“MaLT2 Commands”

Table 1: Avatar’s control commands

Command	Description	Example
Avatar’s movement		
forward/fd *number*	Avatar moves **forward** as many steps as the *number* value.	fd 50
back/bk *number*	Avatar moves **backward** as many steps as the *number* value.	bk 70
Avatar’s orientation		
right/rt *number*	Avatar turns its head to the **right** by as many **degrees** as the *number* value.	right 90
left/lt *number*	Avatar turns its head to the **left** by as many **degrees** as the *number* value.	lt 120
up *number*	Avatar turns its head **upwards (looks up)** by as many **degrees** as the *number* value.	up 50
down/dn *number*	Avatar turns its head **downwards (looks down)** by as many **degrees** as the *number* value.	down 60
roll_right/ rr *number*	Avatar rotates around itself **clockwise** by as many **degrees** as the *number* value.	rr 40
roll_left/rl *number*	Avatar rotates around itself **anticlockwise** by as many **degrees** as the *number* value.	rl 30
Avatar’s position		
setx *number*	Places the avatar at the position where x coordinate equals to the *number.*	setx 100
sety *number*	Places the avatar at the position where y coordinate equals to the *number.*	sety -50

setz *number*	Places the avatar at the position where z coordinate equals to the *number.*	setz 90
setxy *n1 n2*	Places the avatar at the position where x coordinate equals to the *n1* and y coordinate equals to *n2.*	setxy 50 100
setxz *n1 n2*	Places the avatar at the position where x coordinate equals to the *n1* and z coordinate equals to *n2.*	setxz 50 -90
setyz *n1 n2*	Places the avatar at the position where y coordinate equals to the *n1* and z coordinate equals to *n2.*	setyz 50 -90
setpos [*n1 n2 n3***]**	Places the avatar at the position with the coordinates n1 n2 n3.	setpos [0 30 70]
xcor	**Returns** the value of the x coordinate of avatar's current position.	
ycor	**Returns** the value of the y coordinate of avatar's current position.	
zcor	**Returns** the value of the z coordinate of avatar's current position.	
pos	**Returns** the avatar's current position in an array of three numbers [x y z].	
distanceto [x y z]	Calculates and **returns** the distance between the avatar's position and the point given as an array input of [x y z].	distanceto [100 20 30]
Avatar's Trace		
penup/pu	The avatar doesn't leave a trace while moving in the scene.	
pendown/pd	The avatar leaves a trace while moving in the scene.	
setpensize *number*	Sets the width of the trace to the value of *number.* (**Default is 1**)	setpensize 5

setpencolor *[r b g]*	Sets the color of the trace to the color code of the r b g array (red blue green).	setpencolor [0 0 0] (Black)
home	Avatar returns to initial position (0, 0, 0) while leaving a trace.	
cleartrace/ct	Clears the 3D scene and lets the avatar and the camera in their current position.	
clearscreen/cleargraphics/cs /cg	Clears the 3D scene and resets the avatar to its initial position (0, 0, 0).	
showturtle/st	Shows the avatar on the scene.	
hideturtle/ht	Hides the avatar from the scene.	
Other Commands		
cleartext/ct	Clears messages from the message area.	
print/pr *input*	Prints the output of the *input* at the message area. The *input* may be a command, a mathematical expression or a variable.	print 1+1 print xpos print :height
stop	Stops the execution of the code in a repetition or a recursion. It is necessary in the procedures with recursion!	Example with recursion TO wing :a :n :k if :k < 1 [stop] polygon :a :n wing 2*:a/3 :n :k-1 END

Basic color codes RBG for the avatar's change of color

Red 255 0 0

Green 0 255 0

Blue 0 0 255

White 255 255 255

Black 0 0 0

You can find more color codes at MaLT2's color picker.

