



## Creating Colobot User Levels

## 1. Installing user levels

Installation and creation of user levels requires COLOBOT version 1.7 (or higher). If necessary, you can download a patch from [www.colobot.com](http://www.colobot.com).

User levels are accessible with the “User” button on the main screen :



All user levels are contained in the folder `user\` placed in the main Colobot folder. Generally, the location is :

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

If the folder `user\` does not exist, simply create it.

A user level can contain one or several missions. Every user level corresponds to a line in the left list :



A user level is a folder containing several files. In the example above, level "Sample #1" is contained in the folder :

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

Note that the name of the level such as it appears in the left list is not the name of the folder, but the name contained in the file `sample01\scene00.txt`, in the command :

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

The different missions in the right list correspond to files `sample01\sceneNN.txt` :

File	Title
<code>sample01\scene01.txt</code>	Alert
<code>sample01\scene02.txt</code>	Mc Gywer

The `scene` files must have successive numbers. Let's suppose that the following files exist :

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

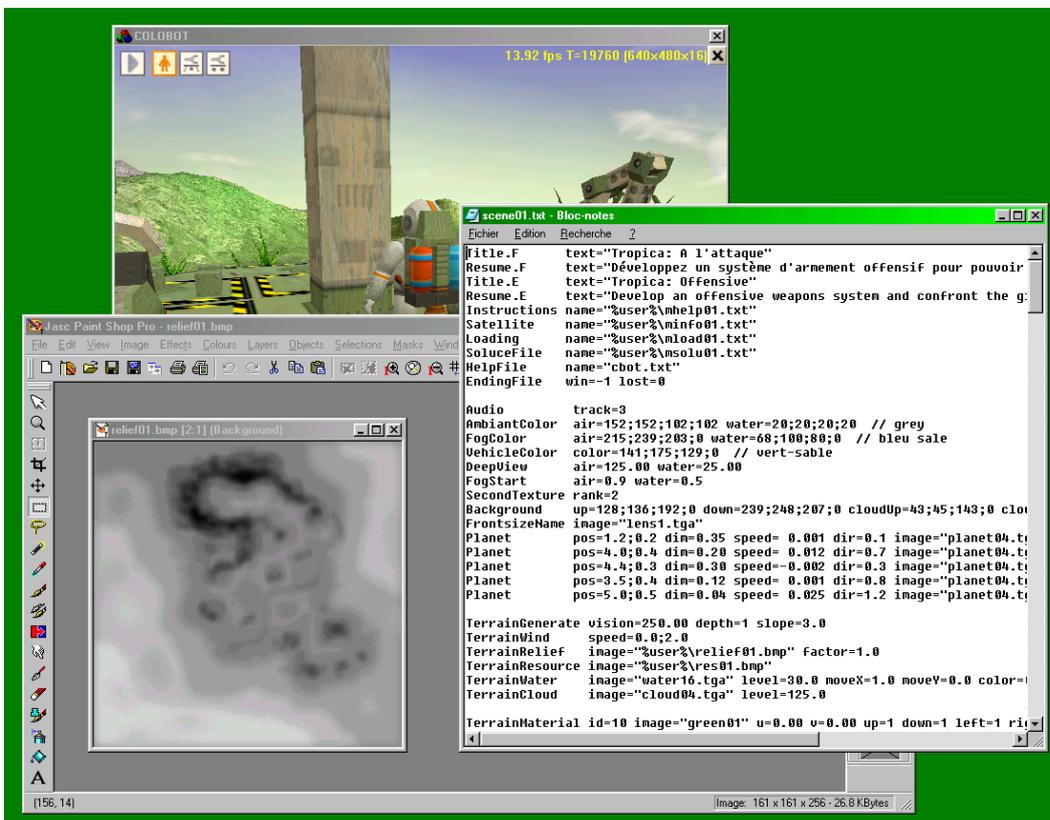
In that case the level will contain only 2 missions, number 01 and 02. Mission number 05 does not appear in the list, because missions number 03 and 04 are missing.

## 2. Working environment

When you create user levels, it is often necessary to activate other programs than COLOBOT, with Alt+Tab. This is much more pleasant when COLOBOT works in windowed mode. In order to run Colobot in windowed mode click "Options", then on the "Device" tab :



Then remove the option "Full screen" and click on "Apply changes". Note that in windowed mode, COLOBOT works always in 640x480 pixels. You cannot change this.



### 3. Create user levels

The file `sample01.zip` contains an example of two complete missions. Unzip this file into the folder `user\` placed in the main folder where COLOBOT has been installed. Generally, the location is :

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

If the folder `user\` does not exist, simply create it.



These screenshots show the two missions in `sample01.zip`.

This sample is used throughout this manual. The `user\sample01\` folder contains following 24 files :

File	Contents	
mhhelp01.txt	Instructions for mission #1	Texts shown in <b>SatCom</b>
mhhelp02.txt	Instructions for mission #2	
minfo01.txt	Satellite info for mission #1	
minfo02.txt	Satellite info for mission #2	
mload01.txt	Explanations of programs sent by Houston for mission #1	
mload02.txt	Explanations of programs sent by Houston for mission #2	
msolu01.txt	Solution for mission #1	
msolu02.txt	Solution for mission #2	
sant01.txt	Script loaded initially into certain ants	Scripts
skill.txt	Script loaded initially into certain Shooter and OrgaShooter bots	
scene00.txt	Title of the level	Scenes
scene01.txt	Description of mission #1	
scene02.txt	Description of mission #2	
back01.bmp	Stars for the background in mission #1	Images
cloud02.bmp	Clouds for the sky in mission #2	
lens02.bmp	Sunbeams in mission #2	
nevada1.bmp	Ground in mission #2 (sand-rock1 materials)	
nevada2.bmp	Ground in mission #2 (rock1-rock2 materials)	
relief01.bmp	Ground relief for mission #1	
relief02.bmp	Ground relief for mission #2	
res01.bmp	Subsoil resources in mission #1	
res02.bmp	Subsoil resources in mission #2	
terra01.bmp	Ground texture for mission #1	
water01.bmp	Water texture for mission #1	

All text files (.txt) can be modified or created with a text editor such as Notepad. Image files (.bmp) can be modified with a graphic editor such as Paint, PaintShop, PhotoShop or PhotoPaint.

The file `user\sample01\scene01.txt` describes the first mission :

Commands	Used files
Title.E text="Alert"	
Resume.E text="Your base is under alert ..."	
Instructions name="%user%\mhhelp01.txt"	Instructions for <b>SatCom</b>
Satellite name="%user%\minfo01.txt"	Satellite report for <b>SatCom</b>
Loading name="%user%\mload01.txt"	Programs sent by Houston for <b>SatCom</b>
SoluceFile name="%user%\msolu01.txt"	Solution for <b>SatCom</b>
HelpFile name="cbot.txt"	Official programming help for <b>SatCom</b>
...	
Background image="%user%\back01.bmp"	Stars shown in the sky
FrontsizeName image="lens1.tga"	Sunbeams shown in frontsize
...	
TerrainRelief image="%user%\relief01.bmp"	Ground Relief
TerrainResource image="%user%\res01.bmp"	Subsoil resources
TerrainWater image="%user%\water01.bmp" ...	Water texture
TerrainInitTextures image="%user%\terra.bmp" dx=1 dy=1 table=1	Ground texture
...	
CreateObject pos=12.50;-112.50 dir=1.0 type=WingedOrgaShooter power=0.2 script1="%user%\skill.txt"	Script loaded initially into the bot
...	

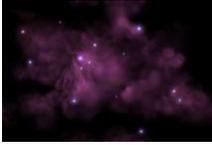
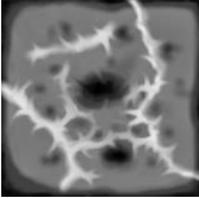
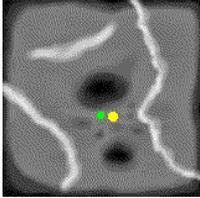
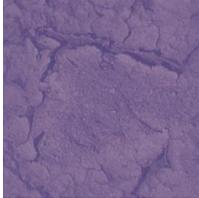
Certain commands contained in the file `user\sample01\scene01.txt` make reference to the other files. For example the command :

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

indicates the file which contains instructions shown within the **SatCom**, to explain what must be done in mission.

When a filename is preceded by `%user%`, the file is taken from the user mission's folder (that is `user\sample01\` in this sample).

The images contained in the `user\sample01\` folder used for the first mission are :

				
back01.bmp	relief01.bmp	res01.bmp	water01.bmp	terra001.bmp
512x347	161x161	161x161	256x256	256x256
16 millions colors	256 grey scale	256 colors	16 millions colors	16 millions colors



### 3.1. Standard files

All level specific files must be in the same folder `user\sample01\`, but it is still possible to make reference to certain standard files of COLOBOT, which are in the standard Colobot folders. For example, the file which contains explanations about the Colobot programming language is always the same. It is useless to duplicate it in every user level. The command :

- `HelpFile name="cbot.txt"`

means that the `cbot.txt` file is located the standard Colobot help folder :

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

Normally, a filename must be preceded with `%user%\` to indicate that it is in the mission specific folder. Without this prefix the file is opened in one of the standard Colobot folders :

Standard folder	Contents
<code>scene\</code>	Description of exercises and missions.
<code>help\</code>	Instructions shown in <b>SatCom</b> .
<code>diagram\</code>	Images shown in <b>SatCom</b> .
<code>textures\</code>	Images for ground relief and subsoil resources.
<code>script\</code>	CBOT programs loaded initially into bots.

