Picture Blocks

Layout 1

Code Tabs
Click through all the different categories of blocks available

Upload
Sends code to Pink-E

Run
Runs virtual code

Stop
Stops either upload or run

Block Library
Shows all the different blocks you can drag in that library

Outcome Space
The outcome of the program or any error messages

Variable Block
A block requiring a variable

Control Block
Controls logical flow

Program Space
The place to drag and drop blocks to run the program

General Blocks
General blocs that connect together by puzzle edges

Layout 2

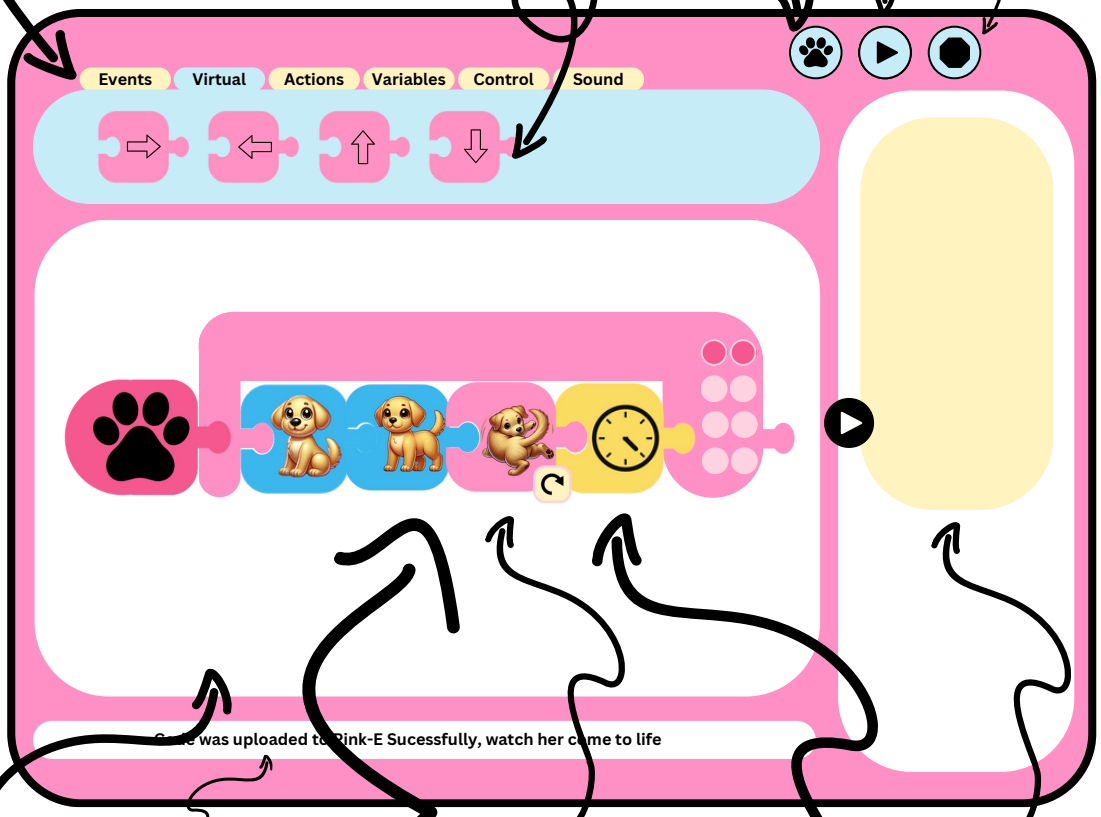
Code Tabs
Click through all the different categories of blocks available

Upload
Sends code to Pink-E

Run
Runs virtual code

Block Library
Shows all the different blocks you can drag in that library

Stop
Stops either upload or run



Outcome Space
The outcome of the program or any error messages

Variable Block
A block requiring a variable

Virtual Challenge
The virtual challenge will be displayed here as well as the translation to the next level of coding

Program Space
The place to drag and drop blocks to run the program

General Blocks
General blocs that connect together by puzzle edges

Control Block
Controls logical flow

Event Block

Overview:
Event Blocks serve as triggers to initiate a sequence of actions. These blocks are essential for defining when and how code begins to execute, whether in virtual environments or with the physical robotic dog. Multiple event blocks can be placed in the code area, and each will wait for its specific condition to be met before executing.

Common Features:

- Shape: All event blocks have a rounded left side with a puzzle connection on the right.
- Execution: Each block waits for a specific condition to be triggered before starting a sequence of code.
- Purpose: Marks the beginning of a code sequence, signaling when the program should run.

Types of Event Blocks:

1. Start Virtual: Triggers code execution for virtual challenges when clicked or when the play button is pressed.
2. Start Physical: Executes code on the physical dog, triggering actions once uploaded and certain conditions are met.
3. Nose Button: Initiates a sequence when the physical dog's nose button is pressed.
4. Back Sensor: Starts a code sequence when the back sensor on the physical dog is activated.
5. Stomach Sensor: Triggers the sequence when the stomach sensor is activated on the physical dog.

Use Cases:
Multiple event blocks can be active in the code area simultaneously, each waiting for its respective trigger condition. In the virtual environment, a sequence may start when the play button is pressed. For the physical dog, sequences can begin based on interactions with sensors such as the nose button, back sensor, or stomach sensor. The code must be uploaded to the physical dog, and when any of these conditions are met, the corresponding sequence will execute. When two or more triggers activate simultaneously, the system processes them one at a time in the order they appear in the code area.

Start Virtual

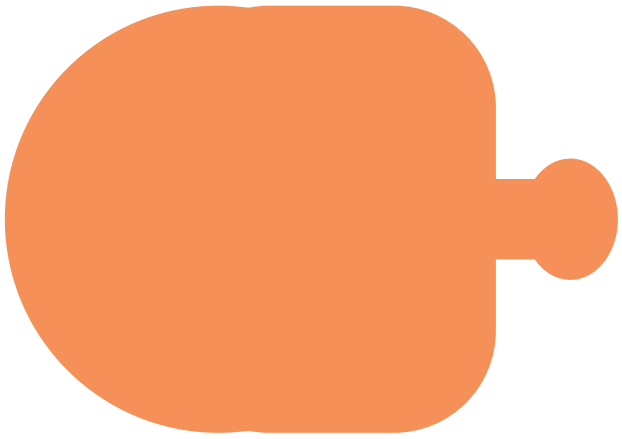
Event Block

Overview
The Start Virtual block is designed to mark the beginning of the code that will be executed in the virtual on-screen environment. It is primarily used during the learning phase of the app, where users are introduced to new coding concepts and lessons.

