1 Introduction

1.1 Minimal requirements

CeeBot requires an up-to-date and powerful computer. It is particularly important to have a good 3D graphic adaptor for maximum performance. This can be a problem with some portable computers.

- 300 MHz CPU
- 64 Mb RAM
- 3D graphic card with 16Mb RAM
- 100 Mb free disk space
- Windows XP, 2000, ME, 98 or 95

Recommended graphics cards :

- nVidia GeForce 2, 3 ou 4
- Matrox G400, G450 ou Parhelia

3dfx Voodoo graphics adapters are not officially supported, but they may work (no guarantee).

1.2 To install CeeBot

- Insert the CeeBot-Teen CD-ROM in the drive

Normally, it should start automatically after ten seconds or so. Click « **Install** ». If the *autorun* doesn't start, follow these steps :

- Double-click **My Computer**.
- Open the **D:** drive (where D: is the letter of your CD-Rom drive).
- Double-click **install.exe**.

The installation process will check if you have DirectX 8a. If this is not the case, you'll be prompted to install it.

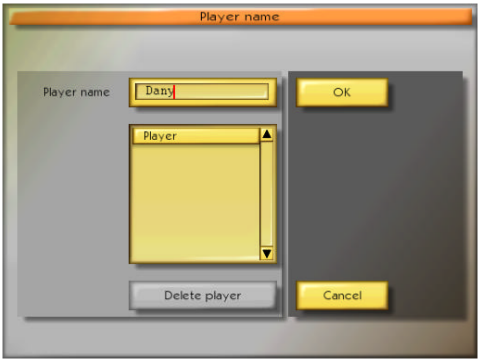
If you have a previous version of CeeBot, a new installation in the same folder will overwrite all data, saved games or programs.

1.3 To uninstall CeeBot

- Double-click **My Computer**.
- Double-click **Control panel**.
- Double-click **Add or Remove programs**.
- Double-click **CeeBot-Teen** in the list.

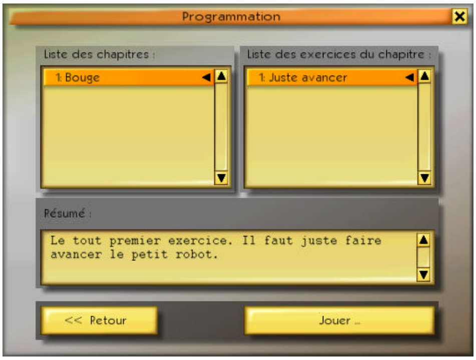
2 First approach

In this section, we're going to take a close look at the first exercise, step by step. The detailed explanations will be found in the following chapters. This will be your very first contact, to help you get familiar with CeeBot-Teen.



The first thing CeeBot needs is the player's name. It is better if each player has a separate identity, because CeeBot saves each program and keeps track of progress. So, just type in your pseudo or name the first time, and it will then appear in the list. Simply select it and click « **OK** ».

Then click :

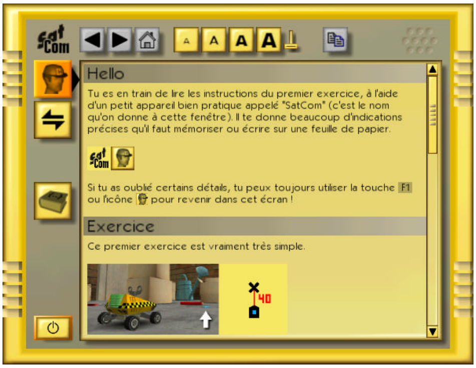


Le tout premier exercice. Il faut juste faire avancer le petit robot.

CeeBot now lists the various exercises. The left column shows the different chapters, and the right column is the list of the exercises within the selected chapter.

When you've finished the first exercise, the second appears automatically in the right column.

After selecting « **1: Move** » et « **1: Forward** », click « **Play** ».



