Programming Principles

Unit 1

Introduction to Ceebot

What is Ceebot?

An Animated Virtual World of Robots

A Programming Environment

A Programming Language

```
fire(1);
```

```
grab();
```

```
move(20);
```

```
message("Program Completed");
```

```
turn(90);
```

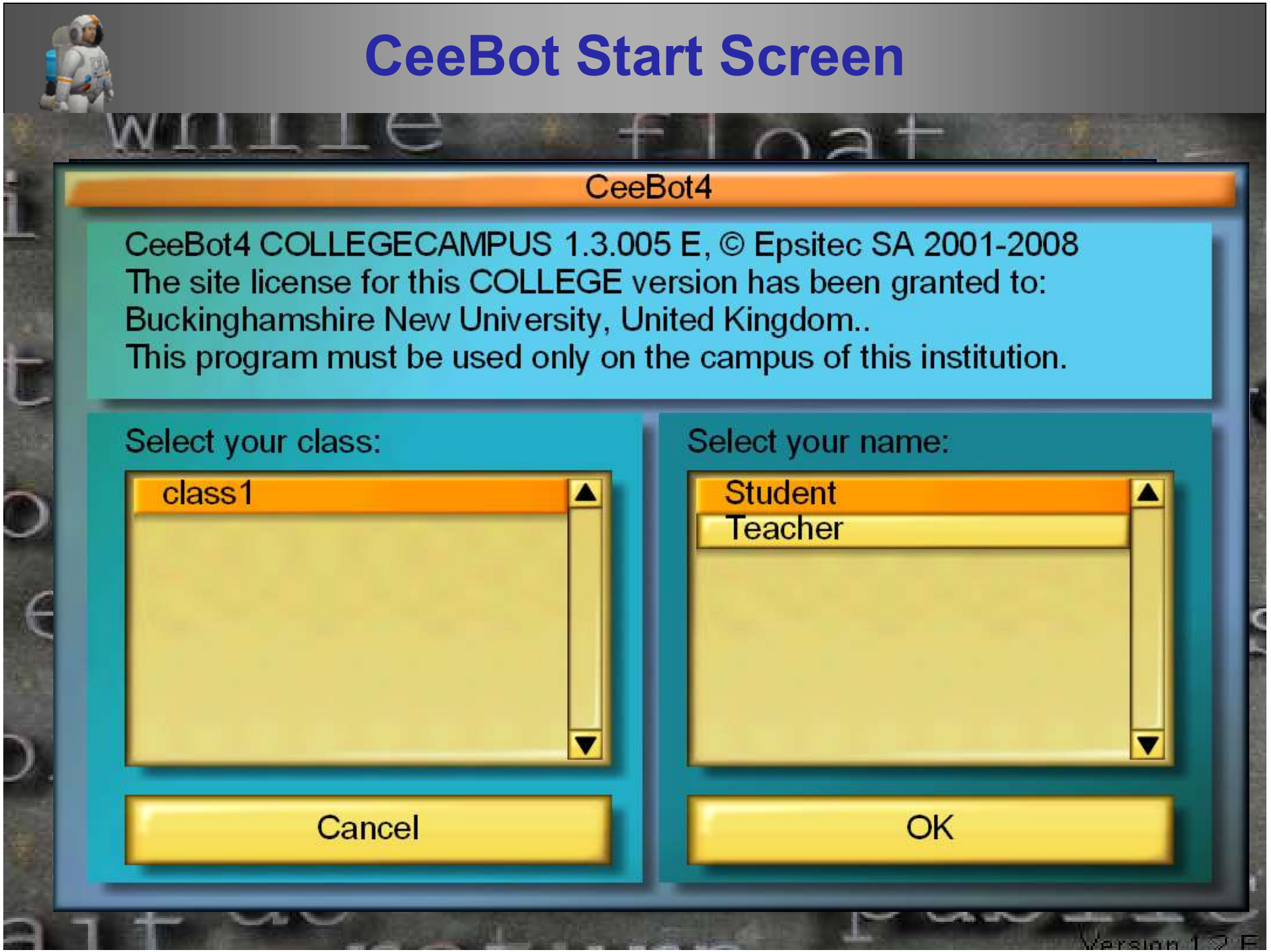
```
while (i <= 5)
{
    GetItem();
    i++;
}
```

```
if (count < 10)
{
    turn(angle);
    wait(0.5);
    drop();
}
```