You can make reference to files in these folders, or copy them in your `user\sample01\` folder and modify them for your own levels. **But never modify a file in a standard folder**, this could compromise the whole Colobot game and your user level would not work once installed on another computer.

All standard files for exercises, missions are in the standard Colobot folder `scene\` :

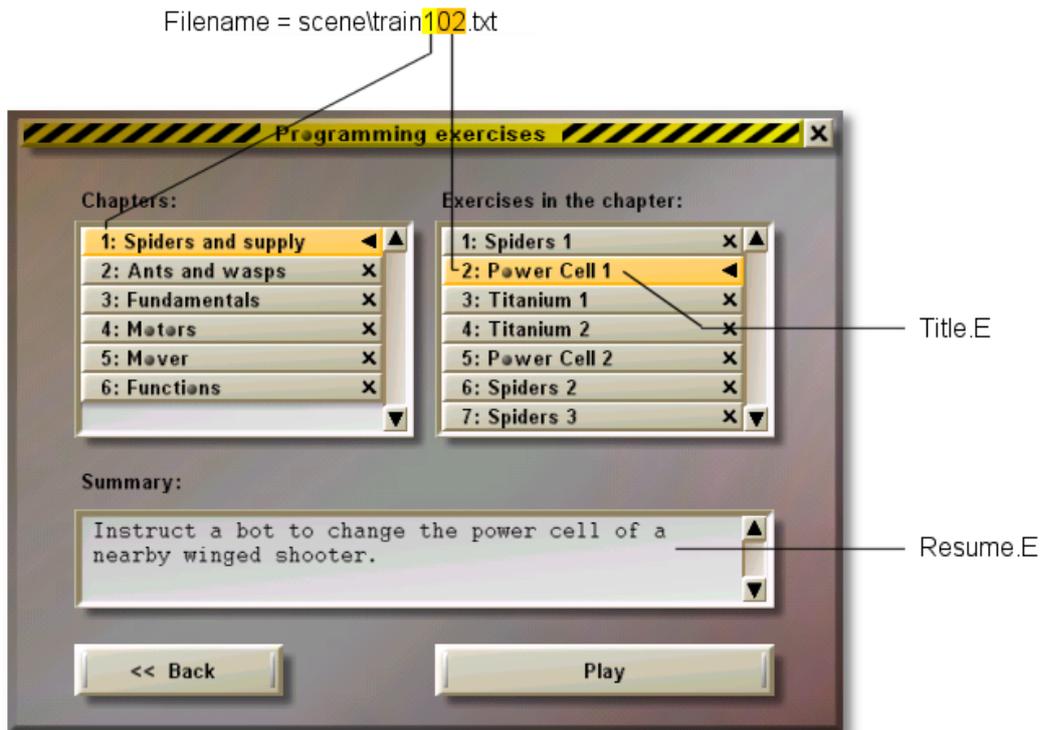
<code>scene\scenexyy.txt</code>	Missions
<code>scene\freexyy.txt</code>	Free games
<code>scene\trainxyy.txt</code>	Exercises
<code>scene\defixyy.txt</code>	Challenges



The 3 digit number is composed of :

<b>x</b>	Chapter number	1..9
<b>yy</b>	Rank in the chapter	01..99

For example, `train102.txt` is the second exercise of the first chapter.



**Remark :** Do not hesitate to consult these files to understand how the "official" missions of COLOBOT are realized.

## 3.2. Description of a mission or an exercise

The description of a mission determines the ground relief, the textures, the initial position of the various bots, raw materials, plants, etc. and it is contained in a file called `user\sample01\sceneNN.txt`. Every line of such a description file contains a command, which begins with a keyword followed by various parameters. Empty lines are ignored. A double slash `//` starts a comment which ends automatically at the end of the line. Colobot is very picky concerning the format of the commands so all commands are case sensitive and you cannot put spaces before and after `=` signs.

Correct syntax :

- `Title.E text="Alert"`

Wrong syntax :

- `title.E text = "Alert" // letter case not respected, spaces before and after "=" sign.`

Colobot does not do much error checking while parsing the mission description file. So wrong commands may result in unpredictable behavior.

### 3.2.1. Header

The header determines the name of the exercise or the mission and names of the explanation files.

```
Title.E text="Alert"
```

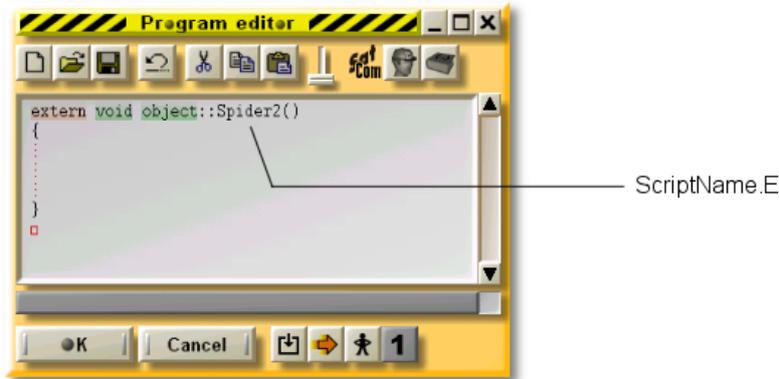
Short name of the mission, as it appears in the right list.

```
Resume.E text="Your base is under alert ..."
```

Summary of the mission, as it appears below the two lists.

**Remark :** Title commands `Title.F` and `Resume.F` allow to specify the texts which will be shown with the French version of COLOBOT.

`ScriptName.E text="Spider2"`  
 Default name given to new programs.



`Instructions name="%user%\mhelp01.txt"`  
 Name of the file which contains the instructions of the mission, which will be shown in **SatCom**. All instructions files should start with the letters "mhelp".

`Satellite name="%user%\minfo01.txt"`  
 Name of the file which contains the Satellite report, which will be shown in **SatCom**. All satellite report files should start with the letters "minfo".

`Loading name="%user%\mload01.txt"`  
 Name of the file which contains programs sent by Houston, which will be shown in **SatCom**. All program files sent by Houston should start with the letters "mload".

`SoluceFile name="%user%\msolu01.txt"`  
 Name of the file which contains the solution of the mission, which will be shown in **SatCom**. All solution files should start with the letters "msolu".

`HelpFile name="cbot.txt"`  
 Name of the file which contains instructions on Colobot programming shown when the F2 key is pressed. The standard file `cbot.txt` is located in the standard folder `help\`. Normally, all missions make reference to the same instructions contained in the standard file `cbot.txt`.



EndingFile win=-1 lost=0

Scene to be used when mission over (successful or not).

- win=-1: there is no specific scene of the end. The mission ends simply as soon as the spaceship takes of.
- win=2: scene\win002.txt is to be used.
- win=123: scene\win123.txt is to be used.
- lost=0: scene\lost000.txt is to be used.

You cannot place the descriptions of these scenes in the user level folder. You must use the standard winxxx.txt files files in the folder scene\, without modifying them.

MessageDelay factor=5

Increases the time during which messages are shown at the top of the screen.

### 3.2.2. Atmosphere

This part describes the general colors, the fog, the sky etc. as well as the background music.

Audio track=0

Number of the audio track of the CD to be played during the mission. You can use following numbers :

Track number	Planet
0	Moon (silence)
2	Earth
3	Tropica
4	Crystallium
5	Saari
6	Volcano
7	Centaury
8	Orpheon
9	Terranova

AmbiantColor air=102;102;102;102 water=20;20;20;20

This is the color of indirect light sources for air and for under water.

Colors are specified by 4 components red/green/blue/alpha. Values are included between 0 (black) and 255 (white). The alpha component is generally ignored. For example :

- color=175;209;215;0 // sand-blue

Planet	Air	water
Earth	120;90;0;0	20;20;20;20
Moon	102;102;102;102	20;20;20;20
Tropica	152;152;102;102	20;20;20;20
Crystallium	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
Orpheon	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

Color of objects which are far away. See also DeepView and FogStart.

Planet	air	water
Earth	100;100;90;0	67;80;100;0
Moon	0;0;0;0	67;80;100;0
Tropica	215;239;203;0	68;100;80;0
Crytanium	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
Orpheon	50;30;10;0	67;80;100;0
Terranova	208;200;223;0	94;153;180;0

VehicleColor color=175;209;215;0

Color of bots and buildings.

			
208;206;196;0	141;175;129;0	234;129;26;0	158;120;82;0
Earth	Tropica	Volcano	Orpheon

GreeneryColor color=250;187;69;0

Color of vegetation.

InsectColor color=189;171;51;0

Color of insects.

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

```
DeepView air=100.00 water=25.00
```

Maximal distance of sight in meters. Anything beyond this distance is not displayed.

```
FogStart air=0.1 water=0.1
```

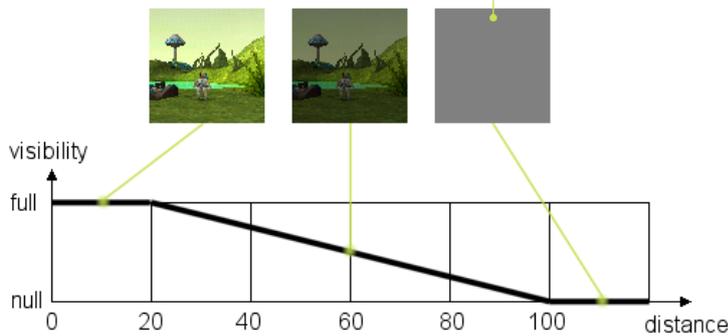
The more you approach the maximal distance of sight (`DeepView`) the more the color of objects merges with the fog color (`FogColor`); this simulates fog. A value of 0.1 indicates dense fog. A value of 0.9 indicates a light fog.