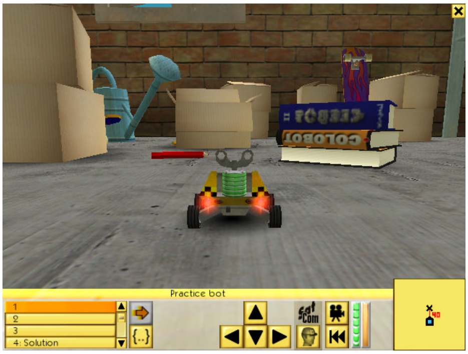
Tu es en train de lire les instructions du premier exercice, à l'aide d'un petit appareil bien pratique appelé "SatCom" (c'est le nom qu'on donne à cette fenêtre). Il te donne beaucoup d'indications précises qu'il faut mémoriser ou écrire sur une feuille de papier.

Si tu as oublié certains détails, tu peux toujours utiliser la touche F1 ou l'icône pour revenir dans cet écran !

Ce premier exercice est vraiment très simple.

Read this carefully, as it explains exactly what you must do. When you've finished, click on the button at the bottom left to exit.

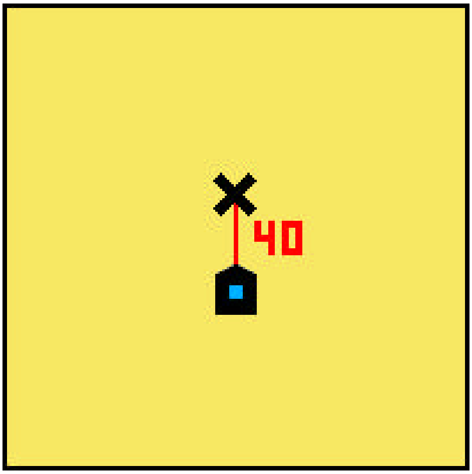
If necessary, you can come back to this screen at any time by pressing « **F1** ».



The miniature robot is on the floor of a cellar which is filled with an odd collection of objects.

Use the arrow keys to check out the surroundings.

In this first exercise, you have to program the robot so that he'll go straight ahead exactly 40 centimeters.

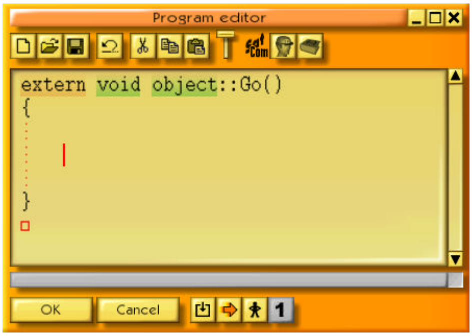


The miniature map at the bottom left of the screen summarises the situation. The bottom symbol is the robot. The cross represents the goal to be reached. The two symbols are 40 cms apart.

If it's not already been done, select program no. 1 from the top of the list. Each bot can store up to 10 programs, which can be useful when you're testing different solutions. All these programs are saved automatically in each player's profile.

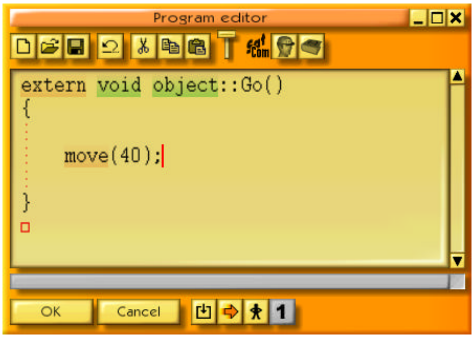
Program no. 4 contains the **solution** so you'll never be stuck. Try not to peek too soon !

To enter the program editor, click **{..}**.



