COLOBOT

COLOBOT

1 Introduction

COLOBOT combines both a real time strategy game and an initiation to programming. You are in charge of a space expedition withonly a few robots to assist you. Your mission consists in successive attempts at the exploration and colonization of various planets.

1.1 The story

Life on earth is threatened by a devastating cataclysm. Mankind has to move out and search for a new home.

A first expedition of only robots was sent to find another habitable planet. For reasons yet unknown, the mission was a disaster and was never to return.

Take up the challenge to succeed where robots failed...

With a few robots for only companions, you will travel to planets unknown. Houston, Earth Mission Control as well as a spy satellite will transmit valuable information to you, but they are so far away …

You'll need to build the infrastructure necessary to gather raw materials and energy supplies. You'll need to produce the weapons necessary to defend yourself.

Then comes the time when you'll have to fight for your survival. Grab the controls of a robot… try to be faster than your opponents ... don't let yourself be overwhelmed. Sometimes there are just too many of them. It all explodes and the fight is over.

Your only escape lies in the programming of your robots in order to hand some tasks over to them. This is the only way you can send four programmed robots to fight a swarm of wasps on the attack while controlling a fifth one to get those that might have gotten through ...

A complete library of ready-to-use programs is at your disposal and Houston will occasionally transmit newly developed ones tailored to the specific needs of a given mission. However, these programs may not always be precisely adapted to your requirements, in which case you can modify them or even discard them altogether and write whole new ones.

The C-Bot programming language used is similar in structure and syntax to C and Java™. The skills you can acquire on this job could be of invaluable use in your professional career ... that is once you get back to your life as a civilian.

1.2 Minimum requirements

- 300 MHz Processor
- 64 MB RAM
- 3D accelerator board with 16 MB RAM
- Voodoo boards are not supported
- 100 Mb of free disk space
- Windows® 95, Windows® 98, Windows® ME or Windows® 2000

1.3 Installation

- Insert the COLOBOT CD-ROM into your drive, wait and follow the instructions.

If nothing happens after you have inserted the CD-ROM:

- Double click on **My Computer**
- Double click on your CD-ROM drive
- Double click on **install.exe** and follow the instructions
- The Colobot installation program may ask you if you want to install DirectX® 8a.

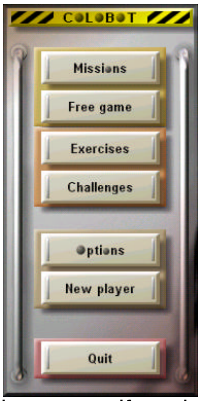
If you have already installed another version of Colobot, a new installation will delete your saved Colobot games.

To play Colobot the CD-ROM must be in the drive.

1.4 Uninstallation

- Double click on **My Computer**
- Double click on **Control panel.**
- Double click on **Add/Remove programs**.
- Double-click on **COLOBOT** in the list.

2 Main menu



2.1 Missions

The missions are the core aspect of COLOBOT. There are 36 missions, taking place on 9 different planets. They must be completed successively in a fixed order.

2.2 Free game

In the free games, you can play freely without any definite objective. Only the planets already visited in the missions and the research already performed are available.

2.3 Exercises

In this part of COLOBOT you can learn how to program the bots, even if you know nothing about programming. The exercises are structured in chapters, and represent a progressive introduction to the basic concepts of programming in a structured and object-oriented language. You can do the exercises in any order, but we advise you to start with the easier ones.

2.4 Challenges

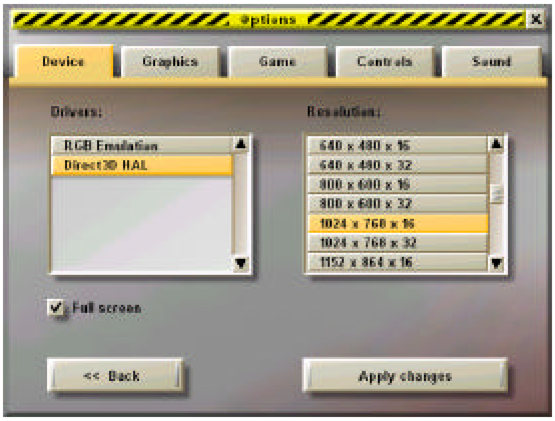
The challenges allow you to apply the skills acquired in the programming exercises, with fewer explanations than in the exercises. Challenges allow you to check if you fully understand the concepts taught in the exercises.

2.5 User

This is for playing user designed missions that can be downloaded from www.colobot.com or that have been designed by yourself. This button is only available on version 1.7 and up. If you don't have version 1.7 or better you can download an upgrade from www.colobot.com. You'll also find lots more documentation on the site.

2.6 Options

2.6.1 Device



Drivers:
RGB Emulation
Direct3D HAL

Resolution:
640 x 480 x 16
640 x 480 x 32
800 x 600 x 16
800 x 600 x 32
1024 x 768 x 16
1024 x 768 x 32
1152 x 864 x 16

✓ Full screen

Drivers:

Better choose a driver with the attribute HAL (Hardware Abstraction Layer), and avoid drivers with the attribute "Emulation" or "T&L".

Resolution:

The first and the second number indicate the number of pixels of the screen on the horizontal and vertical axis. The third number indicates the number of bits used to encode colors: 16 bits allow displaying 65,000 colors, whereas 32 bits allow displaying 4 million colors. The higher you set the parameters, the more detailed the scene will be represented on the screen.

However, if the frame-rate is too low (jerky movements), try a lower display quality. Start with 640 x 480 x 16 and climb slowly higher; most modern graphic boards support at least 1024 x 768 x 16. Just try out the best compromise for your computer.

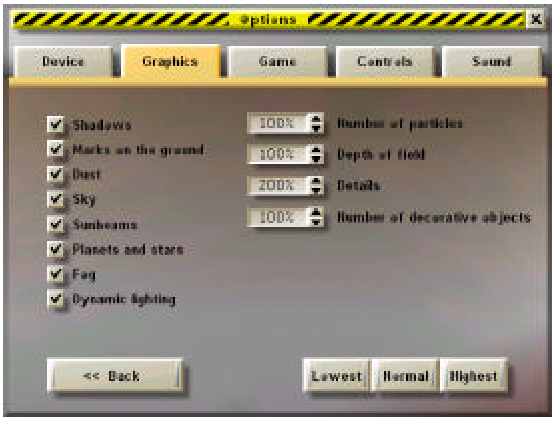
✔ Full screen

Normally COLOBOT runs in full screen mode, whatever resolution you choose. If you deactivate this flag, COLOBOT will be displayed in a fixed size window of approximately 640 x 480 pixels.