When it executes
This block will run when pressed and when the play button is pressed. It must be succeeded by a series of code that will be executed by the virtual dog when one of these actions are performed.

Constrains
This block must be the start of a series of code, it cannot be within a series. It serves only as a starting point for code execution in the virtual environment and should not be included in the uploaded code for the physical robot.

Possible Errors
If the user tries to run the code when no blocks follow this and there are no other event blocks, and error message will be shown
If the upload button is pressed and there is only this event, no other event blocks, an error message will occur.



Start Physical

Event Block

Overview:
The Start Physical block is designed to mark the beginning of the code that will be executed on the physical robotic dog. It is primarily used when applying the learned code to control the physical device after completing the lessons in the app.

When It Executes:
This block will run when pressed and when the upload button is pressed. It must be succeeded by a series of code that will be executed by the physical dog when one of these actions is performed.

Constraints:
This block must be the start of a series of code, it cannot be within a series. It serves only as a starting point for code execution on the physical robot and should not be used in the virtual environment.

Possible Errors:

- If the user tries to play the code when no blocks follow this and there are no other event blocks, an error message will be shown.
- If the play button is pressed (not upload) and there is only this event and no other event blocks, an error message will occur.

Nose Button

Event Block

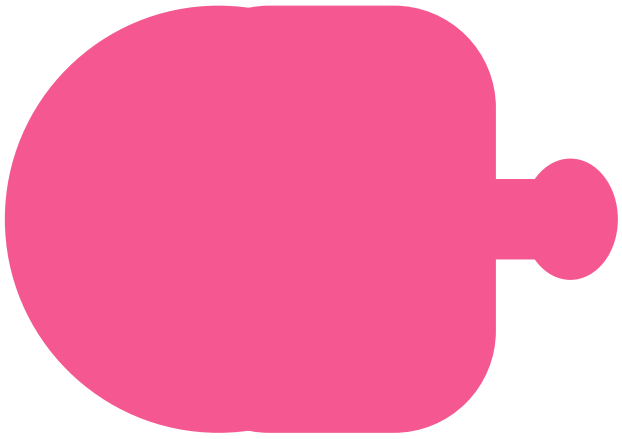
Overview:
The Nose Button block is designed to mark the beginning of code that will be executed on the physical robotic dog when its nose button is pressed. It is primarily used for triggering actions on the physical dog after the code has been uploaded, based on user interaction with the physical button.

When It Executes:
This block will only execute after the code is uploaded to the physical robotic dog and when the nose button on the dog is pressed. It must be succeeded by a series of code that will be executed by the physical dog when the nose button is triggered.

Constraints:
This block must be the start of a series of code, it cannot be placed within a series.

Possible Errors:

- If the user tries to upload the code when no blocks follow this and there are no other event blocks, an error message will be shown.
- If the play button is pressed (not the upload) and there is only this event and no other event blocks, an error message will occur.



Back Sensor

Event Block

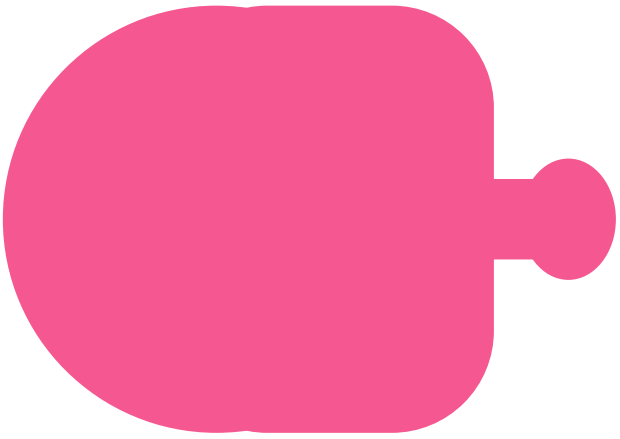
Overview:
The Back Sensor block is designed to mark the beginning of code that will be executed on the physical robotic dog when its back sensor is triggered. It is primarily used for triggering actions on the physical dog after the code has been uploaded, based on user interaction with the physical button.

When It Executes:
This block will only execute after the code is uploaded to the physical robotic dog and when the back sensor on the dog is triggered. It must be succeeded by a series of code that will be executed by the physical dog when the back sensor is triggered.

Constraints:
This block must be the start of a series of code, it cannot be placed within a series.

Possible Errors:

- If the user tries to upload the code when no blocks follow this and there are no other event blocks, an error message will be shown.
- If the play button is pressed (not the upload) and there is only this event and no other event blocks, an error message will occur.



Belly Sensor

Event Block

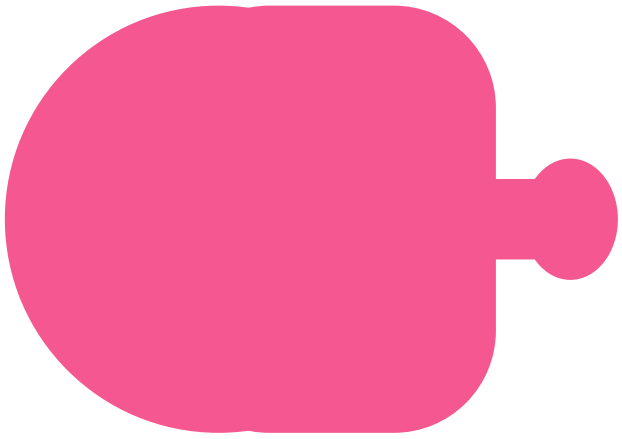
Overview:
The Belly Sensor block is designed to mark the beginning of code that will be executed on the physical robotic dog when its belly sensor is triggered. It is primarily used for triggering actions on the physical dog after the code has been uploaded, based on user interaction with the physical button.

When It Executes:
This block will only execute after the code is uploaded to the physical robotic dog and when the belly sensor on the dog is triggered. It must be succeeded by a series of code that will be executed by the physical dog when the belly sensor is triggered.

Constraints:
This block must be the start of a series of code, it cannot be placed within a series.