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

}
```

The editor comes up with an empty shell ready for you to write your new program. The text cursor is between the braces. Do not alter the first line or the braces.

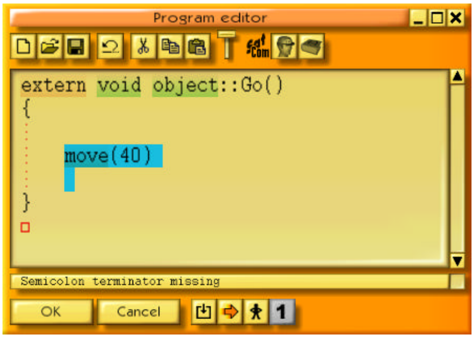


```
extern void object::Go()
{

    move(40);

}
```

The point of the exercise is to move the robot forward 40 cms up to the blue cross. All you need is to type this instruction :

```
move(40);
```

All the characters are in lower case. Don't forget the semi-colon that completes the instruction.



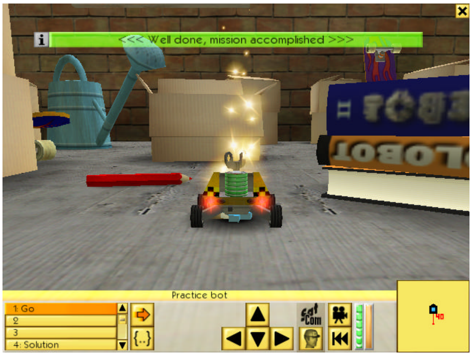
```
extern void object::Go()
{

    move(40)

