DOBOT MAGICIAN LITE

– A MINIATURE INDUSTRIAL ROBOT

Math with a robot

Patrick Liljegren Gregersen and Niels Erik Wegge, Birkerød Gymnasium.

Version 3

Contents:

Notes for the teacher:

- This unit can be used as an introduction to trigonometry (right angled geometry can be motivated by the need to maneuver the arms of the Dobot) and to 2D and 3D coordinate systems.
- The unit is designed for a single module (90 minutes), but it may be necessary to do 2.1.4 and perhaps some of 3.1.2 in a later module.
- If possible, have an assistant teacher!
- We recommend that
  - the Dobot software is installed and tested on a suitable number of computers *before* start.
  - the programs are up and running and robots connected before start.
  - the first 5 minutes is used to explain the aim of the unit.
- Two students per robot is ideal. Consider splitting the unit over two modules with half a class per module.
- The present manual should be handed out on paper: 1 per group (to avoid too many screens).
- Afterwards: Spend a module to consolidate what has been learned.

Introduction to the Dobot

Patrick Liljegren Gregersen and Niels Erik Wegge
Birkerød Gymnasium

1.1 Introduction

The Dobot is a stationary industrial robot *en miniature*. It works with high precision and is actually used for industrial processes, e.g. in the medical industry and for other laboratory work. Typical tasks include grabbing and lifting objects and moving them from one place to another. Our Dobot can also write and draw.

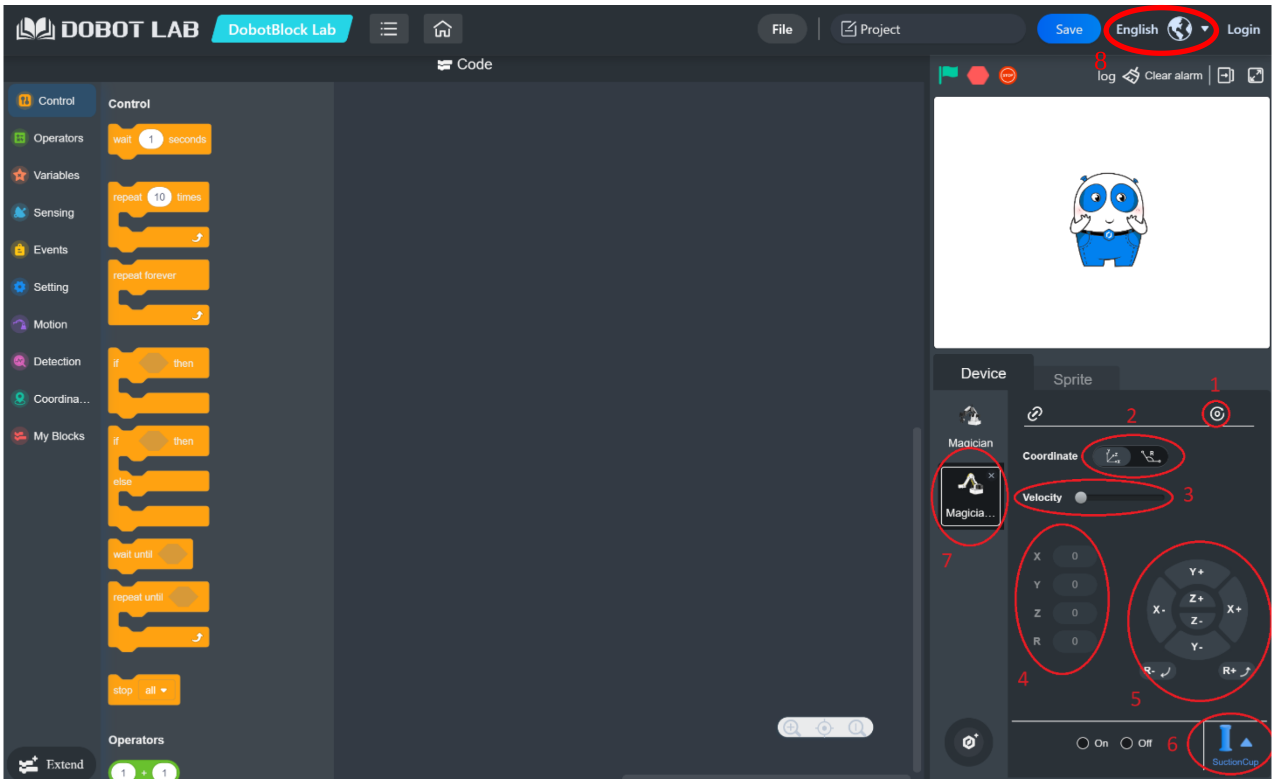
The Dobot is stationary, so there are limits to how high and how far its arm can reach, and to how much it can turn around. The robotics people say the *workspace* is limited – and mathematicians say the *domain* of the parameters for the robot is limited. Our first task is to investigate this, and then we will look into the coordinate system(s) used by the Dobot.