Possible Errors:

- If the user tries to upload the code when no blocks follow this and there are no other event blocks, an error message will be shown.
- If the play button is pressed (not the upload) and there is only this event and no other event blocks, an error message will occur.



Virtual Blocks

Left

Virtual

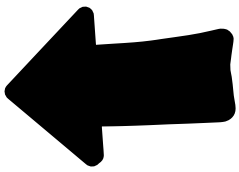
Overview:
The Move Left block is designed to control the movement of the virtual character in the on-screen environment. It is primarily used to navigate the character through a maze during lessons, teaching users basic movement commands and spatial logic.

When It Executes:
This block will execute when the play button is pressed. It moves the virtual character one step to the left. The Move Right block must follow the Start Virtual block.

Constraints:
This block must be in a sequence of code that begins with the Stat Virtual block. This block is for controlling the virtual character only and has no effect on the physical robot.

Possible Errors:

- If the code does not begin with a Start Virtual block, an error message will be shown when the user tries to run the code.
- If the character cannot move further left due to a wall or obstacle, an error message will notify the user.



Right

Virtual

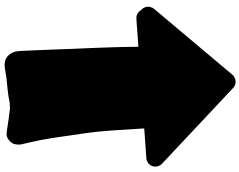
Overview:
The Move Right block is designed to control the movement of the virtual character in the on-screen environment. It is primarily used to navigate the character through a maze during lessons, teaching users basic movement commands and spatial logic.

When It Executes:
This block will execute when the play button is pressed. It moves the virtual character one step to the right. The Move Right block must follow the Start Virtual block.

Constraints:
This block must be in a sequence of code that begins with the Stat Virtual block. This block is for controlling the virtual character only and has no effect on the physical robot.

Possible Errors:

- If the code does not begin with a Start Virtual block, an error message will be shown when the user tries to run the code.
- If the character cannot move further right due to a wall or obstacle, an error message will notify the user.



Up

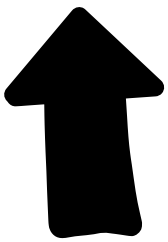
Virtual

Overview:
The Move Up block is designed to control the movement of the virtual character in the on-screen environment. It is primarily used to navigate the character through a maze during lessons, teaching users basic movement commands and spatial logic.

When It Executes:
This block will execute when the play button is pressed. It moves the virtual character one step up. The Move up block must follow the Start Virtual block.

Constraints:
This block must be in a sequence of code that begins with the Stat Virtual block. This block is for controlling the virtual character only and has no effect on the physical robot.

Possible Errors:
- If the code does not begin with a Start Virtual block, an error message will be shown when the user tries to run the code.
- If the character cannot move further up due to a wall or obstacle, an error message will notify the user.



Down

Virtual

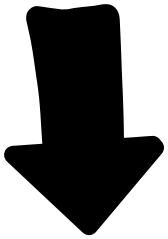
Overview:
The Move Down block is designed to control the movement of the virtual character in the on-screen environment. It is primarily used to navigate the character through a maze during lessons, teaching users basic movement commands and spatial logic.

When It Executes:
This block will execute when the play button is pressed. It moves the virtual character one step down. The Move Down block must follow the Start Virtual block.

Constraints:
This block must be in a sequence of code that begins with the Stat Virtual block. This block is for controlling the virtual character only and has no effect on the physical robot.

Possible Errors:

- If the code does not begin with a Start Virtual block, an error message will be shown when the user tries to run the code.
- If the character cannot move further down due to a wall or obstacle, an error message will notify the user.



Actions

Sit

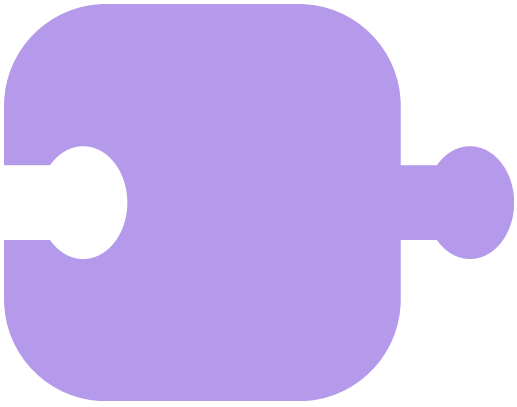
Action

Overview:
The Sit block is designed to control the physical robotic dog to perform the sitting action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Sit block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The Sit block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:
- If the Sit block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Lay Down

Action

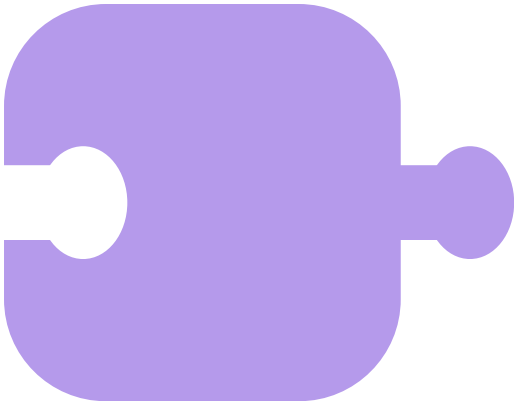
Overview:
The Lay Down block is designed to control the physical robotic dog to perform the laying down action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Lay Down block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The Lay Down block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:

- If the Lay Down block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Stand

Action

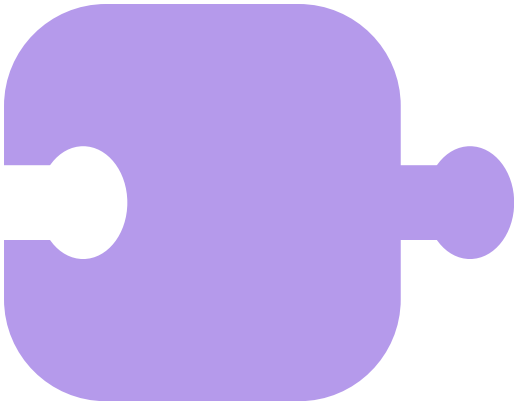
Overview:
The Stand block is designed to control the physical robotic dog to perform the standing action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The stand block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The stand block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:

- If the stand block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Play Dead