[Apply changes]

Hit this key for the changes on this page to be applied.

2.6.2 Graphics



✓ Shadows
✓ Marks on the ground
✓ Dust
✓ Sky
✓ Sunbeams
✓ Planets and stars
✓ Fog
✓ Dynamic lighting

100% Number of particles
100% Depth of field
200% Details
100% Number of decorative objects

✔ Shadows

Normally, all objects (bots, buildings, raw material) project a shadow on the ground. With some older graphic boards, a gray square may appear around the shadow. In this case, you can deactivate the displaying of the shadows.

✔ Marks on the ground

The marks on the ground show up the slopes or flat surfaces. The ground can take a different colors. If you switch off this option, the flat zones around the astronaut cannot be displayed (see chapter 4.1.1).

✔ Dust

Dirt and dust give a more realistic, dirty aspect to bots and buildings.

✔ Sky

If you activate this option, the sky on some planets will show beautiful clouds pushed by the wind. If your frame-rate is too low, deactivate this highly time-consuming option.

✓ Sunbeams

Shows realistic sunbeams in the sky.

✓ Planets and stars

On some planets, nearby planets move around in the sky, and numerous stars adorn the firmament.

✓ Fog

This option allows you to deactivate the fog near the ground.

✓ Dynamic lighting

Dynamic lighting appears when there's an explosion or close to power stations.

Number of particles (0% - 200%)

Particles are used to simulate dust, smoke, fire, explosions, etc.

Depth of field (50% - 200%)

The field depth determines up to what distance the scene is displayed. This distance varies from planet to planet. A high value (for example 200%) allows you to see very far, but requires a powerful 3D graphic board.

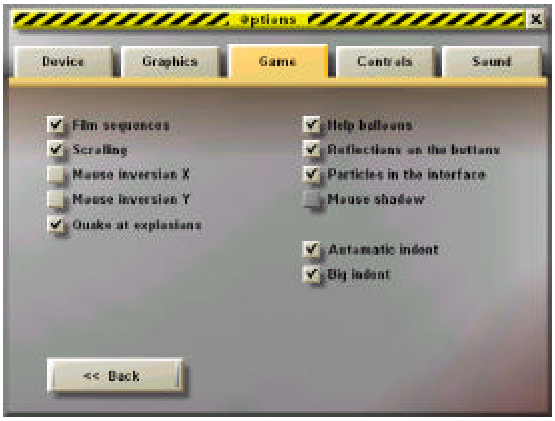
Details (0% - 200%)

Sets the visual quality of 3D objects according to their distance.

Number of decorative objects (0% to 100%)

Set the number of purely ornamental objects like plants, trees, crystals, etc.

2.6.3 Game



- ✓ Film sequences
- ✓ Scrolling
- ☐ Mouse inversion X
- ☐ Mouse inversion Y
- ✓ Quake at explosions
- ✓ Help balloons
- ✓ Reflections on the buttons
- ✓ Particles in the interface
- ☐ Mouse shadow
- ✓ Automatic indent
- ✓ Big indent

✓ Film sequences

Small film sequences appear at the beginning and at the end of most missions. With the Esc key, you can always stop these sequences. If you want these sequences never to appear, deactivate this option.

✓ Scrolling

When the camera is behind the astronaut or a bot, the screen scrolls if the mouse gets close to the edge of the screen.

✓ Mouse inversion X

Inverts the scrolling direction when the mouse gets close to the left or the right edge of the screen.

✓ Mouse inversion Y

Inverts the scrolling direction when the mouse gets close to the upper or lower edge of the screen in the program editor.

✓ Quake when there's an explosion

Explosions or unskillful landings will shake the camera. Deactivate this option in order to keep the camera always stable.

✓ Help balloons

The help balloons appear when the mouse stops on a button or object and display a small text that explains the item.

✓ Reflections on the buttons

The reflections appear when the mouse flies over a button.

✓ Particles in the interface

Sparks and steam clouds appear when you move the mouse over the user interface.

✓ Mouse shadow

When you wish the mouse to have a shadow, the mouse is managed by COLOBOT, whereas the mouse without shadow is managed by Windows ®. When COLOBOT is executed in a window, the mouse cannot have a shadow.

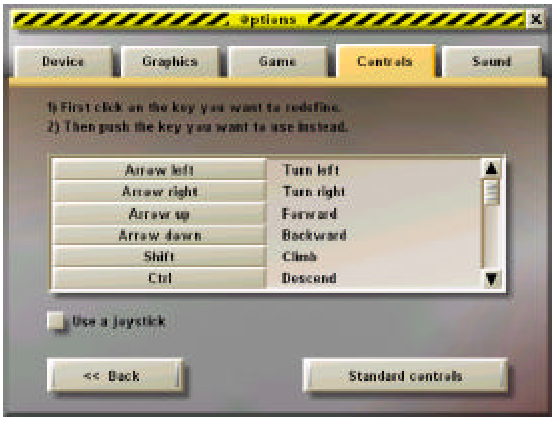
✓ Automatic indent

The automatic indent moves the text in the programming editor to the right according to the number of braces { and }.

✓ Big indent

The big indent moves the text four spaces to the right at every brace. Otherwise, the indent is two spaces.

2.6.4 Controls



1) First click on the key you want to redefine.
2) Then push the key you want to use instead.

Arrow left	Turn left
Arrow right	Turn right
Arrow up	Forward
Arrow down	Backward
Shift	Climb
Ctrl	Descend

Try to use your left hand for the arrow keys of your keyboard and the right hand for the mouse. Use the mouse for firing and for moving the cannon.

Arrow left, right, up and down

Move the astronaut left, right, forward or backward. In the programming exercises, the bots cannot be moved with these keys.

Shift and **Ctrl**

These keys control the jet of the astronaut and the bots. Shift increases the power of the jet (climb); Ctrl decreases the power (descend). On some planets, flying is impossible.

Enter

This key starts the main action of the selected bot (grab/drop, fire, sound, etc.), equivalent to the red-framed button.

Space bar

Changes the viewing angle of the camera. For most bots, this control switches between a rear sight and the onboard camera.