For example :

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

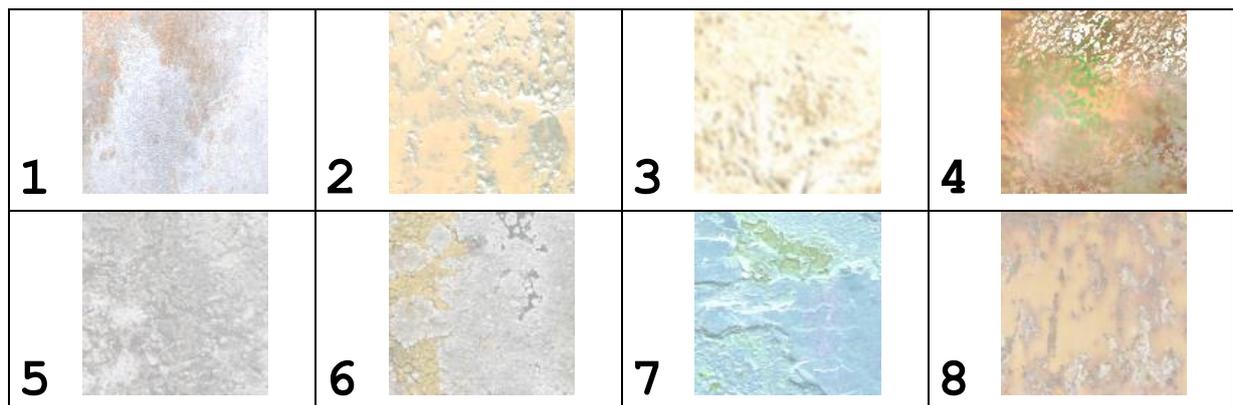
Distance from the observer	
0 to 20 metres	Normal display
20 to 100 metres	Gradually foggy display
100 metres and more	No more display

```
DeepView air=100.00
FogStart air=0.2
FogColor air=128;128;128;128
```

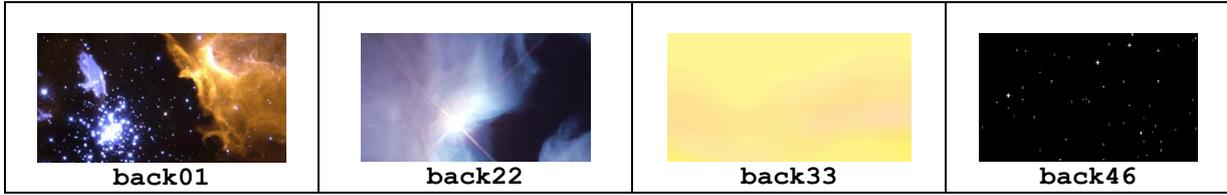


```
SecondTexture rank=3
```

Texture used to simulate dirt on bots and buildings. You can use a value between 1 and 8 :



```
Background image="back01.tga" up=76;105;226;0 down=192;250;255;0
Description of the background (sky). You can use one of the following images :
```



If the image name is preceded by %user%, the image is located in the user level folder (user\sample01\). You can use .bmp or .tga files. Do not exceed an image size of 1280x480.

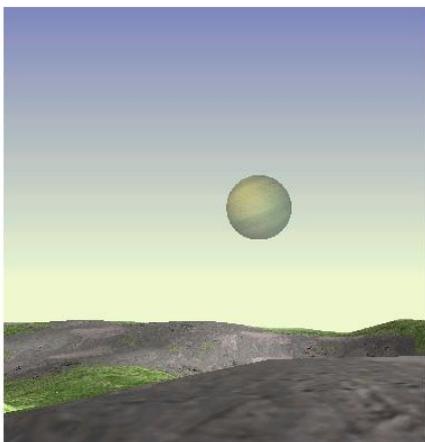
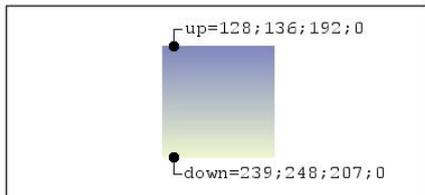
The up and down values are used if no image has been specified or if the "sky" option has been disabled in the "Graphics" tab of the Options in Colobot.

Both up and down are colors specified by their RGBA (red, green, blue, alpha) components. The background is drawn as a color gradation which starts with the up color at the top of the screen and finishes with the down color at the middle of the screen. The lower half of the screen is displayed with the down color and it never visible as it is normally covered by the ground.

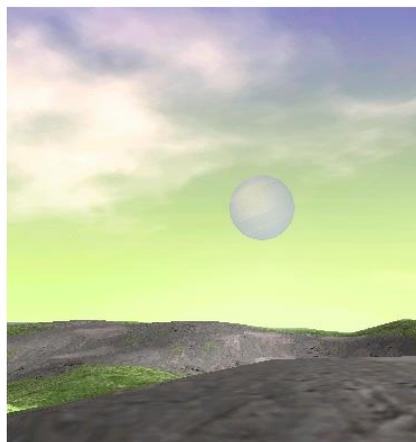
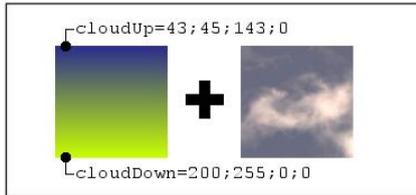
If you don't want to use an image, just omit the image part. Parameters cloudUp and cloudDown are used to specify the colors of the background behind clouds (TerrainCloud command). If clouds are absent (on low cost machines), the colors up and down are used instead. These two colors then should imitate as well as possible the colors of "real" clouds.

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

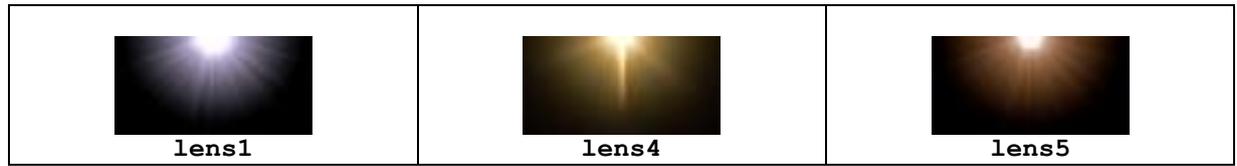
**Sky**



**Sky**



`FrontsizeName image="lens5.tga"`  
 Name of the frontsize texture, which contains a "lens flare" effect more or less visible according to the sight direction.

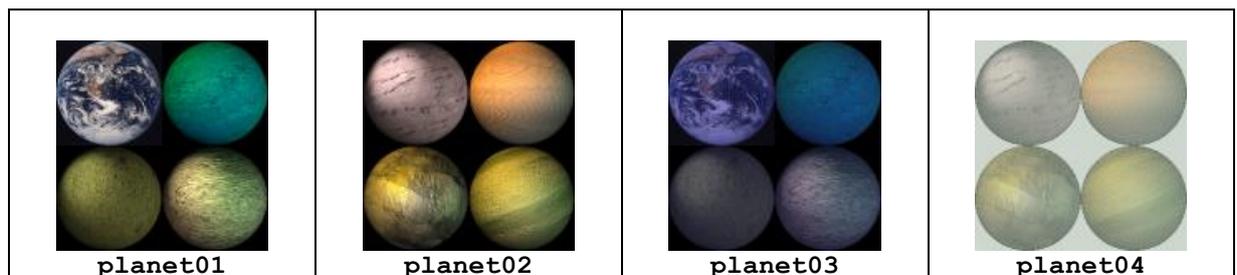
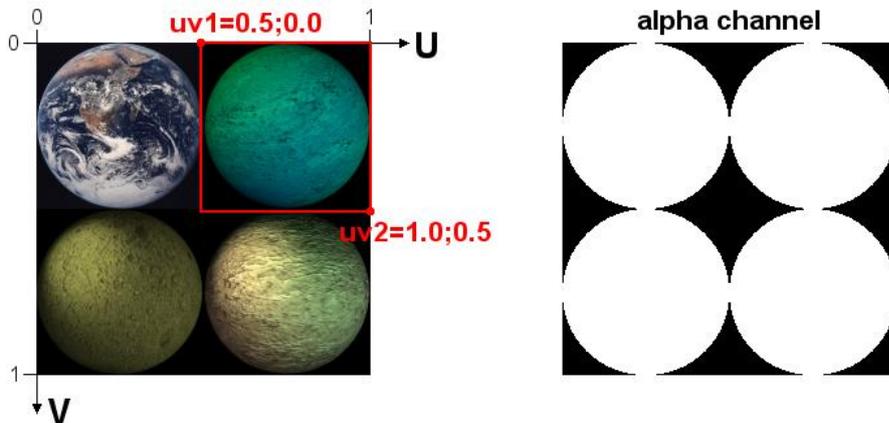


If the image name is preceded by `%user%`, the image is located in the user level folder (`user\sample01\`). You can use `.bmp` or `.tga` files. Do not an image size of 512x256; 128x64 is usually enough.

`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`

Create a steady or a mobile planet in the sky.

