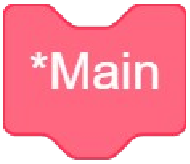
Pengine 5.0

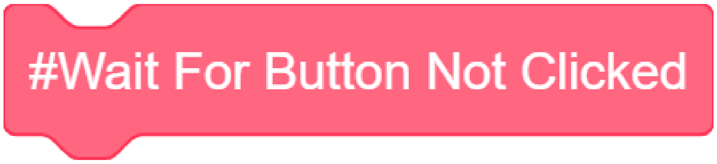
Documentation

This document covers all of the custom blocks in Pengine.



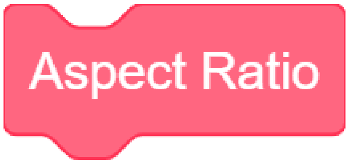
*Main

This block runs during every Scratch tick (unless you've modified the code). All rendering scripts should be placed under this block.



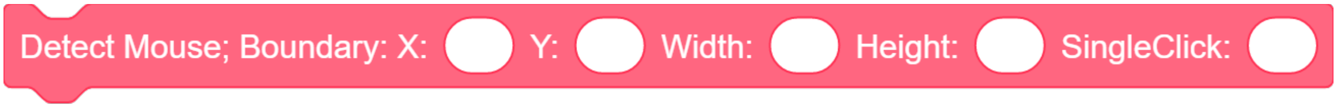
#Wait For Button Not Clicked

The details of this block will be explained in the Detect Mouse section.



Aspect Ratio

This block also runs every tick and sets the variables '!Screen Height' and '!Screen Width' to their respective values.



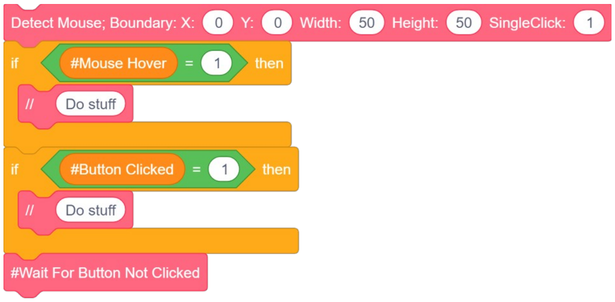
Detect Mouse; Boundary: X: ( ) Y: ( ) Width: ( ) Height: ( ) SingleClick: ( )

This block detects the mouse when it's hovered over or clicked in an area specified by the X, Y, Width, and Height parameters. It's centered, so a square of 50px wide and tall will have its highest point at 25px. When the mouse is clicked within the boundary, it sets the variable '#Mouse Down' to 1.

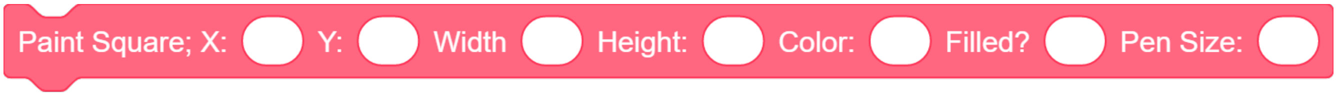
If the mouse is hovered over the boundary, it sets '#Mouse Hover' to 1. If the SingleClick parameter is set to 1 and the mouse is clicked, it will set the variable '#Button Clicked' to 1, but only for a single frame.

When using the SingleClick parameter, make sure there's a '#Wait For Button Not Clicked' block below the scripts.