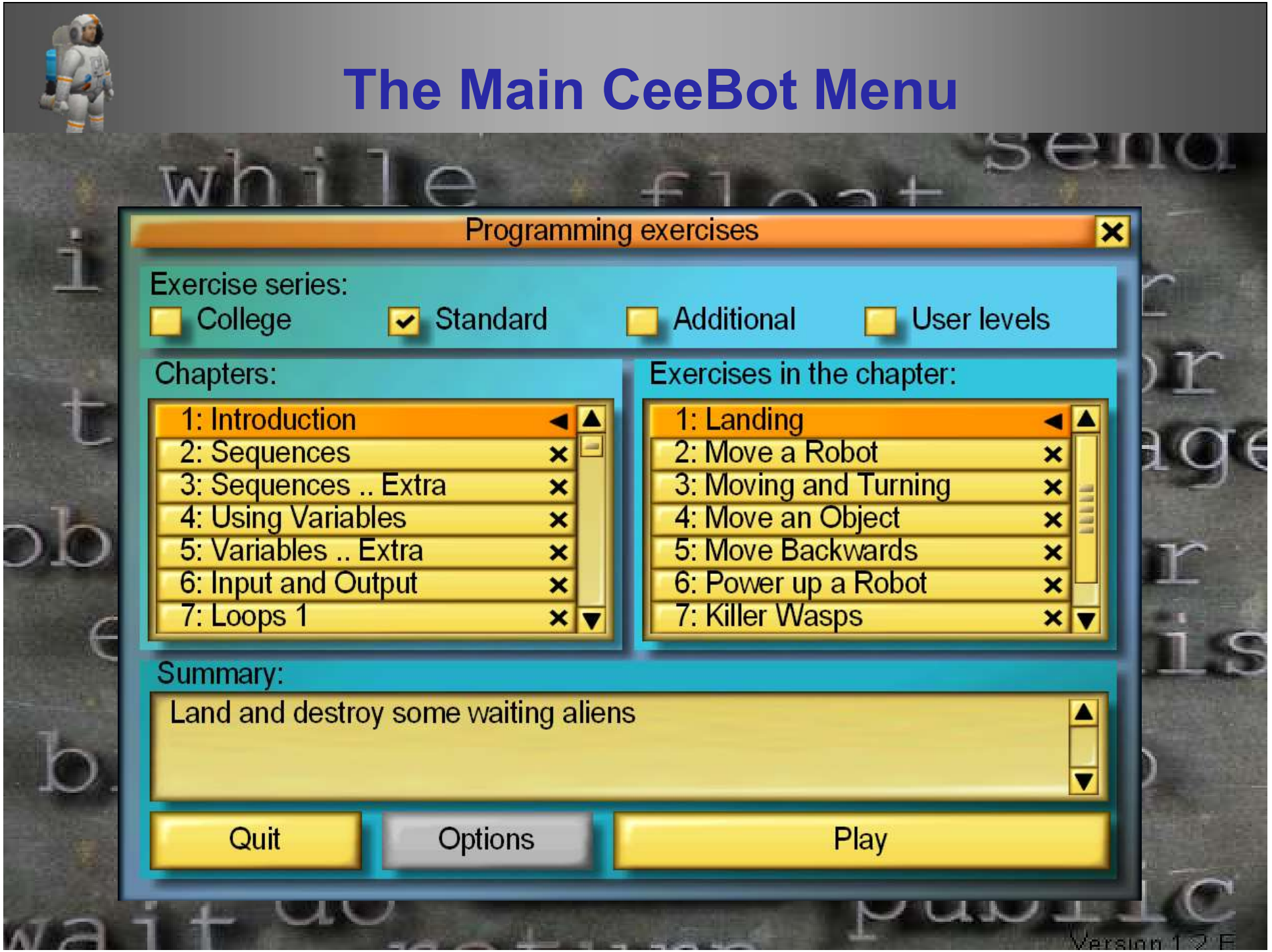
Using C#/C++/Java syntax

How to Use Ceebot

CeeBot Start Screen

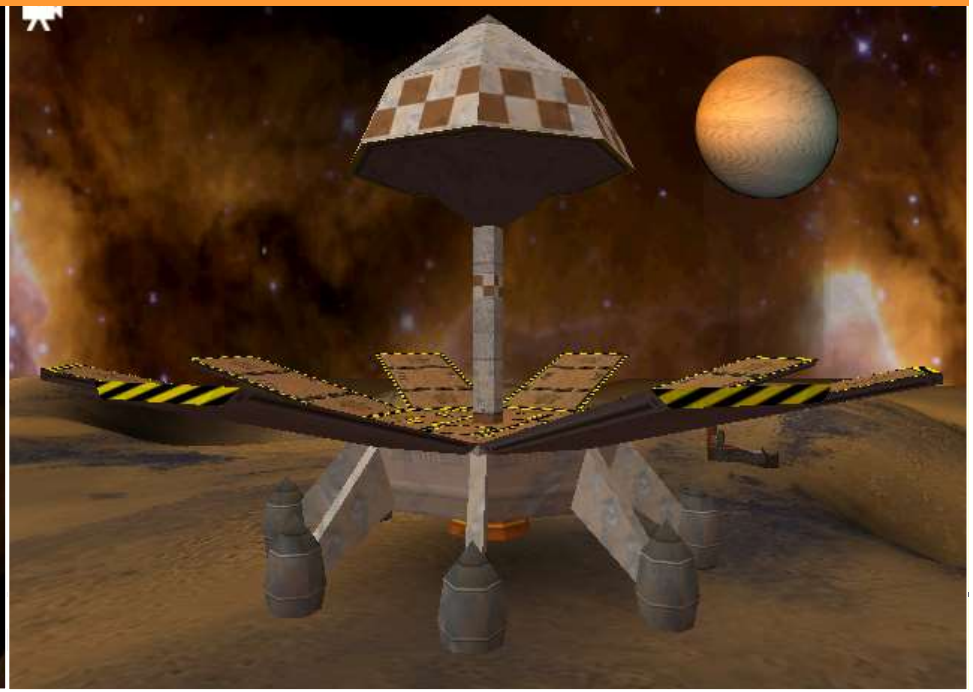


The Main CeeBot Menu



Introductory Video

The introductory video can be skipped by pressing the [esc] key




Push F1 to get your instructions
Click on the shooter
You ==== The astronaut
Student
x.5
x 1
x 2
x 3


access the editor
Wheeled shooter
1
2
3
4
x.5
x 1
x 2
x 3
{..}

The CeeBot Editor/Compiler

Program editor

```
extern void object::Task1_1()
{
    fire(1);
}
```

Enter instructions here

Then click OK

OK Cancel 1


Running the program
click here
Danger!
low robot
shield level
1: Task1_1
2
3
4
x.5
x 2
x 1
x 3
Execute the selected program


Ant fatally wounded
Wheeled shooter
1: Task1_1
2
3
4
x.5
x 2
x 1
x 3
Execute the selected program


<<< Well done, mission accomplished >>>


Bot destroyed
<<< Sorry, mission failed >>>


STOP
CeeBot4
Options
Restart
Abort

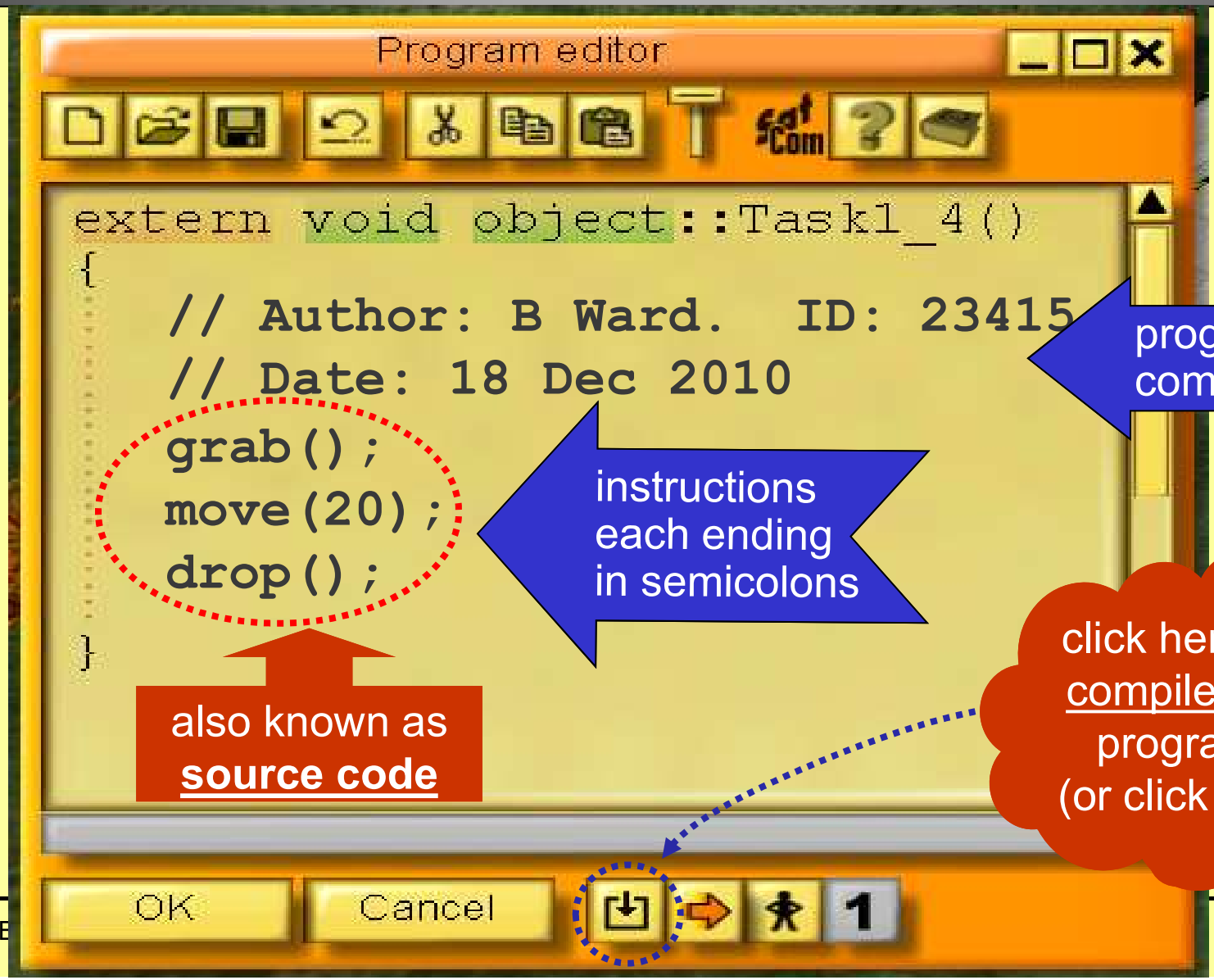
Using Ceebot

to write

larger programs


Transport a Titanium cube to the platform
Select the first program slot .. and then the editor
Wheeled grabber
1
2
3
4
{..}
x.5
x 1
x 2
x 3

Use the Editor




Program editor
extern void object::Task1_4()
{
// Author: Brian Ward
// Date: 28 June 2005
grab();
move(20)
drop();
}
The compiler will report syntax errors
Semicolon terminator missing
OK
Cancel
1


Run
(execute)
the
program
Wheeled grabber
1: Task1_4
2
3
4
x.5
x 1
x 2
x 3

Some Useful Instructions

fire(…);

move(…);

turn(…);

grab();

drop();

wait(…);

message(…);

pendown();

red();

Put them in the right order and use the correct parameters to create your program.