}
```

```
Semicolon terminator missing
```

Click « OK ». If you've made a mistake, it will be highlighted in blue, and a message will be displayed at the bottom of the screen.

In the example to the left, the semi-colon at the end of the instructions is missing.

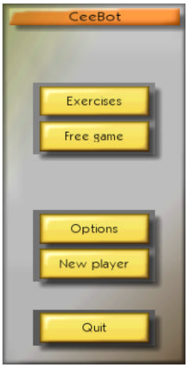
If everything is correct, when you click OK the editor closes.



Click this icon to run your new program. The robot should drive up to the blue cross.

Congratulations, you've just completed your very first program.

3 Main menu



3.1 Exercises

In this part of CeeBot-Teen, you'll learn how to program the bots. The exercises are gathered into chapters, and progressively introduce the basic concepts of programming. You should start with the easy ones and work your way through to the really hard ones.

3.2 Free play

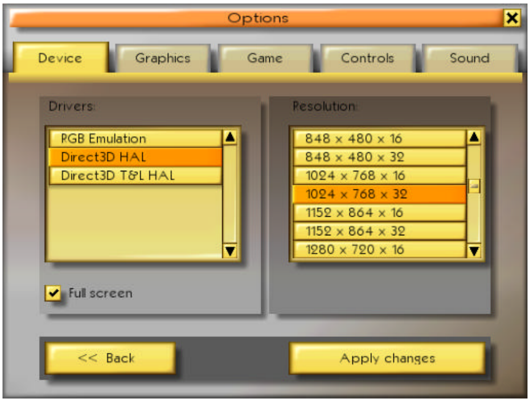
This gives you access to three special levels, « just move », « easy drawing » and « magic programming ». In the last option, you can your own invent programs. Simply move the robot around as you wish in order to draw, and the corresponding program is created at the same time.

3.3 Options

Click « Options » to access various settings separated into 5 categories.

3.3.1 Display

The first time you run CeeBot-Teen, it is set to use a 640 x 480 x 16 display. On many computers, you can get a much better graphic result by changing the default settings.



Drivers:
- RGB Emulation
- Direct3D HAL
- Direct3D T&L HAL

[✓] Full screen

Resolution:
- 848 x 480 x 16
- 848 x 480 x 32
- 1024 x 768 x 16
- 1024 x 768 x 32
- 1152 x 864 x 16
- 1152 x 864 x 32
- 1280 x 720 x 16

Drivers :

It's best to choose a HAL driver (Hardware Abstraction Layer), and avoid « Emulation » or « T&L » drivers. It often happens that drivers with different names produce the same result.

Résolution :

The first and the second figure stand for the size of the image on your screen. The third number influences the number of colours used: 16 bits will display 65'000 colours, whereas 32 bits will display 4 million colours.

The higher you set these parameters, the more detailed the scene will be on the screen. However, high parameters need a lot more processor and graphic power, and the frame-rate will be lower.

✍ Full screen

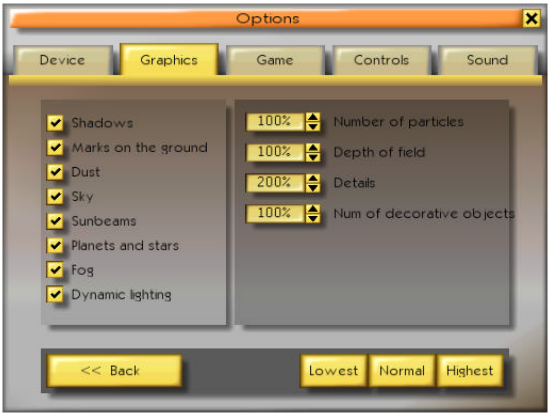
Normally CeeBot-Teen runs in full screen mode, whatever resolution you choose. If you deactivate this flag, CeeBot will be displayed in a fixed size window of approximately 640 x 480 pixels.

[Apply changes]

Click this button to use the new settings.

3.3.2 Graphics

If the game is too slow, or if the frame-rate is too low, you can tune down some of the graphic options. Obviously, the higher the settings, the more beautiful the game!



✓ 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% Num of decorative objects

Number of particles (0% - 200%)

Particles are used to simulate dust, smoke, explosions, bubbles under water, etc.

Depth of field (50% - 200%)

This feature determines up to what distance the scenery is displayed. This distance varies according to the atmosphere on each planet. A high value close to 200% allows you to see very far, but requires a powerful 3D graphic card.

Details (0% - 200%)

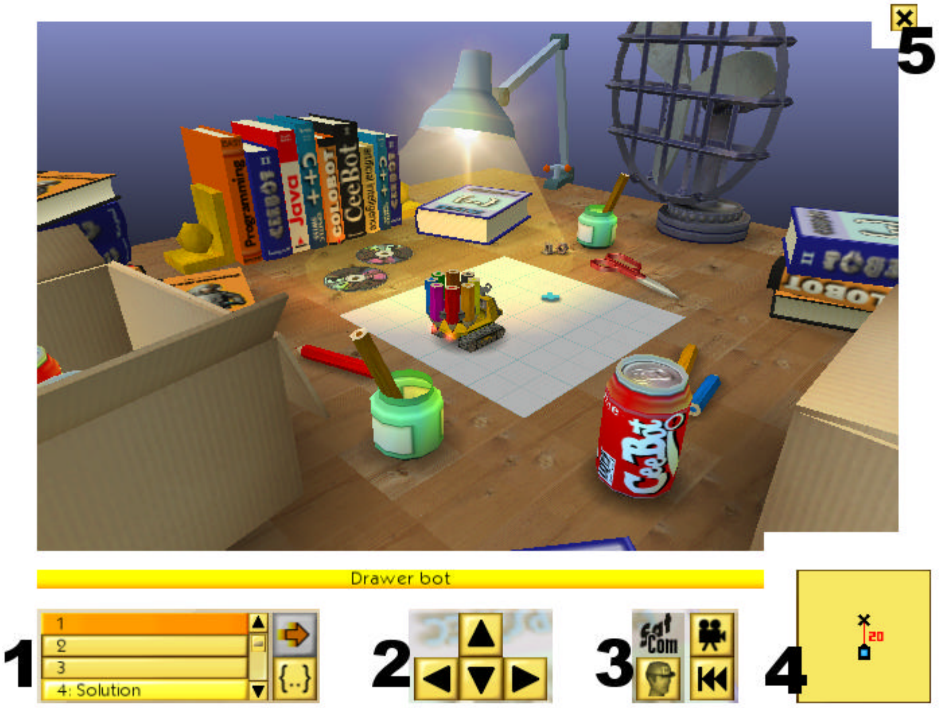
Sets the visual quality of 3D objects according to their distance..

Number of decorative objects (0% - 100%)

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

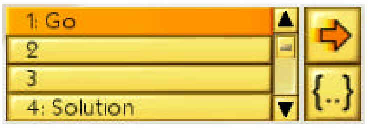
4 The Screen

This is what the screen looks like during an exercise :



1 The robot's 10 programs
2 Control panel for the camera
3 Various commands
4 Mini map
5 Access to the menu

4.1 The 10 programs