Above all, we of course need to learn to communicate with the Dobot. It is programmed with Python, one of the most widespread programming languages in robotics and science. The grammar of Python can be difficult to get exactly right, but fortunately the Dobot works with *block programming*, which makes life much easier. A bit like spell check on your computer.

Remember that the Dobot is very expensive – so treat it gently 😊

1.2 Get connected

1) The Dobot is controlled through the program *Dobot Lab* . Open it and choose *DobotBlock Lab*.
2) Turn on the Dobot and connect it to the computer via a USB cable.
3) Select *Magician Lite* and English language (buttons 7 and 8 on figure on next page).
4) Press the Home button (number 1).



Important functionalities in the Dobot Lab. Numbers in red ellipses are explained below.

1) The ”Home”-button. Is used when connecting to the Dobot at start and whenever the robot’s arm is reset to its home position.
2) We can switch between either the coordinates for the end point of the arm or the angles for the parts (joints) of the arms.
3) Changes the speed.
4) The coordinates (or angles) for the robot in the current state.
5) Maneuvering the robot’s arm.
6) To select the gripping or drawing tool placed at the end of the Dobot’s arm.
7) Our robot is a *Magician Lite* – select this at the very beginning of a session.
8) Selection of language.

When you have connected the Dobot to the computer with the USB cable and have turned on the Dobot, choose *Magician Lite* (button 7) and finally press the Home-button (1). You may need to click “connect” in the pop-up window and close some other pop-ups – then press Home one more time. Now you are connected to the robot!

1.3 Free play

What do the different buttons in *DobotLab* do? Try! Remember that you can always get out of a sticky situation by pressing Home (or calling the teacher).

2.1 The Dobot's workspace and domain

Now we will explore the functionality of the Dobot and the math behind it in a more structured way.

2.1.2 Control the Dobot with the x-y-z-buttons

Use the x-y-z-buttons to move the robot's arm different places. Notice that you can change the speed (Velocity, button 3).

1) Which of the buttons make the tip of the arm move up and down? x, y or z?
2) What do the other two buttons do?

2.1.2 Move the Dobot's arm manually

Did you notice the little button at the end of the Dobot's arm? It has a pictogram of a lock. When activating this button, the arm is released, and you can move to where you want. Never use force – and do not move the arm unless the lock-button is activated!

1) Notice the coordinate numbers for x-y-z when the arm is in different positions. Where should one move the arm to get positive / negative …
    - …z-values?
    - …y-values?
    - …x-values?
2) Move the robot's arm somewhere close to the coordinates $(x, y, z) = (200, -200, 0)$. You can fine tune with the x-y-z-buttons on the screen, but the numbers need not be very precise. Now point where you think the point with coordinates $(x, y, z) = (200, 200, 0)$ is positioned – and then use the y-button to move the tip of the robot's arm over there. Were you right?
3) So, how does the x-y-z coordinate system work? (where is the x-axis, y-axis, z-axis – and where are the positive/negative values?)

2.1.3 Limitations for the x-y-z coordinates

We will now examine the *workspace* of the robot.

1) Release the arm and move it close to the coordinates $(200,0,0)$ – again: it does not have to be very prescise. Then use the z button to move the tip of the arm up and down. Why can it not get all the way up and down?
2) Move the arm back to coordinates $(200,0,0)$ – and this time move the tip of the arm back and forth with the x-button. What is the largest and the smallest value for x that you can get? Does the length of the interval depend on initial value of y and z? (Yes!)