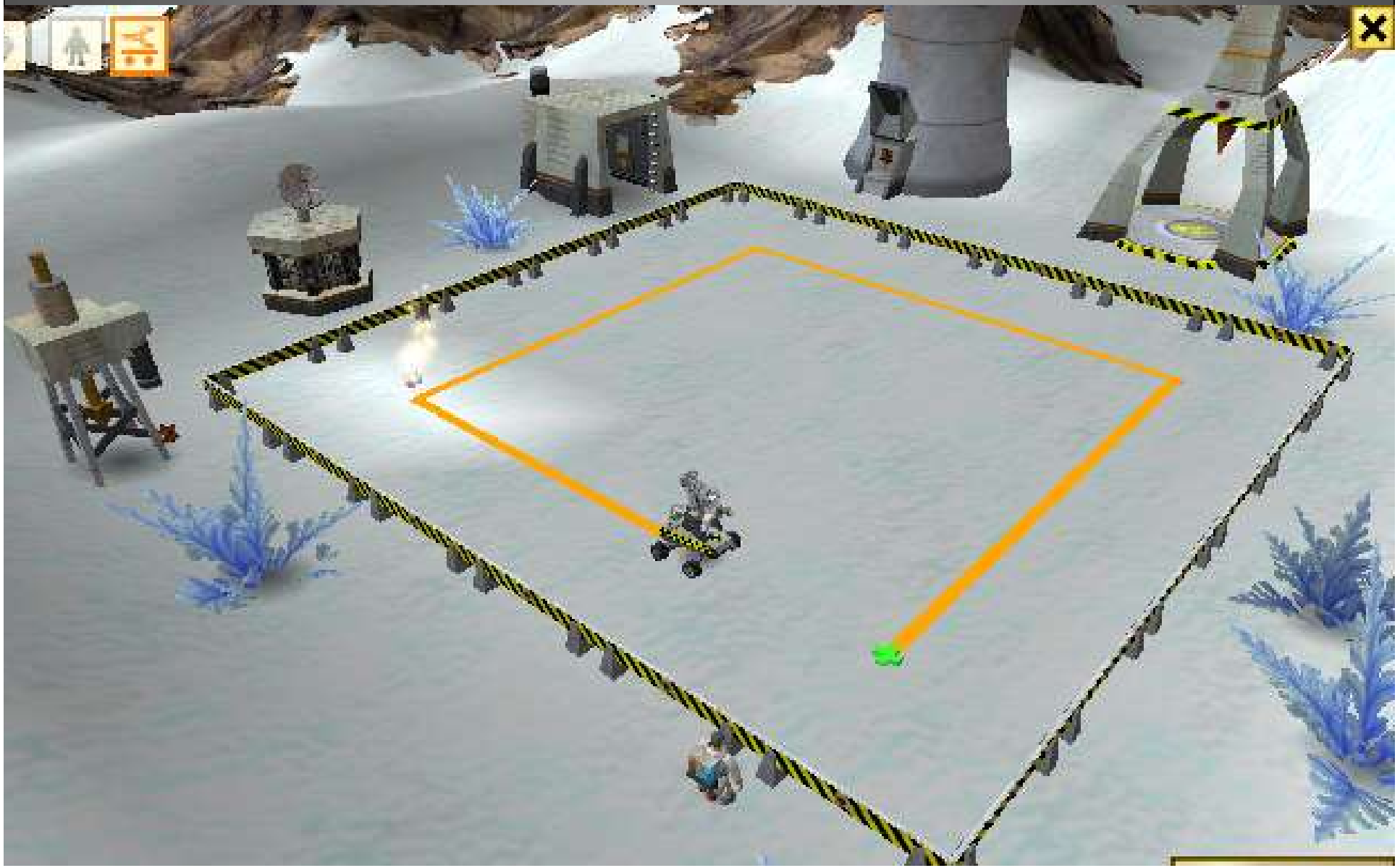
==========

each instruction ends with a semicolon ;

Algorithms

A plan for the program using english-like statements

Draw a Square



Algorithm … then Code

The square is orange and 20 metres in size

the steps are numbered in the order of execution

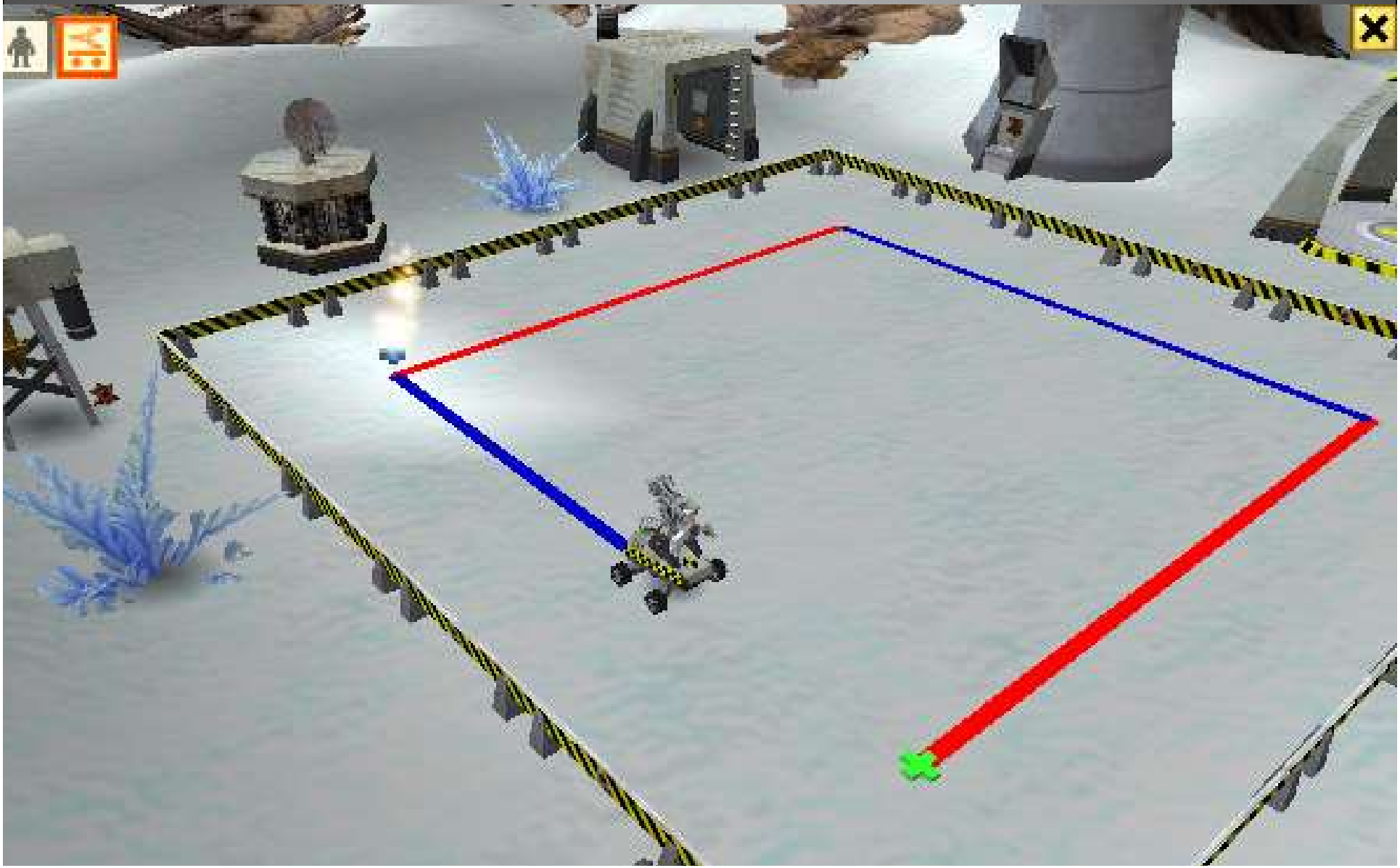
Algorithm

1. Choose orange colour
2. put pen down
3. move 20 metres forward
4. turn 90 degrees anti clockwise
5. move 20 metres forward
6. turn 90 degrees anti clockwise
7. move 20 metres forward
8. turn 90 degrees anti clockwise
9. move 20 metres forward
10. turn 90 degrees anti clockwise
11. raise pen

Program Code

```
extern void object::Task2_1()
{   // Author: BWard.  ID:156874
    // Course:  BSc Comp
    // Date: 18/12/2010
        orange();
        pendown();
        move(20);
        turn(90);
        move(20);
        turn(90);
        move(20);
        turn(90);
        move(20);
        turn(90);
        penup();
}
```