The left bottom corner gives you access to 10 program slots for each robot. The 4th program usually is our suggestion for the **solution** to the exercise. Use the scroll bar to access programs 4 to 10.

Launch or stop the execution of the selected program.

Launch the built-in program editor for the selected program (see section 6.1). When you edit a program, the game pauses so you can take all the time you need to modify programs.

4.2 The Camera

The arrows on your keyboard or the 4 arrow icons let you look at the surroundings by moving the camera. The « up » and « down » arrows advance or recede the camera. The « left » and « right » arrows let you see a full turn around the robot.

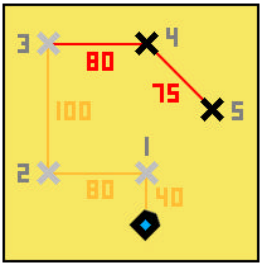
4.3 Various Commands

The ***SatCom*** is a very crafty instrument that displays all the instructions for the current exercise. The « **F1** » key has the same function.

You can change the viewpoint of the camera by clicking this button or by pressing the space bar of your keyboard. Some robots have 2 viewpoints, others have 3.

This resets the exercise to the beginning. It's very useful when your program hasn't worked out as you'd wished. It will put the robots and objects back into their original places and will erase the drawing.

4.4 The mini map



The mini map in the lower right corner offers an overview of the situation. The crosses represent passage points, and distances are always shown in centimetres.

The robot is represented by a pointed rectangle. The point shows the direction the robot is heading.

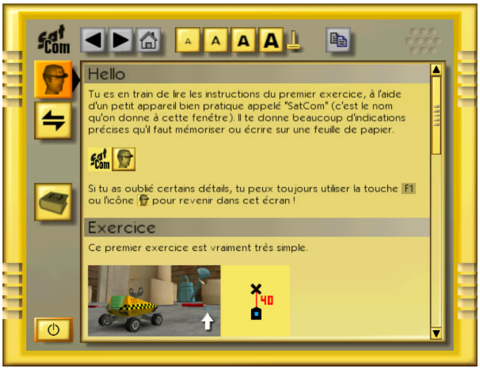
4.5 The Menu

If you click this button in the top right corner of the screen, the menu is displayed :

Continue	Continue the current exercise.
Options	Access 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 in the main menu (see section 3.3).
Restart	Abort the current exercise and restart from the beginning.
Abort	Abort the current exercise and return to the list of exercises.

5 The *Sat*Com

The ***Sat*Com** is a small but useful apparatus. It is used to communicate with the data base and displays all the information about the CeeBot programming language.



Hello

Tu es en train de lire les instructions du premier exercice, à l'aide d'un petit appareil bien pratique appelé "SatCom" (c'est le nom qu'on donne à cette fenêtre). Il te donne beaucoup d'indications précises qu'il faut mémoriser ou écrire sur une feuille de papier.

Si tu as oublié certains détails, tu peux toujours utiliser la touche F1 ou l'icône pour revenir dans cet écran !

Exercice

Ce premier exercice est vraiment très simple.

It works rather like an Internet browser. Whenever a word is underlined in blue, you can click it to display the corresponding page of information.

◀	Previous page.
▶	Next page.
🏠	Display the page that came up when you turned the ***Sat*Com** on.
	Copies the highlighted characters to the clipboard. You can use this feature to copy all or part of a program and paste it into the editor (see section 6.1.4).
	Displays the instructions for the current exercise. « F1 » always accesses this page directly.
	Displays overall help for the CeeBot programming language. It's a gold mine, containing detailed explanations for each instruction. « F2 » always accesses this page directly.
⏻	Turns the ***Sat*Com** off and closes the window.

If one of the icons described here is grey, it can't be used in the current context.

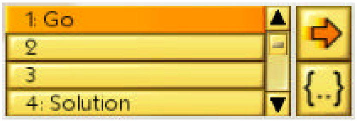
6 Programming

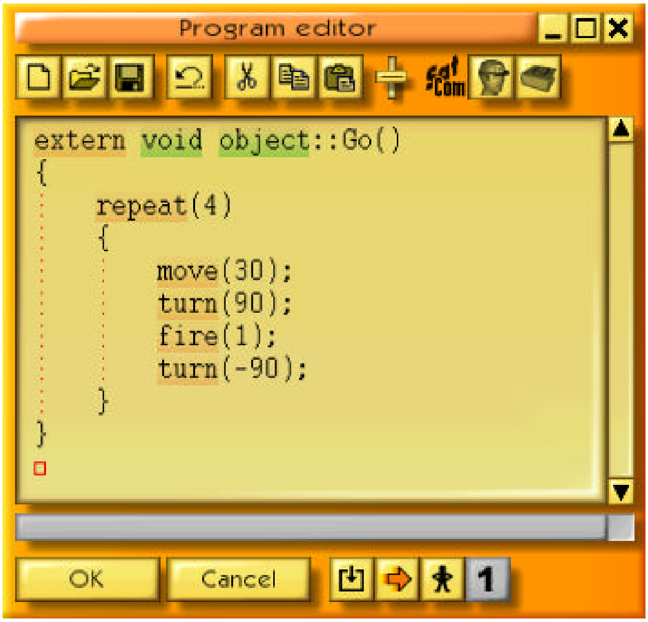
All the robots are programmable.

6.1 Program editor

To enter the program editor proceed as follows :

- Choose one of the 10 program slots.
- Click « Edit the selected program » **{..}**.





