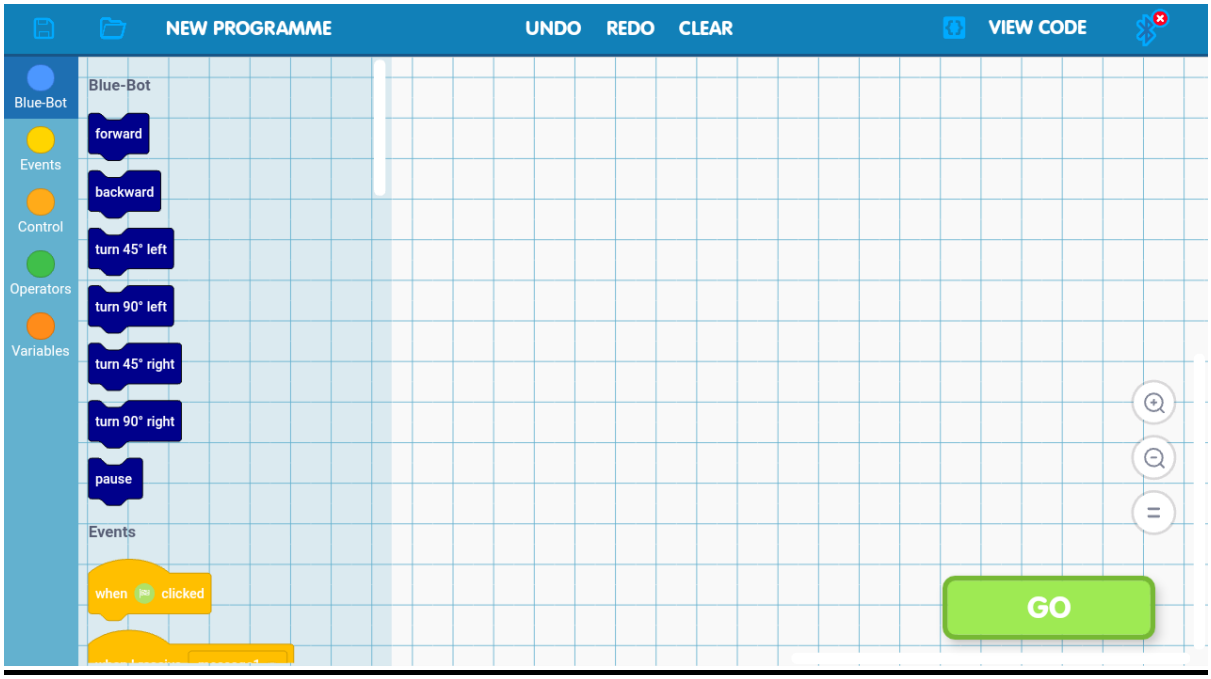
TTS Visual Programming User Guide



Workspace

The workspace refers to the programming interface in its entirety, including the categories (far left), the toolbox (left, next to the categories), canvas (center and right of the screen).

Categories

Fixed to the far left of the screen, allows the user to quickly navigate to sets of blocks they wish to use. For example, the event blocks or the control blocks or the robot specific blocks (Blue-Bot, Ino-Bot basic and Ino-Bot advanced).

Toolbox

To the right of the Categories, the toolbox contains all of the blocks the user will have access to. New blocks can be added to the canvas by pressing and dragging the block from left to right and dropping it onto the canvas. Blocks can also be deleted by dragging and dropping them onto the toolbox.

Canvas

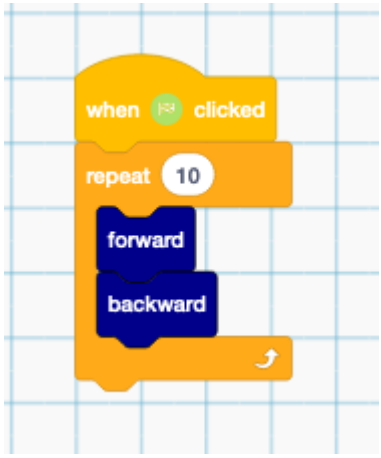
The canvas is where the user will assemble and arrange their blocks in order to create their program. Blocks can be added and removed from the top, bottom or placed within other blocks.