Variation



Draw a different square

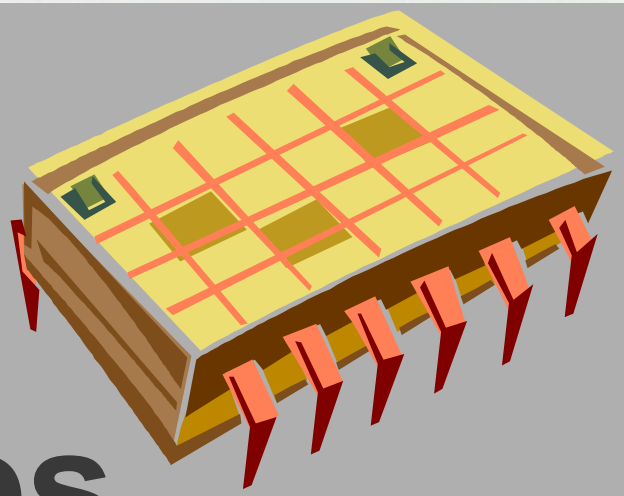
Make left and right sides blue, top and bottom sides red

Algorithm

1. choose red colour
2. put pen down
3. move 20 metres forward
4. turn 90 degrees anti clockwise
5. choose blue colour
6. move 20 metres forward
7. turn 90 degrees anti clockwise
8. choose red colour
9. move 20 metres forward
10. turn 90 degrees anti clockwise
11. choose blue colour
12. move 20 metres forward
13. turn 90 degrees anti clockwise
14. raise pen

Program Code

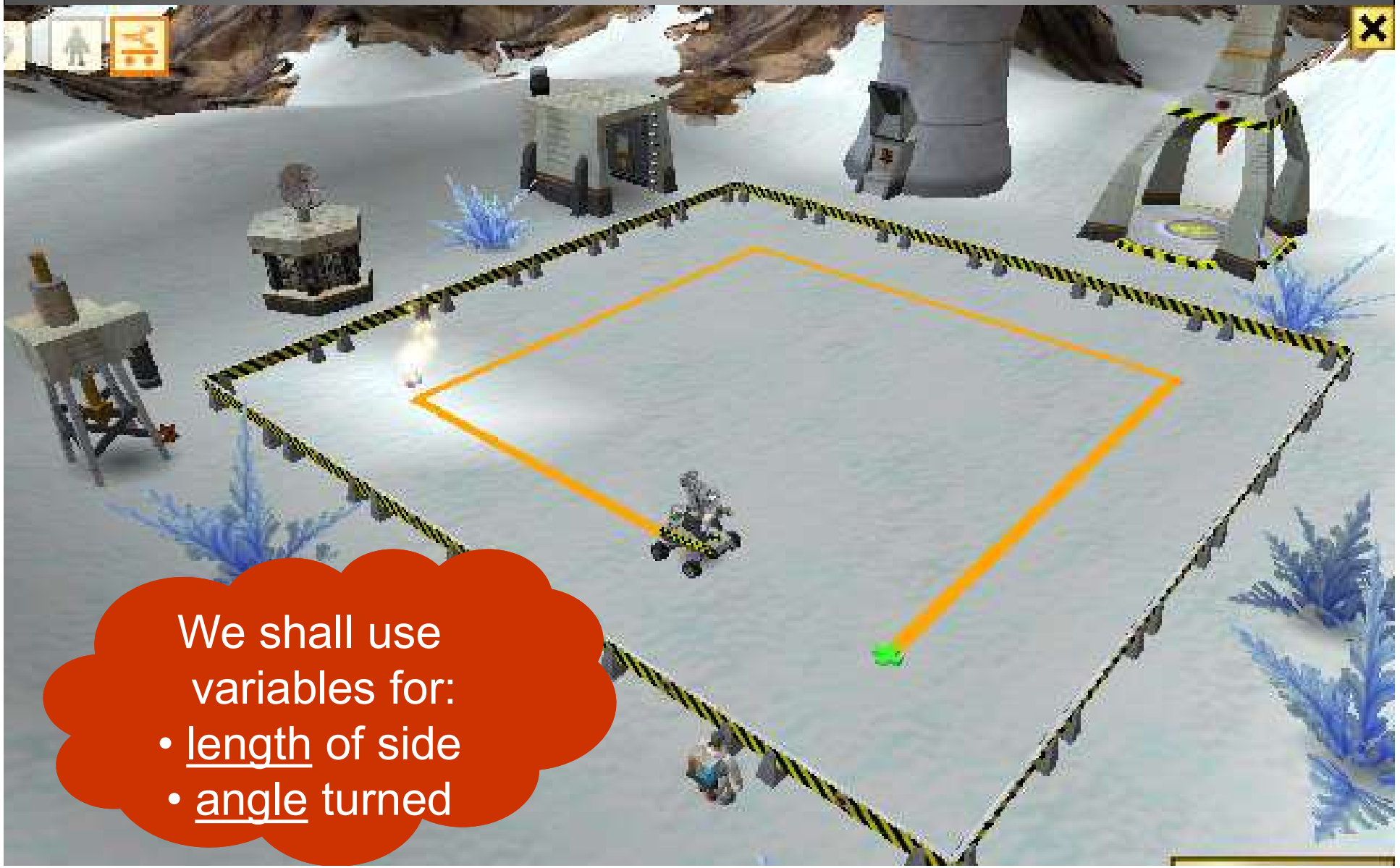
```
extern void object::Task2_1()
{
        red();
        pendown();
        move(20);
        turn(90);
        blue();
        move(20);
        turn(90);
        red();
        move(20);
        turn(90);
        blue();
        move(20);
        turn(90);
        penup();
}
```



Variables

how to store information temporarily in a program

Draw a Square using variables



The Code to Draw a Square

```
extern void object::Task2_1()
{
    // variation of Square program using variables

    float length;    // declare a float variable called length
    float angle;     // declare a float variable called angle
    length = 20;     // store 20 in length
    angle = 90;      // store 90 in angle

    orange();        // set drawing colour
    pendown();       // prepare to draw

    move(length);    // move using length variable
    turn(angle);      //  turn using angle variable

    move(length);  // move using length variable
    turn(angle);     // turn using angle variable
    etc.
}
```

Using a robot to do maths!