Example Script:

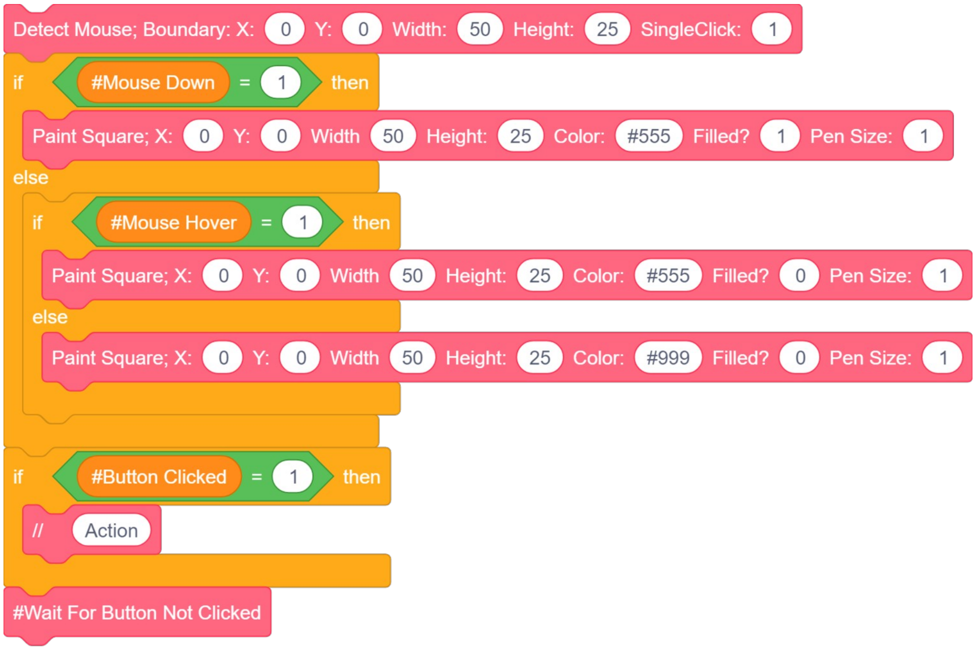


```
Detect Mouse; Boundary: X: (0) Y: (0) Width: (50) Height: (50) SingleClick: (1)
if <(#Mouse Hover) = (1)> then
    // (Do stuff)
if <(#Button Clicked) = (1)> then
    // (Do stuff)
#Wait For Button Not Clicked
```

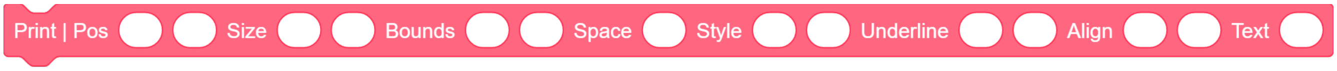


```
Paint Square; X: ( ) Y: ( ) Width ( ) Height: ( ) Color: ( ) Filled? ( ) Pen Size: ( )
```

This block allows you to draw a rectangle at the specified location. Keep in mind that it's also centered. When the 'Filled?' parameter is set to 1, make sure the Pen Size parameter is also set to a value; otherwise, it may slow down and potentially crash. Here's an example script for a button:

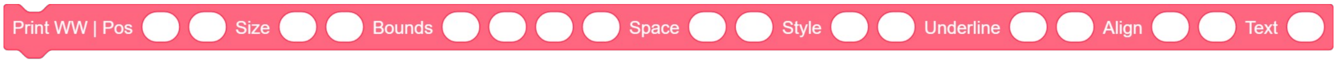


```
Detect Mouse; Boundary: X: (0) Y: (0) Width: (50) Height: (25) SingleClick: (1)
if <(#Mouse Down) = (1)> then
  Paint Square; X: (0) Y: (0) Width (50) Height: (25) Color: (#555) Filled? (1) Pen Size: (1)
else
  if <(#Mouse Hover) = (1)> then
    Paint Square; X: (0) Y: (0) Width (50) Height: (25) Color: (#555) Filled? (0) Pen Size: (1)
  else
    Paint Square; X: (0) Y: (0) Width (50) Height: (25) Color: (#999) Filled? (0) Pen Size: (1)
if <(#Button Clicked) = (1)> then
  // (Action)
#Wait For Button Not Clicked
```



```
Print | Pos ( ) ( ) Size ( ) ( ) Bounds ( ) ( ) Space ( ) Style ( ) ( ) Underline ( ) ( ) Align ( ) ( ) Text ( )
```