Program

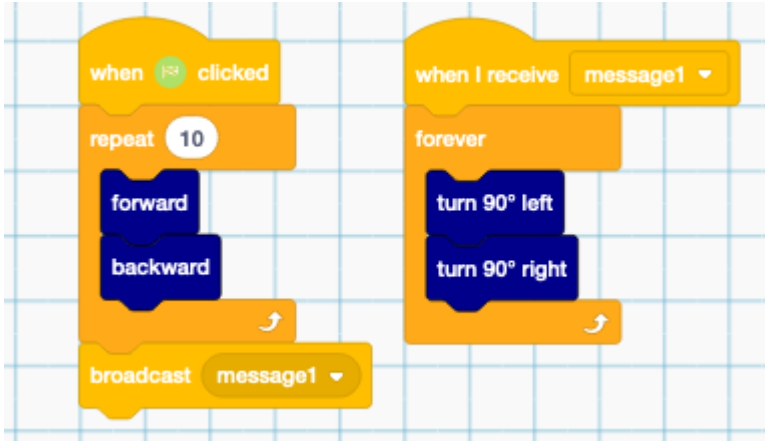
A program refers to all of the blocks placed onto the canvas.

Stacks

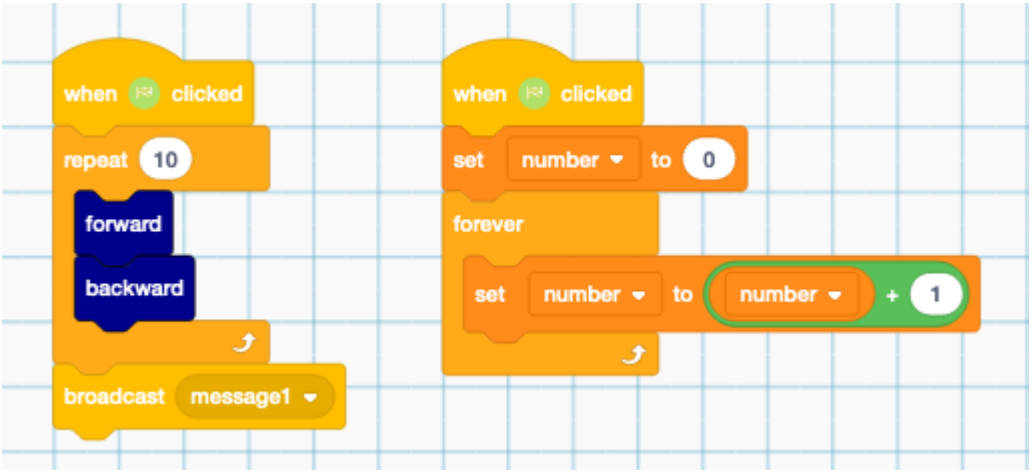
Stacks are individual blocks of code. They can be thought of in the traditional programming sense as functions. All stacks must start with a Top Block.



Here is an example of a single stack. This stack will begin execution when the play button is pressed and will command the robot to move forward then back 10 times, before finishing execution.



The example above shows 2 stacks. Like the first program when the play button is pressed the robot will move forward then back 10 times. However when it reaches the broadcast block it will trigger the execution of the second stack, causing the robot to turn left then right by 90 degrees until the program is stopped.



This final stacks example shows 2 stacks that will be started at the same time. This will lead to the blocks being executed simultaneously. This is probably the most difficult aspect of the TTS Visual Programming (TTS VP) as commands to the robot may execute on top of each other and send conflicting commands to the robot.