- `mode=0` : planet that is visible during the normal phase of game.
- `mode=1` : planet that is visible during the film of the intersidereal journey, when the spaceship is created with `CreateObject run=11`. (see also section 3.2.5.2).
- `pos` is the initial position in the sky. The first value is the direction and second the height.
- `dim` is the size of the planet in the sky. Use values from 0.02 for small stars (like on **Crystalium**) to 0.5 for really big planets (like Earth on the moon).
- `speed` of the planet, generally very slow (0 for a fix planet). Typical values are around 0.001
- `dir` is the vertical fluctuation of the movement. If `dir` is 0 the planet stays at the same height in the sky. Typical values are around 0.1
- `image` is the name of the image to be used (see below).
- The planet image files always contain four different planets as displayed below. `uv1` are the coordinates of the upper left corner and `uv2` are coordinates of the lower right corner of the planet in the image. The coordinates of the the upper left corner the image are 0;0 the coordinates of the lower right corner are 1;1.



If the image name is preceded by %user%, the image is located in the user level folder (user\sample01\). You must use .tga files with an alpha channel to specify the shape of the planets. Do not exceed an image size of 256x256.

### 3.2.3. *Ground*

This part describes the ground relief and its texture, the level and appearance of water as well as the location of resources in the subsoil (energy, titanium- and uranium ore).

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

- `vision` determines until which distance of the viewpoint the ground must be generated. This distance must be at least the double of the larger of the two `DeepView` distances (`air` or `water`).
- `depth` always must be set to 1.
- `hard` determines the hardness of the ground which influences the noise of steps when the astronaut walks.

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

Vector indicating the direction and strength of the wind. On the moon, there is no wind (`speed=0;0`). `speed=10;0` specifies a strong wind which blows from West to East. Wind influences the movement of clouds (`TerrainCloud`), particles of smokes and flags. Wind does however not influence the movement of bots.

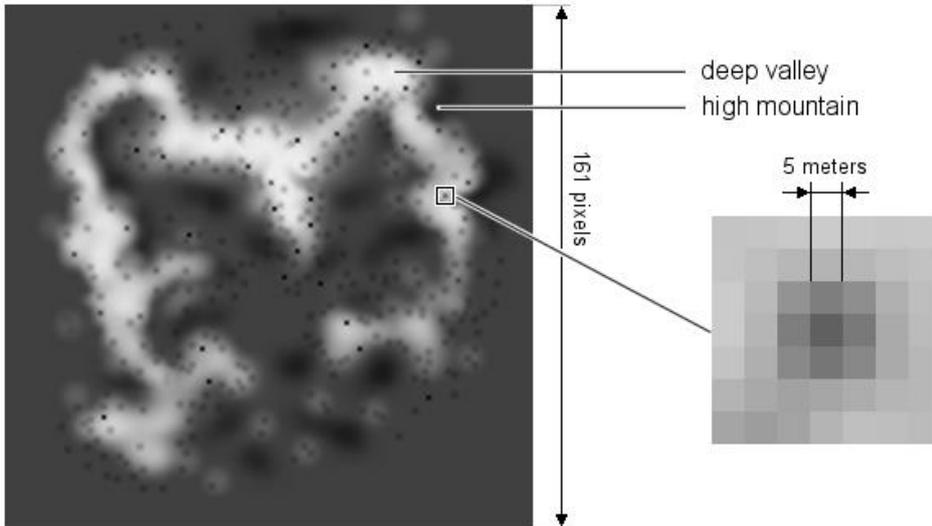
```
TerrainBlitz sleep=60 delay=5 magnetic=100
```

Activates the random flash of lightning generator. In the standard Colobot missions, this command is used only in the **Orpheon** missions.

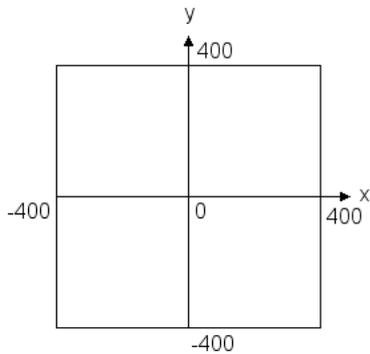
- `sleep` is the delay in seconds before the first lightning occurs. This allows to leave some time to build a first lightning conductor (`PowerCaptor`).
- `delay` gives the average period (in seconds) between two lightnings.
- Every metal object is surrounded by a virtual circular zone which attracts lightning. `magnetic` specifies the radius (in metres) of this zone. If a lightning occurs inside the `magnetic` radius of an object, the latter will be destroyed.

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

The relief of the ground is described by a 256 grey levels .bmp image having a size of exactly 161 x 161 pixels. The white color corresponds to the lowest possible height. The black color corresponds to the highest possible height. *factor* is a stretch factor which is usually set to 1.0. In that case, the 256 gray levels allow an altitude difference of 64 metres. So an intensity difference of 1 corresponds to a difference of height of 0.25 metres.



The coordinates of the central pixel 80;80 of the image corresponds to the coordinates 0;0 metres in COLOBOT. The coordinated 0;0 of the top-left pixel in the image corresponds to the extreme northwest point -400;400 metres in COLOBOT. A pixel in the image corresponds to a square on the ground of 5x5 metres in COLOBOT.



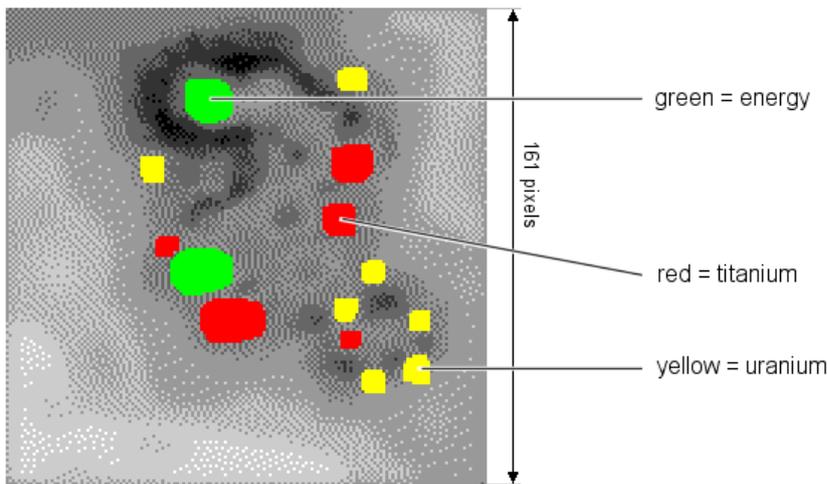
The universe of COLOBOT has a size of 800x800 metres. You can draw new reliefs with software such as PaintShop, or to reuse the numerous files `reliefxx.bmp` in the `textures\` folder.

```
TerrainResource image="%user%\res01.bmp"
```

This image determines the presence of resources in the subsoil. This must be a 256 colors .bmp image measuring 161 x 161 pixels using the standard Windows palette.

Color	RGB	Palette index	Contents of the subsoil
Red	255;0;0	5	titanium
Green	0;255;0	30	energy
Yellow	255;255;0	35	uranium
Green	0;104;0	24	key A
Green	51;104;0	25	key B
Green	102;104;0	26	key C
Green	153;104;0	27	key D

All other colors or gray levels are ignored.

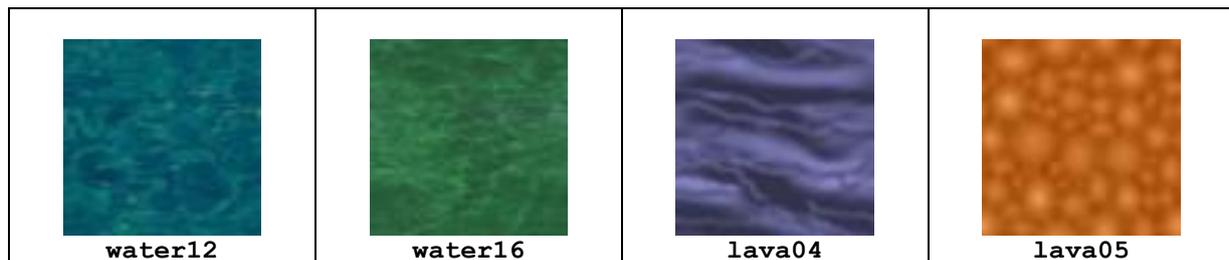


The simplest method is to start with the gray scale image used in `TerrainRelief` and to convert it to 256 colors.

```
TerrainWater image="water16.tga" level=30.0 moveX=1.0 moveY=0.0  
color=0;240;100;0 brightness=0.2
```

This command determines the appearance of water or lava, which is identical throughout one mission. You cannot have a water lake and a lava lake in the same mission.

- `image` specifies the appearance of the water or the lava.
- `level` indicates the water or lava level relative to the absolute zero height that corresponds to the color white in the relief image (`TerrainRelief`). The altitude (z-coordinate) of bots is then always relative to the water (or lava) level. A negative bot altitude means that the bot is under water. The water or lava level of all lakes in one mission is the same. You cannot have a lake at the bottom of a valley and another one on the top of a mountain in the same mission.
- `moveX` determines the amplitude of horizontal movement at the surface.
- `moveY` determines the amplitude of vertical movement. The surface is really deformed according to a bidimensional sinus, and rises and comes down slowly to simulate swell.
- `color` and `brightness` modify the color of the texture.



If the image name is preceded by %user%, the image is located in the user level folder (user\sample01\). You can use use .bmp or .tga files. Do not exceed an image size of 256x256.

```
TerrainLava mode=1
```

- If mode=1 the astronaut dies immediately as soon as he is below the lava level. In this mode the camera never goes below the lava level.

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

Image used for the mobile clouds in the sky. The movement of clouds depends on the wind (see TerrainWind). Following standard images exist :



If the image name is preceded by %user%, the image is located in the user level folder (user\sample01\). You can use use .bmp or .tga files. Do not exceed an image size of 640x480.