Now you have seen that the permitted interval for any of the coordinates depends on the values of the other coordinates. Obviously, this is because of the way the Dobot is constructed, mechanically – and this makes it annoyingly difficult to control the robot. When the arm moves from one place to another, each of the three coordinates must always be within the interval of permitted values, and these intervals continuously change when the arm moves. The good news is that it gets much simpler when we use *angles* instead of x-y-z-coordinates. Read on!

2.1.4 Domains in Mathematics

In 2.1.3 you probably found out that the Dobot's workspace is complicated to describe: the permitted values for each of the variable coordinates depend on the values of the other two coordinated. The workspace – or the domain as mathematicians like to call it – is practically impossible to write down in x-y-z coordinates.

All the functions you will come across in school have much simpler domains. This is not least because they will usually only have one and not three variables.

The examples and exercises below are meant for the next math lesson, not for now – instead, *go to section 2.2.*

1) Consider the linear function $f(x) = 2x - 3$. There is no immediate limitation to which x-values, so the domain is $\mathbb{R}$, the set of all real numbers. But, if Albert (for some reason) wants the values of $f(x)$ to fall between -3 and 5, to what domain must f then be limited?
2) Consider the function $f(x) = \sqrt{4 - x^2}$. We cannot take the squareroot of a negative number, so for this function, x^2 has to stay less than (or equal to) 4, and so only x-values from -2 til 2 are allowed. The domain of f is $[-2; 2]$.
3) Find the domain for the following functions:
    a. $g(x) = \sqrt{100 - x^2}$.
    b. $L(x) = \sqrt{x^2 - 100}$.
    c. $h(x) = \sqrt{64 - (2x)^2}$
    d. $p(x) = \sqrt{27 - x^3}$
    e. $s(x) = \sqrt{a - x^n}$
4) Challenge: The function $f(x, y) = \sqrt{xy}$ has two variables. How would you describe its domain?

Notice that the domains in exercises 1) – 3) are given by fixed intervals, while in 4) the domain for each of x and y are interdependent. This resembles how the workspace of the Dobot is limited.

2.2 The Dobot's coordinate system

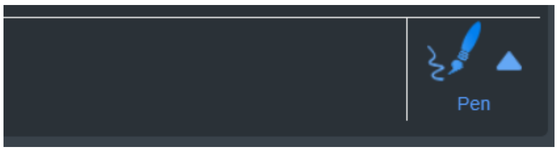
Now it is time to find out exactly what the x-y-z-numbers in *DobotLab* represent. In other words: what coordinate system does the Dobot use? What does, for instance $(x, y, z) = (150, -120, 0)$ mean? Only when we know the answer to that, we will be able to control the robot and have its arm move exactly as we want.

First, we will find the origin of the coordinate system. Where is (0,0,0)? To answer the question, we will need to understand and use the *angles* for the robot's arm.

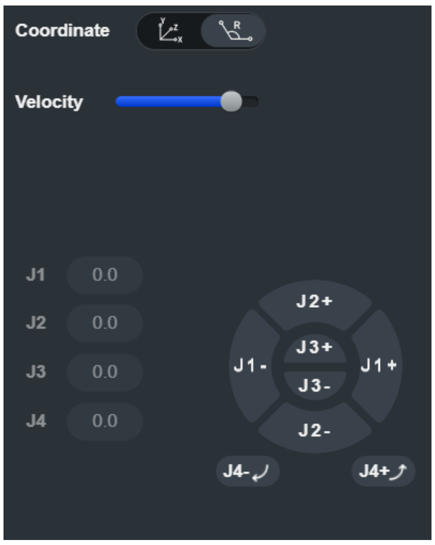
2.2.1 The robot's angular coordinates

Pen Holder

1) Begin by attaching the pen holder and select "Pen" in *DobotLab*.
   It is button 6 on the figure page 3.
   But: do <u>not</u> put a pen in the pen holder yet!