Top Blocks

Top Blocks are the start of any stack. They begin the execution of the stack when triggered. Currently there are only 2 Top Blocks implemented into TTS VP. Top Blocks are denoted by a bump on the top side of the block and a connection at the bottom. All Top Blocks are also part of the events category

When Clicked Block

The When Clicked block will trigger the execution of a stack when the Play button is pressed. This block will cause the execution of the block that is connected below it.

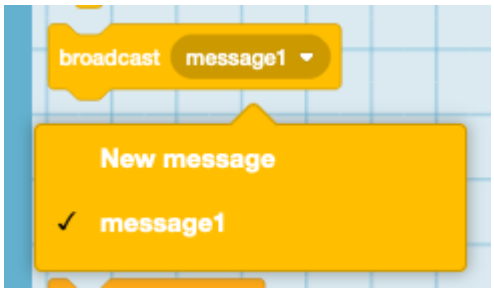
When I receive {message} Block

This Top Block is triggered by the broadcast {message} block if the messages match. When triggered it will cause the execution of the block that is connected below it.

Other Event Blocks

Broadcast {message} Block

This block will broadcast a message to the whole workspace and will trigger any When I receive {message} blocks.



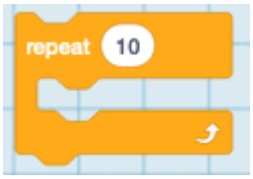
Using the Broadcast Block, the user can create new messages to broadcast and name them. They can also change which message is being broadcast.

Control Blocks

Wait {x} seconds Block

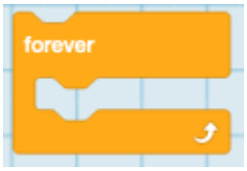
This block will pause the execution of the stack the block is present within for {x} seconds.

Repeat {x} Block



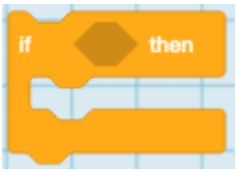
This block will execute the blocks placed within it {x} times. Similar in programming to a for loop.

Forever Block



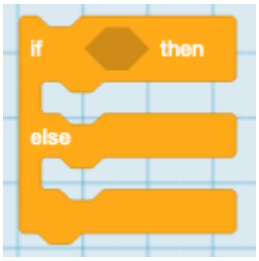
This Block will execute the blocks placed within it in a loop until the program is stopped. Similar in programming to a while(true) loop.

If {condition} Then Block



This block will execute the blocks placed within it if the {condition} is true. Similar in programming to an if statement.

If {condition} Then, Else Block



This block will execute the blocks placed within the first section if the {condition} is true. If the condition is false then it executes the blocks placed within the second section. Similar in programming to an if else statement.

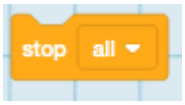
Wait until {condition} Block

This block will pause the execution of the stack it is present within, until the {condition} is true.

Repeat until {condition}

This block will execute the blocks placed within it until the {condition} is true. Similar in programming to a while ({condition}) loop.

Stop Block



This block will stop the execution of either the whole program or the stack it is placed within.

Operators

Add Block

This block will return the result of adding {variable 1} to {variable 2}.

Minus Block

This block will return the result of subtracting {variable 2} from {variable 1}.

Times Block

This block will return the result of multiplying {variable 1} with {variable 2}.

Divide Block

This block will return the result of dividing {variable 1} by {variable 2}.

Pick random {variable 1} to {variable 2}

This block returns a random number between {variable 1} and {variable 2}.

{variable 1} < {variable 2}

This block returns a {condition} based on if {variable 1} is less than {variable 2}.

{variable 1} = {variable 2}

This block returns a {condition} based on if {variable 1} is equal to {variable 2}.

{variable 1} > {variable 2}

This block returns a {condition} based on if {variable 1} is greater than {variable 2}.

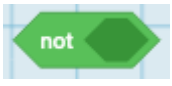
{condition 1} and {condition 2}

This block returns a {condition} based on if {condition 1} and {condition 2} are true.