- level indicates the maximal height of the cloud layer in meters. The level height must exceed the highest mountain by at least several dozens of meters.

Use the Background command with cloudUp and cloudDown to specify colors behind clouds.

### 3.2.4. Ground textures

There are two different systems for applying a textures to the ground :

	Commands	Meaning
1	TerrainInitTextures	The texture is applied "blindly" to the ground allover the map.
2	TerrainMaterial TerrainInit TerrainLevel	Various materials are used according to the altitude. This allows for example to have sand under water, the herb on average heights and some snow on summits.

For example, the missions on the moon uses the first system, while the missions on Terranova use the second one.

The mapping commands are complex. You should copy the commands of an existing standard mission them as they are. Here are just some incomplete explanations.

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

Apply an uniform texture to the set of the ground. In this example the texture is composed of following 8 images, juxtaposed in this order :

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

This checkerboard of 8 textures is repeated in all directions.

So, the right edge of mars002.tga touches the left edge of mars003.tga.

The right edge of mars004.tga touches the left edge of mars001.tga.

The upper edge of mars002.tga touches the lower edge of mars006.tga. etc.

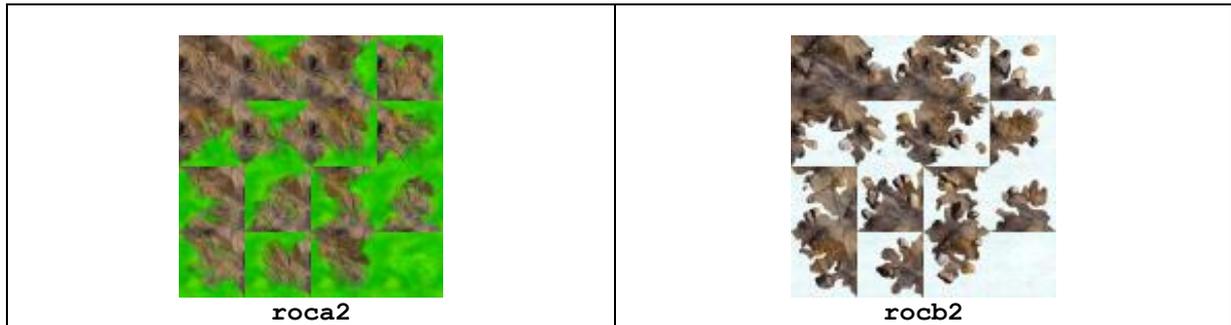
The filename is composed of the base image name followed by a 3 digit number as explained above. If the name is preceded by %user%, the image is located in the user\sample01\ folder of the user level. You can use .bmp or .tga images. For example :

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

Here the image user\sample01\moon015.bmp is used.

```
TerrainMaterial id=1 image="roca2" u=0.00 v=0.00 up=1 down=1 left=1 right=1
hard=0.8
```

Definition of a ground type having an id. number. For example, on **Terranova**, the following two textures are used :



Each texture is cut into 16 pieces (4x4) as shown above.

The following four main ground types are defined, with id 1 to 4 :

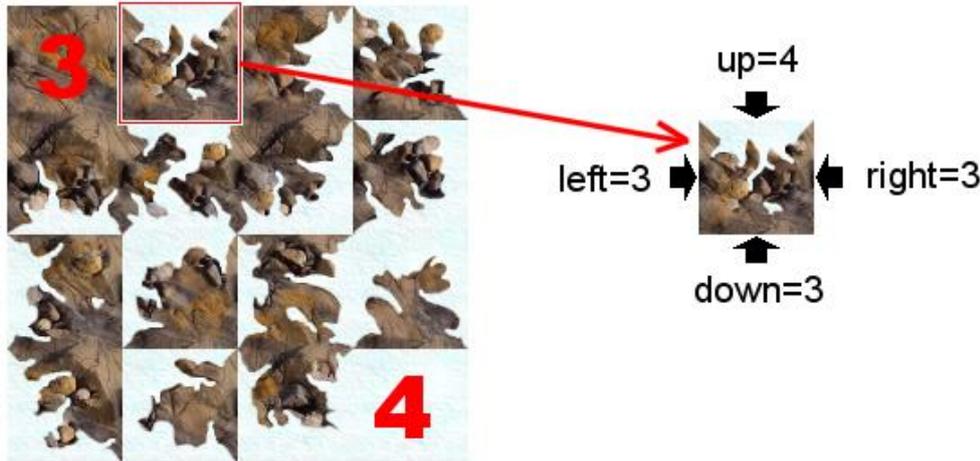
- id=1 image="roca2" u=0.00 v=0.00 up=1 down=1 left=1 right=1 // rock
- image="roca2" u=0.25 v=0.00 up=2 down=1 left=1 right=1
- image="roca2" u=0.50 v=0.00 up=1 down=1 left=1 right=2
- image="roca2" u=0.75 v=0.00 up=2 down=1 left=1 right=2
- image="roca2" u=0.00 v=0.25 up=1 down=2 left=1 right=1
- image="roca2" 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 // grass
- id=3 image="rocb2" u=0.00 v=0.00 up=1 down=1 left=1 right=1 // rock
- ...
- id=4 image="rocb2" u=0.75 v=0.75 up=3 down=3 left=3 right=3 // snow

The other materials (without id) are defined to allow transitions. Commands up, down, left and right give the id of the nearby ground types which can be used next to the ground type.

For example, the line :

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

corresponds to the second piece of the first line in the following image (id=3:rock and id=4:snow) :



hard determine the hardness of a ground type and must be between 0 and 1. The hardness influences the noise of the astronaut walking.

**Remark :** Ground types are used on the missions on Earth to draw roads on the ground (for example `scene\scene101.txt`). Three different ids correspond to roads East/West (id=3), North/South (id=4) and to crossroads (id=5).

If the image name is preceded by `%user%`, the image is located in the user level folder (`user\sample01\`). You can use `.bmp` or `.tga` files. Do not exceed an image size of 512x512.

```
TerrainInit id=3
```

Initialize the whole ground with a single ground type.

```
TerrainLevel id=10;11;10;11;12;13 min=0 max=99 slope=0.0 freq=100
```

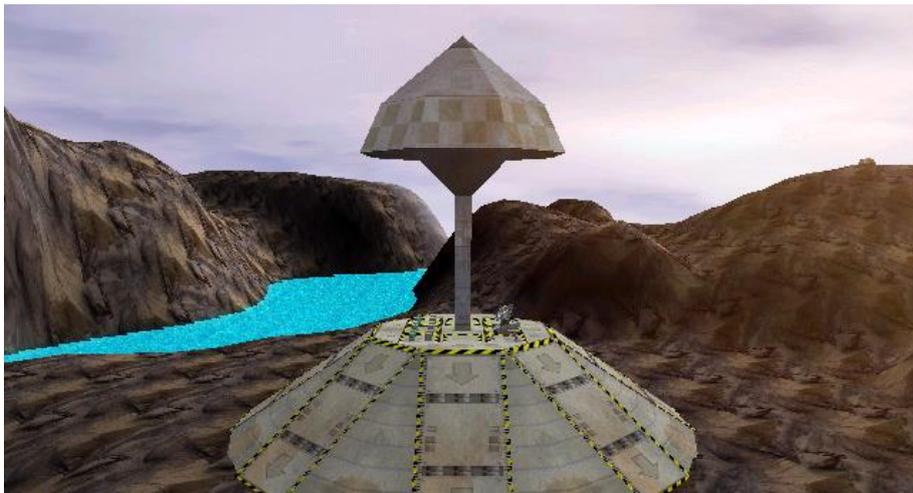
Apply a ground type on the ground, according to certain criteria.

- `id` ground type to be used.. If more than one ground type id is given, they will be chosen randomly. If you repeat an id you can vary the proportions of the ground types to be used. Example `id=10;10;10;11` uses three times more ground type 10 than 11.
- `min` and `max` determine the minimal and maximal height where the ground type can be used. You must give the absolute height (not the height above the water/lava level).
- `slope` determines the slope on which the ground type should be applied or not. For example, `slope=9` will apply the ground type only if the ground is almost flat which is useful for snow as snow won't stay on steep slopes. On the other hand negative slope values allow to put the ground type only if the slope is steep.
- `center` determines the center of an optional circular zone. If `center` and `radius` are defined, the ground type is only applied to the circular zone defined by these two values
- `radius` determines the radius of the optional circular zone.
- `freq` allows you to apply the ground type only on a given percentage of the surface. For example, with `freq=25`, only 25 % of the surface (corresponding the `min`, `max`, `slope`, etc.) will be covered with the ground type.

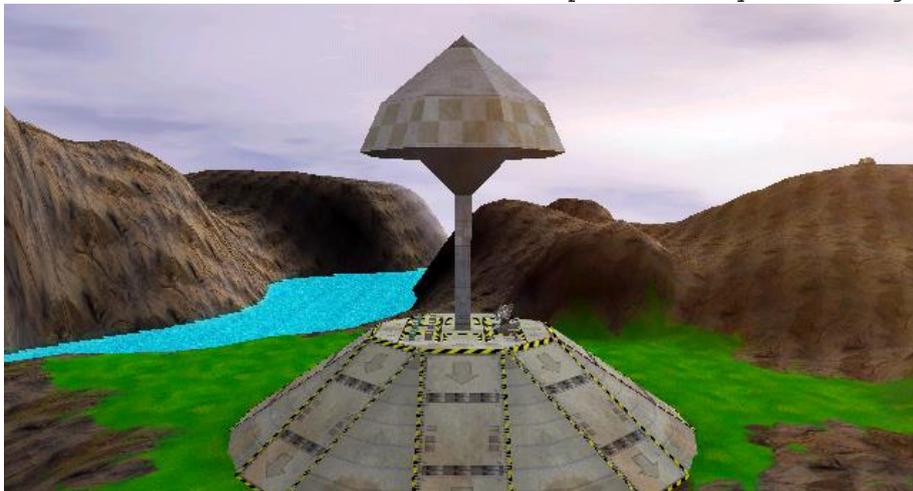
You can use several `TerrainLevel` commands which will be used in the given order. Each `TerrainLevel` command be s "superimposed" on the previous ground; the ground is so composed of several layers.