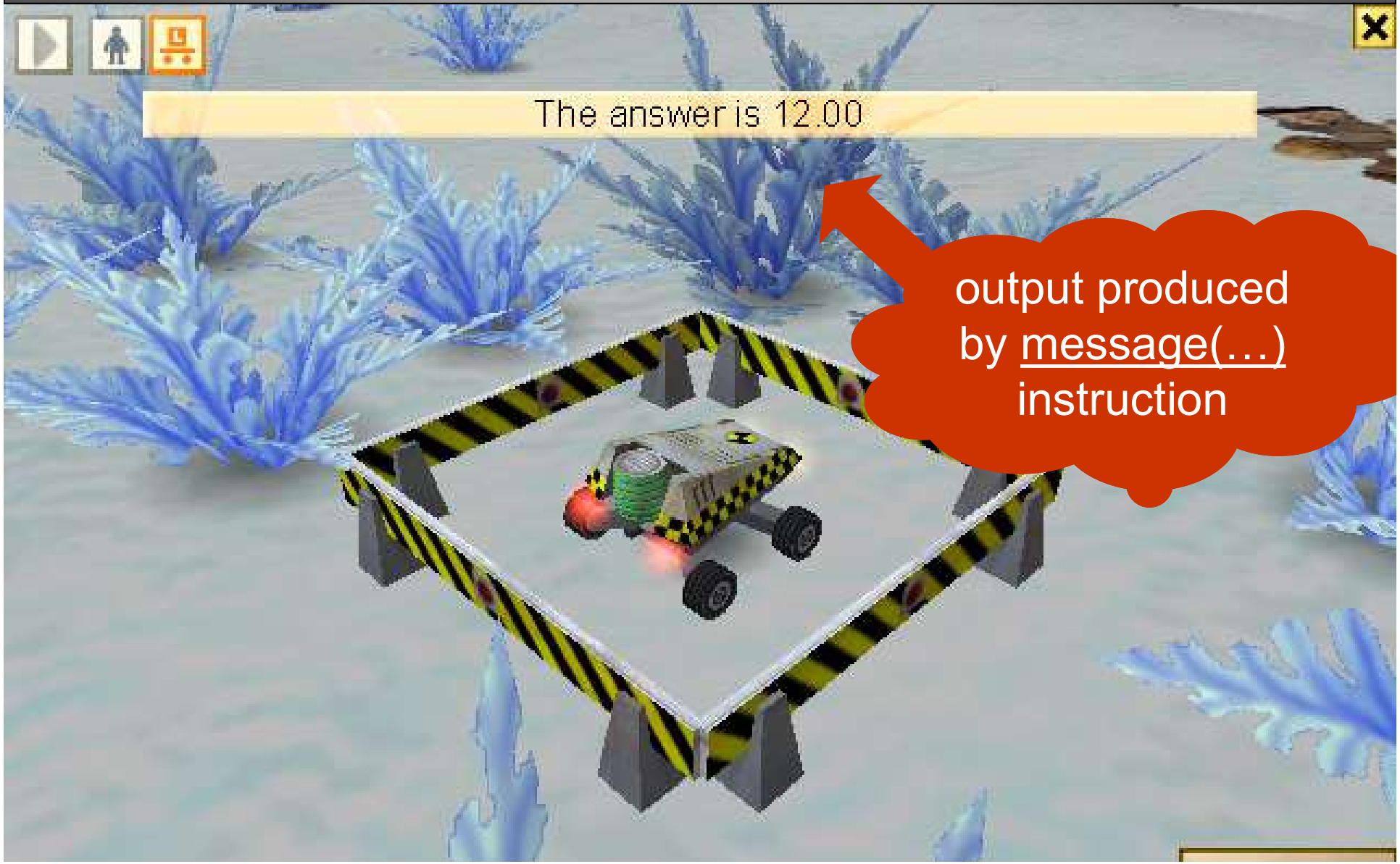
using variables

Using a robot to do maths

```
extern void object::Task5_3()
{
        float a, b;          // declare 2 variables a and b
        a = 5;               // store 5 in a
        a = a + 2;           // add 2 to a and store again
        b = a*2 - 10;        // multiply by 2 and subtract 10
        b = b/2;             // divide by 2
        a = b + a + 3;       // add b, a, 3
        message("The answer is " + a);      // display result
}
```

New instruction for screen output display (see next week)

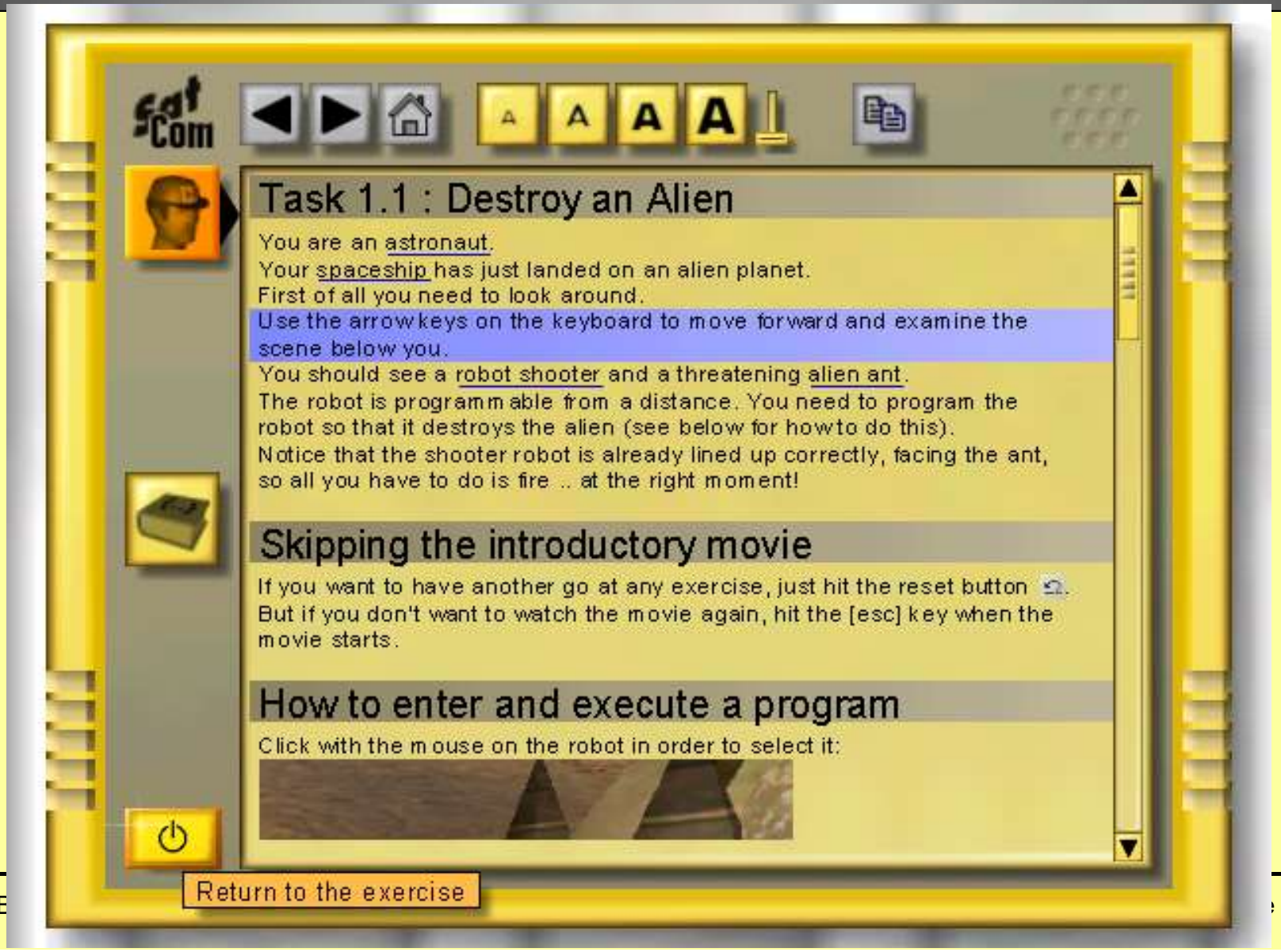
Result




What I learned today
1. The Ceebot editor
2. Compiling and running programs
3. Program comments
4. Ceebot instructions
5. Syntax errors
6. Instruction Sequences
7. Algorithms
8. Variables for storing data
9. Doing arithmetic
10.message() instruction
THE END