. (num pad)

Pauses the game and shows where the last message displayed on top of the screen was sent from. If several messages were displayed, a new hit on the key will show the previous message, and so forth. Hit the Esc key to get back to the game.

Tab

Selects the next object, according to the order of the buttons on top of the screen.

Home

Selects the astronaut.

0 (num pad)

Selects the bot or building that was previously selected.

+ and **-** (num pad)

Moves the camera closer or further away from the selected object.

F1

Displays the instruction for the current mission or exercise on the **SatCom**.

F2

Displays the general help about programming on the **SatCom**.

F3

During program edition, this key displays help about the current instruction.

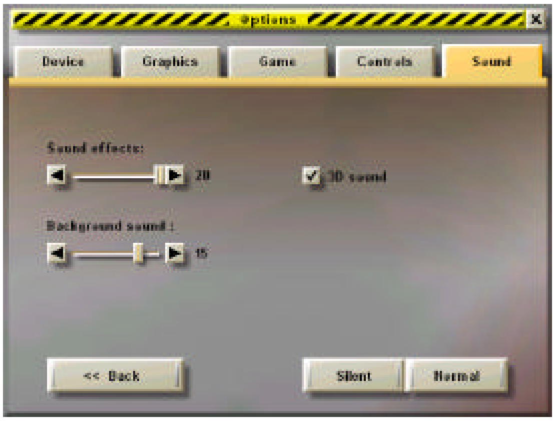
F4, F5 and F6

Selection of game speed. F4 sets normal speed (x1). F5 (x1.5) and F6 (x2) allow you to increase the speed of the whole game, in order to cut down waiting time.

Esc

Interrupts the current mission, and displays a menu where you can choose to quit the mission, save it, load a saved game, restart the mission, or set the options.

2.6.5 Sound



Sound effects: 20

✓ 3D sound

Background sound : 15

Sound effects:

The game sound is generated by the current action, i.e. motor, voice and shooting sounds.

Background sound:

The background sound depends on the planet. Every background sound corresponds to a soundtrack on the CD. They are made of a specific musical atmosphere and are independent of the current action. On the Moon and in the exercises, there is no background sound.

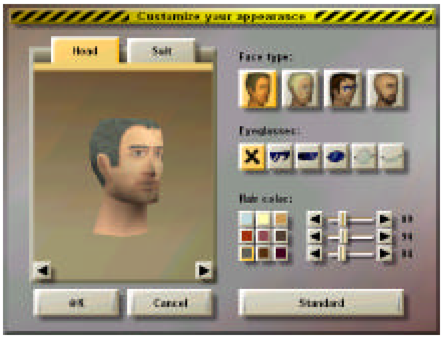
✔ 3D sound

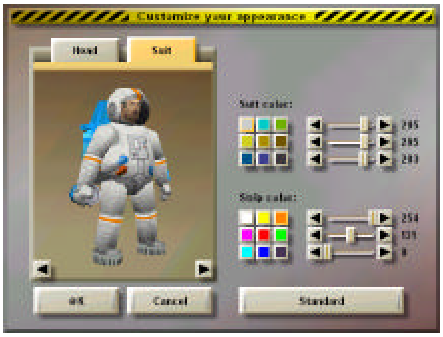
Some soundboards allow the positioning of sound in space with 4 loudspeakers, increasing the realism of the sound effects. If your soundboard does not support this possibility, the button is gray.

2.7 *New player*

Once you have installed Colobot on a computer, several different persons can play. For each player the progression in the game as well as written programs are stored separately.

Each player can also customize the look of the astronaut:



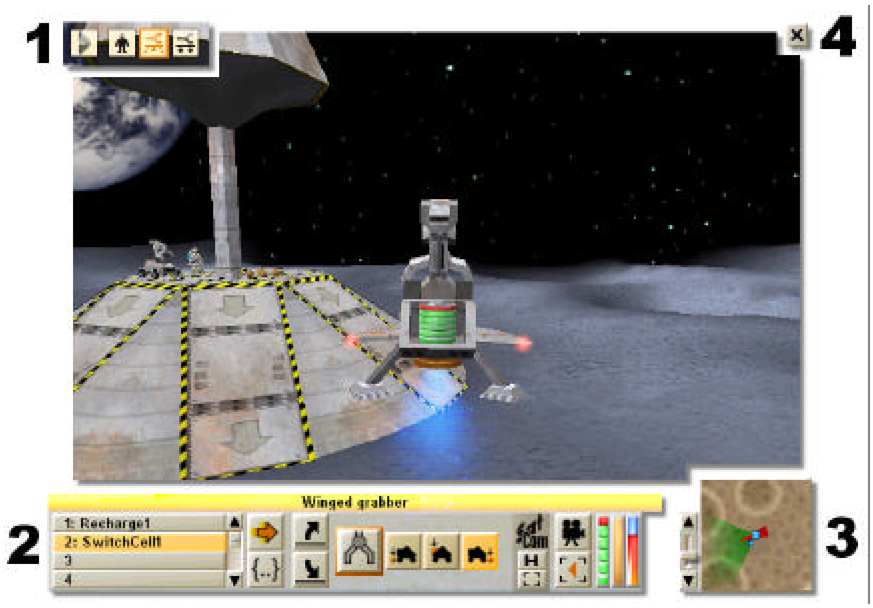


You can customize following elements:

- Face
- Glasses
- Hair color
- Space suit color
- Color of the strips on the space suit

3 The screen

During a mission or an exercise the screen looks like this:



1 Existing robots and buildings
2 Possible actions for the selected object
3 The mini map
4 Show the menu

3.1 *Existing robots and buildings*

Shows all existing robots and buildings in the mission. The selected object appears in orange. If a program is running in a robot, the frame of its button will blink.

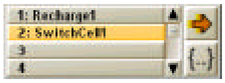
You can click on one of these buttons in order to select the object. You can also press the «Tab» key to select the next object.

The first button allows you to display robots or buildings.

3.2 *The selected object*

The lower left part of the screen s hows all possible actions for the selected object.

3.2.1 The ten programs

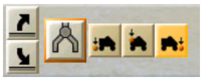


Each robot can store up to 10 programs. Select the program you want to load in the list on the left. Use the scroll bar to access to programs 5 to 10.

Launch or stop the execution of the selected program.

Launch the built-in program editor for the selected program (see chapter 5.1). During program edition, the game pauses so you have all the time you need to modify programs.