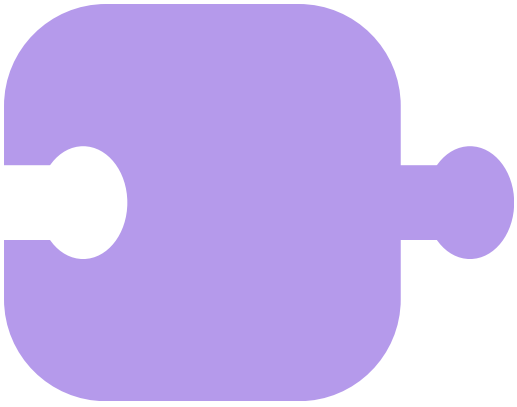
Action

Overview:
The Play Dead block is designed to control the physical robotic dog to perform the playing dead action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Play Dead block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The Play Dead block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:
- If the Play Dead block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Bow

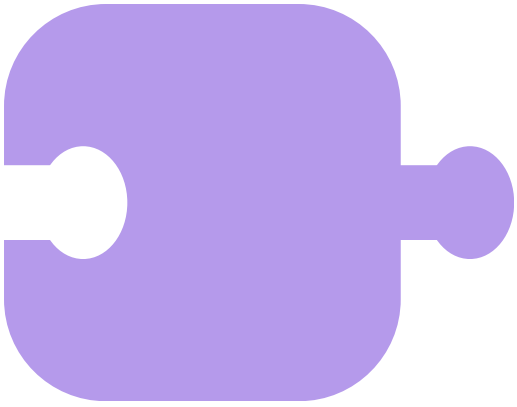
Action

Overview:
TheBow block is designed to control the physical robotic dog to perform the bowing action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Bow block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The Bow block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:
- If the Bow block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Beg

Action

howl

Action

Wag Tail

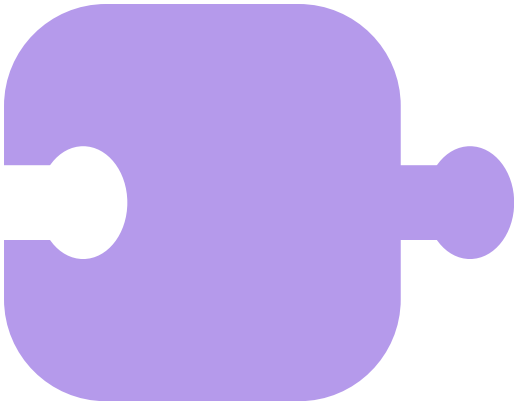
Action

Overview:
The Wag Tail block is designed to control the physical robotic dog to perform the wagging tail action. It is used in sequences that involve real-world interactions with the robotic dog.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Wag Tail block can be part of a sequence following any event block except the Start Virtual block.

Constraints:
The Wag Tail block must be placed after an event block (e.g., Start Physical or Nose Button) but cannot follow the Start Virtual block.
It only controls the physical robotic dog and has no effect on the virtual character.

Possible Errors:
- If the Wag Tail block is placed in a sequence following the Start Virtual block, an error message will be shown when the user tries to run or upload the code.
- If no event block is present at the start of the sequence, an error will occur.



Variables

Walk

Variable

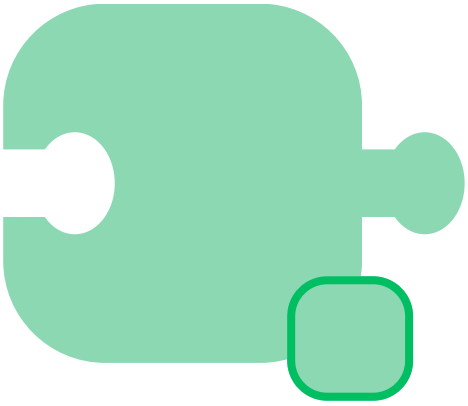
Overview:
The Walk block is designed to control the physical robotic dog to walk either forward or backward, depending on the variable chosen.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Walk block can be part of a sequence following any event block designed for the physical dog.

Constraints:
The Walk block must be placed after an event block (e.g., Start Physical or Nose Button).
It only controls the physical robotic dog and does not affect the virtual character.
This block includes a variable input that must be set to either "forward" or "backward" to specify the direction the dog should walk.
The block cannot be left without a variable; it will produce an error if no variable or an invalid variable is set.

Possible Errors:

- If the Walk block is placed in a sequence without an event block (e.g., following the Start Virtual block), an error message will be shown when the user tries to run or upload the code.
- If the variable is left empty or not set to either "forward" or "backward", an error message will be displayed.



Turn

Variable

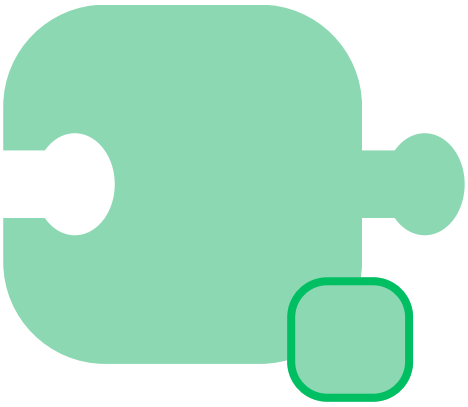
Overview:
The Turn block is designed to control the physical robotic dog to turn either right or left, depending on the variable chosen.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Turn block can be part of a sequence following any event block designed for the physical dog.

Constraints:
The Turn block must be placed after an event block (e.g., Start Physical or Nose Button).
It only controls the physical robotic dog and does not affect the virtual character.
This block includes a variable input that must be set to either "right" or "left" to specify the direction the dog should turn.
The block cannot be left without a variable; it will produce an error if no variable or an invalid variable is set.

Possible Errors:

- If the Turn block is placed in a sequence without an event block (e.g., following the Start Virtual block), an error message will be shown when the user tries to run or upload the code.
- If the variable is left empty or not set to either "right" or "left", an error message will be displayed.



Lift Leg

Variable

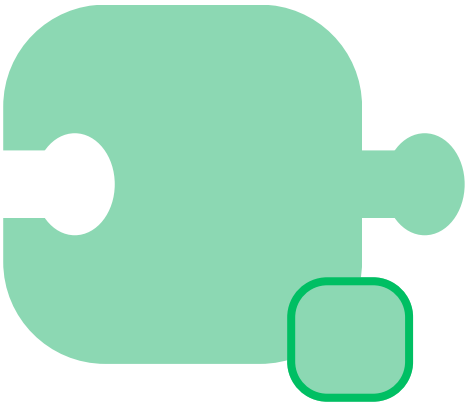
Overview:
The Lift Leg block is designed to control the physical robotic dog to lift up and bend one of its legs, depending on the variable chosen.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Lift Leg block can be part of a sequence following any event block designed for the physical dog.