{condition 1} or {condition 2}

This block returns a {condition} based on if {condition 1} or {condition 2} are true.

not {condition 1}



This block returns a {condition} and will invert {condition 1}. Example if {condition 1} is true will return false and vice versa.

{variable 1} mod {variable 2}

This block will return the remainder of the division between {variable 1} and {variable 2}.

Round {variable}

This block will return the nearest whole number of {variable}

Variable blocks

Make a variable button

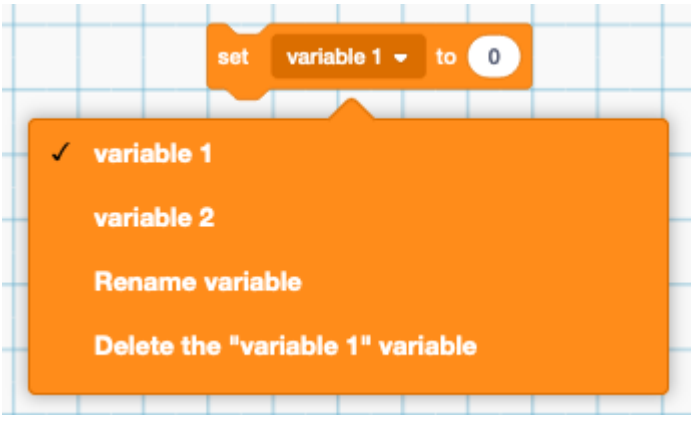
This button will allow the user to create and name a new variable.

Variable block

This block will allow the user to add their variable to another block with a variable slot.

Set {variable} to {variable} block

This block will allow the user to set what is stored inside their variable

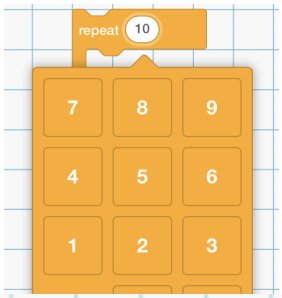


This block will also allow the user to swap variables, rename their variable or delete their variable.

A note about {variable} slots

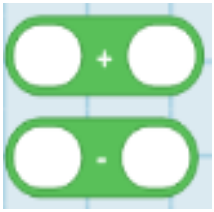
A variable slot is denoted by having rounded edges and can be seen in the wait {variable} seconds block.

The user can fill a {variable} slot in a block in 3 ways:



The first way is simple by pressing the empty slot which will produce a number pad.

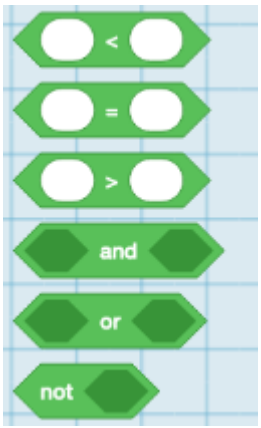
The second way is by adding a custom variable block.



The third way is by using an operator block. The blocks like the custom variable blocks have rounded edges. For example the addition and subtraction blocks.

A note about {condition} slots

Condition slots are denoted by having pointed edges and can be seen in the wait until {condition} block.



The user can fill a {condition} slot in a block in 1 way, by adding one of the 6 condition blocks.

Blue-Bot Blocks

Forward Block

This block will instruct a connected Blue-Bot to move forward once. This will take approximately 4 seconds.

Backward Block

This block will instruct a connected Blue-Bot to move backward once. This will take approximately 4 seconds.

Turn 45 left

This block will instruct a connected Blue-Bot to turn on the spot 45 degrees left once.
This will take approximately 3 seconds.

Turn 90 left

This block will instruct a connected Blue-Bot to turn on the spot 90 degrees left once. This will take approximately 3 seconds.

Turn 45 right

This block will instruct a connected Blue-Bot to turn on the spot 45 degrees right once. This will take approximately 3 seconds.

Turn 90 right

This block will instruct a connected Blue-Bot to turn on the spot 90 degrees right once. This will take approximately 3 seconds.

Pause

This block will pause a connected Blue-Bot for 3 seconds.