2) Reset the position of the robot's arm with the "Home"-button – the arm will nicely and slowly go back to its initial position.
   a. Does it make sense that y is now precisely 0?
   b. Notice the z-value. If this number is a distance in millimeters, where are then the points with $z = 0$? Are they at the level of the table? (No!)
   c. Look at the x-coordinate. Where (approximately) will we find the zero for the x-axis?
   d. The coordinate values for x and z in the "home"-position are not very simple numbers – but look at the "Joint"-coordinates. Can you now see what values the "Home"-button resets?



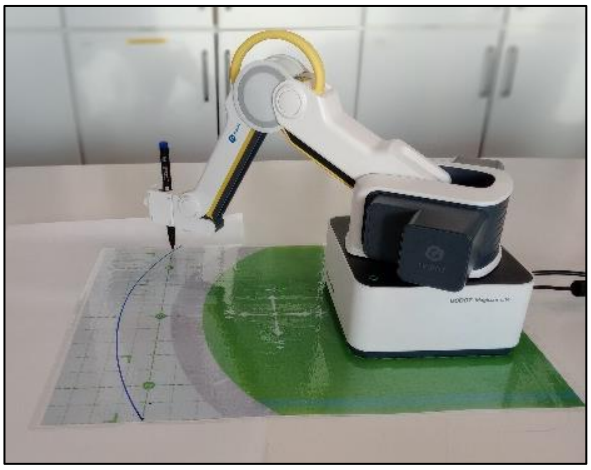
3) "Joint" literally refers to the joints of the Dobot's arm, and the two numbers J1and J2 represent the angles for the arm's two joints – but the angle J3 is not really a joint angle. Move the arm into different positions manually (remember to unlock the arm with the lock-button) and try to make sense of the significance of each angle.
   Then test your findings by pressing the joint-buttons in *Dobot lab* one at a time.
4) You probably did not get anything for the angle J4 – this angle is not used with the pen holder mounted. Instead, it is used to rotate the suction cup or soft grip that can also be mounted at the end of the arm.

You have now seen how the angles can be used to specify the position of the robot's arm – and maybe you also noticed that their domains are simpler than those of the x-y-z-coordinates. In fact, each of the motors inside the robot directly changes one (and only one) of the J1-J2-J3-J4-values. Therefore, to get to a specific (x, y, z)-point, the robot must transform those three x-y-z-numbers to three suitable J-numbers, and then tell the motors to adjust their angles accordingly. This not entirely simple mathematical task we will focus on in a later unit.

2.2.2 The origin of the coordinate system

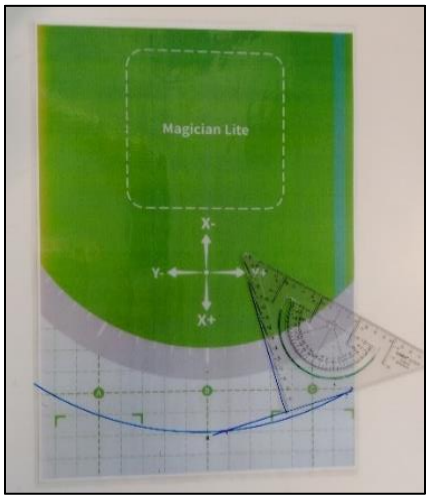
In 2.2.1 you probably found out that the origin, i.e. the point $(x, y, z) = (0,0,0)$, sits somewhere inside the middle of the body of the robot. It is of course impossible to move the end of the arm to this point, so to find it, we need some *measurements*. We will do it in an elegant way involving a few mathematical properties of circles.

Materials: For this task you need the laminated sheet, two sheets of A3 paper, and a permanent marker with a soft tip.

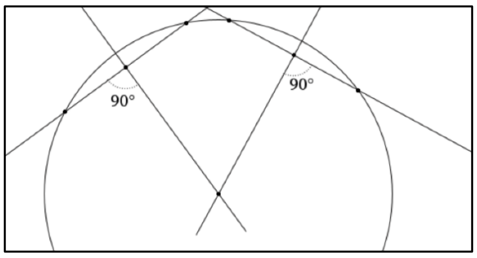


1) Place the Dobot exactly in the dotted white square. Place the two A3 sheets *under* the laminated sheet so that the Dobot will not accidentally make permanent drawings on the desk.
2) Use one of the J-buttons in *Dobot-Lab* to make the Dobot draw a circle on the laminated sheet (see photo).
3) Then remove the Dobot.