Here are some examples of commands on **Terranova** :

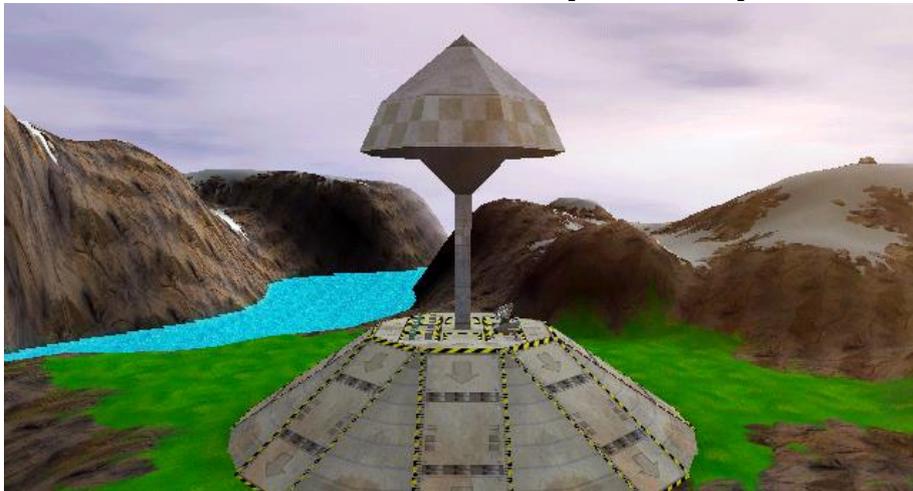
```
TerrainInit id=1 // rock
```



```
TerrainInit id=1 // rock  
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**

This command must end all the Terrain commands and creates actually the ground.

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

With this command you can change the ground color at certain spots. For example you can apply a greenish color to a zone containing many plants. `pos` and `radius` determine the circular zone where the color should be applied.

Examples :

	
	<p>radius=30 color=150;255;150</p>
	
<p>radius=50 color=255;0;0</p>	<p>radius=10 color=255;0;0</p>

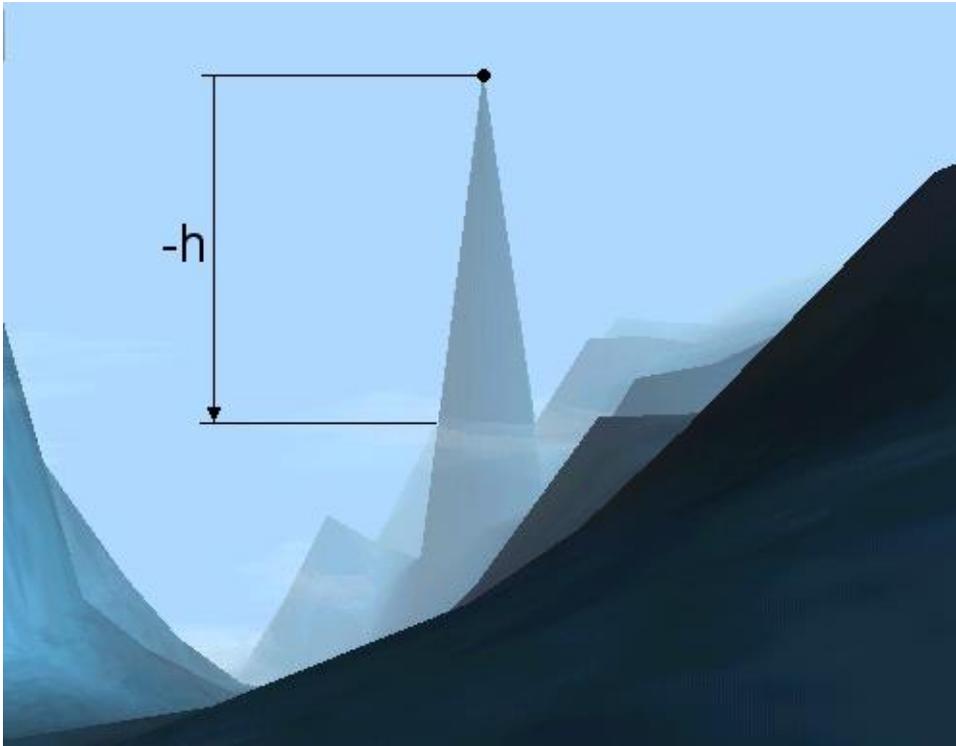
`GroundSpot color=0;128;255 min=-10 max=24 smooth=10`

You can also apply a special color to the banks of lakes, by specifying a minimal and a maximal height.

		
	<p>color=0;255;0 min=-10 max=28.75</p>	<p>color=0;0;255 min=-10 max=28.75</p>

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

Create a horizontal layer of fog. These layers are most visible on Crystalium #4 (The Lost Valley). Height is relative to the ground at `pos`. If the position of the zone is at the top of a peak, `height` can be negative, so that the fog layer surrounds the peak. `delay` is the rotation speed of the layer.



Type	Appearance
0 and 1	Bluish layer.
2 and 3	Red-orange colored ring.
4 and 5	Grey layer.
6 and 7	Yellowish layer.

```
MaxFlyingHeight max=40
```

Maximal absolute altitude (ignoring the water level) at which flying is possible for bots and the astronaut.

### 3.2.5. Objets

In this section various 3D objects such as buildings, plants, trees, astronaut, bots, transportable objects, etc. are created. Some objects are immobile (buildings, plants, ...) where as others are mobile (bots, astronaut).

#### BeginObject

This command starts the Objects section and must be used prior to the first `CreateObject` command.

#### CreateObject pos=7;-10 dir=1.5 type=Me

Create an object, that is a bot, a building, a raw material, a plant, etc. If `position` is on the platform of the `SpaceShip`, the object will land with the spaceship at the beginning of the mission.

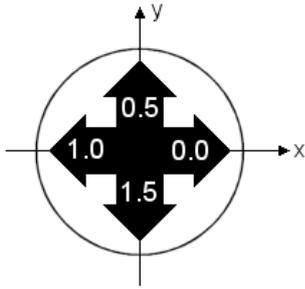
To determine the position of an object, the easiest way is to switch on a special debug mode and then move the astronaut to the desired spot.

To switch on this debug mode hit the keys `Ctrl+Break` and type the command `showstat` and validate with the `Enter` key. Then hit again `Ctrl+Break` and type the command `showpos`. Now the position of the selected object is shown permanently on the screen :



`CreateObject pos=-9.96;47.45 dir=1.7 type=TitaniumOre`

The direction is a number included between 0 and 2 :



List of all possible object types :

**General :**

```
type=Me // astronaut
type=SpaceShip
```

**Bots :**

```
type=PracticeBot // training bot
type=TargetBot
```

```
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
```

**Buildings :**

```
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 // for the training in the flight
type=Target2
type=Houston // mission control

```

**Transportable objects :**

```

type=TitaniumOre
type=UraniumOre
type=Titanium
type=PowerCell
type=NuclearCell
type=OrgaMatter
type=BlackBox
type=KeyA..D
type=TNT // box of explosive

```

**Plants and decorations :**

```

type=Greenery0..4 // small standard plants
type=Greenery5..7 // small clover
type=Greenery10..14 // rising succulent plant
type=Greenery15..19 // fern
type=Tree0..3 // high tree
type=Mushroom1 // harmless mushroom
type=Mushroom2 // corrosive mushroom
type=MegaStalk0..5 // huge strange plant

type=Quartz0..3 // small to big quartz
type=Barrier0 // short barrier
type=Barrier1 // long barrier
type=ApolloLEM // on the moon only :
type=ApolloJeep
type=ApolloFlag
type=ApolloModule
type=ApolloAntenna

```

**Recyclable wrecks or derelicts:**

```

type=WreckBotw1..2 // bot with wheels
type=WreckBott1..2 // bot with small tracks
type=WreckBotr1..2 // bot with big tracks

```

**Ruins :**

```

type=RuinBotFactory
type=RuinDoor // converter door
type=RuinSupport // radar support
type=RuinRadar // radar pedestal
type=RuinConvert
type=RuinBaseCamp // spaceship pedestal
type=RuinHeadCamp // spaceship roof

```

**Enemies :**

```

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

```

**Indicators :**

```

type=PowerSpot // presence of energy in the subsoil
type=TitaniumSpot // presence of titanium in the subsoil
type=UraniumSpot // presence of uranium in the subsoil
type=KeyA..DSpot // presence of key in the subsoil
type=WayPoint // cross for exercises
type=BlueFlag
type=RedFlag
type=GreenFlag
type=YellowFlag
type=VioletFlag
    
```

**Miscellaneous :**

```

type=Mine // fixed mine
type=Portico // crane (on the earth)
type=Bag // survival kit
type=Home // small nice house (on terranova)
type=Tech // Houston engineer
type=Firework
    
```

CreateObject can contain more parameters :

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

Name of the CBOT program to be loaded into index 1 in a bot or an insect. You can load up to 10 different programs by using the commands `script1` to `script10`. To make reference to a general program placed in the folder `script\` (for example for an insect), just omit `%user%\` before the name of the script.

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

Index of the program to be executed when the mission starts. This is useful for enemies or for a bot that should do something else than just stand around.