Constraints:
The Lift Leg block must be placed after an event block (e.g., Start Physical or Nose Button).
It only controls the physical robotic dog and does not affect the virtual character.
This block includes a variable input that must be set to either "front left leg" or "back left leg" or "front right leg" or "back right leg".
The block cannot be left without a variable; it will produce an error if no variable or an invalid variable is set.

Possible Errors:

- If the Lift Leg block is placed in a sequence without an event block (e.g., following the Start Virtual block), an error message will be shown when the user tries to run or upload the code.
- If the variable is left empty or not set to either "front left leg" or "back left leg" or "front right leg" or "back right leg", an error message will be displayed.



Eyes

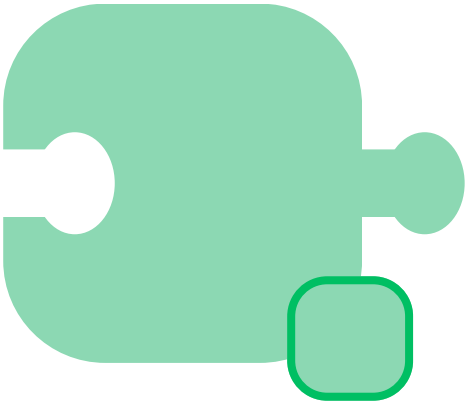
Variable

Overview:
The Eyes block is designed to control the physical robotic dog to open or close its eyes, depending on the variable chosen.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Eyes block can be part of a sequence following any event block designed for the physical dog.

Constraints:
The Eyes block must be placed after an event block (e.g., Start Physical or Nose Button).
It only controls the physical robotic dog and does not affect the virtual character.
This block includes a variable input that must be set to either "open" or "close" to specify if the eyes are opened or closed.
The block cannot be left without a variable; it will produce an error if no variable or an invalid variable is set.

Possible Errors:
- If the Eyes block is placed in a sequence without an event block (e.g., following the Start Virtual block), an error message will be shown when the user tries to run or upload the code.
- If the variable is left empty or not set to either "open" or "closed", an error message will be displayed.



Mouth

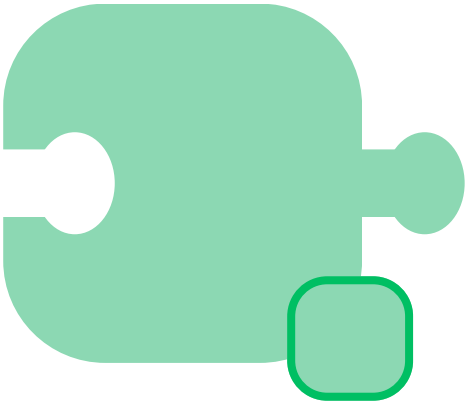
Variable

Overview:
The Mouth block is designed to control the physical robotic dog to open or close its eyes, depending on the variable chosen.

When It Executes:
This block will execute when the corresponding event occurs (e.g., nose button press or Start Physical). The Mouth block can be part of a sequence following any event block designed for the physical dog.

Constraints:
The Mouth block must be placed after an event block (e.g., Start Physical or Nose Button).
It only controls the physical robotic dog and does not affect the virtual character.
This block includes a variable input that must be set to either "open" or "close" to specify if the mouth is opened or closed.
The block cannot be left without a variable; it will produce an error if no variable or an invalid variable is set.

Possible Errors:
- If the Mouth block is placed in a sequence without an event block (e.g., following the Start Virtual block), an error message will be shown when the user tries to run or upload the code.
- If the variable is left empty or not set to either "open" or "closed", an error message will be displayed.



Forwards

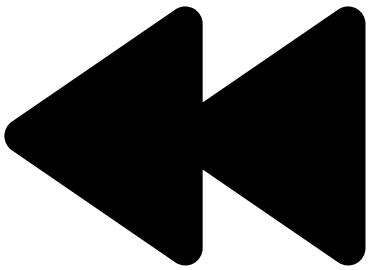
Variable

Overview:
The Forward Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to forward and can only be used in conjunction with blocks that require a directional input.

When It Executes:
The Forward Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Walk block).

Constraints:
The Forward Variable block can only be placed in the variable input field of a block from the variable category (e.g., Walk block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Backwards

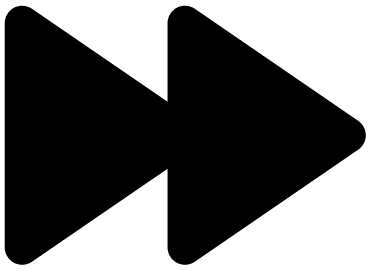
Variable

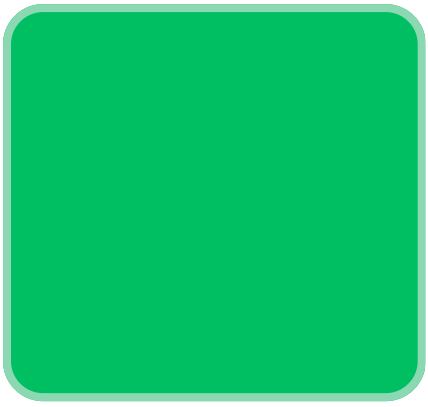
Overview:
The Backward Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to backward and can only be used in conjunction with blocks that require a directional input.

When It Executes:
The Backward Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Walk block).