The x-y-origin of the coordinate system must be the center of the circle you just drew – but where is this center? We only have a small arc, not the entire circle. However, the center of a circle is where any two diameters intersect, so we ”just” need to construct two diameters.

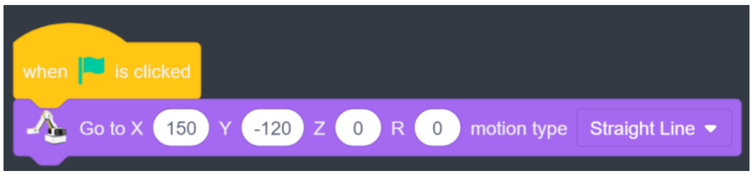


4) The *first diameter* is constructed in this way:
   a. Mark two arbitrary points P_1 and P_2 on the arc.
   b. With a ruler, draw a straight line through the two points. This is called a *secant*.
   c. Find – by simple measurement – the midpoint of the secant.
   d. Draw a straight line through the secant’s midpoint and perpendicularly to the secant. This line is called the *perpendicular bisector* of the two points P_1 and P_2. More importantly for us: the line is a diameter of the circle and will therefore pass through its center, wherever it might be.
5) The *second diameter* is constructed in the same way.
6) Now you have located the center of the circle – it is where the two diagonals intersect. And this should be in the middle of the dotted square, i.e. in the middle of the Dobot
7) <u>Show your teacher before you proceed</u>!
8) *Extension*: The two perpendicular bisectors can be constructed without using measurements. This was already known in Greece several thousand years ago. You can ask your teacher to show how to do this by the use of a ruler *without numbers* and a protractor.



So, now we have located the origin of the Dobot's coordinate system. Let us test it by moving to some specific coordinates. For that, you will need to do some programming – read on!

9) Assume that we need to move the robot's arm to the point with coordinates $(x, y, z) = (150, -120, 0)$. Might those numbers be *distances from the origin* measured in millimeters? Use a ruler to locate this point and mark it on the laminated sheet.
10) Remove the pen from the pen-holder and place the Dobot back in the dotted square. Make the following little program:



```
when [flag] is clicked
Go to X (150) Y (-120) Z (0) R (0) motion type [Straight Line ▾]
```

11) Press "Play" (upper right hand corner of the screen) and see if the robot moves to the correct place.
12) Remove the robot – and use a ruler to measure the (x, y)-coordinates for the point on the laminated sheet called A. Can you use the program above to make the robot's arm go to A? How many tries did you need?

2.3 What have we learned?

About the robot:

- The permitted intervals for each of x, y and z depend on the values of each of the two other: the workspace (domain) is complicated!
- The origin for the x-y-z-coordinate system is in the center of the robot.
- The robot measures distances in millimeters and angles in degrees.
- The position for "Home" is complicated in x-y-z-coordinates, but simple when using angles.

About mathematics:

- The domains for linear and squareroot functions.
- The meaning of the mathematical terms *arc*, *secant*, *perpendicular bisector*.
- How to draw a secant on a circle arc and use it to construct a diameter for the circle.
- How to use two diameters to find the circle's center.
- Maybe how to construct perpendicular bisectors with ruler and protractor (without doing any measurements).

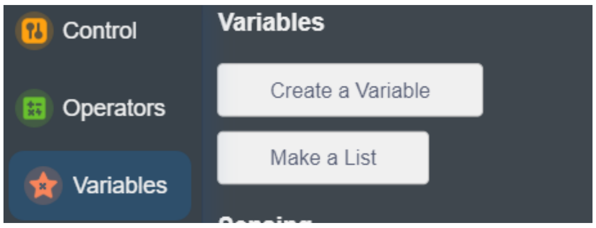
3.1 The Dobot draws rectangles and triangles

Now we will introduce some more complex programming of the robot. The first task is to have it draw simple geometric figures with given measurements. The programs will be using variables and mathematical calculations to make the pen draw the lines we want it to draw.