```
CreateObject ... select=1
```

One and only one object can have parameter `select=1`. This object will then be selected at the start of the mission and the camera will aim at it.

```
CreateObject ... power=0.5
```

Specify the initial level of a bot's the power cell. Following values are possible :

Value	Meaning
-1	The bot has no power cell at all.
0	The bot has a normal empty power cell.
0..1	The bot has a more or less filled normal power cell.
1..100	The bot has a more or less filled nuclear power cell.

```
CreateObject ... range=100
```

Autonomy of flight. Larger values allow longer flights before reactor overheating occurs. The default value is 30. Bots constructed *during* the mission always have the default autonomy of flight.

```
CreateObject ... shield=1
```

Initial strength of the shield. 1 corresponds to a totally effective shield. 0 corresponds to an almost destroyed shield; at the next hit, the bot will be destroyed.

```
CreateObject ... magnifyDamage=2
```

Amplifies or to reduces the damages when the bot gets hit by enemy fire. Bots constructed *during* the mission always have the default resistance.

Value	Effect
0.5	Doubled resistance.
1	Default resistance.
2	Doubled damages.

```
CreateObject ... proxyActivate=1 proxyDistance=10
```

The bot or the building is not accessible, neither on the mini-map, nor with icons at top of the window. The object must first be "found" by approaching it to a certain distance. These commands are useful for "abandoned" objects of previous expeditions. `proxyDistance` is the distance the object must be approached in order to "find" it.

```
CreateObject ... selectable=0
```

The object cannot be selected. Useful for robots that just drive around or for certain exercises.

### 3.2.5.1. The command line

The command line allows to send arguments to a program. Let's create an ant create with :

```
CreateObject pos=-55;243 cmdline=-55;243;-62;234 dir=0.0 type=AlienAnt
script1="ant04.txt" run=1
```

The program `ant04.txt` will receive 4 arguments -55, 243, -62 and 234. These arguments can be used within the program. Here the parameters are two coordinates (x;y) to indicate two spots between which the ant will go forth and back.:

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

The variable `nav1` will be (-55;243) and `nav2` be (-62;234). With the command line you can use the same program for different behaviors according to the arguments. You can use up to 10 arguments in a program.

### 3.2.5.2. Spaceship

```
CreateObject pos=0.00;0.00 dir=0.0 type=SpaceShip run=1
```

The `SpaceShip` does not contain any CBOT programs. But the command `run` can take the following values :

Command	Action at the beginning of the mission
<code>run=0</code>	The spaceship has landed and its doors are opened
<code>run=1</code>	The spaceship lands, then its doors are opened
<code>run=2</code>	The spaceship is brought with the crane (Earth #3)
<code>run=3</code>	Special mode for <code>win</code> and <code>lost</code>
<code>run=11</code>	Interstellar journey and then landing

During an interstellar journey, you can choose the visible planets with the `Planet mode=1` command (see command `Planet` in section 3.2.2).

### 3.2.5.3. *Egg*

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

An egg gives birth to an enemy after a certain time. This is very practical when numerous enemies should launch attacks by successive assaulting waves. As long as the egg has not hatched, the enemy does not physically exist, and so doesn't overload neither the memory nor the processor.

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

Delay in seconds before the hatching occurs.

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

Type of the insect that will hatch.

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

CBOT program that will be executed within the insect. The name can be preceded by %user%, as usual.

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

Indicate an active egg; an insect will hatch after a certain time. If you put `run=0` the egg will never hatch.

It is possible to send arguments in the program of the insect with `cmdline` like within the `CreateObject` command (see section 3.2.5.1). The arguments will be sent to the program of the insect during hatching.

### 3.2.6. *Lighting*

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

Create a omnidirectionnal light source. Different light sources light the ground and the objects. `dir` is a vector of direction ( $x; z; y$ ). The amplitude of the vector is ignored. Most lights sources have a -1 z component, so the light goes from top to bottom ( $x; -1; y$ ).

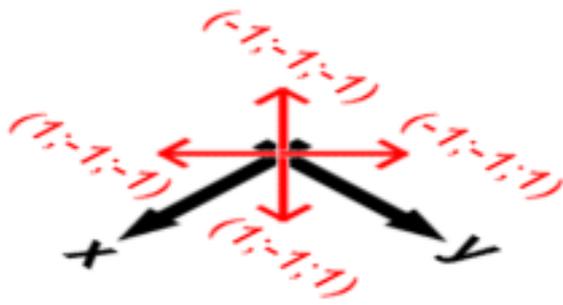
`color` determines the color of the light given by its three components red, green and blue included generally between -1 and 2 :

Value	Effect
-1.0	A negative value actually absorbs light (does not exist in reality).
0.0	No light at all
1.0	Normal lighting.
2.0	Overexposure.

`Type` can take the following values :

Type	Effect
Terrain	Light the ground.
Object	Light objects (bots, buildings, plants, etc.).
Quartz	Light quartz (for example on the planet <b>Crystalium</b> ) with a slow movement to simulate reflections.

The table below shows different Object lighting types on a robot :



<pre>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</pre>	<pre>dir=-1;-1; 1 <b>color=2.0;0.0;0.0</b> 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</pre>	<pre>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 <b>color=2.0;0.0;0.0</b> dir= 1;-1;-1 color=0.2;0.2;0.2</pre>
<pre>dir= 1;-1; 1 color=0.3;0.3;0.3</pre>	<pre>dir=-1;-1; 1 color=0.6;0.6;0.6</pre>	<pre>dir=-1;-1;-1 color=0.3;0.3;0.3</pre>

```
WaterColor color=-0.6;-0.1;-0.1
```

Modify the color under water. The values above give a dark blue tint. The three components are added to all lights created with CreateLight when the camera is under the water.

For example with color=-0.6;-0.1;-0.1, the colors under water are modified like this :

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
```

This rarely used command puts a spotlight light on the ground. You should rather use the GroundSpot command.

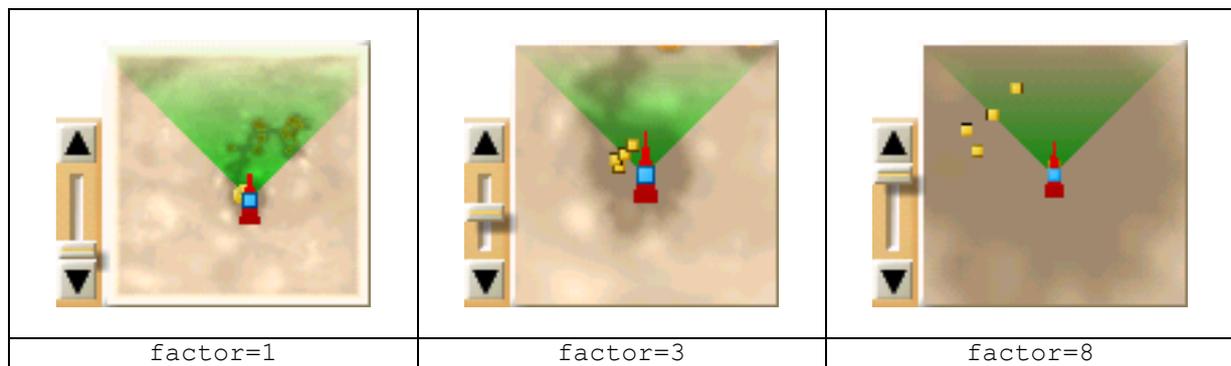
### 3.2.7. Mini-map

```
MapColor floor=104;185;205 water=154;235;255
```

Colors to be used in the mini-map for ground and water.

```
MapZoom factor=4
```

Default zoom factor for the mini-map. Possible values are included between 1 (the whole map is visible map) and 8 ( the strongest zoom).



### 3.2.8. Options

Options that determine what it is possible to make in the exercise or the mission.

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

Bots placed initially in a mission get programs with the command `CreateObject script1="name"`. If bots should automatically get programs after they have been built in a `BotFactory` you must use this command.

- `name` is the name of the file containing the program.
- `type` determines the type of bot which get the program. The type `All` allows to load a program in any constructed bot type.

```
EnableBuild type=PowerStation
```

List of buildings which can be built by the astronaut. Type can take one of the following values :

ResearchCenter	Research center
BotFactory	Bot factory
Converter	Converter
PowerStation	Power station
RadarStation	Radar station
RepairCenter	Repair center
DefenseTower	Defense tower
PowerPlant	Power cell Factory
Derrick	Derrick
NuclearPlant	Nuclear power plant
AutoLab	Laboratory of analysis
PowerCaptor	Lightning conductor
ExchangePost	Information exchange post

List of available special actions for the astronaut :

FlatGround	Can verify if the ground is flat
Flag	Can put or to remove flags



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

EnableBuild type=FlatGround
EnableBuild type=Flag
```

In the example above, the `NuclearPlant` button does not appear, because the research has not yet been conducted.