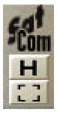
3.2.2 Actions



This part depends on the selected object.

Usually one of these buttons is always bigger than the other ones. This means that you can press the « Enter » key on your keyboard rather than clicking on the button.

3.2.3 The SatCom

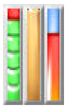


The first button activates the **SatCom** and displays directly the goal of the mission as well as some explanations. The second button activates the **SatCom** and displays directly an explanation about the selected object.

3.2.4 The camera

You can change the viewpoint of the camera by clicking on this button or by pressing the space bar of your keyboard. Usually this switches between a rear sight and the onboard camera.

3.2.5 Gauges



Shows the state of the selected object. From left to right: energy level of the power cell, the shield level and the reactor temperature (only for flying robots and the astronaut).

3.2.5.1 Energy level

This gauge shows the energy level of the power cell. The green part represents the remaining energy. When the gauge is completely red, the power cell is completely empty or there is simply no power cell on the selected object. As soon as there is less than 10 percent of energy available, an audible alarm starts ringing.

3.2.5.2 Shield level

This gauge displays the object's shield level. As soon as the shield level falls to zero, the object will be destroyed if it is hit. The green balls fired by ants will destroy the shield very quickly. A violent collision also affects the shield as well as the shield of the object you bumped into.

You can regenerate a robot's shield of a by bringing it onto a repair center. The only way to regenerate the shield of a building is to bring a shielder robot close to the building. As soon as a building is inside the blue protection zone of the shielder, its shield will be quickly regenerated.

The astronaut's shield automatically regenerates slowly. There is no way to accelerate this process, even if you bring the astronaut onto a repair center.

3.2.5.3 Reactor temperature

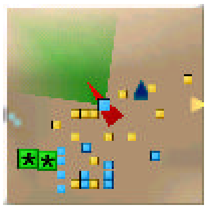
This gauge shows the temperature of a robot's or the astronaut's reactor. When overheating occurs, the reactor fails and the astronaut or the robot will fall to the ground. As soon as the temperature is close to the overheating temperature, an audible alarm starts ringing. Pay attention to this gauge if you fly a robot over water as it will be destroyed if it hits water.

If the ambient temperature on a planet is very high, overheating occurs faster.

3.2.6 Demolition

Buildings have a button that allows you to perform auto destruction.

3.3 The mini map



The mini map on the lower right corner of the screen gives you an overview of the situation. The darker the zones the lower is the altitude. Robots are yellow, buildings are blue and enemies are green.

Enemies are displayed only if there is a radar. When you pass over an object on the mini map, the name of the object is shown in a small popup window. You can click on an object on the mini map to select it.

The slider on the left side of the map allows you to zoom in or out.

3.4 The menu

If you click on this button on the upper right corner of the screen, the menu is displayed:

Continue	Continue the current mission.
Save	Save the current mission

Load	Load a previously saved mission. The current mission will be aborted.
Options	Access to certain game options. You don't have access to all options from here. If you want to have access to all options use the Options button of the main menu (see chapter 2.6).
Restart	Abort the current mission and restart from the beginning.
Abort	Abort the current mission.

4 Objects

4.1 *Astronaut*

You are the astronaut. If you die, the mission immediately fails.

In most missions, the survival kit allows you to fly, thanks to the reactor, or to swim and dive without time limit.

You can carry most objects, but the weight will slow down your movements considerably. Flying, swimming and walking underwater is impossible if you are carrying an object. It is better to use a grabber robot or a subber robot. You cannot carry uranium ore because of the risks due to radioactivity.

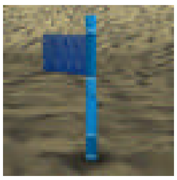
4.1.1 Neutron gun

With the neutron gun holstered to your survival kit you can construct various buildings depending on the mission and the research already conducted with the research center.

A special button allows you to ensure that the ground around you is flat. This is important since building is only possible on a flat surface. A flat surface is displayed in green while a slope is red.

Note: If the « Marks on the ground » option is switched off (see chapter 2.6.2), you cannot highlight flat zones.

4.1.2 Flags



The « Flags + » button allows you to mark a location with a colored flag. This can prevent you from getting lost, to find important positions again, or to indicate a position to a bot. 5 flags of 5 different colors are at your disposal. You can remove a flag and store it for further use by using the « Flags - » button.

Category: `BlueFlag`, `RedFlag`, `GreenFlag`, `YellowFlag` and `VioletFlag`

4.1.3 SatCom

The **SatCom** is a very valuable assistant you wear like a watch on your left wrist. It displays various information such as the goals of the current mission or the satellite report. You can consult the **SatCom** at any time by pressing the F1 key.

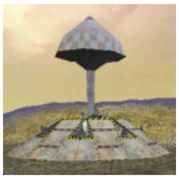
4.2 Buildings

4.2.1 Houston Mission Control

This is the earth control center for all your space missions. Though still named after the well-known mission control of the old days, "Houston" is actually located in the middle of the Nevada Desert, the new center for space exploration activities.

The Mission Control Center watches over you day and night, and a whole team of scientists and engineers are working hard trying to find a solution to all the problems that you encounter.

4.2.2 Spaceship



This is your means of transport from one planet to the next, the only way to travel safely across the cosmos and accomplish your missions.

As well as yourself, the spaceship can carry bots and raw materials. Whenever a mission is completed, you must select the spaceship, then click on the "takeoff" command.

Category: `SpaceShip`

4.2.3 Research center

The research center is an enormous computer. Its most useful feature is its ability to come up with new technologies as well as improvements on existing ones regarding buildings and bots in particular.

The 65'536 processors it contains use up a large amount of energy. Each research program needs a brand new and fully charged power cell.

Category: `ResearchCenter`

4.2.4 Bot factory

This building is intended for the manufacturing of bots using titanium.

- Place the titanium cube inside the factory.
- Step back out.
- Select the factory.
- Click on the button showing the diagram of the bot you want the factory to assemble.

The finished bot does not include an onboard power cell. You'll need to supply it with one for it to be able to leave the factory.

Note: The list of possible bots will depend upon the findings your research center has made at this time.

Category: `BotFactory`

4.2.5 Converter

This building was designed to convert chunks of titanium ore into usable titanium cubes. All you need to do is place a chunk on the center of the platform and step back. The converter takes care of the rest.