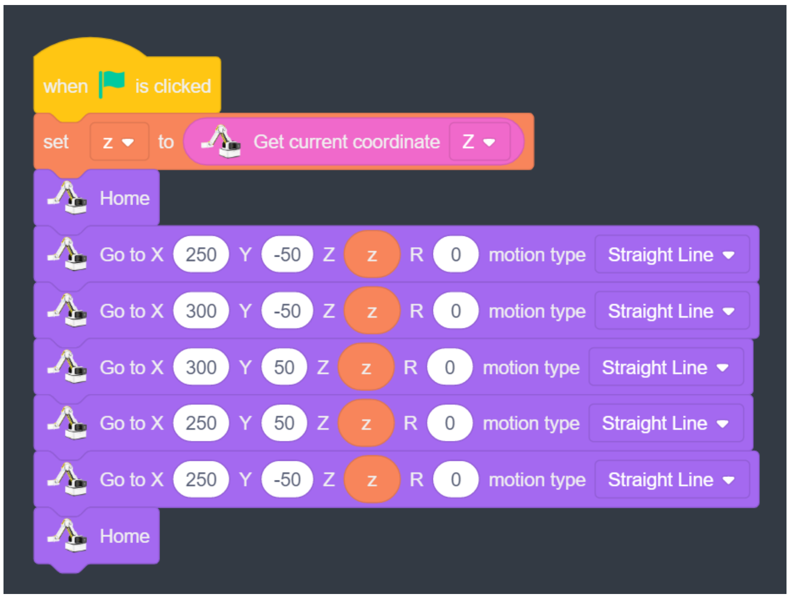
3.1.1 Drawing a rectangle in two different ways

The first task is to have the Dobot draw a rectangle with length 10 cm and height 5 cm.

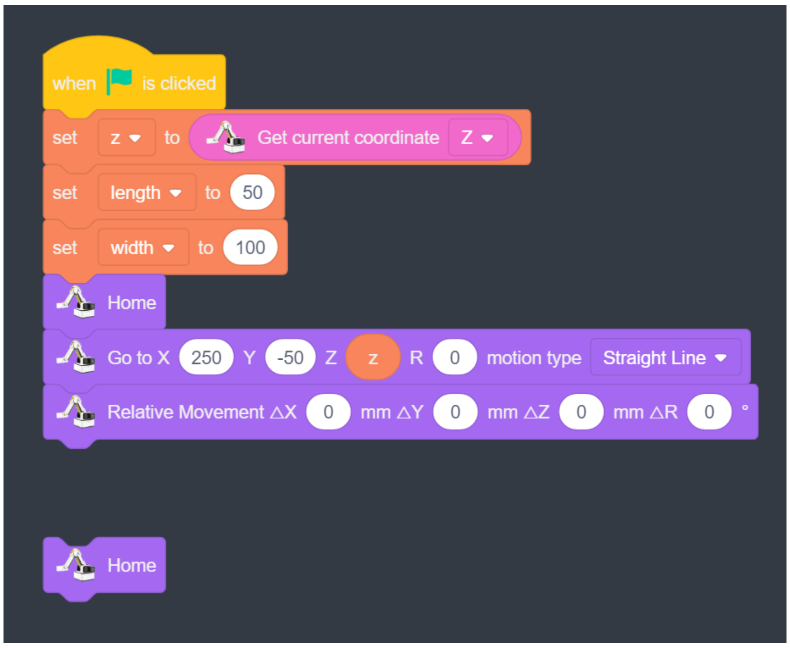
To define a variable that we can conveniently refer to several times in a program, choose ”Variables” and ”Create a Variable” in *DobotBlock Lab* →



1) Create a variable called *z* (more variables will follow).
2) Make the block program shown →
3) Clean the laminated sheet with ethanol.
4) Place the pen in the pen holder.
5) Position the robot’s arm so that the tip of the pen just touches point *A* on the laminated sheet.
6) Run the program!
7) Use a ruler to measure the rectangle. Is the length 10 cm and the height 5 cm?
8) Can you predict what would happen if one of the three middle ”Go to” commands were removed? Try it!
9) Why do you think the program began by assigning *z* the value “Get current coordinate”? ½
10) We now introduce two new variables ”length” and ”width”. Using these variables, write a program that draws the same rectangle as before. The program should start as shown →
11) Experiment to find out how large the rectangle can get (do not draw on the table!).



```
when [flag] is clicked
set z to Get current coordinate Z
Home
Go to X 250 Y -50 Z z R 0 motion type Straight Line
Go to X 300 Y -50 Z z R 0 motion type Straight Line
Go to X 300 Y 50 Z z R 0 motion type Straight Line
Go to X 250 Y 50 Z z R 0 motion type Straight Line
Go to X 250 Y -50 Z z R 0 motion type Straight Line
Home
```