```
EnableResearch type=WINGER
```

List of researches that can be conducted with a `ResearchCenter` :

TRACKER	Tracked bots*
WINGER	Winged* bots
THUMPER	Thumper bots
SHOOTER	*Shooter bots
TOWER	DefenseTower building
HAZER	PhazerShooter bots
SHIELDER	Shielder bots
ATOMIC	NuclearPlant building

List of researches that can be conducted with an AutoLab :

iPAW	Legged* bots
iGUN	*OrgaShooter bots

```
DoneResearch type=WINGER
```

List of researches that have already been conducted at the beginning of the mission.

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

Exemples

- DoneResearch type=SUBBER

Use this to be able to make a submarine bot SUBBER in a BotFactory.

- DoneResearch type=WINGER
- DoneResearch type=SHOOTER

Use these two commands to be able to make a WingedShooter in a BotFactory.



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

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

In the example above, the researches TRACKER, SHOOTER and TOWER have already been conducted. Research ATOMIC still has to be done.

### 3.3. Ending conditions

```
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
```

Criteria to determine at which moment the mission is ended or if the spaceship can takeoff. You must use one line for each object type you want to test.

For example, if the 2nd line decides that the mission is lost and the 3rd one decides that it is won, the mission will be "lost". See also command `EndingFile` in section 3.2.1.

The example above means :

The mission is lost if *one* of the following conditions is true :

- The number of astronauts is 0  
(`EndMissionTake pos=0;0 dist=1000 type=Me lost=0`)
- The number of WheeledGrabber objects is 0  
(`EndMissionTake pos=0;0 dist=1000 type=WheeledGrabber lost=0`)

The mission is won or the spaceship can take off if *all* of the following conditions are true :

- There are at least 4 chunks of titanium.  
(`EndMissionTake pos=0;0 dist=1000 type=Titanium min=4`)
- There are exactly zero ants.  
(`EndMissionTake pos=0;0 dist=1000 type=AlienAnt min=0 max=0`)

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

Central position and radius of the circular zone in which the presence of objects is tested.

`Pos=0;0 dist=1000` means that objects are tested in the whole map. This command is useful to test if certain items are on the spaceship.

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

Object type to be tested (see `CreateObject` command for the list of possible types).

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

If the number of objects reaches this lower limit the mission is immediately lost.

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

If the number of objects is lower than this limit, it is impossible to takeoff (and so to end mission). This is useful, for example, to verify that a certain quantity of a resource has been produced.

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

If the number of objects is higher than this limit, it is impossible to takeoff (and so to end mission). This is useful to verify that all the enemies have been eliminated.

### 3.4. Initial camera position

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

At the beginning of the mission the camera aims at the selected object (`CreateObject select=1`). With this command you can choose another position direction. The camera will then travel (with a speed according to `delay`) towards the given position and direction.

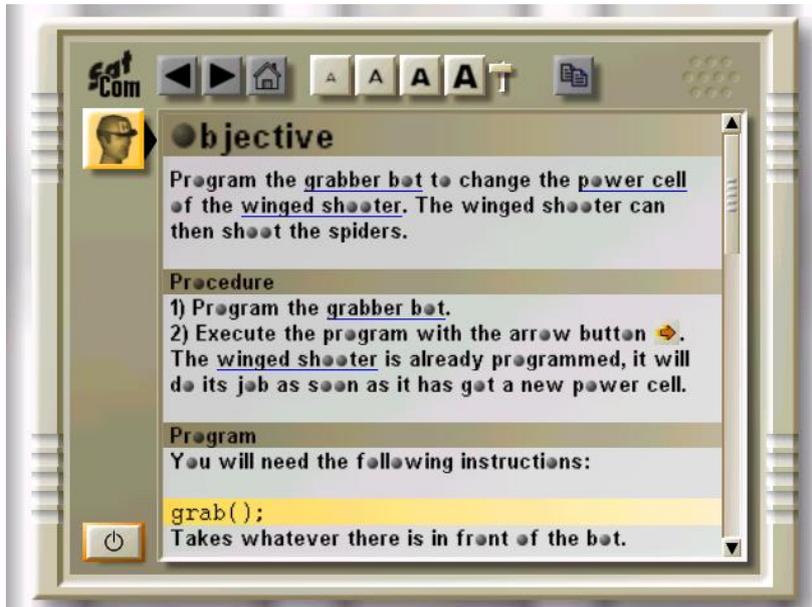
## 4. Formatting text for *SatCom* messages

The files containing official Colobot Help texts and instructions shown in the **SatCom** are located in the `help\` folder. These texts contain special formatting commands which begin with a back slash "`\`" and end by a semicolon "`;`". A paragraph is delimited by an end of line "`\n`".

Here is an example of a source text :

```
\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.
\n
\n;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.
\n
\n;t;Program
You will need the following instructions:
\n;c;
\n;s;grab();\n;
\n;Takes whatever there is in front of the bot.
\n;c;
```

And this will be shown on the **SatCom** :



`\b;`

Big title with brown background. This command must be at the beginning of the line and ends automatically at the end of the line.

`\t;`

Subtitle with brown background. This command must be at the beginning of the line and ends automatically at the end of the line.

`\s;`

Subtitle with yellow background. This command must be at the beginning of the line and ends automatically at the end of the line. It is mostly used for displaying CBOT programs.

```
\tab;
```

Table with orange background, used for satellite reports. This command must be at the beginning of the line and it ends automatically at the end of the line.

```
\l;hypertext\u file;
```

Hypertext link. The word surrounded by `\l;` and `\u;` is the link name that is displayed. The `\u;` command indicates the file to which the link points. For example, `\u object\botgr;` points to the standard file `help \object\botgr.txt`. Another example: `\u %user%\mlink;` points to the user level specific file `user\sample01\mlink.txt`.

```
\c;
```

Use the Courier fixed pitch font, used for CBOT programs.

```
\n;
```

Use the normal Colobot font.

```
\image %user%\filename dx dy;
```

Show a .bmp image contained in the user mission specific folder. If the `%user%` is omitted, image will be fetched from the general folder `diagram\`.

**Example:** `\image %user%\sample01 8 8;` displays `user\sample01\sample01.bmp`.

**Example:** `\image sniff1 12 12;` displays `diagram\sniff1.bmp`.

`dx` is width in characters, and `dy` is height in characters. The image is stretched to fit into the specified size. The values 12 12 used here result in a square image.

```
\token;
```

Use an orange background. Used generally for instructions of the CBOT language. The command `\norm;` returns to standard background.

```
\type;
```

Use a green background. Used generally for variable types of the CBOT language (for example `int`, `float`, `point`, etc.). The command `\norm;` returns to the standard background.

```
\const;
```

Use a red background. Used generally for CBOT language constants. The command `\norm;` returns to the standard background.

```
\key;
```

Use a grey background. Used generally for key names as F1, F2. The command `\norm;` returns to the standard background.

```
\norm;
```

Return to the standard background after following commands: `\token;`, `\type;`, `\const;` `\key;`.

```
\key name;
```

Display a key name. For example `\key help;` shows the name of the key which activates the **SatCom**, generally F1. But if the user has modified the key mapping in the Colobot options, the new name will be shown. Never to write F1 directly but rather use `\key help;` instead.

The following key names exist :

Name	Default key	Action
Left	Left arrow	Turn to the left
Right	Right arrow	Turn to the right
Up	Up arrow	Move forward
Down	Down arrow	Move backward
Gup	Shift	Go up
Gdown	Ctrl	Go down
Camera	Space bar	Change the viewpoint of the camera
Desel	numpad 0	Select the previous object
Action	Enter	default action for the selected bot
Near	numpad +	Approach the camera
Away	numpad -	Move back the camera
Next	Tab	Select the next bot or building
Human	Home	Select the astronaut
Quit	Esc	Quit action
Help	F1	General instructions on the <b>SatCom</b>
Prog	F2	CBOT language help on the <b>SatCom</b>
Cbot	F3	Help on current instruction on the <b>SatCom</b>
Visit	numpad .	Show error location
speed10	F4	Speed x1.0
speed15	F5	Speed x1.5
speed20	F6	Speed x2.0

As keys are generally shown with a gray background, you should use the combination of following commands to indicate a key name :

```
\key;\key help;\norm;
```

## 5. Problems

To resolve certain problems, you may modify the `colobot.ini` file which is located in

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

Do not add new lines, but simply modify existing values. Do not to insert spaces. You must leave COLOBOT prior to modifying this file.

If vegetation displays poorly, or even not at all, you may have set to zero the number of ornamental objects in the options. To put back 100% :

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

If vegetation is still not displayed try :

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

or

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

If a square appears on the ground around shadows, try :

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

or

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

If this does not work you can remove all shadows with :

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

When an object is between the selected object and the camera, it normally becomes transparent. If the object is not transparent enough, try :

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

or

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

## 6. Fine tuning

In order to allow you to find the best settings for your computer, you can instruct the computer to display the current frame-rate. Hit the keys `Ctrl+Break`, and type the command "showstat" and validate with the `Enter` key :

- `Ctrl+Break showstat Enter`

The frame-rate and some other parameters will then be displayed on the upper part of the screen, for example :

- 32.46 fps T=11558 (640x480x16)

The first number is frame-rate (`fps` = "frames per second"), i.e. the number of images per second displayed. The second number is the number of triangles displayed in the scene. The three numbers in brackets indicate the display resolution and the number of bits per pixel.

All options are saved in the file `colobot.ini`. You can edit this file in any text editor, for example Notepad.

## 7. Disclaimer

EPSITEC AND/OR ITS SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THIS DOCUMENT AND RELATED GRAPHICS. THIS DOCUMENT AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. EPSITEC AND/OR ITS SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL EPSITEC AND/OR ITS RESPECTIVE SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION AVAILABLE IN THIS DOCUMENT. THIS DOCUMENT AND RELATED GRAPHICS COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS.

## 8. Development team

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

### 8.1. *Beta tester core team*

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

### 8.2. *Developer*

EPSITEC SA  
Mouette 5  
CH-1092 Belmont

colobot@epsitec.ch  
www.colobot.com