Extra Reading

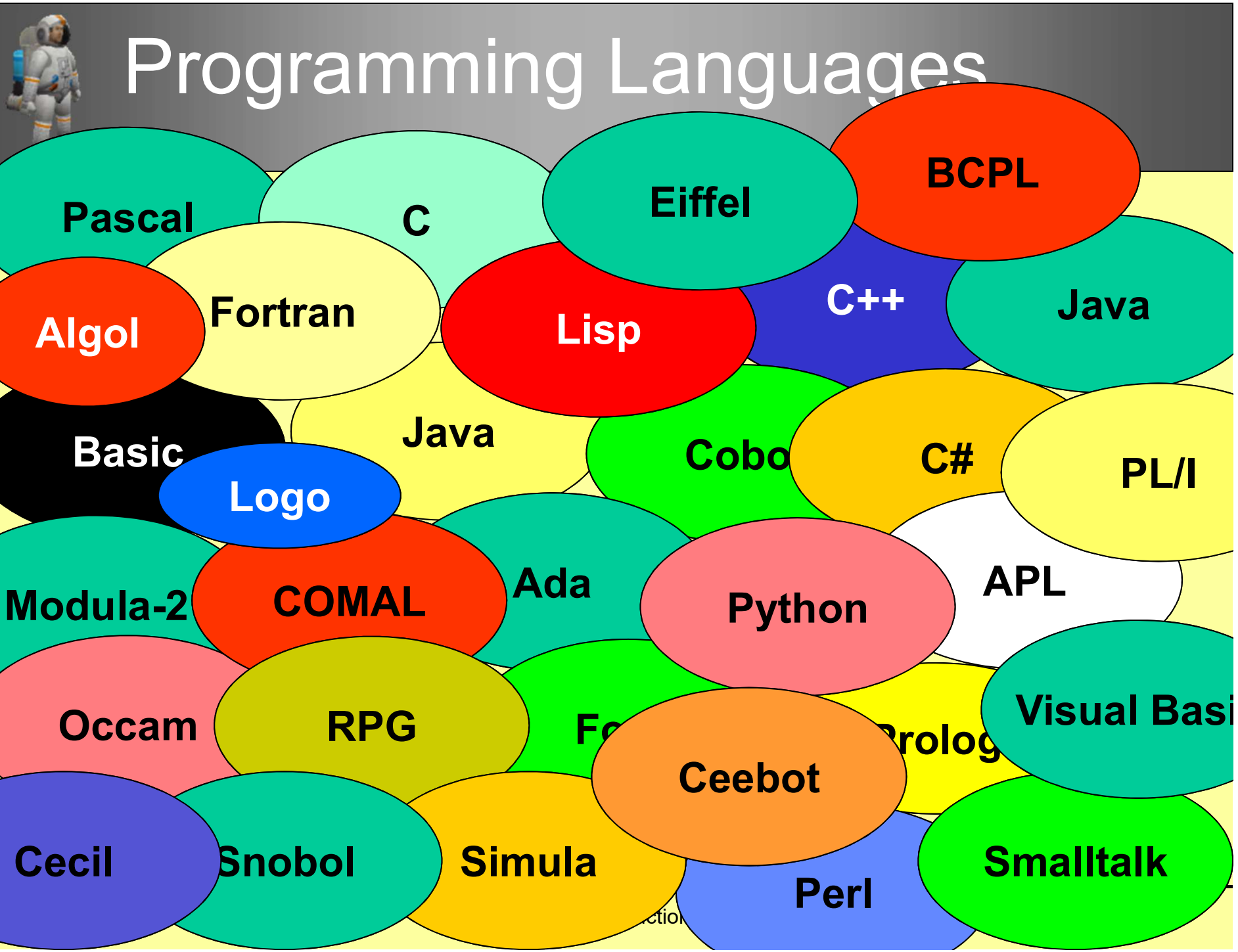
Push [F1] to get instructions



What is a computer program ???

A Program is:

- A set of instructions to the computer
- To make the computer do something useful
- Designed by programmers
- Written in a language like Ceebot, C++, Java
- Other high-level languages could be used .. e.g. Cobol, Fortran, Pascal, Basic, etc, etc.


Programming Languages
Pascal
C
Eiffel
BCPL
Algol
Fortran
Lisp
C++
Java
Basic
Java
Cobo
C#
PL/I
Logo
Modula-2
COMAL
Ada
Python
APL
Occam
RPG
Prolog
Visual Basi
Ceebot
Cecil
Snobol
Simula
Perl
Smalltalk

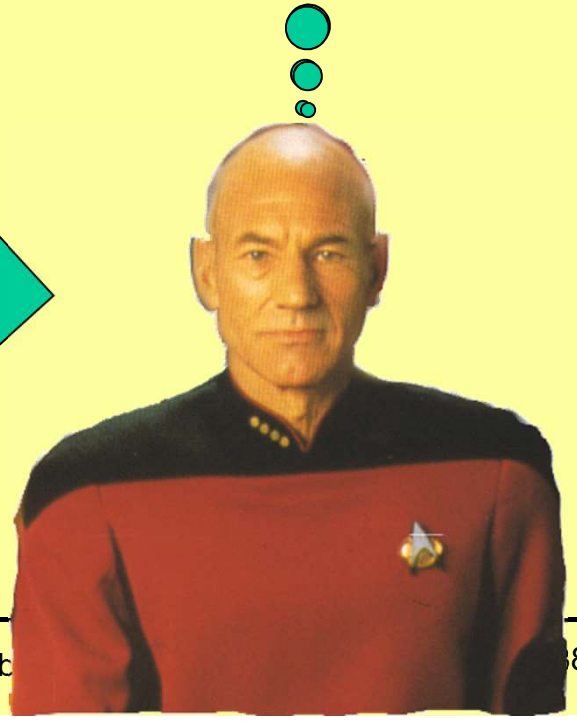
Star Trek Talk

How do StarTrek officers understand aliens?

