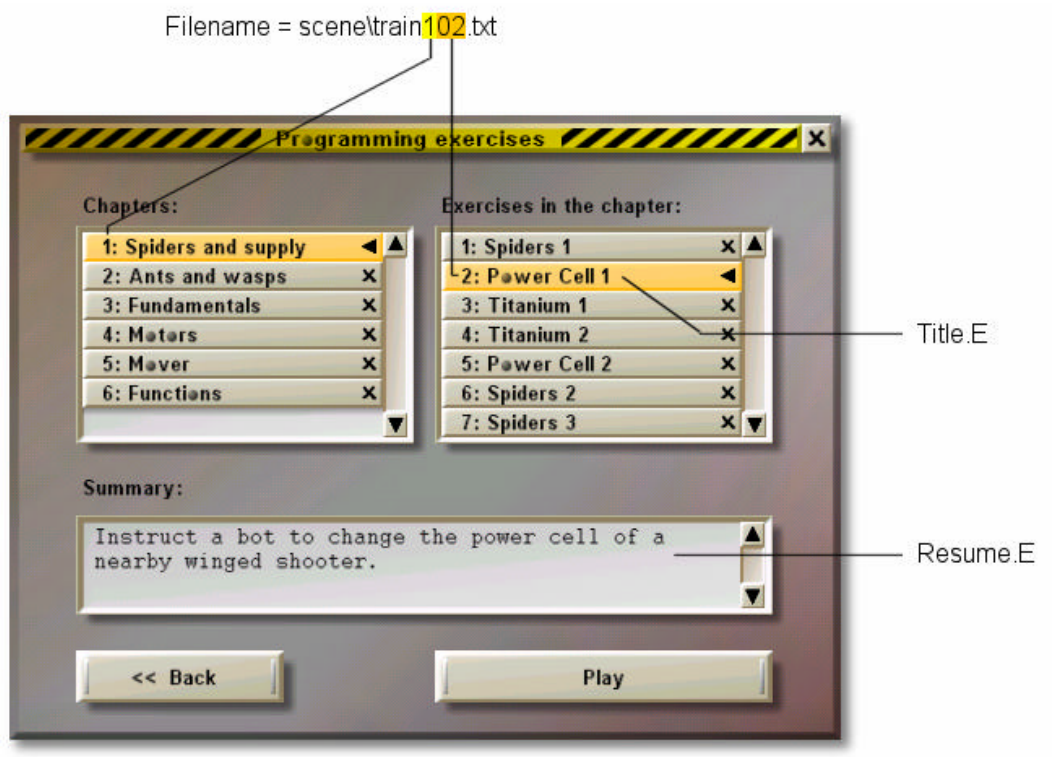
`scene\scene`**`xyy`**`.txt`	missions
`scene\free`**`xyy`**`.txt`	jeux libres
`scene\train`**`xyy`**`.txt`	exercices
`scene\defi`**`xyy`**`.txt`	défis



Le numéro de 3 chiffres est composé de :

x	numéro du chapitre	1..9
yy	rang dans le chapitre	01..99

Par exemple, `train`**`102`**`.txt` est le deuxième exercice du premier chapitre.



Remarque : N'hésitez pas à consulter ces fichiers pour comprendre comment les missions « officielles » de CoLoBoT sont réalisées.

3.2. Description d'une mission ou d'un exercice

Un fichier `user\sample01\sceneNN.txt` de description de mission détermine le relief du terrain, les textures utilisées, la position initiale des différents robots, matières premières, plantes, etc. Chaque ligne contient une commande, qui commence par un mot-clé, suivi des paramètres. Les lignes vides sont ignorées. Une double barre oblique `//` débute un commentaire qui se termine à la fin de la ligne.

3.2.1. En-tête

L'en-tête détermine le nom de l'exercice ou de la mission et les fichiers d'explication associés.

```
Title.E text="Alert"
```
Nom court de la mission, tel qu'il apparaît dans la liste de droite.

```
Resume.E text="Your base is under alert ..."
```
Résumé de la mission, tel qu'il apparaît en dessous des deux listes.

Remarque : Les commandes `Title.F` et `Resume.F` permettent de donner les textes qui seront affichés avec la version française de CoLoBoT.

`ScriptName.E text="Spider2"`
Nom par défaut donné à un nouveau programme créé.

`Instructions name="%user%\mhelp01.txt"`
Nom du fichier qui contient les instructions de la mission, qui seront affichées dans le ***Sat*Com**. Tous les fichiers d'instructions commencent par les lettres « `mhelp` ».

`Satellite name="%user%\minfo01.txt"`
Nom du fichier qui contient le rapport du satellite d'observation, qui sera affiché dans le ***Sat*Com**. Tous les fichiers de rapports commencent par les lettres « `minfo` ».

`Loading name="%user%\mload01.txt"`
Nom du fichier qui contient les programmes envoyés par Houston, qui seront affichés dans le ***Sat*Com**. Tous les fichiers des programmes envoyés par Houston commencent par les lettres « `mload` ».

`SoluceFile name="%user%\msolu01.txt"`
Nom du fichier qui contient la solution de la mission, qui sera affichée dans le ***Sat*Com**. Tous les fichiers de solution commencent par les lettres « `msolu` ».

`HelpFile name="cbot.txt"`
Nom du fichier qui contient les instructions sur la programmation, affichées lorsque l'on presse sur F2. Le fichier `cbot.txt` est placé dans le dossier `help\`. Normalement, toutes les missions font référence aux mêmes instructions générales contenues dans `cbot.txt`.

`EndingFile win=-1 lost=0`
Scène à utiliser lorsque la mission est réussie ou ratée.

- `win=-1` signifie qu'il n'y a pas de scène spécifique de fin. La mission se termine simplement par le décollage du vaisseau spatial.
- `win=2` signifie qu'il faut utiliser `scene\win002.txt`.
- `lost=0` signifie qu'il faut utiliser `scene\lost000.txt`.

Il n'est pas possible de placer ces descriptions de scènes dans le dossier du niveau supplémentaire. Il faut donc se contenter de faire référence aux fichiers généraux du dossier `scene\`, sans les modifier.

`MessageDelay factor=5`
Augmente le temps pendant lequel les messages sont affichés en haut de l'écran.

3.2.2. Ambiance

Cette partie décrit les couleurs générales, le brouillard, le ciel, etc.

`Audio track=0`
Numéro de la piste audio du CD à jouer pendant la mission. Normalement, les exercices restent silencieux, en donnant le numéro de piste 0.Il est possible de donner les numéros suivants :

Numéro de piste	Planète
0	Lune (silence)
2	Terre
3	Tropica
4	Crystalium
5	Saari
6	Volcano
7	Centaury
8	Orphéon
9	Terranova

`AmbiantColor air=102;102;102;102 water=20;20;20;20`
Couleur ambiante utilisée lorsqu'on est à l'air libre ou sous l'eau.
Les couleurs sont spécifiées à l'aide de 4 composantes rouge/vert/bleu/alpha. Les valeurs sont comprises entre 0 (noir) et 255 (blanc). La composante alpha est généralement ignorée. Par exemple :

- `color=175;209;215;0 // bleu-sable`

Planète	air	water
Terre	120;90;0;0	20;20;20;20
Lune	102;102;102;102	20;20;20;20
Tropica	152;152;102;102	20;20;20;20
Crytalium	102;102;102;102	20;20;20;20
Saari	136;136;102;102	20;20;20;20
Volcano	102;102;102;102	20;20;20;20
Centaury	102;102;102;102	20;20;20;20
Orphéon	102;68;68;102	20;20;20;20
Terranova	102;102;102;102	20;20;20;20

`FogColor air=180;222;255;0 water=10;20;100;0`
Couleur que prennent les objets lorsqu'ils sont au loin. Voir aussi `DeepView` et `FogStart`.

Planète	air	water
Terre	100;100;90;0	67;80;100;0
Lune	0;0;0;0	67;80;100;0
Tropica	215;239;203;0	68;100;80;0
Crytalium	180;222;255;0	10;20;100;0
Saari	254;245;146;0	67;80;100;0
Volcano	205;86;21;0	200;100;0;0
Centaury	176;176;181;0	10;100;20;0
Orphéon	50;30;10;0	67;80;100;0
Terranova	208;200;223;0	94;153;180;0

`VehicleColor color=175;209;215;0`
Couleur des robots et des bâtiments. Sur chaque planète, les couleurs changent. Voici 4 exemples, parmi les 9 planètes existantes :

			
208;206;196;0	141;175;129;0	234;129;26;0	158;120;82;0
Lune	Tropica	Volcano	Orphéon

`GreeneryColor color= 250;187;69;0`
Couleur des végétaux (`Greenery`).

`InsectColor color=189;171;51;0`
Couleur des insectes.
Résumé des couleurs utilisées sur toutes les planètes :

Planète	VehicleColor	GreeneryColor	InsectColor
Terre	168;158;118;0	161;151;41;0	
Lune	208;206;196;0		
Tropica	141;175;129;0		
Crytalium	175;209;215;0		
Saari	158;143;68;0		
Volcano	234;129;26;0	250;187;69;0	
Centaury	117;158;64;0		
Orphéon	158;120;82;0		189;171;51;0
Terranova	200;196;174;0		

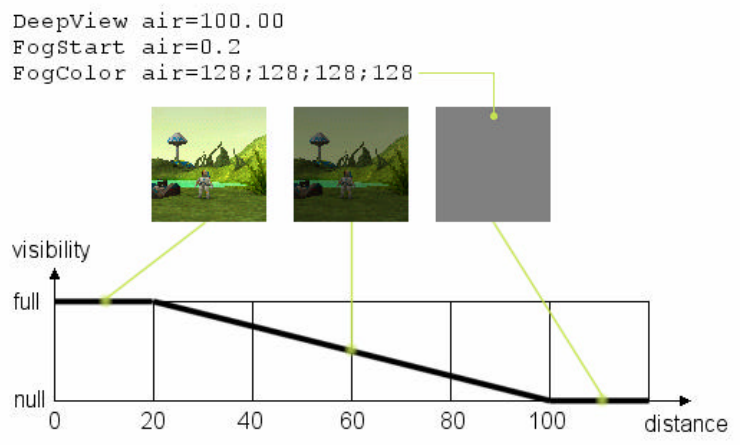
`DeepView air=100.00 water=25.00`
Distance en mètres jusqu'où porte la vue. Au delà de cette distance, plus rien n'est affiché.