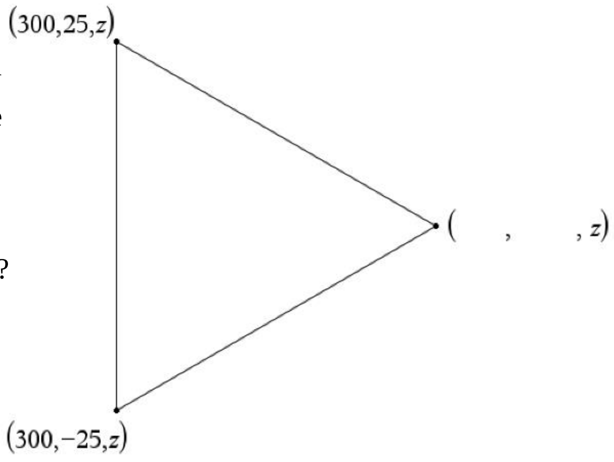
```
when [flag] is clicked
set z to Get current coordinate Z
set length to 50
set width to 100
Home
Go to X 250 Y -50 Z z R 0 motion type Straight Line
Relative Movement ΔX 0 mm ΔY 0 mm ΔZ 0 mm ΔR 0 °

Home
```

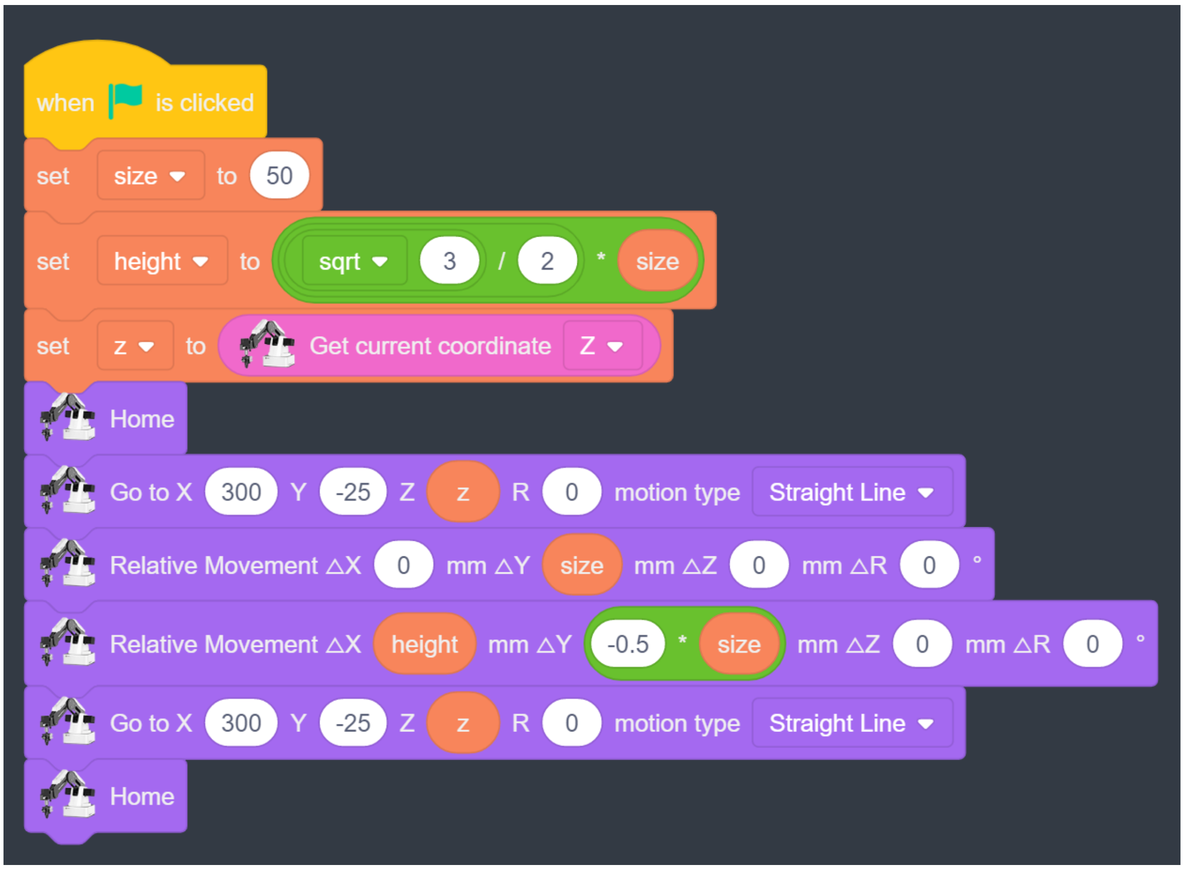
3.1.2 Drawing equilateral triangles

In 3.1.1 you probably found out that the modified program sometimes drew triangles. The next task is to draw a equilateral triangle (all three sides are equal).

1) We want an equilateral triangle with two vertices in $(300, -25, z)$ and $(300, 25, z)$, where x, y and z are measured in millimeters.
   a. How long is each side? (Remember unit…)
   b. What are the coordinates for the third vertex? [Hint: Pythagoras…]



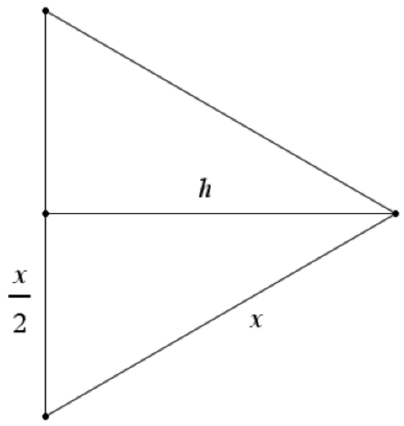
2) Write a program so that the Dobot draws the above triangle.
3) Measure the sides. Is the triangle equilateral? And is the side length what it should be?
4) It would be nice that the coordinate set for the last point was automatically calculated by the program and that the side length could be chosen arbitrarily. Try the program below!



```
when [flag] is clicked
set size to 50
set height to sqrt(3) / 2 * size
set z to Get current coordinate Z
Home
Go to X 300 Y -25 Z z R 0 motion type Straight Line
Relative Movement ΔX 0 mm ΔY size mm ΔZ 0 mm ΔR 0 °
Relative Movement ΔX height mm ΔY -0.5 * size mm ΔZ 0 mm ΔR 0 °
Go to X 300 Y -25 Z z R 0 motion type Straight Line