Jaggh kapra plaakh

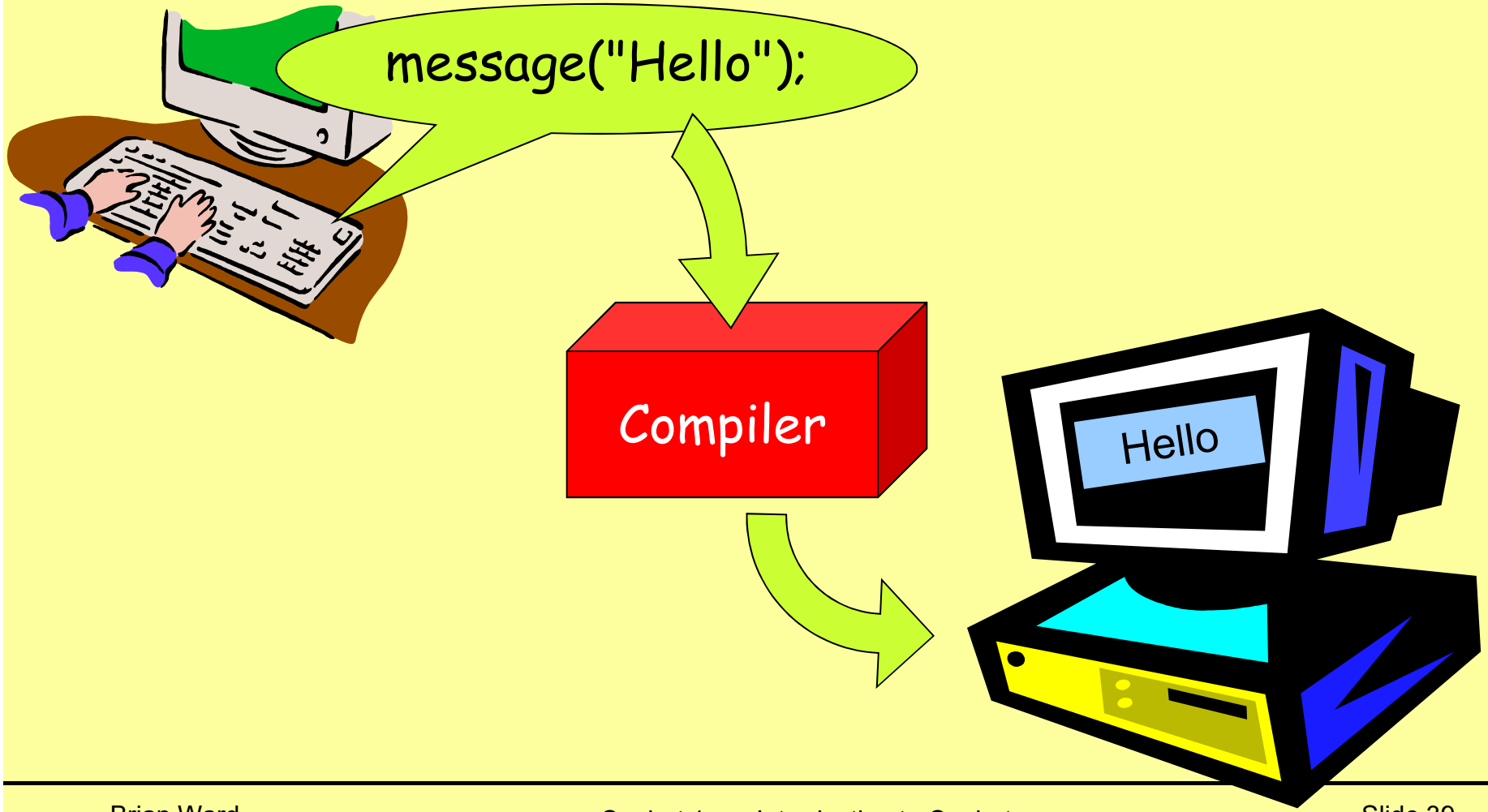
Universal Translator

Prepare for disembowelling

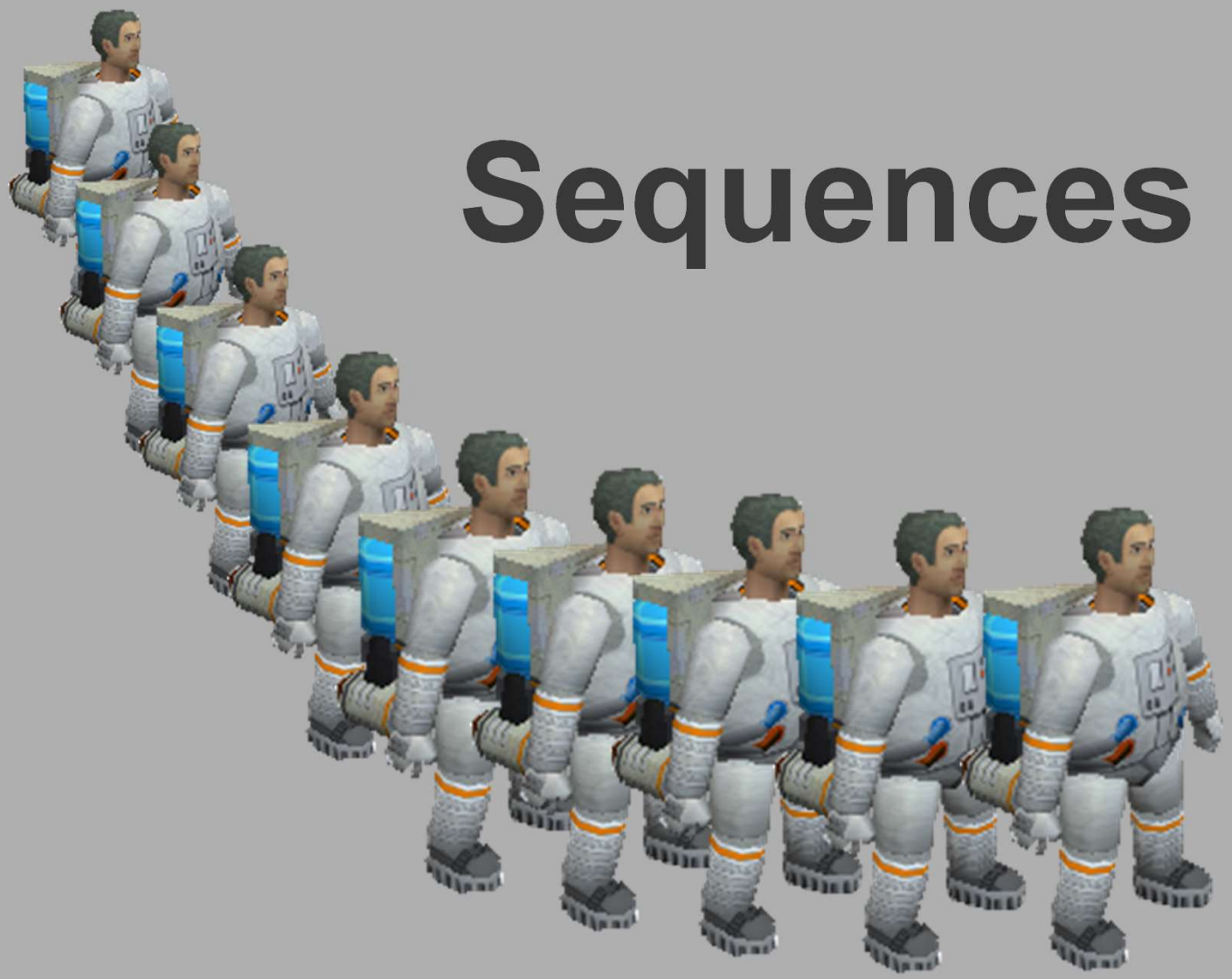


Computer Translation

How do computers understand our instructions?



Sequences



How a Sequence of instructions is Programmed

a sequence block is enclosed in curly brackets { }

```
{
    move(20);
    grab();
    turn(-90);
    move(10);
    drop();
}
```

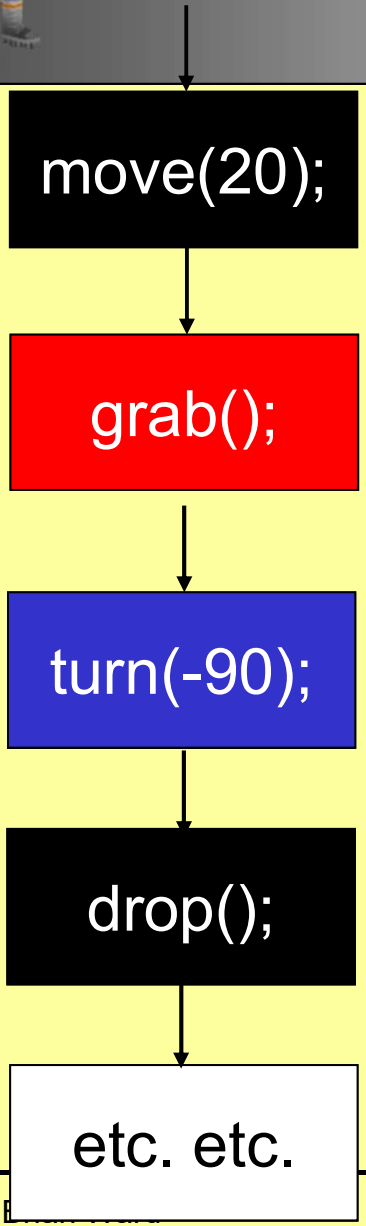
instructions are all in lowercase and end with a semicolon ;

The instructions are **indented** (using tab key or spaces) and placed in the order in which they are to be executed (from top to bottom)

A typical sequence

```
{
    grab();
    move(20);
    drop();
}
```

Sequence



The **sequence** is :

- a block of instructions .. one after the other
- with no deviation or repetition

The **order** of the instructions in a sequence is very important ……
if the order is changed, so is the logic of the program

The sequence is a basic construct of all programming languages

Why use algorithms?

Algorithms can be used to <u>design</u> programs before coding starts

This is especially important when writing larger and more complicated programs

We use algorithms from the start, so you learn Good Practice

Program comments

Good Practice

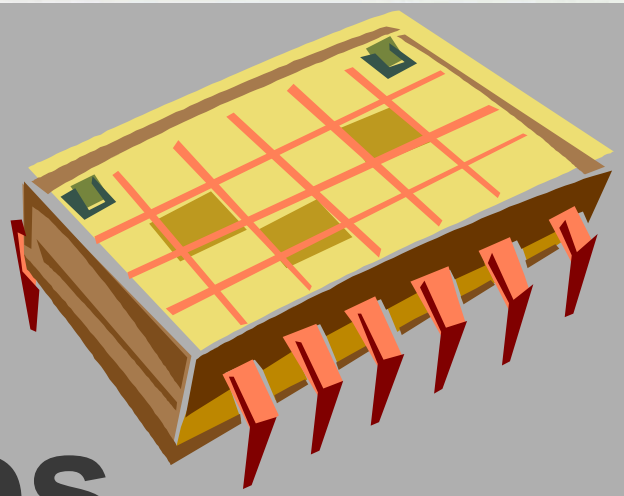
Use plenty of comments in all your programs

```
// this is a one line comment
// the compiler ignores this line

/*  this is a multiple-line
comment and the compiler will
ignore everything here
*/
```

Use comments to:
- identify author and program
- explain trickier parts

Example Uses

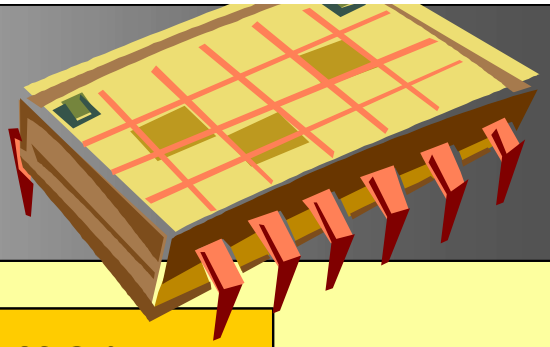
```
extern void object::Task0_7()
{
    /* Author: B Ward
       Date: 14/08/05
       Task: install power cell  */

       grab();      // pick up object
       turn(-90);  // turn clockwise
       wait(0.5);  // pause for 0.5 sec
       drop();      // drop object
}
```



Variables

how to store information temporarily in a program

What is a Variable?