`FogStart air=0.1 water=0.1`
Plus on s'approche de la distance maximale de vue (`DeepView`) et plus la couleur des objets fusionne avec la couleur du brouillard (`FogColor`), ce qui simule du brouillard. Une valeur de 0.1 indique un brouillard qui commence proche du point de vue, donc un brouillard dense. Une valeur de 0.9 indique un brouillard très peu dense qui débute au loin.

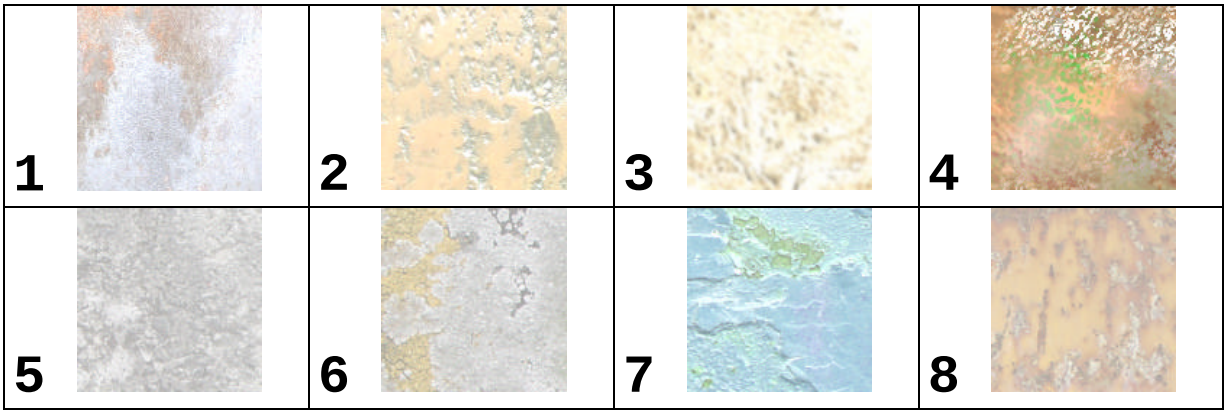
Par exemple :

- `DeepView air=100.00  // total clipping at 100 meters`
- `FogStart air=0.2`
- `FogColor air=128;128;128;128  // grey`

Distances à partir de l'observateur	
0 à 20 mètres	affichage normal
20 à 100 mètres	affichage de plus en plus brouillardeux
100 mètres et plus	plus rien n'est affiché

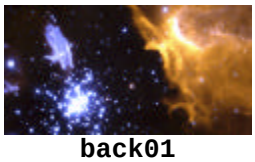
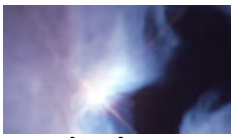


`SecondTexture rank=3`
Texture utilisée pour salir les robots et les bâtiments. Vous pouvez utiliser une valeur comprise entre 1 et 8 :



```
Background image="back01.tga" up=76;105;226;0 down=192;250;255;0
```

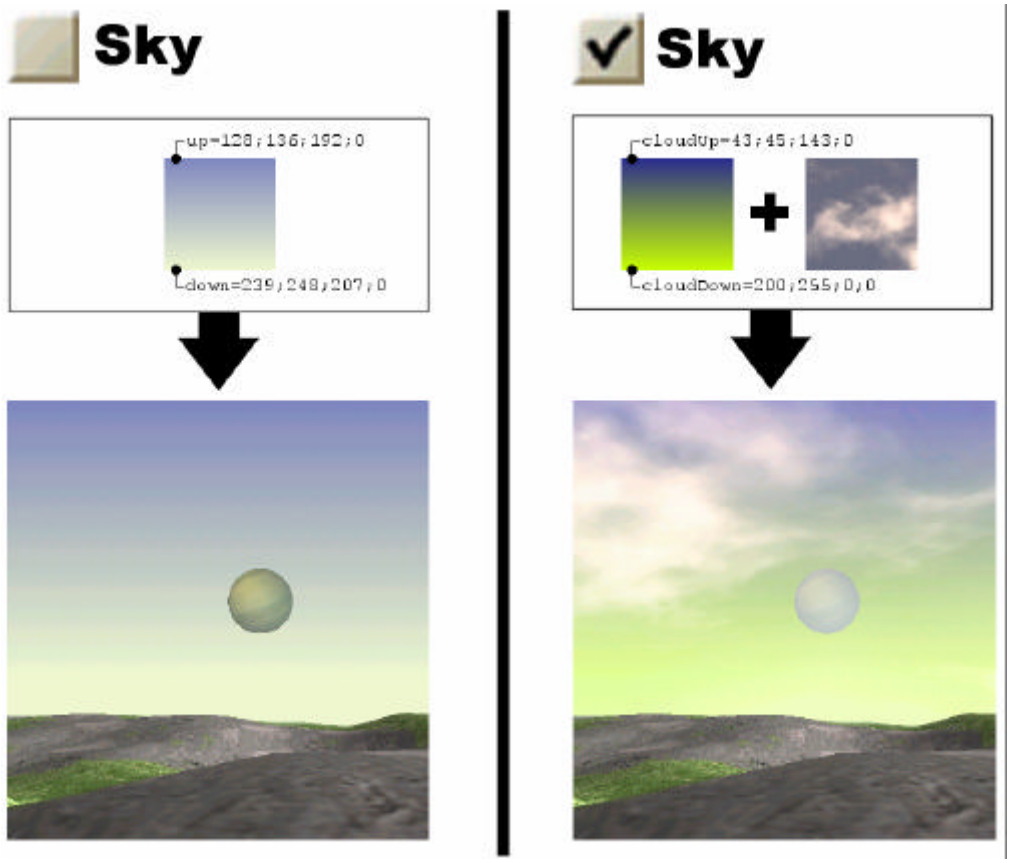
Fond de l'écran au lointain. Il peut s'agir d'une image ou d'un dégradé de couleurs. Les images suivantes sont disponibles :

 **back01**	 **back22**	**back33**	 **back46**

Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Il ne faut pas dépasser les dimensions 1280x480, si possible.

Un dégradé de couleur passe de "up" tout en haut de l'écran progressivement jusqu'à "down" en milieu d'écran. La moitié inférieure de l'écran prend la couleur unie "down". Cette moitié inférieure n'est en principe jamais visible, puisqu'il y a toujours une partie de terrain qui la recouvre.
S'il n'y a qu'un dégradé, il suffit d'omettre la partie `image`. S'il y a une image, elle couvre normalement complètement le fond. Il faut toutefois quand même spécifier le dégradé, car l'image sera peut-être ignorée, en fonction des réglages (sur des machines peu performantes).

```
Background up=128;136;192;0 down=239;248;207;0 cloudUp=43;45;143;0
cloudDown=200;255;0;0
```



Les paramètres `cloudUp` et `cloudDown` sont utilisés pour spécifier les couleurs de l'arrière-plan derrière les nuages (commande `TerrainCloud`). Si les nuages sont absents (sur des machines peu performantes), c'est les couleurs `up` et `down` qui sont utilisées. Ces deux couleurs doivent alors imiter le mieux possible les couleurs des « vrais » nuages.

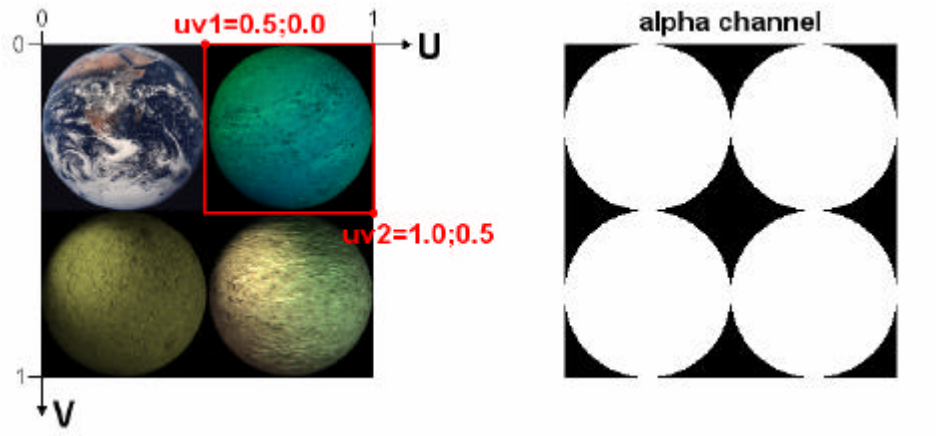
```
FrontsizeName image="lens5.tga"
```
Nom de la texture d'avant-plan, qui contient un effet de "lens flare", plus ou moins visible selon l'orientation.

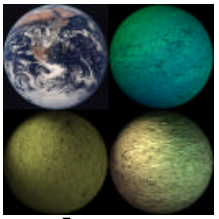
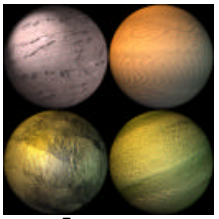
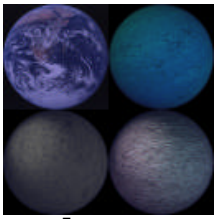
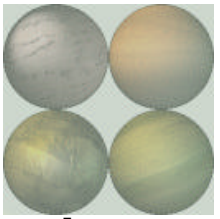
 **lens1**	 **lens4**	 **lens5**

Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Il ne faut pas dépasser les dimensions 512x256, mais les dimensions 128x64 sont généralement suffisantes.

```
Planet mode=0 pos=1.2;0.2 dim=0.35 speed=0.001 dir=0.1 image="planet04.tga"
uv1=0.0;0.5 uv2=0.5;1.0
```
Crée une planète fixe ou mobile dans le ciel.