Constraints:
The Backward Variable block can only be placed in the variable input field of a block from the variable category (e.g., Walk block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:





Right

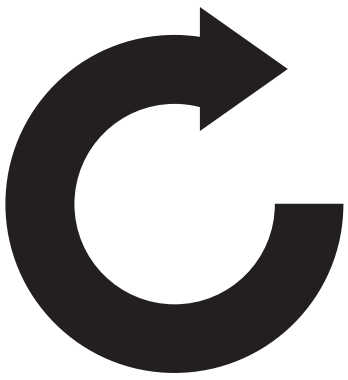
Variable

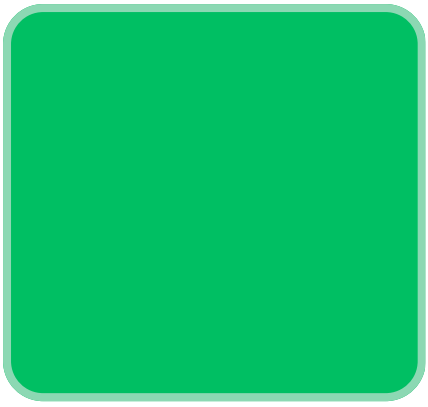
Overview:
The Right Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to backward and can only be used in conjunction with blocks that require a directional input.

When It Executes:
The Right Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Turn block).

Constraints:
The Right Variable block can only be placed in the variable input field of a block from the variable category (e.g., Turn block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:





Left

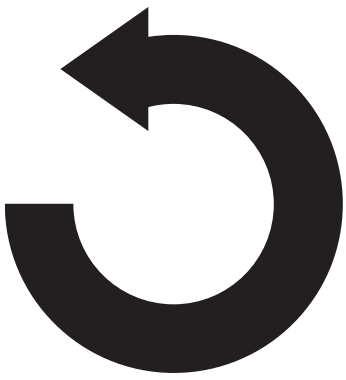
Variable

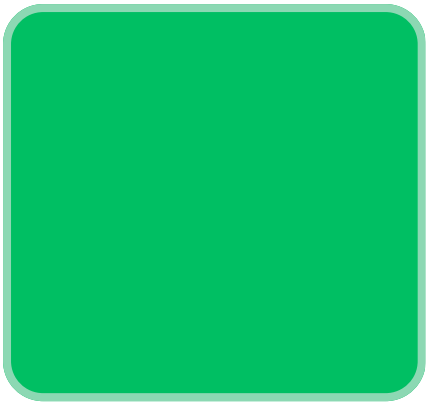
Overview:
The Left Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to backward and can only be used in conjunction with blocks that require a directional input.

When It Executes:
The Left Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Turn block).

Constraints:
The Left Variable block can only be placed in the variable input field of a block from the variable category (e.g., Turn block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:





Front Left Leg

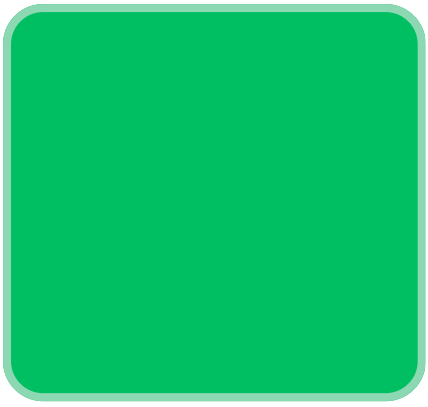
Variable

Overview:
The Front Left Leg Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the Leg to Front Left Leg and can only be used in conjunction with blocks that require a body part input.

When It Executes:
The Front Left Leg Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Lift Leg block).

Constraints:
The Front Left Leg Variable block can only be placed in the variable input field of a block from the variable category (e.g., Lift Leg block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Front Right Leg

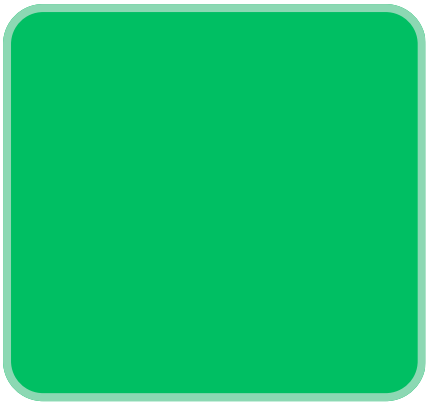
Variable

Overview:
The Front Right Leg Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the Leg to Front Right Leg and can only be used in conjunction with blocks that require a body part input.

When It Executes:
The Front Right Leg Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Lift Leg block).

Constraints:
The Front Right Leg Variable block can only be placed in the variable input field of a block from the variable category (e.g., Lift Leg block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Back Right Leg

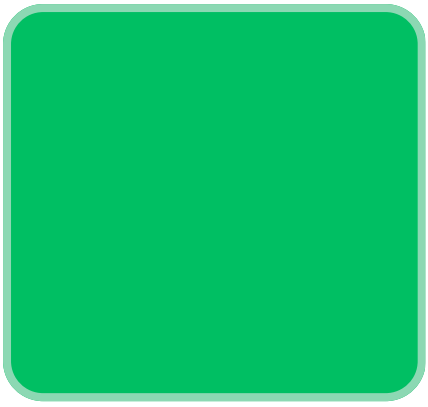
Variable

Overview:
The Back Right Leg Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the Leg to Back Right Leg and can only be used in conjunction with blocks that require a body part input.

When It Executes:
The Back Right Leg Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Lift Leg block).

Constraints:
The Back Right Leg Variable block can only be placed in the variable input field of a block from the variable category (e.g., Lift Leg block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Back Left Leg

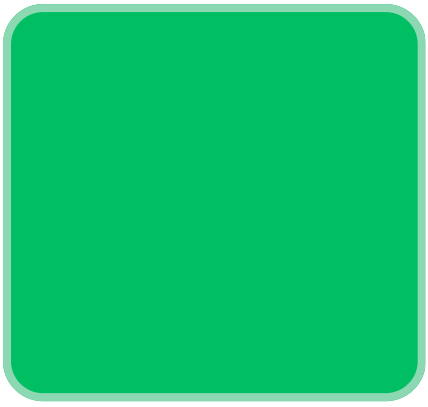
Variable

Overview:
The Back Left Leg Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the Leg to Back Left Leg and can only be used in conjunction with blocks that require a body part input.

When It Executes:
The Back Left Leg Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Lift Leg block).

Constraints:
The Back Left Leg Variable block can only be placed in the variable input field of a block from the variable category (e.g., Lift Leg block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Open

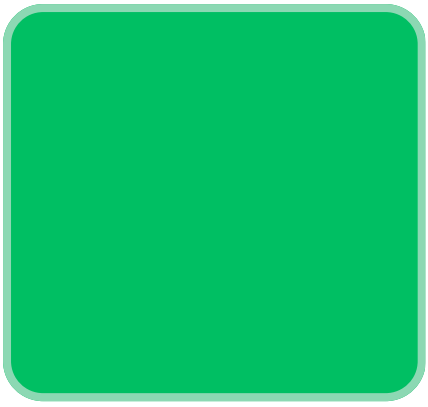
Variable

Overview:
The Open Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to backward and can only be used in conjunction with blocks that require a positional input.