```
extern void object::Go()
{
    repeat(4)
    {
        move(30);
        turn(90);
        fire(1);
        turn(-90);
    }
}
```

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. Next time you call up the editor, it will be exactly as you left it.

Reduced size, to reduce the window to a bar at the bottom of the screen.

Maximum size, to enlarge the window to full screen.

Close, equivalent to « OK ».

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 section 6.1.11).

The keywords of the language are displayed in different colours :

Colour	***Kind***	***Example***
Orange	Instruction	`move`, `turn`, `pendown`, `repeat`, etc.
Green	Variable type	`int`, `object`, etc.
Red	Constants, Category	`Red`, `Scrap`, `Metal`, etc.

When the cursor is on a keyword, the status bar at the bottom of the program editor window displays the word and some information. You can the click on the status bar or press « F3 » to bring up the ***Sat*****Com**, which will display further information.

You can double-click a word to select the whole word quickly. If you keep the button pressed and move the mouse, the letters are highlighted. 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 also available.

6.1.1 New

Clears the whole program and creates and empty skeleton program :

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

}
```

`New` is the default name of the programme. You can change it to whatever you like, but you can't use blank spaces or special characters, only letters and digits. For example `Search`, `PrettyDrawing`, `SmallFlower`, etc.

Note : If you've pressed this button by mistake, you can cancel its effect by pressing Ctrl-Z or by clicking the Undo button (see section **Erreur ! Source du renvoi introuvable.**).

6.1.2 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.

Public The program is saved in a public folder and 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/Player/Program** means that the program will be saved in **c:/Program Files/CeeBot-Teen/Savegame/Player/program/** (assuming

CeeBot-Teen has been installed in the folder **c:/Program Files/CeeBot-Teen/**. You can use this option to send a program to a friend by email.

Shortcuts : « Ctrl+O » and « Ctrl+S »

6.1.3 Undo

Cancels the last modification. You can undo the last 20 modifications.

Shortcuts : « Ctrl+Z »

6.1.4 Cut, Copy, Paste

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

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

6.1.5 Font size

Use this slider to change the font size for the editor and the ***Sat*Com**. The font size you chose will be the norm until you decide to change it

6.1.6 SatCom : Instructions

Displays information about the goal of the exercise.

Shortcuts : « F1 ».

6.1.7 SatCom : Programming help

Shows general information about programming robots. Use the hypertext links (underlined in blue) to navigate through the pages.

Shortcut : « F2 »

6.1.8 OK

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

6.1.9 Cancel

Quits the editor without compiling but all modifications are saved, so you can come back to the program you're writing at a later time.