- `mode=0` indique une planète visible pendant la phase normale de jeu.
- `mode=1` indique une planète visible pendant le film du voyage intersidéral, lorsque le vaisseau spatial est créé avec `CreateObject ... run=11`.
- `pos` donne la position initiale dans le ciel. La première valeur est la direction et la deuxième la hauteur.
- `dim` est la dimension de la planète dans le ciel. De 0.02 pour une étoile minuscule (sur Crystalium) à 0.5 pour la terre visible depuis la lune.
- `speed` est la vitesse de la planète, généralement très lente (0 si immobile).
- `dir` donne la fluctuation verticale du mouvement.
- `image` donne le nom de l'image à utiliser (voir plus bas).
- `uv1` et `uv2` donnent les coordonnées dans l'image de la partie à utiliser. 0;0 correspond au coin supérieur gauche.



 **planet01**	 **planet02**	 **planet03**	 **planet04**

Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format doit être `.tga` avec un canal alpha pour définir la zone circulaire de la planète. Il ne faut pas dépasser les dimensions 256x256, si possible.

3.2.3. Terrain

Cette partie décrit le relief du terrain et son texturage, le niveau et l'apparence de l'eau ainsi que les ressources en sous-sol.

```
TerrainGenerate vision=250 depth=1 hard=0.6
```

- `vision` détermine jusqu'à quelle distance du point de vue le terrain est généré. Cette distance doit être au moins le double de la distance `DeepView` la plus grande (`air` ou `water`).
- `depth` doit toujours valoir 1.
- `hard` détermine la dureté du sol qui influence le bruit des pas lorsque le cosmonaute marche.

```
TerrainWind speed=0;-5
```

Vecteur indiquant la direction et la force du vent. Sur la lune, il n'y a pas de vent, donc `speed=0;0`. `speed=10;0` spécifie un vent fort qui souffle d'ouest en est. Le vent influence le mouvement des nuages (`TerrainCloud`), des particules de fumées et des drapeaux. Le vent est sans effet sur le mouvement des robots.

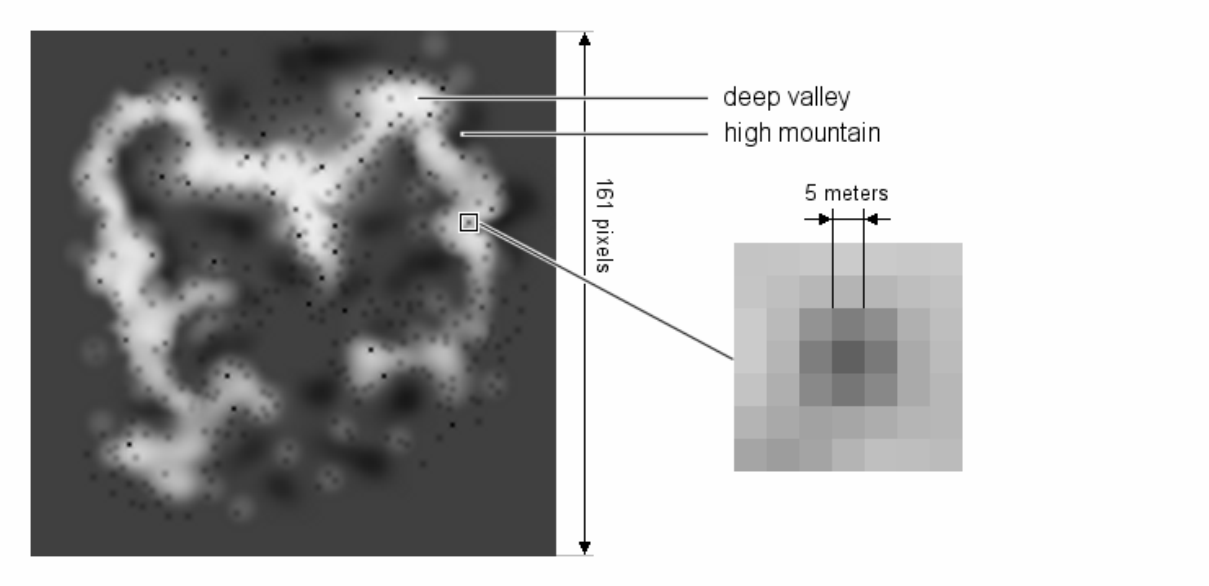
```
TerrainBlitz sleep=60 delay=5 magnetic=100
```

Active le générateur aléatoire d'éclairs. Dans les missions officielles, la seule planète sur laquelle sévissent des orages magnétiques est **Orphéon**.

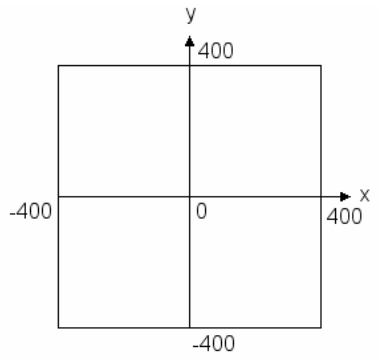
- `sleep` donne la durée en secondes avant le premier éclair. Ceci permet, par exemple, de laisser du temps pour construire un premier paratonnerre (`PowerCaptor`).
- `delay` donne le délai moyen (en secondes) entre deux éclairs.
- Chaque objet métallique est entouré d'une zone circulaire qui attire la foudre. `magnetic` spécifie le rayon (en mètres) de cette zone. Plus ce rayon est grand et plus un éclair risque de provoquer des dégâts. Inversement, avec une petite valeur, la probabilité qu'un éclair tombe sur une zone vierge est plus grande.

```
TerrainRelief image="%user%\relief01.bmp" factor=1.0
```

Le relief du terrain est décrit dans une image `.bmp` à 256 niveaux de gris mesurant exactement 161 x 161 pixels. La couleur blanche correspond à l'altitude la plus basse. La couleur noire correspond à l'altitude la plus haute. Normalement, on utilise "`factor=1.0`". Dans ce cas, les 256 niveaux permettent de s'élever de 64 mètres. Une différence d'intensité de gris de 1 correspond donc à une différence d'altitude de 0.25 mètres.



La coordonnée du pixel central 80;80 de l'image correspond à la coordonnée 0;0 mètres dans CoLoBoT. La coordonnée 0;0 du pixel en haut à gauche dans l'image correspond au point à l'extrême nord-ouest -400;400 mètres dans CoLoBoT. Un pixel dans l'image correspond à un carré au sol de 5x5 mètres dans CoLoBoT.

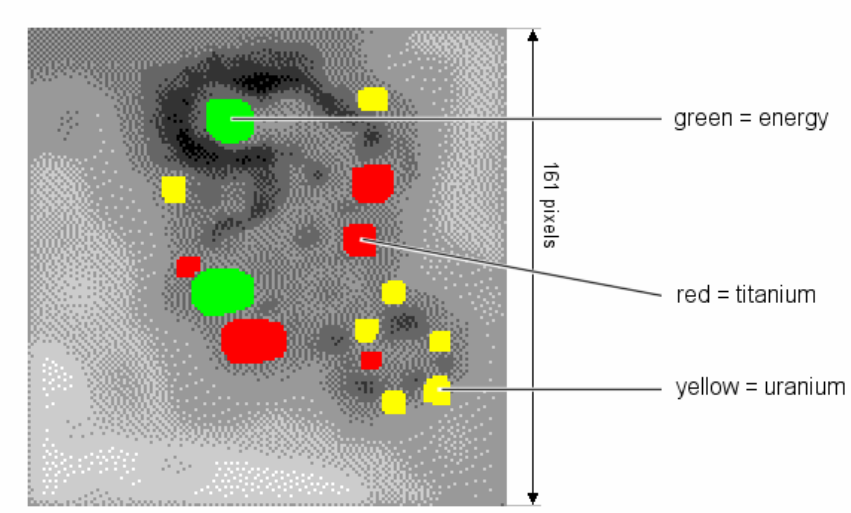


L'univers de CoLoBoT mesure donc 800x800 mètres, soit près d'un kilomètre carré !
Vous pouvez dessiner de nouveaux reliefs avec un logiciel tel que PaintShop, ou réutiliser les nombreux fichiers `relief`**`xx`**`.bmp` placés dans le dossier `textures\`.

```
TerrainResource image="%user%\res01.bmp"
```
Cette image détermine la présence des ressources dans le sous-sol. Il s'agit d'une image `.bmp` en 256 couleurs de 161 x 161 pixels, utilisant la palette de couleurs standard de Windows.

Couleur	RVB	Index	Contenu du sous-sol
Rouge	255;0;0	5	titanium
Vert	0;255;0	30	énergie
Jaune	255;255;0	35	uranium
Vert	0;104;0	24	clé A
Vert	51;104;0	25	clé B
Vert	102;104;0	26	clé C
Vert	153;104;0	27	clé D

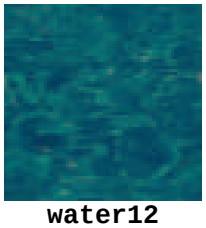
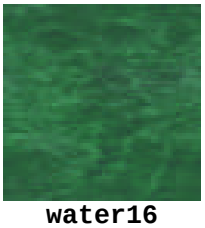
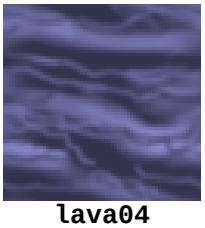
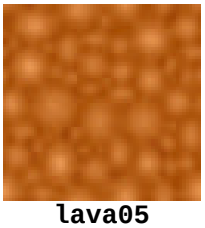
Toutes les autres couleurs ou niveaux de gris sont ignorés.



Généralement, un truc pour placer les zones de couleur au bon endroit est de partir de l'image à niveaux de gris du relief `TerrainRelief` et de la convertir en 256 couleurs.

```
TerrainWater image="water16.tga" level=30.0 moveX=1.0 moveY=0.0
color=0;240;100;0 brightness=0.2
```
Cette commande détermine l'apparence de l'eau, qui est forcément identique dans toute la mission. Il n'est pas possible, par exemple, d'avoir un lac d'eau claire à une certaine altitude, et un autre lac de lave à une altitude différente.