When It Executes:
The Open Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Mouth block).

Constraints:
The Open Variable block can only be placed in the variable input field of a block from the variable category (e.g., Mouth block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Closed

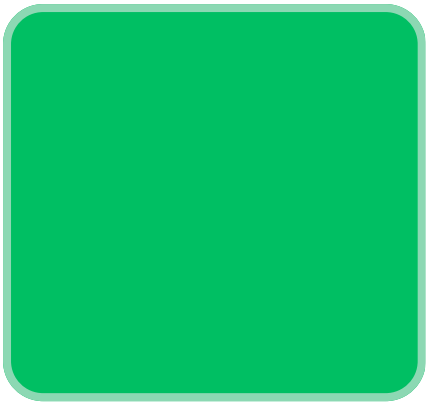
Variable

Overview:
The Close Variable block is designed to define the direction for movement-related actions within the physical robotic dog's code. This block specifically sets the movement to backward and can only be used in conjunction with blocks that require a positional input.

When It Executes:
The Close Variable block is not executed on its own; instead, it acts as a parameter for movement or action blocks that require direction. It must be placed within a variable input field of a variable category block (e.g., Mouth block).

Constraints:
The Close Variable block can only be placed in the variable input field of a block from the variable category (e.g., Mouth block).
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:



Control

Repeat

Control

Overview:
The Repeat block is designed to loop over a sequence of blocks, repeating their execution for a specified number of times. This block allows users to efficiently control actions or movements in repeated cycles.

When It Executes:
The Repeat block runs when the code is executed, and it will repeat the series of blocks within its loop as many times as specified by the user. On the right side of the block, there is a segment resembling a block with 8 spaces, which defines how many times the loop will run.

How It Works:

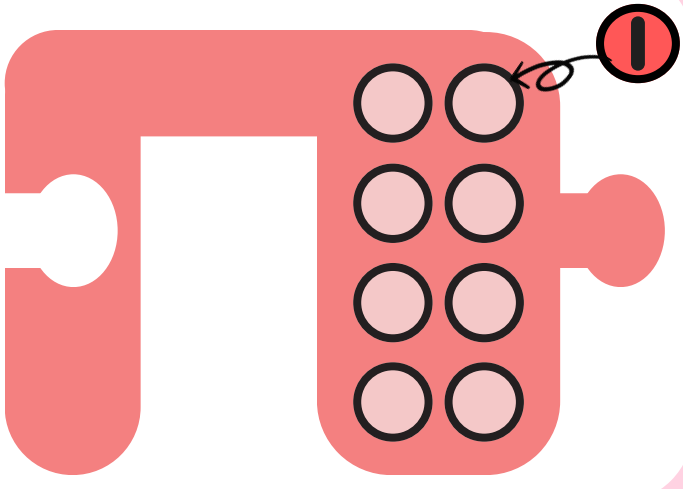
- The number of repetitions is determined by placing a token into one of the black spots on the right side of the block.
- The token defines the exact number of iterations, with each space representing a different repeat count.
- The user can drag and drop the token into the desired spot to set the number of times the enclosed code will repeat.

Constraints:

- The Repeat block must enclose other blocks within it, and it cannot be used as a standalone block.
- The user must select a repeat count by placing a token into one of the spaces; otherwise, the loop will not function properly.

Possible Errors:

- If the user does not place a token in any of the spaces, an error message will be shown, indicating that the repeat count has not been set.



Wait

Control

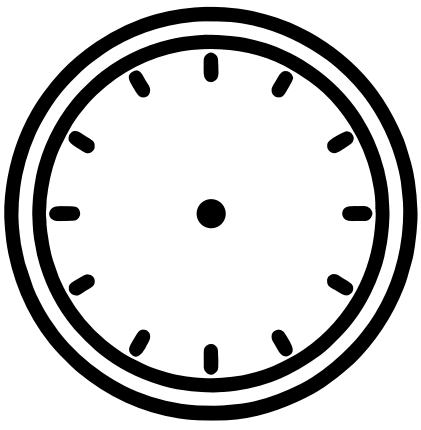
Overview:
The Wait block is designed to create a delay in the code before the next sequence is executed. This block allows user to specify the amount of time they would like the code to pause for.

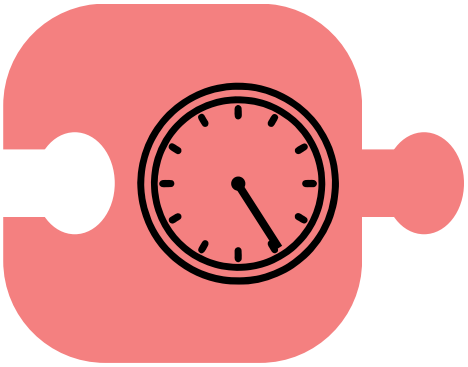
When It Executes:
The wait block executes when it is read in a sequence of code as long as it has a starting event block.

How It Works:
- The amount of time to wait is specified by moving the hand on the clock.
- Each increment (number on the clock) is 1 additional second waited
- 12 or more seconds require a second wait block or repeat of this block

Constraints:

Possible Errors:
- If the user specifies a 0 second wait time the code will produce an error message





If Else

Control

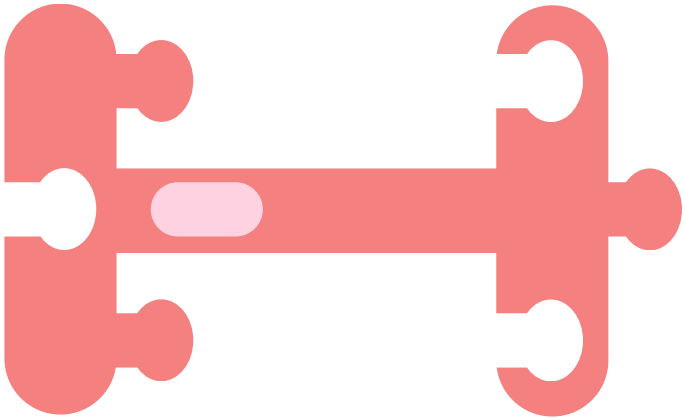
Overview:
The if else block allows conditional execution in the coding language. The block allows the user to input a condition. If the condition is is true the top area of code will execute. If not the bottom section of code will execute.