Category: `Converter`

4.2.6 Power station

The power station extracts energy from the underground and recharges regular power cells through induction. To charge up a power cell loaded at the back of a bot, just move the bot to the center of the platform and wait for a few seconds. A power cell carried at arm's length can also be recharged. Nuclear power cells are non-rechargeable.

A power plant needs energy from the subsoil. If the satellite report says that energy is available only in some places, you will need a sniffer bot to prospect the subsoil. The marks it lays down provide specific information about its findings: a green cross means that there is an energy deposit at this location beneath the surface, which is indispensable to build a power station or a power plant.

If the large power cell at the top of the station remains red after the construction is completed, this means that the site is not geologically adequate.

Category: `PowerStation`

4.2.7 Radar station

The radar indicates the direction of the nearest enemy. The radar also provides information to the mini-map at the bottom right corner of your screen so it can show the position of all bots, buildings and enemies as small squares and triangles of different colors.

Radar stations also act as relays for communication with the Earth from distant planets.

Category: `RadarStation`

4.2.8 Repair center

The repair center regenerates the shield on damaged bots. A bot's shield will absorb a certain amount of enemy hits. If there is no shield left, this means that the bot itself, when hit next, will be destroyed.

Category: `RepairCenter`

4.2.9 Defense tower

The tower is the best defense against enemy attacks, whether they originate on the ground or from the skies.

The tower requires either a regular or a nuclear power cell. A regular power cell provides a capacity of eight shots. A nuclear power cell is of course preferable. The tower starts to blink when it runs out of power.

The range is 40 meters. To visualize it, select the tower then hit the "range" button. Red dots outline the circular zone for 20 seconds.

Category: `DefenseTower`

4.2.10 Power cell factory

The power plant is a power cell factory. It transforms a titanium cube into a regular power cell fully charged and ready for use.

A power plant requires energy from the subsoil. If the satellite report says that energy is available only in some places, you will need a sniffer bot to prospect the subsoil. The marks it lays down provide specific information about its findings: a green cross means that there is an energy deposit at this location beneath the surface, which is indispensable to build a power plant or a power station.

If the large power cell at the base of the plant remains red after the construction is completed, this means that the site is not geologically adequate.

Category: `PowerPlant`

4.2.11 Derrick

The derrick is used to extract raw materials. In order to determine the best site on which to erect a derrick, a sniffer should be used to prospect the subsoil. The marks it lays down indicate what the derrick will be able to extract:

- Red cross ➔ titanium ore.
- Yellow circle ➔ uranium ore.
- Green cross ➔ energy, not suitable for the construction of a derrick.

Category: `Derrick`

4.2.12 Nuclear power station

The nuclear plant is a nuclear power cell factory. It transforms a chunk of uranium ore into a nuclear power cell fully charged and ready for use.

Category: `NuclearPlant`

4.2.13 Autolab

The lab is intended for the analysis of organic matter. It will help you become familiar with the insects' very own technology and perhaps even use it to your advantage. Place a chunk of organic matter on the platform, select the lab, and click the button corresponding to the desired research program.

Category: `AutoLab`

4.2.14 Power captor

The power captor acts both as a lightning conductor and a power converter. It offers protection within a radius of 50 meters against the perilous lightning bolts of magnetic storms. Additionally, when the captor is hit by lightning, all bots and power cells placed underneath, except nuclear power cells, are recharged.

To visualize the zone shielded by a power captor, select it and then hit the "range" button. Red dots outline the circular zone for 20 seconds.

In the vastness of space, few planets suffer from magnetic storms. Among the ones you'll be visiting, only Orpheon is subject to this phenomenon.

Note that your spaceship can also act as a lightning conductor. However, it will not recharge power cells.

Category: `PowerCaptor`

4.2.15 Information exchange post

This building stores digital information. A post can contain up to 10 pieces of information, each one referenced by a name. For example, a post can contain 3 pieces of information:

- "Position.x" 23.45
- "Position.y" -102.70
- "Quantity" 3.00

Use the `send` command to store a new piece of information. To read a piece of information from a post, use the `receive` command.

Category: `ExchangePost`

4.2.16 Vault

This dome-shaped building was conceived by the first expedition. Its purpose was to offer protection to a new type of advanced bot. To protect it from insect attacks, the vault was locked by four keys. To access the bot, you must find the four keys and place each one in its corresponding slot.

Category: `Vault`

4.3 Transportable objects

4.3.1 Titanium ore

Titanium is indispensable for the construction of buildings, bots and regular power cells. The titanium you see in its usable, cubic shape has been converted from raw chunks of titanium ore.

Titanium ore can be found either directly on the surface or in the subsoil. In this case, it needs to be located and identified by a sniffer, then extracted using a derrick. If a sniffer identifies a titanium ore deposit, it lays down a red cross.

Category: `TitaniumOre`

4.3.2 Uranium ore

Uranium ore is essential to produce of nuclear power cells. It can be found on the surface or in the subsoil. In this case, it needs to be located and identified by a sniffer, then extracted using a derrick. If a sniffer identifies a deposit of uranium ore, it lays down a yellow circle.

A nuclear plant will then take care of converting the chunk into a new and fully charged nuclear power cell. The astronaut cannot carry uranium ore because of the radioactivity hazard.

Category: `UraniumOre`

4.3.3 Titanium

Titanium is indispensable for the construction of buildings, bots and regular power cells. There is usually some left on the spaceship when you embark on a new mission.

If you need more, you will have to convert chunks of titanium ore into titanium cubes.

Category: `Titanium`

4.3.4 Power cell

A power cell supplies bots and some buildings with energy. An entirely red power cell is empty. The green section indicates the remaining capacity. A power cell can be recharged using a power station. A power plant is needed to produce a new and fully charged power cell.

The capacity of a regular power cell is 100 times smaller than that of a nuclear power cell produced by a nuclear plant using uranium ore.

Category: `PowerCell`

4.3.5 Nuclear power cell

A nuclear power cell supplies bots and some buildings with energy. Its capacity is 100 times that of a regular power cell.

Nuclear power cells cannot be recharged or recycled. A nuclear plant and some uranium ore is needed to produce a new and fully charged nuclear power cell.

Category: `NuclearCell`

4.3.6 Black box

A black box is actually orange to be easier to identify and locate. Each spaceship is equipped with a black box that records and stores information about the flight.

A black box can also be used to leave important information. On every planet, the first expedition has left a black box containing the coordinates of the planet it was heading towards next.