- `image` spécifie l'apparence de l'eau ou de la lave.
- `level` indique le niveau de l'eau ou de la lave, par-rapport au niveau zéro qui correspond à la couleur blanche dans l'image du relief (`TerrainResource`). L'altitude des robots est par la suite toujours calculée par-rapport au niveau de la mer (`level`). Une altitude négative indiquera donc que le robot est sous l'eau.
- `moveX` détermine l'amplitude du mouvement horizontal à la surface.
- `moveY` détermine l'amplitude du mouvement vertical. La surface se déforme réellement selon une sinusoïde bidimensionnelle, et monte et descend lentement pour simuler la houle.
- `color` et `brightness` modifient la couleur de la texture.

 **water12**	 **water16**	 **lava04**	 **lava05**

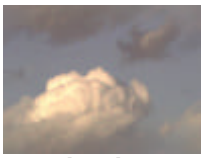
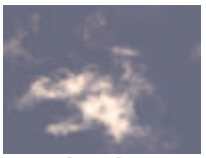
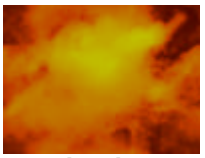
Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Il ne faut pas dépasser les dimensions 256x256, si possible.

```
TerrainLava mode=1
```

- `mode=1` signifie une mort instantanée lorsque le cosmonaute est sous le niveau de l'eau. Dans ce mode, la caméra ne va jamais en-dessous du niveau de l'eau.

```
TerrainCloud image="cloud05.tga" level=125.0
```

Image utilisée pour les nuages mobiles dans le ciel. Le mouvement des nuages dépend de la force et de la direction du vent (voir `TerrainWind`). Les différentes images possibles sont :

 **cloud02**	 **cloud03**	 **cloud04**	 **cloud05**	 **cloud09**

Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Il ne faut pas dépasser les dimensions 640x480, si possible.

- `level` indique l'altitude maximale de la couche nuageuse, en mètres, qui doit dépasser de quelques dizaines de mètres (au minimum) la montagne la plus haute.

Utilisez la commande `Background` avec `cloudUp` et `cloudDown` pour spécifier les couleurs derrière les nuages.

3.2.4. Texturage du terrain

Il existe deux systèmes différents pour texturer le sol :

	Commandes	**Signification**
1	`TerrainInitTextures`	La texture est appliquée « aveuglément » sur l'ensemble du terrain.

2	`TerrainMaterial`
	`TerrainInit`
	`TerrainLevel`
		Différentes matières sont utilisées en fonction de l'altitude. Ceci permet d'avoir du sable sous l'eau, de l'herbe en moyenne altitude et de la neige sur les sommets, par exemple.

Par exemple, la **lune** utilise le premier système, alors que **Terranova** utilise le second.
Les commandes de texturage sont complexes. Il est conseillé de copier les commandes d'une planète à choix et de les utiliser telles quelles. Nous donnons ici juste quelques pistes, incomplètes !

```
TerrainInitTextures image="mars" dx=4 dy=2 table=1;2;3;4;5;6;7;8
```

Applique une texture uniforme sur l'ensemble du terrain. La texture est constituée dans cet exemple des 8 images suivantes, juxtaposées dans cet ordre :

`mars001.tga`	`mars002.tga`	`mars003.tga`	`mars004.tga`
`mars005.tga`	`mars006.tga`	`mars007.tga`	`mars008.tga`

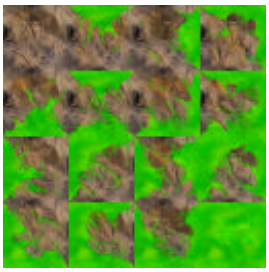
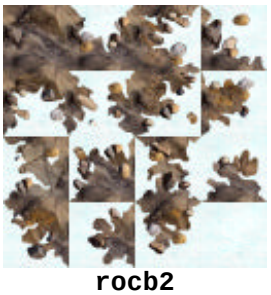
Le damier de 8 textures est répété dans toutes les direction.
Donc, le bord droite de `mars002.tga` touche le bord gauche de `mars003.tga`.
Le bord droite de `mars004.tga` touche le bord gauche de `mars001.tga`.
Le bord supérieur de `mars002.tga` touche le bord inférieur de `mars006.tga`.

Le nom de fichier est composé du nom de base suivi d'un numéro à 3 chiffres pris dans la table.Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Par exemple :

- `TerrainInitTextures image="%user%\moon.bmp" dx=1 dy=1 table=15`

Ici, c'est l'image `user\sample01\moon015.bmp` qui est utilisée.

`TerrainMaterial id=1 image="roca2" u=0.00 v=0.00 up=1 down=1 left=1 right=1 hard=0.8`
Définition d'une matière ayant un numéro d'identification. Par exemple, sur Terranova, les deux textures suivantes sont utilisées :

 **roca2**	 **rocb2**

Chaque texture est forcément découpée en 16 morceaux (4x4).
Les quatre matières principales suivantes sont définies, avec les identificateurs 1 à 4 :

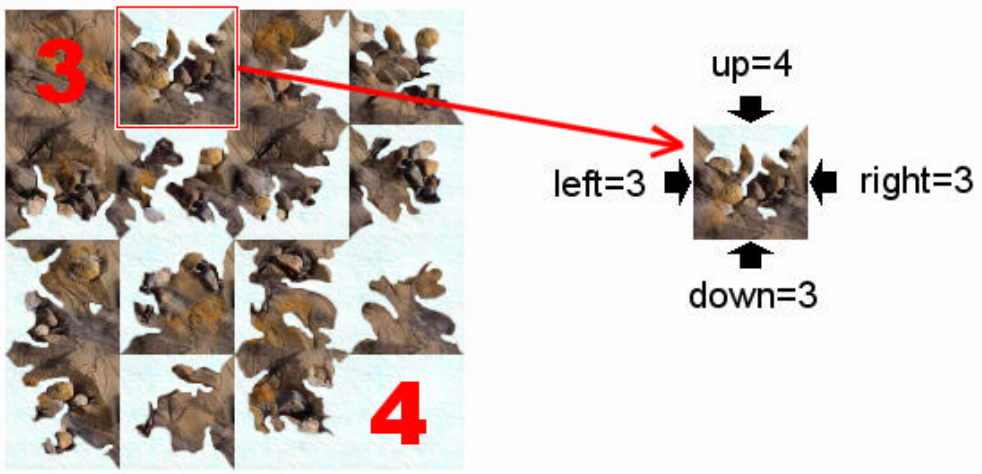
- `id=1 image="roca2" u=0.00 v=0.00 up=1 down=1 left=1 right=1 // roche`
- `image="roca2" u=0.25 v=0.00 up=2 down=1 left=1 right=1`
- `image="roca2" u=0.50 v=0.00 up=1 down=1 left=1 right=2`
- `image="roca2" u=0.75 v=0.00 up=2 down=1 left=1 right=2`
- `image="roca2" u=0.00 v=0.25 up=1 down=2 left=1 right=1`
- `image="roca2" u=0.25 v=0.25 up=2 down=2 left=1 right=1`
- `image="roca2" u=0.50 v=0.25 up=1 down=2 left=1 right=2`
- `image="roca2" u=0.75 v=0.25 up=2 down=2 left=1 right=2`
- `image="roca2" u=0.00 v=0.50 up=1 down=1 left=2 right=1`
- `image="roca2" u=0.25 v=0.50 up=2 down=1 left=2 right=1`
- `image="roca2" u=0.50 v=0.50 up=1 down=1 left=2 right=2`
- `image="roca2" u=0.75 v=0.50 up=2 down=1 left=2 right=2`
- `image="roca2" u=0.00 v=0.75 up=1 down=2 left=2 right=1`
- `image="roca2" u=0.25 v=0.75 up=2 down=2 left=2 right=1`
- `image="roca2" u=0.50 v=0.75 up=1 down=2 left=2 right=2`
- `id=2 image="roca2" u=0.75 v=0.75 up=2 down=2 left=2 right=2 // herbe`
- `id=3 image="rocb2" u=0.00 v=0.00 up=1 down=1 left=1 right=1 // roche`
- `...`
- `id=4 image="rocb2" u=0.75 v=0.75 up=3 down=3 left=3 right=3 // neige`

D'autres matières (sans identificateurs) sont définies pour permettre les transitions. `up`, `down`, `left` et `right` donnent les identificateurs des matières voisines qu'il est possible de mettre à côté.

Par exemple la ligne :

- `image="rocb2" u=0.25 v=0.00 up=4 down=3 left=3 right=3`

Correspond au morceau suivant de l'image (`id=3`:roche et `id=4`:neige) :



`hard` détermine la dureté d'une matière, comprise entre 0 et 1, qui influence le bruit des pas lorsque le cosmonaute marche à cet endroit.

Remarque : Les matières sont utilisées sur la terre pour dessiner les routes sur le sol (voir par exemple `scene\scene101.txt`). Trois identificateurs différents correspondent aux routes est/ouest (`id=3`), nord/sud (`id=4`) et aux carrefours en croix (`id=5`).

Si le nom est précédé de `%user%\`, l'image est cherchée dans le dossier `user\sample01\` du niveau supplémentaire. Le format `.bmp` ou `.tga` peut être utilisé. Il ne faut pas dépasser les dimensions 512x512, si possible.

`TerrainInit id=3`
Initialise l'ensemble du terrain avec une première matière de base.

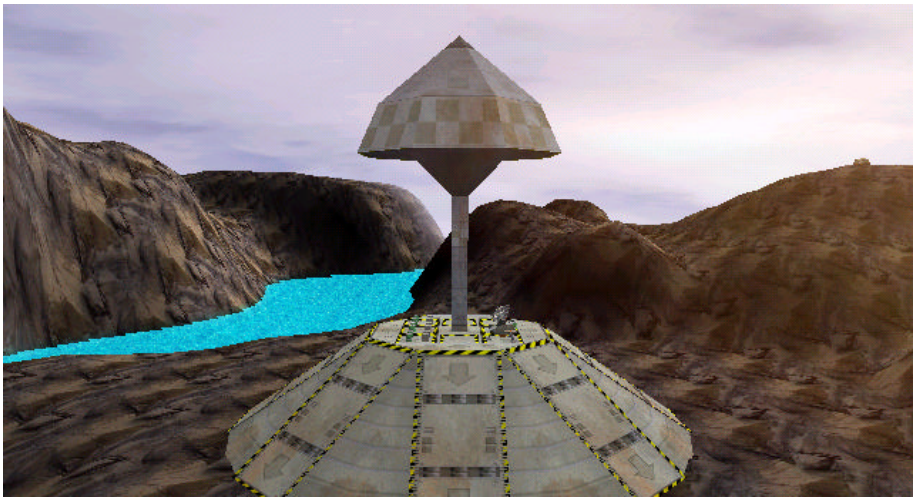
`TerrainLevel id=10;11;10;11;12;13 min=0 max=99 slope=0.0 freq=100`
Pose une matière sur l'ensemble du terrain, selon certains critères.