6.1.10 Compile

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

6.1.11 Execute/stop

Starts or stops program execution without closing the editor. This is useful for debugging purposes as you can follow the progress of your program. If the button on the right side shows a little 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.

6.1.12 Pause/continue

Toggles between step by step and continuous execution.

6.1.13 One step

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

6.2 The CeeBot programming language

The CeeBot programming language is very close to C++, C# and Java? . It has been designed especially for CEEBOT and is very well adapted to learning. In this manual, we only give some very simple examples. For a more complete description, use the ***Sat*****Com** by pressing« F2 ».

The CeeBot language is case sensitive, for example, « `Detect` » is not the same as « `detect` ».

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

Comments start with `//` and end at the end of the line. This has no incidence on the instruction, but serves to make the sentence explicit within the program.

6.2.1 Instruction `move`

This instruction lets you move forwards or backwards. Distance is in centimetres. Use positive numbers to advance, and negative numbers to recede. Take into account that the robot won't avoid obstacles with this instruction..

```
        move(40);    // Advance 40 cms
        move(-10);   // Recede 10 cms (negative)
```

6.2.2 Instruction `turn`

Use this instruction `turn();` to rotate the robot a certain number of degrees. 90 degrees is a quarter turn, 180 a half turn, 360 a complete circle. Here are some examples :

```
        turn(90);    // Quarter turn to the left
        turn(-90);   // Quarter turn to the right (negative number)
        turn(180);   // Half turn
```

6.2.3 *Instruction* `pendown`

With this instruction, you can use one of the robot's 8 crayons. Once the crayon is lowered, any of the robot's moves using the instruction `move` will leave a trace. A change of direction using the `turn` instruction won't leave a mark, as the crayon is exactly in the middle of the robot.

```
       pendown(Red);    // Lowers the red crayon
       pendown(Blue);   // Lowers the blue crayon
```

The 8 available colours are :

Black
Yellow
Orange
Red
Purple
Blue
Green
Brown

6.2.4 *Instruction* `penup`

This instruction raises the drawing robot's crayon. While you need to write the brackets for the instruction, you don't need to include the colour.

```
       penup();  // Raises the crayon
```

6.2.5 *Instruction `repeat`*

This instruction doesn't need a semicolon at the end! When you use repeat, the instructions contained in the brackets `{ }` will be repeated.

```
        repeat(4)
        {
                move(20);
                turn(90);
        }
```

6.2.6 *Instruction `fire`*

You can fire the robots canon with this instruction in bursts of one or more seconds.

```
        fire(1);
```

6.2.7 *Instructions `grab` et `drop`*

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

```
        grab();     // take object
        move(15);   // move forward 15 cms
        drop();     // put object down
```

6.2.8 *Instruction `find`*

The robot will search for objects like scrap and factories when you use the instruction `find();`. Put the name of the object in the brackets, and if it exists in the area, the robot will find it immediately, avoiding all eventual obstacles. Here's how to find the nearest scrap, for example :

```
        find(Scrap);
```

6.2.9 *Instructions `if` et `else`*

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 `{}`.

Note : you mustn't type a semi-colon `;` to complete the instruction. The instruction else doesn't need either a semi-colon or brackets.

Here's an example of how to disintegrate a metal object, or, if the object is made of something else, how to destroy it :

```
	if( load.material == Metal )
	{
		find(Disintegrator);
		drop();
		move(-10);
	}
	else
	{
		find(Destroyer);
		drop();
		move(-20);
	}
```

6.2.10 Instruction `detect`

With the instruction `detect`, the robot can check to see if a specific object is in front of it. You can use this in combination with the instructions `if` and `else`.

```
	if ( detect(Barrier) )
	{
		turn(90);
	}
	else
	{
		move(20);
	}
```

7 And afterwards

CeeBot-Teen is an introduction to computer programming. Lots of other instructions in the game will lead you to learn more complex notions of programming. And when you've mastered CeeBot-Teen, check out CeeBot-A.

8 Development team

Daniel Roux, Denis Dumoulin, Otto Kölbl, Michael Walz

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