When It Executes:
The if else block executes when it is read in a sequence of code as long as it has a starting event block no matter which one.

How It Works:
- The condition being asses will be placed in the pink section
- The top line of code is the true statement
- The bottom is the false statement

Constraints:
The block cannot be left without a conditional input; it will produce an error if no variable or an invalid variable is set.

Possible Errors:
- If the condition is left empty or not set to either "open" or "closed", an error message will be displayed.



Dog Conditions

Control

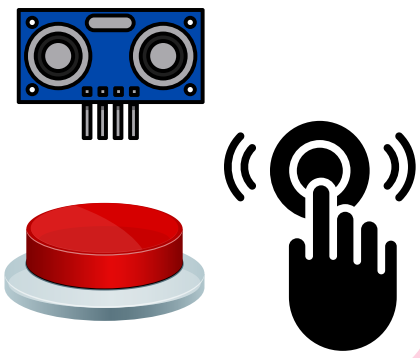
Overview:
The condition block works in conjunction with the if else block to specify the condition tested.

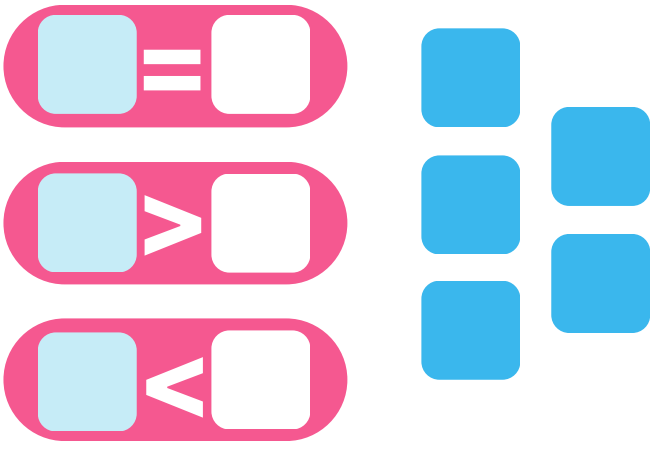
When It Executes:
The conditional blocks can only be used inside an if else block that is in a sequence of code that doe snot start with the virtual start event block

How It Works:
- There are 5 different sensor types, ie, ultrasonic left. represented as dark blue squares
- These sensors go into the light blue squares
- The white square is typed into to make the condition
- These can be dragged into if else

Constraints:
The Close Variable block can only be placed in the condition input field of a if else block
It cannot be placed independently in a sequence of code or anywhere outside of a variable spot.
This block is specifically for controlling the physical robotic dog.

Possible Errors:
- If it is placed outside the condition field an error will occur
- If it is placed in a sequence of code starting with the virtual start block an error will occur





Sounds

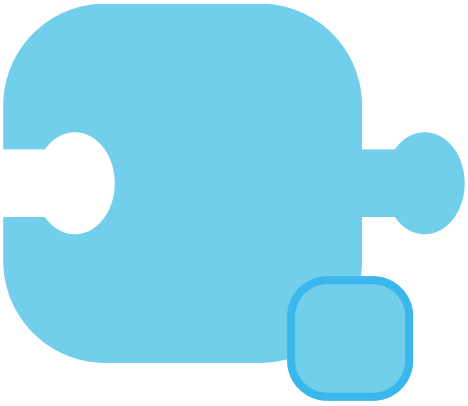
Sound Block

Overview:
he Sound block is crafted to control the sounds made by the physical robotic dog, allowing it to emit different sounds based on the variable chosen.

When It Executes:
This block activates when triggered by a corresponding event (e.g., nose button press or Start Physical). The Sound block can be incorporated into a sequence following any event block tailored for the physical dog.

Constraints:
The Sound block must be positioned after an event block (e.g., Start Physical or Nose Button). It exclusively controls the physical robotic dog and has no impact on the virtual character. This block includes a variable input for selecting the sound to be played. It cannot operate without a variable; failing to set a valid sound variable will lead to an error.

Possible Errors:
- Placing the Sound block in a sequence without a preceding event block (e.g., after the Start Virtual block) will trigger an error message when attempting to run or upload the code. If the variable is left unset or set incorrectly, an error message will be displayed.



Sounds

Overview:
The Bark Variable block is designed to specify the type of bark sound within the physical robotic dog's code. This block allows the selection of different bark sounds, enhancing the interactivity of the dog.

When It Executes:
The Bark Variable block is not executed independently; instead, it serves as a parameter for sound-related blocks. It must be placed within the variable input field of a block like the Sound block that requires sound selection.

Constraints:
The Bark Variable block can only be placed in the variable input field of a block from the variable category (e.g., Sound block). It cannot function independently in a sequence of code or be placed anywhere outside of a variable spot. This block is specifically configured for use with the physical robotic dog, controlling its sound outputs.

Possible Errors:
If the Bark Variable block is placed outside of a variable input field or used without a corresponding event block, it will result in an error during code execution. An error message will be displayed if the Bark Variable is left unset or set to an unrecognized bark type.

Customisation

Speed

Overview:
The Speed block is designed to control the movement speed of the physical robotic dog, affecting all actions following its placement in the code. It includes a dropdown tab that allows users to choose the speed setting.

When It Executes:
This block will execute when triggered by a corresponding event (e.g., nose button press or Start Physical). It can be integrated into a sequence following any event block designed for the physical dog, influencing the speed of subsequent actions.

Constraints:
The Speed block must be placed after an event block (e.g., Start Physical or Nose Button). It solely affects the physical robotic dog and does not impact the virtual character. This block includes a variable input with a dropdown menu to select the speed (e.g., slow, medium, fast). The block cannot be left without setting a speed; it will produce an error if the dropdown is left unselected or if an invalid speed is chosen.

Possible Errors:
If the Speed block is placed in a sequence without a preceding event block (e.g., following the Start Virtual block), an error message will be displayed when the user attempts to run or upload the code. If the speed is not specified or an invalid option is selected from the dropdown, an error message will be displayed.