- `id` détermine la matière à utiliser. Si plusieurs matières sont spécifiées (`id=10;11;12;13`), elles sont choisies au hazard. `id=10;10;10;11` permet de mettre trois fois plus de matière 10 que de 11.
- `min` et `max` déterminent l'altitude minimale et maximale où doit être appliquée la matière. Il s'agit ici d'altitudes absolues, qui ne tiennent pas compte du niveau de l'eau.
- `slope` détermine la pente. Par exemple, `slope=9` ne mettra de la neige que si le sol est presque plat. Dès que la pente est trop grande, la neige n'est plus déposée. A l'inverse, une valeur négative permet de mettre la matière seulement si la pente est forte.
- `center` détermine le centre d'une zone circulaire facultative.
- `radius` détermine le rayon de la zone circulaire facultative.
- freq permet de ne pas mettre partout la matière. Par exemple, avec `freq=25`, seul 25% de la surface (répondant aux critères `min`, `max`, `slope`, etc.) sera recouverte avec la matière.

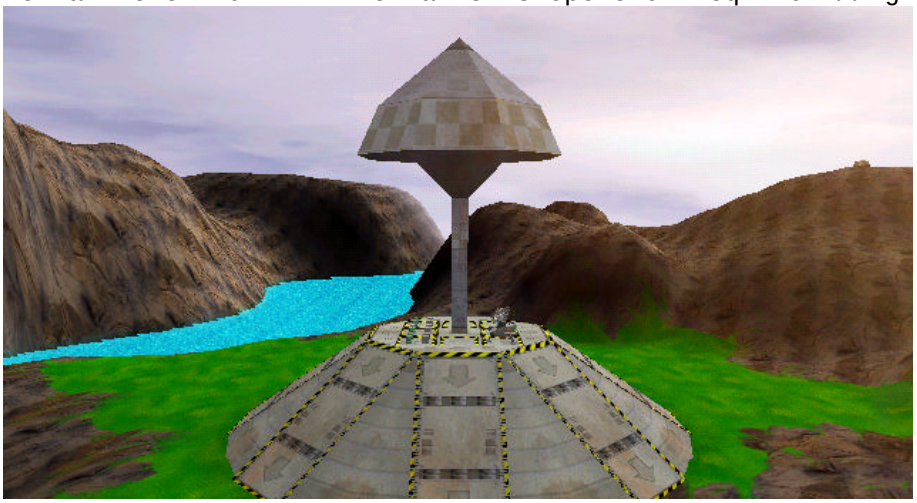
Il faut bien comprendre que les différentes commandes `TerrainLevel` sont exécutées dans l'ordre. La deuxième commande `TerrainLevel` modifie donc le terrain dans l'état laissé par `TerrainInit` suivi du premier `TerrainLevel`. Il est important de bien réfléchir à la stratégie d'application des différentes « couches ».

Voici quelques exemples de commandes sur Terranova, avec les effets obtenus :

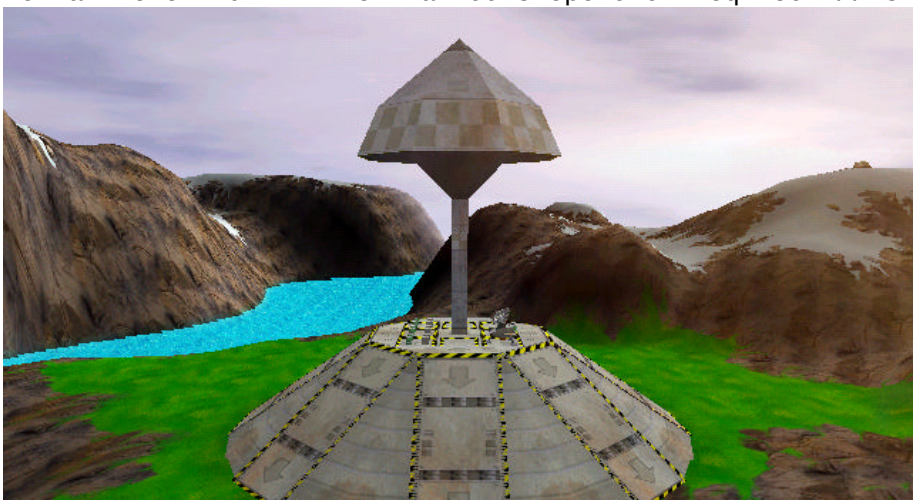
```
TerrainInit  id=1  // roc
```



```
TerrainInit  id=1  // roc
TerrainLevel id=2 min=25 max=37 slope=3.0 freq= 70  // grass
```



```
TerrainInit  id=1  // roc
TerrainLevel id=2 min=25 max=37 slope=3.0 freq= 70  // grass
TerrainLevel id=4 min=37 max=99 slope=9.0 freq= 80  // snow
```



`TerrainCreate`

Cette commande termine toutes les commandes de description du terrain. C'est à ce moment que le terrain est effectivement généré.

`GroundSpot pos=-125;100 radius=40 color=190;220;160`

Colorie spécialement certaines zones du terrain, afin de donner une ambiance particulière. Cela permet par exemple de donner une teinte verdâtre à une zone contenant beaucoup de plantes. `pos` et `radius` déterminent une zone circulaire.

	
	`radius=30` `color=150;255;150`
	
`radius=50` `color=255;0;0`	`radius=10` `color=255;0;0`

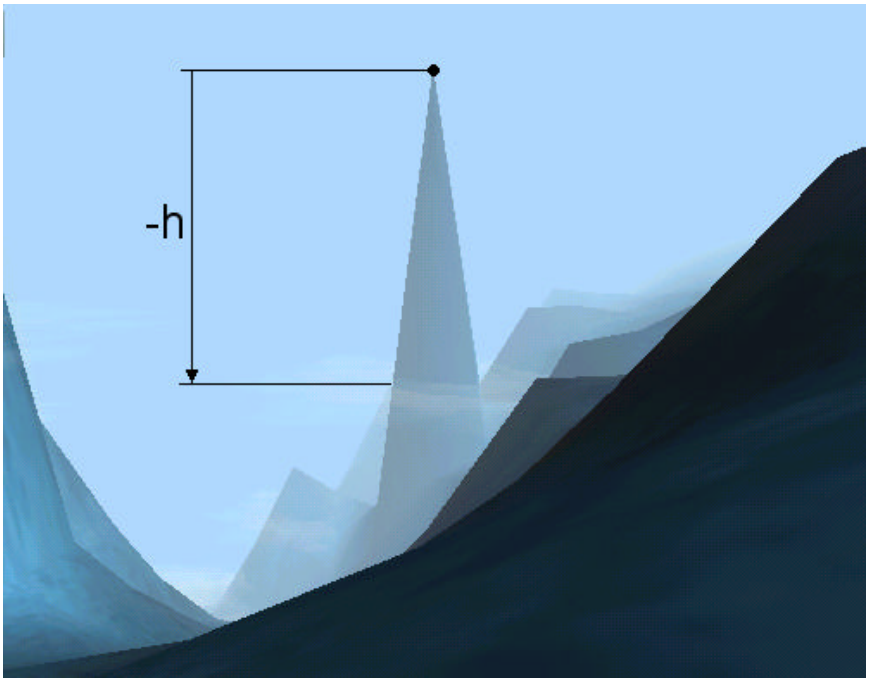
`GroundSpot color=0;128;255 min=-10 max=24 smooth=10`

Il est également possible de colorier toutes les berges, en spécifiant une altitude minimale et une altitude maximale.

		
	`color=0;255;0` `min=-10 max=28.75`	`color=0;0;255` `min=-10 max=28.75`

```
CreateFog pos=57;69 height=3 dim=20 delay=4.0 type=4
```

Crée une nappe de brouillard horizontale platte. La mission où ces nappes sont le plus visibles est Crystalium #4 (The Lost Valley). La hauteur est relative au sol à cet endroit. Si le centre de la zone est au sommet d'un pic, la hauteur peut être négative, afin que la nappe entoure le pic. Le délai détermine la lente rotation.



Type	Apparence
0 et 1	Nappe bleutée.
2 et 3	Anneau rouge-orangé.
4 et 5	Nappe grise.
6 et 7	Nappe jaunâtre.

```
MaxFlyingHeight max=40
```

Hauteur maximale à laquelle il est possible de voler, avec le cosmonaute ou avec un robot volant. Cette hauteur ne tient pas compte du niveau de l'eau.

3.2.5. Objets

C'est ici que sont créés les différents objets 3D utiles ou décoratifs posés sur le terrain au départ de l'exercice ou de la mission. Certains sont fixes (bâtiments, plantes, arbres, etc) alors que d'autres sont mobiles (cosmonaute, robots, objets transportables, etc.).

`BeginObject`
Cette commande doit précéder le premier `CreateObject`.

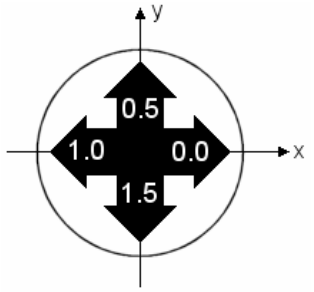
`CreateObject pos=7;-10 dir=1.5 type=Me`
Création d'un objet dans la mission. Il peut s'agir d'un robot, d'un bâtiment, d'une matière première, d'une plante, etc. Si la position est sur la plate-forme du vaisseau spatial `SpaceShip`, l'objet atterrira avec le vaisseau, au début de la mission.