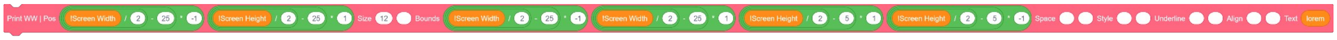
This block is straightforward and used to print text. The Size parameters determine the size of the text. The first parameter sets the scale, and the second sets the stretch of the text.
The bounds parameters determine what text to print. If the position is set to 0 and the second bound is set to 15, text after x position 15 won't print; the same rule applies to the first parameter but in the opposite direction.
The first Style parameter sets the boldness of the text, and the second parameter sets how slanted (italicized) the text is.
The first Underline parameter sets the y position of the underline (0 is at the bottom of the text, and 1 is above), and the second parameter sets the boldness.
The Align parameters set the alignment of the text (L, C, R) for the first, and (T, C, B) for the second; you can also use numbers to offset the alignment.



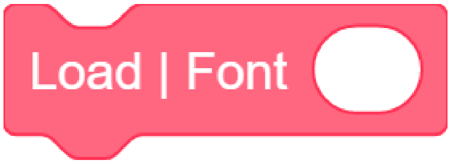
This block performs the same function as the first print block, with a few additional parameters for word wrapping. There are now two Space parameters; the first one controls horizontal spacing, and the second controls vertical spacing.

There are four Bounds parameters; the first two determine the starting and maximum x positions, while the third and fourth parameters clip the text vertically (if the text goes beyond the fourth parameter, it won't render).

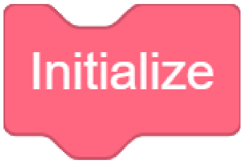
Here's a script that renders lorem ipsum across the entire screen:



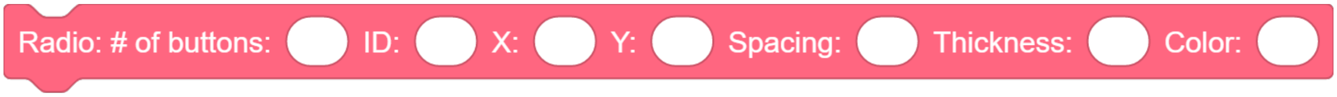
```
Print WW | Pos (!Screen Width / 2 - 25 * -1) (!Screen Height / 2 - 25 * 1) Size 12 ( ) Bounds (!Screen Width / 2 - 25 * -1) (!Screen Width / 2 - 25 * 1) (!Screen Height / 2 - 5 * 1) (!Screen Height / 2 - 5 * -1) Space ( ) ( ) Style ( ) ( ) Underline ( ) ( ) Align ( ) ( ) Text lorem
```



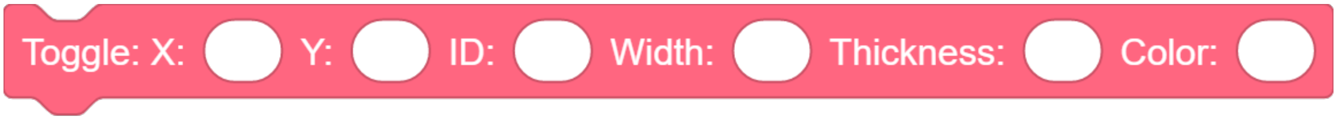
This block allows you to set the font, and included fonts are: Parabolic, Curvature, Geometric, Angular.



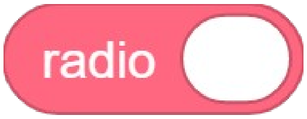
Runs at the beginning of the project

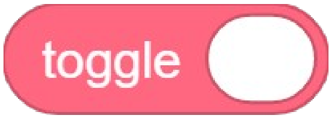


A radio displays buttons in a column, and you can select only one button at a time. Each radio set has a unique ID, and two sets cannot share the same ID.



A toggle or switch, as the name suggests, can switch between two states: 1 or 0. The same rule about unique IDs applies to toggles.





(Only in the Turbowarp version)
Return the value of the set ID

Radio would return something like 010 depending on the amount of buttons you have set
Toggle will only return a 1 or 0