Category: `BlackBox`

4.3.7 Organic matter

The organic matter is the insects' secretion. Wasps often use balls of organic matter to bombard you. We suggest you try to get hold of some since it could lead you to discover new and strange technologies that will improve the efficiency of your bots if analyse it.

Category: `OrgaMatter`

4.3.8 Keys

The keys give you access to the vault. You'll need four of them.

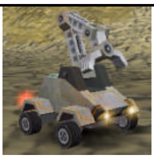
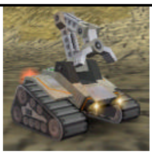
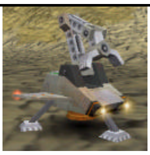
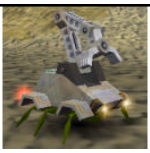
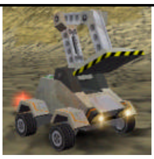
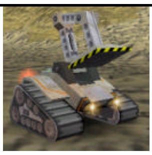
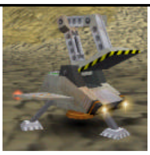
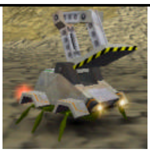
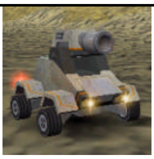
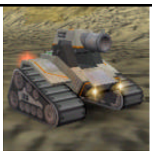
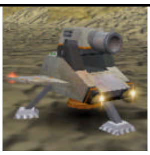
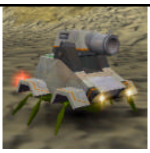
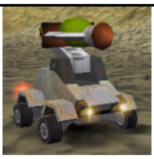
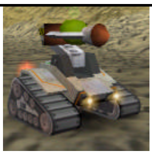
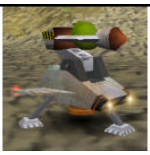
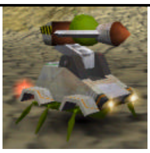
- Key A, blue and triangle-shaped
- Key B, red and pentagon-shaped
- Key C, green and shaped like a 6-pointed star
- Key D, yellow and circular

Category: `KeyA`, `KeyB`, `KeyC` et `KeyD`

4.4 Bots

4.4.1 Propulsion modes

Each of the four robot types (grabber, sniffer, shooter and orga shooter) can be equipped with four different propulsion technologies. You will discover them along your progression through the different missions.

	Wheeled	Tracked	Winged	Legged
Grabber				
Sniffer				
Shooter				
OrgaShooter				

4.4.1.1 Wheeled bots

Wheels are a standard, fast and energy-saving mode of propulsion, which is perfectly adapted for a relatively flat terrain. Whenever the terrain gets sloped, it is advised to use a winged robot instead, or, if this is impossible, a tracked bot.

4.4.1.2 Tracked bots

Tracked bots can climb steep slopes but they are quite slow and use a lot of energy. On flat ground for short distances, a wheeled grabber is a better option. When it is possible to build winged bots, these represent the best solution for long distances.

4.4.1.3 Winged bots

Winged bots can fly over natural obstacles such as mountains or lakes but their energy supply is used up quickly. They're slow on the ground. To cover a short distance that don't necessarily require flying, it is recommended to use a wheeled bot instead.

The display at the bottom of your screen indicates the temperature of the reactor. Keep an eye on it. If the reactor overheats, the engine will stop and the bot will crash (see chapter 3.2.5.3).

4.4.1.4 Legged bots

A moving legged bot uses up half as much energy as a wheeled bot. A legged bot is also perfectly adapted to climb the steepest slopes.

4.4.2 Bot functions

A bot can have four different functions.

4.4.2.1 Grabber function

A grabber can move around small objects such as ore, titanium cubes, power cells etc.

This is the complete list of all transportable objects:

- Titanium ore
- Uranium ore
- Titanium
- Regular power cell
- Nuclear power cell
- Black box
- Keys
- Organic matter

Grabs an object or puts it down again, at a position that is determined by one of the options below. You can also press the "Enter" key instead of clicking on this button.

Three icons specify where the bot must pick up or put down the object :

In front of the bot, on the ground or on the back of a second bot,.

The object is the bot's own power cell.

Behind the bot, on the ground.

4.4.2.2 Sniffer function

Wheeled bot equipped to prospect the geological structure of the subsoil. Whenever it locates something of use, the sniffer lays down the following marks:

- Red cross ➔ titanium ore.
- Yellow circle ➔ uranium ore.
- Green cross ➔ energy deposit, useful for a power station or a power plant.

4.4.2.3 Shooter function

The fireball cannon is an efficient weapon against most kinds of enemies. Use it sparingly though for it requires large amounts of energy. A regular power cell will only allow you to shoot four fireball bursts.

Tip: Move the mouse while shooting, to sweep a larger area.

4.4.2.4 OrgaShooter function

The orgaball cannon is more efficient than the fireball cannon. It shoots small spheres of corrosive organic matter. A regular power cell will allow you to shoot 11 orgaball bursts.

Tip: Move the mouse while shooting, to sweep a larger area.

4.4.3 Thumper

Tracked bot designed to hit the ground with tremendous force to turn ants and spiders belly up within a radius of 100 meters. An insect on its back is not dead but will struggle to get right side up again. After approximately 60 seconds of effort, it will usually succeed.

To visualize the zone of impact, hit the "range" button. Little red dots outline the circular zone for 20 seconds.

This bot uses up a large amount of power. A thump will drain 40% of a regular power cell.

4.4.4 Recycler

Tracked bot designed to convert a derelict bot back into a reusable titanium cube.

4.4.5 Shielder

Tracked bot designed to protect and defend against all enemy attacks within a perimeter of 10 to 25 meters. The individual shields of bots and buildings are re-energized through the shielder's defensive actions. This bot is the only way to get through narrow passages covered with poisonous green mushrooms.

A regular power cell allows for a 20-second activity span with a radius of 25 meters, much too short in most cases. A nuclear power cell is of course more suited to this bot.

The energy consumption is proportional to the radius of the protective sphere. With a radius of 10 meters, the bot can work 2.5 times longer than with the maximum radius of 25 meters.

4.4.6 Subber

Amphibious tracked bot equipped with an operating claw. The subber is the only bot capable of moving and running operations underwater.

It is best to check the power cell readings prior to immersion since replacing or recharging it underwater is impossible. The subber can only pick up objects from the ground, as opposed to, for example, a battery from the back of another bot.