Table 2: Programming structures

Command	Description	Example
Conditional Structures		
if *condition* **[commands]**	If the *condition* is true, the group of commands inside the brackets [ ] is executed.	if :x > 10 [forward 100 right 90]
ifelse *condition* **[commands1] [commands2]**	If the *condition* is true, the group of commands1 of the first brackets is executed. Else if the condition is false, the group of commands2 of the second brackets is executed.	ifelse :x > 10 [forward 100 right 90] [left 90 forward 100]
if and *condition* **[commands]**	If both parts of the *condition* are true, the group of commands inside the brackets [ ] is executed.	if and :x>3 :y>5 [fd 100]
Iterative structures		
repeat *n* **[commands]**	The group of commands inside the brackets [ ] is repeated **n times.**	repeat 4 [forward 100 rt 90]
while *condition* **[commands]**	**While the *condition* is true** the group of commands inside the brackets [ ] is repeated.	make "x 1 while :x<5 [fd 100 rt 90 make "x :x+1]
until *condition* **[commands]**	**Until the condition becomes true,** the group of commands inside the brackets [ ] is repeated.	make "x 0 until :x = 4 [fd 100 rt 90 make "x :x+1]
repcount	**Returns** the current repetition number. It is used only in "repeat n" structure.	repeat 4 [fd 40 print repcount] It will print 1, 2, 3, 4 in sequence.
Operators		
or *Expr1 Expr2*	Returns *true* if at least one of the two expressions is true.	if or 2>3 4<5 [print 'true'] (it is true)

and *Expr1 Expr2*	Returns *true* if both expressions are true.	if and 2>3 4<5 [print 'true'] (it is false)
not *Expr1*	Returns *true* if Expr1 is not true.	if not 2>3 [print 'true'] (it is true)
equal? *Value1 Value2*	Returns *true* if value1 is equal to value2.	if equal? :a :b [print 'equal']
notequal? *Value1 Value2*	Returns *true* if value1 is not equal to value2.	if notequal? :a :b [print 'not equal']
greater? *Value1 Value2*	Returns *true* if value1 is greater than value2.	if greater? :a :b [print 'a bigger']
less? *Value1 Value2*	Returns *true* if value1 is less than value2.	if less? :a :b [print 'a smaller']
greaterequal? *Value1 Value2*	Returns *true* if value1 is greater or equal to value2.	
lessequal? *Value1 Value2*	Returns *true* if value1 is less or equal to value2.	
make *"variable number*	**Defines** the *variable* and assigns to the variable the **value** of the *number*. Then it can be used as :variable	make "height 30 (:height will have the value 30)
rand/random *α b*	**Returns** a random number between a and b-1.	rand 0 4 (returns randomly a number among 0, 1, 2, 3)
output value	Stops the procedure and returns the *value*. **It is used inside procedures.**	TO add :a :b return :a + :b END

Table 3: Mathematical Commands

Command	Description	Example	Result
sum/add *a b*	Returns the sum of the two numbers set in its input, i.e., it performs a+b.	sum 3 5	8
difference/sub *a b*	Returns the difference of the two numbers set in its input, i.e., it performs a-b.	difference 8 3	5

product/mul ***a b***	Returns the product of the two numbers set in its input, i.e., it performs a*b.	product 2 4	8
divide/div ***a b***	Returns the division of the two numbers set in its input, i.e., it performs a/b.	divide 6 3	2
remainder/modulo/mod ***a b***	Returns the remainder of the division of the two numbers set in its input.	remainder 11 2	1
sqrt ***number***	Gives the square root of the number set in its input.	sqrt 36	6
power/pow ***x n***	Raises the x number to the n power and returns the result. Thus, it is x^n.	power 2 4	16
cos ***degrees***	It returns the cosine of the angle set as an input.	cos 60	0.5
sin ***degrees***	It returns the sine of the angle set as an input.	sin 60	0.866
tan ***degrees***	It returns the tangent of the angle set as an input.	tan 180	0
arccos ***number***	It returns the angle that it is calculated by the inverse cosine based on the argument set as an input.	arccos 0.5	60
arcsin ***number***	It returns the angle that it is calculated by the inverse sine based on the argument set as an input.	arcsin 0.5	30
arctan ***number***	It returns the angle that it is calculated by the inverse tangent based on the argument set as an input.	arctan 1	45
radcos ***rads***	It returns the cosine of the angle given in radius (rads).	radcos 1	0.5403023058681398
radsin ***rads***	It returns the sine of the angle given in radius (rads).	radsin 1	0.8414709848078
exp ***number***	It returns the exponential function with a base of e and a power of the number set in its input (e^{number}).	exp 1	2.718

ln ***number***	It returns the ln value of the number set as an input.	ln 1	0
log10 ***number***	It returns the log10 set as an input.	log10 10	1
integer/int ***number***	It returns the integer part of the number set as an input.	integer 2.8	2
round ***number***	It returns the rounding of the number set in its input.	round 2.3 round 3.8	2 4
minus ***number***	It returns the minus of the number set as an input.	minus 10	-10
abs ***number***	It returns the absolute value of the number set as an input.	abs -3	3
pi	It returns the pi (3,14) number.	pi	3.14