Pour déterminer la position d'un objet, un bon moyen consiste à déplacer le cosmonaute à l'endroit souhaité, puis de taper les commandes :

- `Ctrl+Pause showstat Entrée`
- `Ctrl+Pause showpos Entrée`

La partie inférieure de l'écran indique alors les coordonnées de l'objet sélectionné, qu'il n'y a plus qu'à reporter dans le fichier de description de la mission :

La direction est un nombre compris entre 0 et 2 :



Les différents types possibles pour les objets sont :

Base :

```
type=Me      // cosmonaute
type=SpaceShip
```

Robots :

```
type=PracticeBot  // robot d'entraînement
type=TargetBot    // robot cible

type=WheeledGrabber
type=TrackedGrabber
type=WingedGrabber
type=LeggedGrabber

type=WheeledShooter
type=TrackedShooter
type=WingedShooter
type=LeggedShooter

type=WheeledOrgaShooter
type=TrackedOrgaShooter
type=WingedOrgaShooter
type=LeggedOrgaShooter

type=WheeledSniffer
type=TrackedSniffer
type=WingedSniffer
type=LeggedSniffer

type=Thumper
type=PhazerShooter
type=Recycler
type=Shielder
type=Subber
```

Bâtiments :

```
type=Derrick
type=BotFactory
type=PowerStation
type=Converter
type=RepairCenter
type=DefenseTower
type=AlienNest
type=ResearchCenter
type=RadarStation
type=ExchangePost
type=PowerPlant
type=AutoLab
```

```
type=NuclearPlant
type=PowerCaptor
type=Vault
type=StartArea
type=GoalArea
type=Target1    // pour l'entraînement au vol
type=Target2
type=Houston    // centre de contrôle
```

Objets transportables :

```
type=TitaniumOre
type=UraniumOre
type=Titanium
type=PowerCell
type=NuclearCell
type=OrgaMatter
type=BlackBox    // boîte noire
type=KeyA..D
type=TNT         // caisse d'explosif
```

Plantes et décors :

```
type=Greenery0..4    // plante standard basse
type=Greenery5..7    // petit trèfle bas
type=Greenery10..14  // plante grasse montante
type=Greenery15..19  // fougère
type=Tree0..3        // arbre haut
type=Mushroom1       // champignon inoffensif
type=Mushroom2       // champignon corrosif
type=MegaStalk0..5   // plante étrange géante

type=Quartz0..3      // quartz petit..grand
type=Barrier0        // barrière courte
type=Barrier1        // barrière longue
type=ApolloLEM       // sur la lune uniquement :
type=ApolloJeep
type=ApolloFlag
type=ApolloModule
type=ApolloAntenna
```

Epaves de robots recyclables :

```
type=WreckBotw1..2   // robot à roues
type=WreckBott1..2   // robot à petites chenilles
type=WreckBotr1..2   // robot à grosses chenilles
```

Bâtiments en ruine :

```
type=RuinBotFactory
type=RuinDoor      // porte de convertisseur
type=RuinSupport   // support de radar
type=RuinRadar     // socle de radar
type=RuinConvert
type=RuinBaseCamp  // socle du vaisseau spatial
type=RuinHeadCamp  // coiffe du vaisseau spatial
```

Ennemis :

```
type=AlienQueen
type=AlienEgg
type=AlienAnt
type=AlienSpider
type=AlienWasp
type=AlienWorm
```

Indicateurs :

```
type=PowerSpot     // indique la présence d'énergie en sous-sol
type=TitaniumSpot  // indique la présence de titanium en sous-sol
type=UraniumSpot   // indique la présence d'uranium en sous-sol
type=KeyA..DSpot   // indique la présence de clé en sous-sol
type=WayPoint      // croix pour les exercices
type=BlueFlag
type=RedFlag
type=GreenFlag
type=YellowFlag
type=VioletFlag
```

Divers :

```
type=Mine      // bombe fixe à éviter
type=Portico   // portique géant (sur la terre)
type=Bag       // sac de survie
type=Home      // petite maison sympa (sur terranova)
type=Tech      // technicien de Houston
type=Firework  // feu d'artifice
```

La commande `CreateObject` peut contenir des paramètres supplémentaires :

`CreateObject ... script1="%user%\scharge2.txt"`

Nom du programme CBOT à charger à la position 1 dans le robot ou l'insecte. Il est possible de charger jusqu'à 10 programmes en utilisant les commandes `script1` à `script10`. Pour faire référence à un programme général placé dans le dossier `script\` (par exemple pour un insecte), il faut omettre `%user%\` avant le nom.

`CreateObject ... run=1`

Numéro du programme à exécuter directement lorsque la mission démarre. Cela peut être utile, par exemple, pour un robot `TargetBot` que l'élève devra suivre, ou pour un ennemi. Si plusieurs programmes sont chargés (avec `script1`, `script2`, etc.), un seul pourra être exécuté, bien entendu.

`CreateObject ... select=1`

Un seul objet peut avoir le paramètre `select=1`. C'est cet objet qui sera sélectionné au départ de l'exercice ou de la mission, avec la caméra braquée sur lui.

`CreateObject ... power=0.5`

Spécifie le niveau initial de la pile. Les valeurs suivantes sont possibles :

Valeur	Signification
-1	Le robot n'a pas de pile.
0	Le robot a une pile normale vide.
0..1	Le robot a une pile normale plus ou moins remplie.
1..100	Le robot a une pile atomique plus ou moins remplie.

`CreateObject ... range=100`

Autonomie de vol. Plus la valeur est grande et plus le réacteur surchauffe lentement. La valeur par défaut est 30.

`CreateObject ... shield=1`

Etat initial du bouclier. 1 correspond à un bouclier totalement efficace. 0 correspond à un bouclier presque détruit; au prochain choc, le robot sera détruit.

```
CreateObject ... magnifyDamage=2
```
Permet d'accentuer ou de diminuer les dommages lors des chocs ou lorsque des ennemis tirent sur le robot.

Valeur	Effet
0.5	Résistance doublée.
1	Résistance standard.
2	Dommages doublés.

```
CreateObject ... proxyActivate=1 proxyDistance=10
```
Le robot ou le bâtiment n'est pas accessible, ni sur la mini-carte, ni à l'aide des icônes en haut de la fenêtre. Il faudra s'approcher à moins d'une certaine distance pour que l'objet soit « vu » et donc utilisable. On utilise ces commandes pour simuler des objets abandonnés par la première expédition.

3.2.5.1. La ligne de commande

La ligne de commande permet de passer des arguments à un programme. Supposons par exemple qu'une fourmi soit crée avec :

```
CreateObject pos=-55;243 cmdline=-55;243;-62;234 dir=0.0 type=AlienAnt
script1="ant04.txt" run=1
```
Le programme `ant04.txt` recevra les 4 arguments –55, 243, -62 et 234. Ces arguments peuvent être exploités pour faire n'importe quoi. Ici, il s'agit de deux coordonnées (x;y) pour indiquer deux bornes entre lesquelles la fourmi effectuera des aller-retours :

- `point  nav1, nav2;`
- `nav1.x = cmdline(0);`
- `nav1.y = cmdline(1);`
- `nav2.x = cmdline(2);`
- `nav2.y = cmdline(3);`

La variable `nav1` vaudra donc (-55;243) et `nav2` vaudra (-62;234). L'énorme avantage de la ligne de commande et de pouvoir écrire un seul programme qui aura différents comportements selon les arguments passés. Il est possible de passer un maximum de 10 arguments à un programme.

3.2.5.2. Vaisseau spatial

```
CreateObject pos= 0.00; 0.00 dir=0.0 type=SpaceShip run=1
```
Le vaisseau spatial `SpaceShip` ne contient pas de programme CBOT. Mais la commande `run` peut tout de même prendre les valeurs suivantes :

Commande	Action au début de la mission
`run=0`	Le vaisseau est posé et ses portes sont ouvertes
`run=1`	Le vaisseau spatial atterrit puis ouvre ses portes
`run=2`	Le vaisseau spatial est amené par le portique géant de transport (terre #3)
`run=3`	Mode spécial pour les scènes `win` et `lost`
`run=11`	Voyage intersidéral puis atterrissage

Lors d'un voyage intersidéral, il est possible de choisir les planètes visibles avec la commande `Planet ... mode=1`.

3.2.5.3. Œuf

```
CreateObject pos=38;298 dir=1.5 type=AlienEgg autoValue1=20
autoType=AlienAnt autoString="ant10.txt" run=1
```

L'œuf donne naissance à un ennemi après un temps variable. Ceci est très pratique lorsque de nombreux ennemis doivent attaquer par vagues successives. Tant que l'œuf n'est pas éclot, l'ennemi n'existe physiquement pas, et donc ne surcharge ni la mémoire ni le processeur.

```
CreateObject ... autoValue1=20
```

Délai en secondes avant l'éclosion.

```
CreateObject ... autoType=AlienAnt
```

Type de l'insecte qui naîtra.

```
CreateObject ... autoString="ant10.txt"
```

Programme CBOT qui sera exécuté par l'insecte. Le nom peut être précédé de `%user%\`, comme d'habitude.

```
CreateObject ... run=1
```

Indique que l'œuf est actif, et qu'un insecte naîtra après le temps fixé.

Il est possible de passer des arguments au programme de l'insecte avec `cmdline`. Les arguments sont stockés temporairement dans l'objet `AlienEgg`, puis passé au programme de l'insecte lors de l'éclosion.

3.2.6. Eclairages

```
CreateLight dir=0.0;-1.0;0.0 color=0.6;0.6;0.6 type=Terrain
```

Crée une lumière omnidirectionnelle qui éclaire la scène. Des lumières différentes éclairent le terrain ou les objets. Il faut évidemment veiller à une cohérence globale.
`dir` est un vecteur de direction `(x;z;y)`. L'amplitude du vecteur n'a aucune importance. La plupart des lumières ont la composante `z` à –1, afin de diriger la lumière de haut en bas `(x;-1;y)`.
`color` détermine la couleur de la lumière, sous forme de trois composantes rouge, verte et bleue comprises généralement entre –2 et 2 :

Valeur	Effet
-1.0	Une valeur négative absorbe la lumière ambiante (ce qui n'existe pas dans la réalité).
0.0	Absence de lumière.
1.0	Eclairage normal.
2.0	Sur-exposition.

