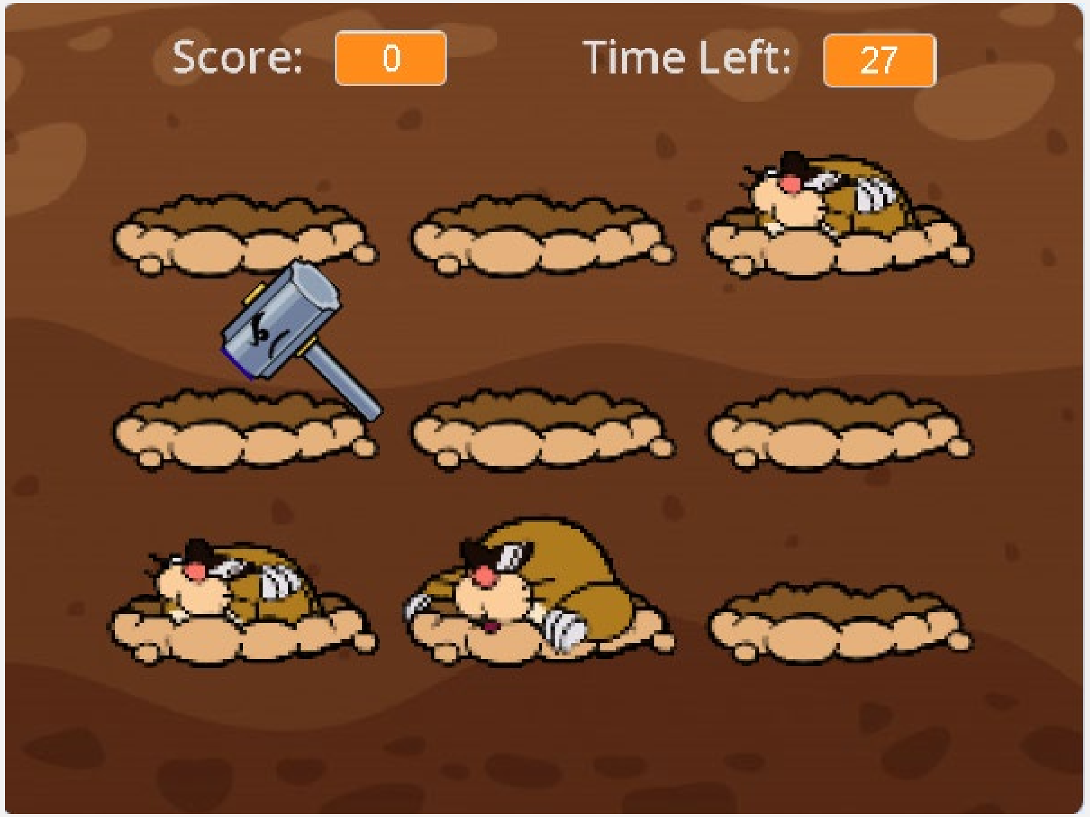
Scratch Programming

Lesson 2-3

Whack-a-mole Game III

Presented by Advaspire Team

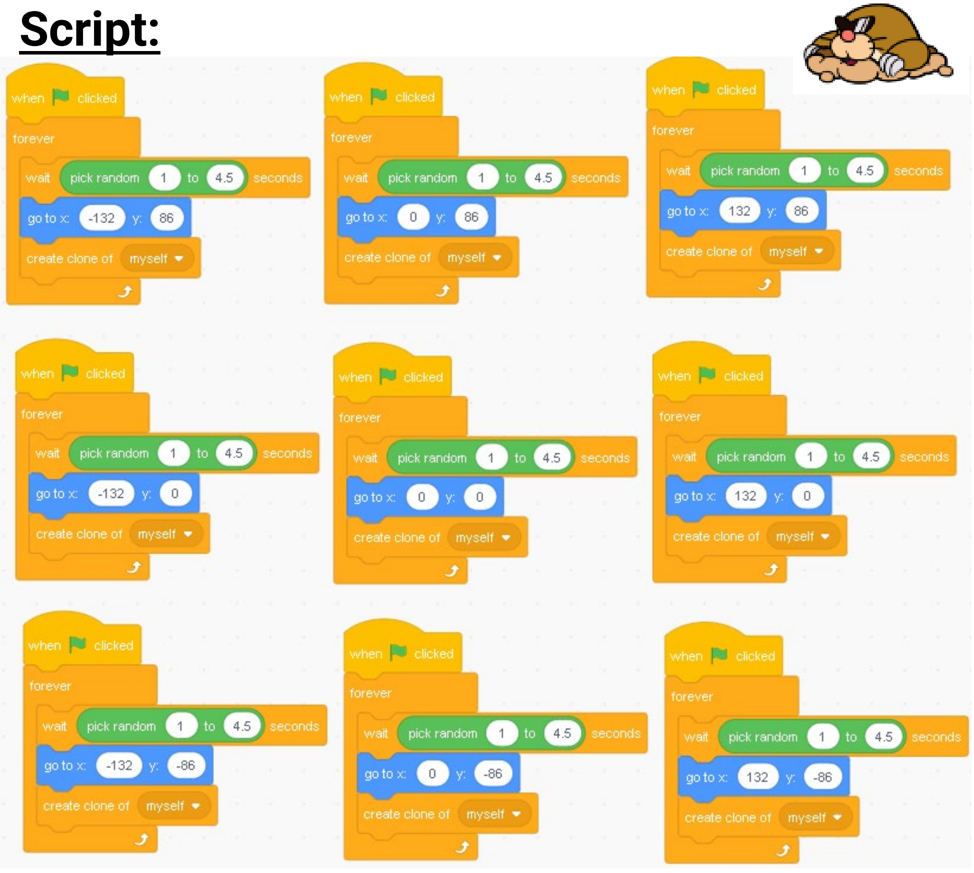
Review - Multiple Moles coming out



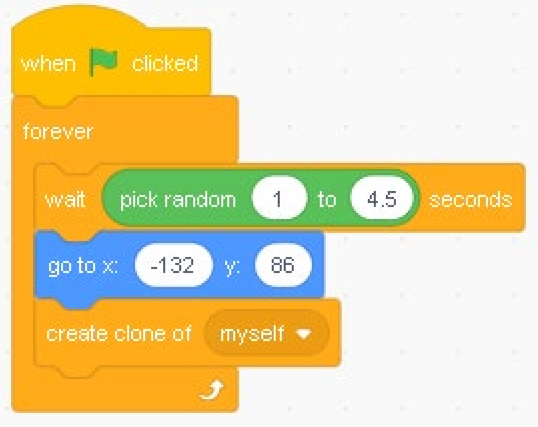
If we want to allow multiple moles coming out, we will have to make our moles into clones, and we don't need to use the real hole to track for random position anymore.

Review – Clones for Mole (Multiple spawn)

Script:



```
when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (-132) y: (86)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (0) y: (86)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (132) y: (86)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (-132) y: (0)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (0) y: (0)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (132) y: (0)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (-132) y: (-86)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (0) y: (-86)
    create clone of (myself ▾)