4.4.7 Phazer shooter

Tracked bot equipped with a very powerful phazer cannon, efficient against most enemies. When you aim upwards, it can shoot up to 60 meters. This is the only weapon that can kill the Alien Queen.

4.5 Aliens

4.5.1 Ant

Ants shoot tiny corrosive balls that eventually gnaw into the protective shielding of bots and buildings, causing them to explode.

You can resist their attack much longer than most buildings and bots, and your protective shield will be restored as your wounds heal. However, if you stay too long within their range or if there are too many of them, your life is in great danger.

Category: `AlienAnt`

4.5.2 Spider

Spiders work like suicide commandoes: whenever a spider gets close enough to its target, it inflates its abdomen and bursts into a multitude of tiny burning fragments. These fragments will set fire to everything they get in contact with. Luckily, the spider won't survive.

Category: `AlienSpider`

4.5.3 Wasp

The wasp is a form idable enemy, very hard to shoot down since its flying is particularly swift. Wasps carry balls of organic matter that they will hurl at your bots and buildings and at you from above.

The defense tower is the easiest way to get rid of them.

Category: `AlienWasp`

4.5.4 Worm

Worms transmit viruses to programmed bots. The virus alters the programming to such an extent that the bot either functions in a very erratic way or stops functioning altogether.

Worms live in the subsoil but sometimes emerge and crawl on the surface. When they are underground, they are undetectable and indestructible.

Category: `AlienWorm`

4.5.5 Alien Queen

This huge insect is the mother of them all. It lays eggs that turn into all the various kinds of insects you're likely to encounter. All of them are extremely hazardous to yourself and your mission. The queen's protective shell is particularly resistan.

Category: `AlienQueen`

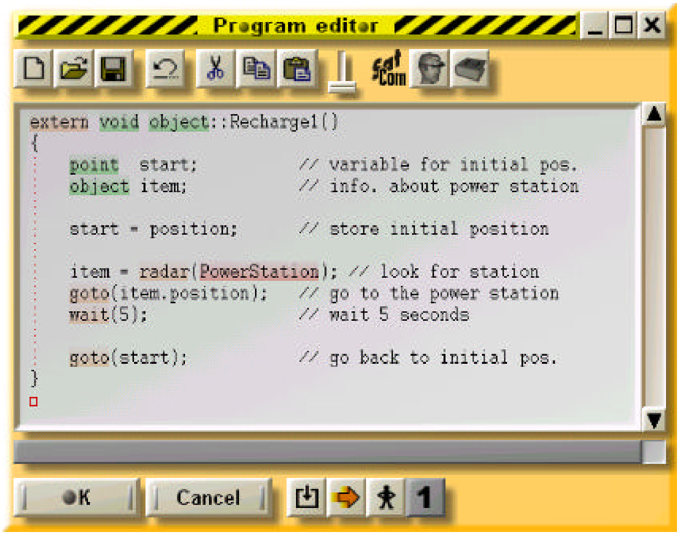
5 Programming

All robots are programmable so they will execute certain tasks automatically without intervention.

5.1 Program editor

To enter the program editor proceed as follows:

- Select a robot
- Choose one of the 10 program slots at the lower left corner of the screen
- Click on the « Edit the selected program » button.



```
extern void object::Recharge1()
{
    point  start;           // variable for initial pos.
    object item;            // info. about power station

    start = position;       // store initial position

    item = radar(PowerStation); // look for station
    goto(item.position);    // go to the power station
    wait(5);                // wait 5 seconds

    goto(start);            // go back to initial pos.
}
```

During program edition the game is paused. If you move the mouse pointer close to the left or the right edge of the screen, the viewpoint camera will rotate. If you move the mouse pointer close to the upper or the lower edge of the screen, the viewpoint moves forward or backward. The three buttons at the top right corner of the program editor window are the same as in most Windows programs.

If you enter the program editor while the program is being executed the game is not paused but you can observe the progress of the program (see chapter 5.1.11). The program editor window can be moved by dragging the title bar. You can also change its size by dragging the edges or the corners.

The keywords of the language are displayed in different colors:

Color	***Kind***	***Example***
Orange	Instruction	`aim`, `fire`, `turn`, `goto`, `while`, etc.
Green	Variable type	`object`, `float`, `int`, etc.
Red	Constants, Category	`TitaniumOre`, `Converter`, `BlackBox`, etc.

When the cursor is on a keyword, the status bar at the lower edge of the program editor window displays some information. You can then click on the status bar or press the F3 key to bring up the **SatCom** , which will display further information.

You can double click on a word to select the whole word. Shift-arrow also selects text. Ctrl-arrow moves word by word and Shift-Ctrl-arrow selects text word by word just like any standard editor. The usual Ctrl-X, Ctrl-V, Ctrl-C shortcuts for copying and pasting text are available like in any other standard editor.

5.1.1 New

Clears the whole program and creates an empty skeleton program :

```
extern void object::New( )
{

}
```

`New` is the default name of the program. You can change it to what ever you like, but you may not use white spaces or special characters but only letters and digits. For example `Search`, `BringTitanium`, `ComeHome`, etc.

Note: You can revert to the previous situation by pressing Ctrl-Z or by clicking on the Undo button (see below).

5.1.2 Open and save

All programs you create are automatically saved within the mission or the exercise. On the other hand, if you want to reuse a program in another mission you must save it explicitly.

Private	The program is saved in a private folder attached to the current player (see chapter 2.7).
Public	The program is saved in a public folder and thus will be available to all other players.

The name of the folder where the program will be saved appears in the title bar of the Save dialog. For example, **Savegame\Player1\Program\** means that the program will be saved in **c:\Program Files\Colobot\Savegame\Player1\Program\** (assuming Colobot has been installed in the folder **c:\Program Files\Colobot\**.

Shortcuts « Ctrl+O » and « Ctrl+S ».

5.1.3 Undo

Undoes the last modification. You can Undo the 20 last modifications.

Shortcut: « Ctrl+Z ».

5.1.4 Cut, Copy and Paste

Cuts or *copies* the selected text to the clipboard. If no text is selected, the whole line will be taken. The content of the clipboard can be pasted into the program you are editing.

Shortcuts: « Ctrl+X », « Ctrl+C » and « Ctrl+V ».

5.1.5 Font size

Use this slider to change the font size for the editor and the **SatCom**.

5.1.6 SatCom: Instructions

Shows information about the goal of the mission or the exercise.