Le `type` peut prendre les valeurs suivantes :

Type	Effet
`Terrain`	Eclaire le terrain.
`Object`	Eclaire les objets (robots, bâtiments, plantes, etc.).
`Quartz`	Eclaire les crystaux (par exemple sur la planète Crystalium) avec un mouvement lent pour simuler les reflets.

Le tableau ci-dessous montre différents éclairages de type `Object` sur un robot orienté comme suit :

		
dir=-1;-1; 1 color=0.6;0.6;0.6 dir= 1;-1; 1 color=0.3;0.3;0.3 dir=-1;-1;-1 color=0.3;0.3;0.3 dir= 1;-1;-1 color=0.2;0.2;0.2	dir=-1;-1; 1 **color=2.0;0.0;0.0** dir= 1;-1; 1 color=0.3;0.3;0.3 dir=-1;-1;-1 color=0.3;0.3;0.3 dir= 1;-1;-1 color=0.2;0.2;0.2	dir=-1;-1; 1 color=0.6;0.6;0.6 dir= 1;-1; 1 color=0.3;0.3;0.3 dir=-1;-1;-1 **color=2.0;0.0;0.0** dir= 1;-1;-1 color=0.2;0.2;0.2
		
dir= 1;-1; 1 color=0.3;0.3;0.3	dir=-1;-1; 1 color=0.6;0.6;0.6	dir=-1;-1;-1 color=0.3;0.3;0.3

`WaterColor color=-0.6;-0.1;-0.1`
Modification de la couleur ambiante lorsqu'on est sous l'eau. Les valeurs ci-dessus donnent une teinte bleue foncée. Les trois composantes données ici sont additionnées à toutes les lumières créées avec `CreateLight` lorsque la caméra est sous l'eau.

Par exemple, avec `color=-0.6;-0.1;-0.1`, les couleurs sont modifiées comme suit sous l'eau :

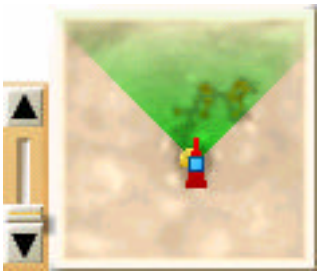
Air	**Water (**-0.6;-0.1;-0.1)
color= 0.63; 0.63; 0.63 type=Terrain	color= 0.03; 0.53; 0.53 type=Terrain
color=-0.70;-0.70;-0.70 type=Terrain	color=-1.50;-0.80;-0.80 type=Terrain
color= 1.40; 1.40; 1.40 type=Terrain	color= 0.80; 1.30; 1.30 type=Terrain
color= 0.56; 0.56; 0.56 type=Object	color=-0.04; 0.46; 0.46 type=Object
color= 0.32; 0.32; 0.32 type=Object	color=-0.28; 0.22; 0.22 type=Object
color= 0.32; 0.32; 0.32 type=Object	color=-0.28; 0.22; 0.22 type=Object
color= 0.16; 0.16; 0.16 type=Object	color=-0.44; 0.06; 0.06 type=Object

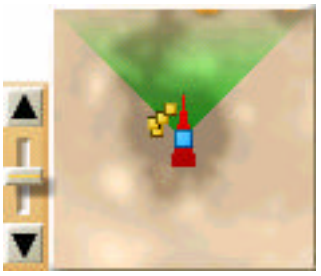
`CreateSpot pos=12;50;-75 color=1.00;-0.20;0.10 type=Terrain`
Cette commande, rarement utilisée, place une véritable lumière fixe sur le terrain, de type « spot ». Généralement, il vaut mieux utiliser `GroundSpot`.

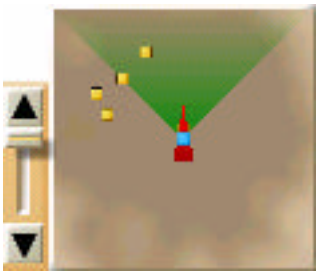
3.2.7. Mini-carte

`MapColor floor=104;185;205 water=154;235;255`
Couleurs à utiliser dans la mini-carte, en bas à droite de la fenêtre, pour représenter le sol ou l'eau.

`MapZoom factor=4`
Zoom par défaut à utiliser pour la mini-carte, en bas à droite de la fenêtre. Les valeurs possibles sont comprises entre 1 (toute la carte visible) et 8 (zoom le plus fort).


factor=1


factor=3


factor=8

3.2.8. Options

Les options déterminent ce qu'il est possible de faire dans l'exercice ou la mission.

```
NewScript name="%user%\tower1.txt" type=WheeledGrabber
```

Les robots placés initialement dans une mission reçoivent les programmes avec la commande `CreateObject ... script1="nom"`. Si un robot est construit par la suite dans une `BotFactory` et qu'il doit recevoir un programme, il faut utiliser cette commande.

- `name` détermine le nom du programme.
- `type` détermine le type de robot qui reçoit le programme. `All` permet de charger un programme dans tous les futurs robots construits.

```
EnableBuild type=PowerStation
```

Liste des bâtiments que le cosmonaute peut construire :

`ResearchCenter`	Centre de recherches
`BotFactory`	Fabrique de robots
`Converter`	Convertisseur
`PowerStation`	Station de recharge
`RadarStation`	Radar
`RepairCenter`	Centre de réparation
`DefenseTower`	Tour de défense
`PowerPlant`	Fabrique de piles
`Derrick`	Derrick
`NuclearPlant`	Centrale nucléaire
`AutoLab`	Laboratoire d'analyse
`PowerCaptor`	Paratonnerre
`ExchangePost`	Borne d'information

Liste des actions spéciales disponibles pour le cosmonaute :

`FlatGround`	Vérifier si le sol est plat
`Flag`	Mettre ou enlever un drapeau

```
EnableBuild type=ResearchCenter
EnableBuild type=BotFactory
EnableBuild type=Converter
EnableBuild type=PowerStation
EnableBuild type=RadarStation
EnableBuild type=NuclearPlant

EnableBuild type=FlatGround
EnableBuild type=Flag
```

Dans l'exemple ci-dessus, le bouton pour construire une centrale nucléaire `NuclearPlant` n'apparaît pas, car la recherche correspondante n'est pas encore effectuée.

```
EnableResearch type=WINGER
```

Liste des recherches qu'il est autorisé de faire en construisant un centre de recherche `ResearchCenter` :

TRACKER	Robots Tracked*
WINGER	Robots Winged*
THUMPER	Robots Thumper
SHOOTER	Robots *Shooter
TOWER	Bâtiments DefenseTower
PHAZER	Robots PhazerShooter
SHIELDER	Robots Shielder
ATOMIC	Bâtiments NuclearPlant

Liste des recherches qu'il est autorisé de faire en construisant un laboratoire AutoLab :

iPAW	Robots Legged*
iGUN	Robots *OrgaShooter

```
DoneResearch type=WINGER
```

Liste des recherches déjà effectuées :

TRACKER	Robots Tracked*
WINGER	Robots Winged*
THUMPER	Robots Thumper
SHOOTER	Robots *Shooter
TOWER	Bâtiments DefenseTower
PHAZER	Robots PhazerShooter
SHIELDER	Robots Shielder
ATOMIC	Bâtiments NuclearPlant
iPAW	Robots Legged*
iGUN	Robots *OrgaShooter
RECYCLER	Robots Recycler
SUBBER	Robots Subber
SNIFFER	Robots *Sniffer

- `DoneResearch type=SUBBER`

Par exemple, il faut mettre cette ligne de commande pour pouvoir fabriquer un robot sous-marin SUBBER dans la BotFactory.

- `DoneResearch type=WINGER`
- `DoneResearch type=SHOOTER`

Il faut mettre ces deux lignes de commande pour pouvoir fabriquer un robot WingedShooter dans la BotFactory.


```
EnableResearch type=TRACKER
EnableResearch type=SHOOTER
EnableResearch type=TOWER
EnableResearch type=ATOMIC

DoneResearch type=TRACKER
DoneResearch type=SHOOTER
DoneResearch type=TOWER
```

Dans l'exemple ci-dessus, les recherches `TRACKER`, `SHOOTER` et `TOWER` sont effectuées. La recherche `ATOMIC` est disponible.

3.3. Conditions de fin

```
EndMissionTake pos=0;0 dist=1000 type=Me lost=0
EndMissionTake pos=0;0 dist=1000 type=WheeledGrabber lost=0
EndMissionTake pos=0;0 dist=1000 type=Titanium min=4
EndMissionTake pos=0;0 dist=1000 type=AlienAnt min=0 max=0
```

Critères pour déterminer à quel moment la mission est terminée. Il faut utiliser une ligne pour chaque type d'objet différent.
Les lignes sont vérifiées les unes après les autres, de haut en bas. Par exemple, si la 2ème ligne décide que la mission est perdue et que la 3ème décide que c'est gagné, la décision prise sera « perdu ».

```
... pos=x;y dist=radius
```

Position centrale et rayon de la zone circulaire dans laquelle la présence des objets est vérifiée. `pos=0;0 dist=1000` indique que l'on prend en compte toute la carte.

```
... type=Me
```

Type de l'objet vérifié (voir la commande `CreateObject` pour la liste des types possibles).

```
... lost=0
```

Si le nombre d'objets atteint cette limite inférieure, la mission est immédiatement échouée.

```
... min=4
```

Si le nombre d'objets est inférieur à cette limite, il n'est pas possible de décoller (et donc de terminer la mission). Utile par exemple pour vérifier qu'un certain type de ressource a été produit dans une quantité donnée.

```
... max=0
```

Si le nombre d'objets est supérieur à cette limite, il n'est pas possible de décoller (et donc de terminer la mission). Utile par exemple pour vérifier que tous les ennemis ont été éliminés.