1. A storage area in the computer memory
2. Can store information for use later in the program
3. A variable can be set up to store different types of data: numbers, words, etc.
3. The contents may change as the program runs (hence the name: variable)
4. Variables need to be given unique **names**
5. A variable name is also known as an **identifier**

Declaring Variables

Before you can use a variable, you must declare it

This means giving the variable a type and a name

```
string  firstname;
int  count;
float  number ;
```

type

name

identifiers (variable names)

Rules for identifiers

1. Name must start with a letter
2. No spaces in the name
3. Can only have letters, digits, underscore
4. No reserved words (move, turn, etc.)
5. Length, length and LENGTH are all different variables
   (i.e. Ceebot is case-sensitive)
6. Good Practice: always choose names that are meaningful

Name OK or not?	
My_Name	✓
my-name	✗
1stname	✗
D2	✓
Number4	✓
%cost	✗
first name	✗

Data Types for Ceebot Variables

There are 5 main data types for variables

int
Can store whole numbers e.g.
3 0 -261 46 -7

float
Can store numbers with decimal places e.g.
10.67 -0.05 13.0 176.4

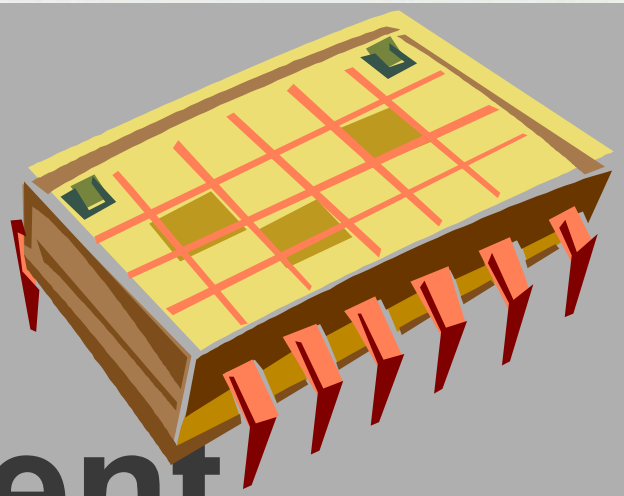
string
Can store text (strings of characters) e.g.
"High Wycombe" "Brian"

object
Can store details of an object e.g. Titanium, PowerCell

boolean
Can only be true or false

point
Can hold position coordinates

Each type needs a different amount of storage space

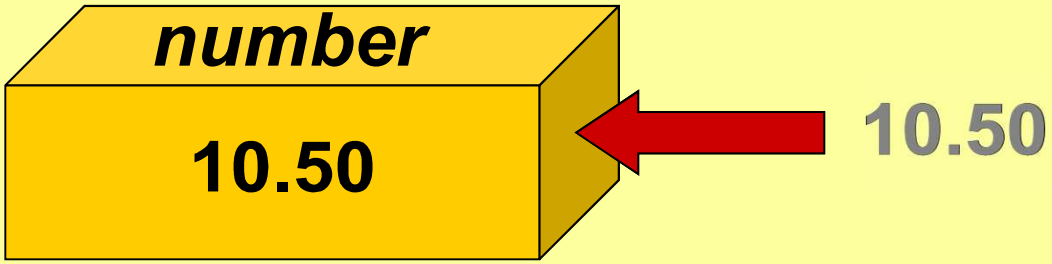


Assignment

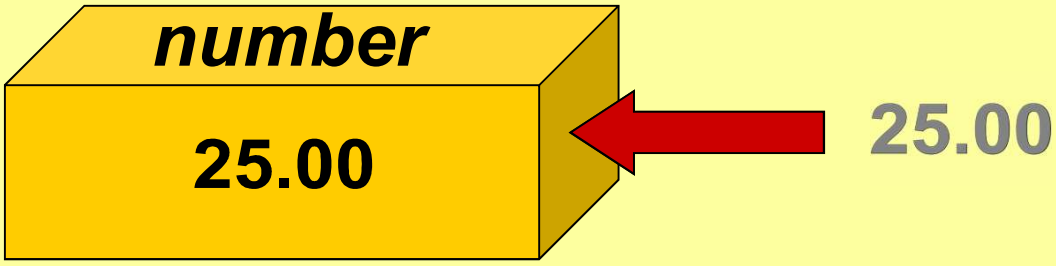
how to put values into variables

Store values in a variable

```
float number ;          // declare variable
number = 10.50 ;        // store 10.50
```




```
number = 25.00 ;        // store 25.00
```



Assignments

Information can be stored in a variable using:

the assignment statement
and assignment operator (=)

e.g:

```
age = 25 ;
wage = 15.50 ;
choice = "A" ;
name = "Brian Ward" ;
title = "Menu List" ;
```

Computer Memory

Variable	Contents
age	25
choice	A
wage	15.50
name	Brian Ward
title	Menu List

Initialising Variables

Variables can be given an initial value at the same time as they are declared

e.g.

```
int  count = 0 ;

float price = 7.54 ;

string name = "Joe Smith" ;
```

Instructions and parameters

Which of these instructions have parameters?

move(20); ✓

fire(1); ✓

grab();

pendown();

wait(0.5); ✓

drop();

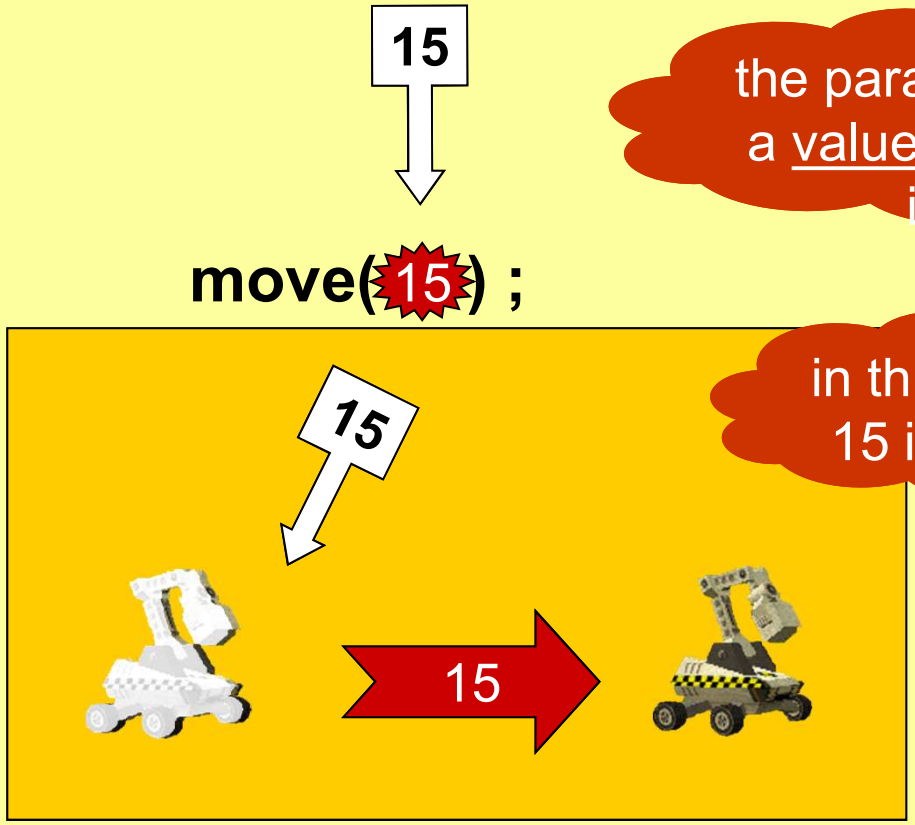
red();

turn(90); ✓

Note: most instructions have brackets, but not all use them

Parameters

Brackets ( ) act like a doorway into the instruction



The parameter is used to complete the instruction: **move(15);**