Shortcut: « F1 ».

5.1.7 SatCom: Programming help

Shows information about the programming of robots. Use the hypertext links (underlined in blue) to navigate around.

Shortcut: « F2 ».

5.1.8 OK

Compiles the program and quits the editor. If there is a syntax error in your program, the error is shown and an appropriate error message is displayed in the status bar.

5.1.9 Cancel

Quit the editor without compiling but all modifications are saved.

5.1.10 Compile

Compiles the program without quitting the editor. This is useful to check your program for syntax errors.

5.1.11 Execute/stop

Starts or stops program execution without quitting the editor. This is useful for debugging purposes as this allows you to follow the progress of your program. If the button on the right side depicts a standing man, the program will be executed step by step.

Note: During program execution the content is displayed in orange and you cannot modify the program.

5.1.12 Pause/continue

Switches from step by step execution to continuous execution and back.

5.1.13 One step

1 Executes the next instruction in step by step mode. The lower part of the program editor shows the content of the different variables which change during the progress of the program.

5.2 The CBOT programming language

The CBOT programming language is very close to C++ and Java™. It has been designed specially for COLOBOT and is very well adapted to learning. In this manual, we only present some very simple examples . For a more complete description, use the **SatCom** by pressing « F2 ».

The CBOT language is case sensitive, for example, `Radar` is not the same as `radar`.

Each instruction must be terminated by a semicolon "`;`".

Comments start with `//` and end at the end of the line.

5.2.1 Instruction `Radar`

With this instruction, you can look for objects like enemies, bots, buildings or raw materials.

Write the name of the object that you are looking for between brackets. Put the result in a variable of the type object. Here is an example that looks for the closest ant:

```
        // At the beginning of the program:
        object  item; // variable declaration

        // Look for the closest ant
        item = radar(AlienAnt);
```

5.2.2 Instruction `goto`

`goto()` instructs the bot to reach a given position.

The most current use consists in moving the bot to an object located with the instruction `radar()`. If the information returned by `radar()` has been stored in a variable, write the name of the variable followed by `.position` in order to get the position of the object. Here is an example of a program that looks for a titanium cube and goes to its position:

```
        object item;
        item = radar(Titanium);
        goto(item.position);
```

5.2.3 Instructions `grab` and `drop`

The instruction `grab()` instructs the bot to use the operating arm to grab an object located on the ground, on the platform of a building or on the power cell location of a bot. The instruction `drop()` on the other hand instructs the bot to drop whatever the operating arm is carrying on the ground, on the platform of a building or on the power cell location of a bot.

After finding titanium in the previous example, this is how to tell your bot to pick it up and drop it 5 meters further:

```
        grab();   // grab the object
        move(5);  // move forward by 5m
        drop();   // drop it
```

5.2.4 Instruction `fire`

Use this instruction to fire the bot's onboard cannon. You can specify how long the burst must last. Generally, this instruction is used to shoot one-second bursts:

```
        fire(1);
```

5.2.5 Instruction `while`

`while(){}` is used to repeat a set of instructions several times. The most frequent use of while consists in repeating a set of instructions again and again. In order to achieve this, write `while(true){}` and put the instructions to be repeated in braces `{}`. As an example, here is a program that repeats again and again the following actions:

- look for a spider,
- turn towards it,
- shoot.

```
        while (true)
        {
                item = radar(AlienSpider);
                turn(direction(item.position));
                fire(1);
        }
```

Just execute this program once, and it will kill all spiders around it.

5.2.6 Instruction `distance`

With `distance( , )` you can calculate the distance between two positions. If you use `position` alone, this gives you the position of the bot that is executing the program. A variable followed by `.position`, gives you the position of the object described in the variable.

Here is a program that moves forward, covering exactly the distance between the bot and the closest ant:

```
        item = radar(AlienAnt);
        move(distance(position, item.position));
```

This is of course pure suicide. Better to stop 40 meters before, in order to be at shooting range:

```
	item = radar(AlienAnt);
	move(distance(position, item.position) - 40);
```

5.2.7 Instruction `if`

Use `if(){}` to execute a set of instructions only if a certain condition is true. Write the condition in brackets (), and the instructions in braces {}.

Here is an example: The bot will shoot only if the target is closer than 40 meters:

```
	item = radar(AlienAnt);
	if (distance(position, item.position) < 40)
	{
		fire(1);
	}
```

You can also test if an object exists at all. If the instruction `radar()` does not find the requested object, it returns the special value `null`. So you can test if an object does not exist with the condition `(item == null)`, or test if it exists with `(item != null)`. Two equal signs == test equality, an exclamation mark followed by an equal sign `!=` test inequality. Here is a test that will go to recharge the power cell only if there is a power station:

```
	station = radar(PowerStation);
	if (station != null)
	{
		goto(station.position);
		wait(5);
	}
```

5.2.8 Instruction `motor`

The instruction `motor( , );` sets the speed for the left-hand and the right-hand motor of the bot. The speed given to the motors will remain constant during the execution of the following instructions and therefore it is possible to perform a rotation during the instruction `fire();`. This will sweep a whole zone with only one burst. Here is an example that will sweep the zone in front of the bot:

```
	turn(45);          // turns 45 degrees left
	motor(0.5, -0.5); // slow rotation to the right
	fire(2);           // fire
	motor(0,0);        // stops the rotation
```

With the left-hand motor turning half-speed forward and the right-hand motor turning half-speed backward, the bot will turn slowly on itself during the 2-second-burst.

5.2.9 Instruction `turn`

Use the instruction `turn()` to instruct the bot to perform a rotation of a certain number of degrees. 90 degrees is a quarter turn, 180 degrees is a half turn. A positive angle will perform a counterclockwise rotation, a negative angle means a clockwise rotation. Here are some examples with `turn()`:

```
	turn(90);   // quarter turn to the left
	turn(-90);  // quarter turn to the right
	turn(180);  // half turn
```

In order to turn the bot towards an object found with the instruction `radar()`, you must calculate the rotation angle with the instruction `direction()`:

```
	TheThing = radar(AlienSpider);
	turn(direction(TheThing.position));
```

After these lines, just fire the cannon, and you'll have gotten rid of a hostile element.

6 Copyrights and trademarks

7 Development team

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

EPSITEC SA, Mouette 5, CH-1092 Belmont
colobot@epsitec.ch
www.colobot.com

7.1 Beta tester core team

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