3.4. Caméra initiale

```
Camera eye=0.00;5.00;0.00 lookat=0.00;1.00;0.00 delay=0
```

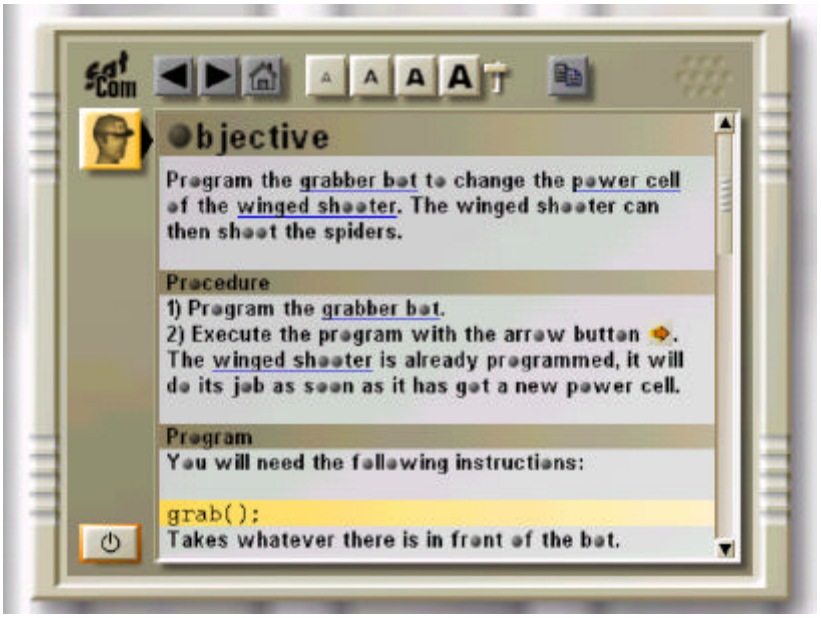
Initialement, la caméra vise l'objet sélectionné (`CreateObject ... select=1`). Cette commande permet de choisir une position et une direction de visée autre. La caméra effectuera alors un traveling plus ou moins rapide (selon `delay`) jusqu'à l'objet sélectionné.

4. Mise en page pour le *Sat*Com

Les textes d'aide ou d'instructions « officiels » affichés dans le ***Sat*Com** sont tous contenus dans le dossier `help\`. Ces textes contiennent des commandes de mise en page qui commencent par une barre oblique arrière « \ » et se terminent par un point-virgule « ; ». Un paragraphe doit être écrit « d'une traite », sans mettre de fins de lignes « ¶ » .
Voici un exemple de texte brut :

```
\b;Objective¶
Program the \l;grabber bot\u object\botgr; to change the \l;power cell\u
object\power; of the \l;winged shooter\u object\botfj;. The winged shooter
can then shoot the spiders.¶
¶
\t;Procedure¶
1) Program the \l;grabber bot\u object\botgr;. ¶
2) Execute the program with the arrow button \button 21;. ¶
The \l;winged shooter\u object\botfj; is already programmed, it will do its
job as soon as it has got a new power cell.¶
¶
\t;Program¶
You will need the following instructions:¶
\c;¶
\s;grab();\n;¶
\n;Takes whatever there is in front of the bot.¶
\c;¶
```

Il sera affiché comme suit dans le ***Sat*Com** :




```
\b;
```
Grand titre avec fond brun. Cette commande doit être au début de la ligne. Elle se termine automatiquement à la fin de la ligne.

```
\t;
```
Sous-titre avec fond brun. Cette commande doit être au début de la ligne. Elle se termine automatiquement à la fin de la ligne.

`\s;`
Sous-titre avec fond jaune, utilisé pour mettre en évidence les programmes CBOT. Cette commande doit être au début de la ligne. Elle se termine automatiquement à la fin de la ligne.

`\tab;`
Tableau avec fond orange, utilisé pour les rapports du satellite d'observation. Cette commande doit être au début de la ligne. Elle se termine automatiquement à la fin de la ligne.

`\l;hypertext\u file;`
Lien hypertexte. Le mot encadré par `\l;` et `\u;` est le lien. La commande `\u;` indique vers quel fichier pointe le lien. Par exemple, `\u object\botgr;` pointe vers le fichier général `help\object\botgr.txt`. Autre exemple, `\u %user%\mlink;` pointe vers le fichier spécifique au niveau supplémentaire `user\sample01\mlink.txt`.

`\c;`
Utilise la fonte courrier, utilisée pour les programmes CBOT.

`\n;`
Utilise la fonte normale, utilisée pour tout ce qui n'est pas un programme CBOT.

`\image %user%\filename dx dy;`
Affiche une image `.bmp` contenue dans le dossier spécifique au niveau supplémentaire. Si le nom `%user%\` est omis, l'image sera cherchée dans le dossier général `diagram\`.
Par exemple, `\image %user%\sample01 8 8;` affiche `user\sample01\sample01.bmp`.
Par exemple, `\image sniff1 12 12;` affiche `diagram\sniff1.bmp`.
`dx` est la largeur en caractères, et `dy` la hauteur également en caractères. Les valeurs `12 12` conviennent donc pour une image carrée. L'image est déformée en fonction des valeurs données.

`\token;`
Affiche les caractères suivant sur fond orangé. Cela indique une instruction du langage CBOT. La commande `\norm;` termine.

`\type;`
Affiche les caractères suivant sur fond vert. Cela indique un type de variable du langage CBOT (par exemple `int`, `float`, `point`, etc.). La commande `\norm;` termine.

`\const;`
Affiche les caractères suivant sur fond rouge. Cela indique une constante du langage CBOT. La commande `\norm;` termine.

`\key;`
Affiche les caractères suivant sur fond gris. Cela indique une touche d'action. La commande `\norm;` termine.

`\norm;`
Termine le fond coloré d'une commande `\token;`, `\type;`, `\const;` ou `\key;`.

`\key name;`
Affiche le nom d'une touche d'action. Par exemple, `\key help;` affiche le nom de la touche qui actionne le ***Sat*Com**, généralement `F1`. Mais si l'utilisateur a modifié cette action dans les réglages, c'est un autre nom qui sera affiché. Veillez donc à ne jamais écrire `F1` en toutes lettres !
Les noms d'actions suivants sont possibles :

Nom	Touche par défaut	Action
`left`	Flèche gauche	Tourne à gauche
`right`	Flèche droite	Tourne à droite
`up`	Flèche haute	Avance
`down`	Flèche basse	Recule
`gup`	Shift	Monte
`gdown`	Ctrl	Descend
`camera`	Barre d'espace	Change le point de vue de la caméra
`desel`	0 pavé numérique	Sélectionne l'objet précédent
`action`	Entrée	Action par défaut du robot sélectionné
`near`	+ pavé numérique	Approche la caméra
`away`	- pavé numérique	Eloigne la caméra
`next`	Tabulateur	Sélectionne le robot ou le bâtiment suivant
`human`	Home	Sélectionne le cosmonaute
`quit`	Esc	Affiche le menu pour quitter, enregistrer, etc.
`help`	F1	***Sat*Com** avec les instructions générales
`prog`	F2	***Sat*Com** avec les explications du langage CBOT
`cbot`	F3	***Sat*Com** avec la syntaxe de l'instruction éditée
`visit`	. pavé numérique	Montre le lieu d'une erreur
`speed10`	F4	Vitesse x1.0
`speed15`	F5	Vitesse x1.5
`speed20`	F6	Vitesse x2.0

Comme les touches sont généralement affichées avec un fond gris, il faut utiliser la combinaison de commandes suivantes pour indiquer une touche d'action, ici `help` :
`\key;\key help;\norm;`

5. Le fichier de configuration

Pour résoudre certains problèmes, il est possible de modifier le fichier `colobot.ini` avec un éditeur de texte (par exemple avec le bloc-notes de Windows). Généralement, ce fichier se trouve à l'emplacement suivant :

- `C:\Program Files\Colobot\colobot.ini`

N'ajoutez pas de nouvelles lignes, mais modifiez simplement les valeurs existantes. Attention à ne pas insérer d'espace. Il faut quitter CoLoBoT avant de modifier ce fichier.

6. Problèmes

Pour résoudre certains problèmes, il est possible de modifier le fichier `colobot.ini`.
Si la végétation s'affiche mal, ou même pas du tout, vous avez peut-être mis à zéro le nombre d'objets décoratifs dans les options. Pour remettre 100% :

- `[Setup]`
- `GadgetQuantity=1.00`

Si la végétation ne s'affiche toujours pas, essayez :

- `[Engine]`
- `AlphaMode=0`

ou

- `[Engine]`
- `AlphaMode=2`

Si un carré apparaît autour des ombres, essayez :

- `[Engine]`
- `WhiteSrcBlend=9`
- `WhiteDestBlend=6`

ou

- `[Engine]`
- `WhiteSrcBlend=6`
- `WhiteDestBlend=3`

Si cela ne fonctionne pas, il faut supprimer les ombres :

- `[Engine]`
- `WhiteSrcBlend=0`
- `WhiteDestBlend=0`
- `[Setup]`
- `GroundShadow=0`

Lorsque un objet s'interpose entre l'objet sélectionné et la caméra, il devient transparent. Si l'objet n'est pas assez transparent, essayez :

- `[Engine]`
- `StateColor=0`

ou

- `[Engine]`
- `StateColor=1`

7. Réglages

Pour afficher le nombre d'images par seconde, il faut appuyer sur `Ctrl+Pause` puis taper la commande "`showstat`" et valider avec la touche `Entrée` :

- `Ctrl+Pause showstat Entrée`

La partie supérieure de l'écran affiche alors, par exemple :

- `32.46 fps T=11558 (640x480x16)`

Le premier chiffre correspond au nombre d'images par seconde (fps = frame per second).
Le deuxième chiffre indique le nombre de triangles affiché dans la scène.
Les 3 derniers chiffres entre parenthèses sont la résolution (largeur x hauteur) et le nombre de bits pour les couleurs.

Tous les réglages sont mémorisés dans le fichier `colobot.ini`, présent dans le dossier principal où CoLoBoT a été installé. Ce fichier peut être modifié (avec prudence et après en avoir fait une copie) avec un éditeur de texte, comme par exemple le bloc-notes de Windows.

8. Equipe de développement

- Daniel Roux
- Denis Dumoulin
- Otto Kölbl
- Michael Walz
- Didier Gertsch

8.1. Beta testeurs

- Adrien Roux
- Didier Raboud
- Nicolas Beuchat
- Joël Roux
- Michael Jubin
- Daniel Sauthier
- Nicolas Stubi
- Patrick Thévoz

8.2. Copyright

8.3. Développeur

EPSITEC SA
Mouette 5
CH-1092 Belmont

colobot@epsitec.ch
www.colobot.com