Home
```

5) How does the math in the program work? See the explanation below – and, in your group, take turns explaining each step!

The figure shows an equilateral triangle with side length x.
The triangle can be divided into two right angled triangles as shown.
In each of the two right angled triangles, the hypotenuse is x and the two shorter sides are h and $\frac{x}{2}$.



With Pythagoras we get

$$h^2 + \left(\frac{x}{2}\right)^2 = x^2.$$

Now solve for h:

$$h = \pm\sqrt{x^2 - \left(\frac{x}{2}\right)^2}$$

The height h cannot be negative, so

$$h = \sqrt{x^2 - \left(\frac{x}{2}\right)^2} = \sqrt{\frac{3x^2}{4}} = \frac{\sqrt{3}}{2}x$$

6) Now analyse the program on the previous page – can you see how it works?
7) Experiment with equilateral triangles of different sizes. How large can they get?
8) Experiment with other geometric figures!

3.2 What have we learned?

About the robot:

- How to create and use variables in a program.
- How to do mathematical calculations in block programming.

About Mathematics:

- How to calculate the height in an equilateral triangle.
- How to calculate the coordinates for the three vertices in an equilateral triangle.

4.1 What's next?

Further teaching units will explore…

- The conversion between x-y-z-coordinates and angular coordinates.
- Calculations with sine and cosine.
- How to construct regular polygons (e.g. a 17-gon).
- How to use the Dobot as an industrial robot to manipulate, move an dsort objects.