when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (132) y: (-86)
    create clone of (myself ▾)
```



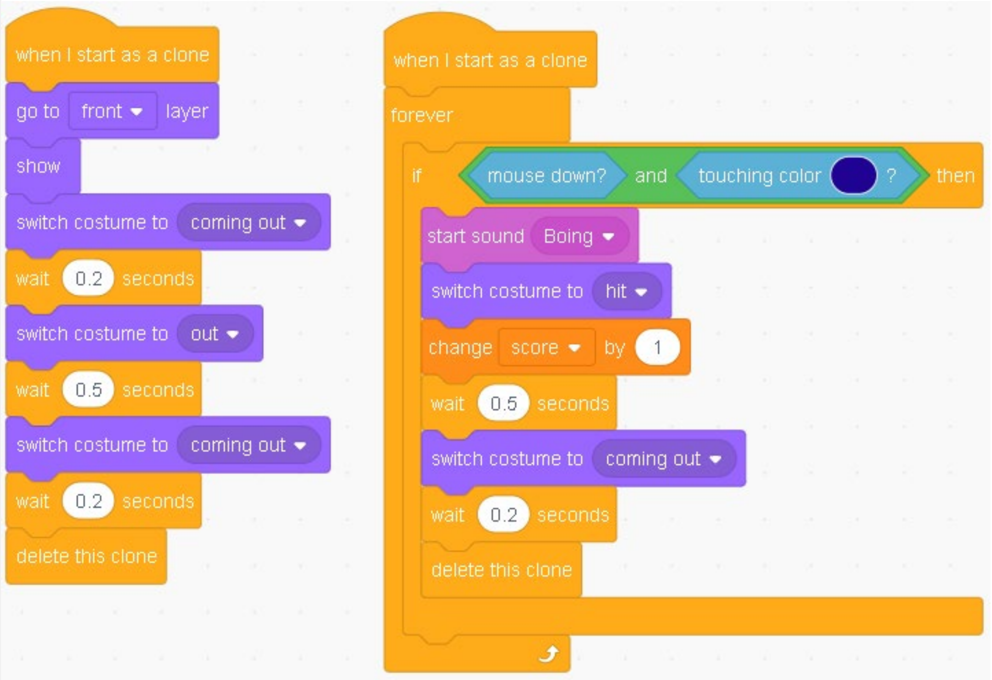
```
when [flag] clicked
forever
    wait (pick random (1) to (4.5)) seconds
    go to x: (-132) y: (86)
    create clone of (myself ▾)
```

Each hole will have 1 mole, so we need to set up 9 moles (clone) running concurrently.

Then we add pick random from 1 to 4.5 seconds of waiting time for the interval of creating clone.

Review - Function of the mole (For Clone)

Script:



```
when I start as a clone
go to front ▾ layer
show
switch costume to coming out ▾
wait 0.2 seconds
switch costume to out ▾
wait 0.5 seconds
switch costume to coming out ▾
wait 0.2 seconds
delete this clone
```

```
when I start as a clone
forever
    if < <mouse down?> and <touching color ● ?> > then
        start sound Boing ▾
        switch costume to hit ▾
        change score ▾ by 1
        wait 0.5 seconds
        switch costume to coming out ▾
        wait 0.2 seconds
        delete this clone
```

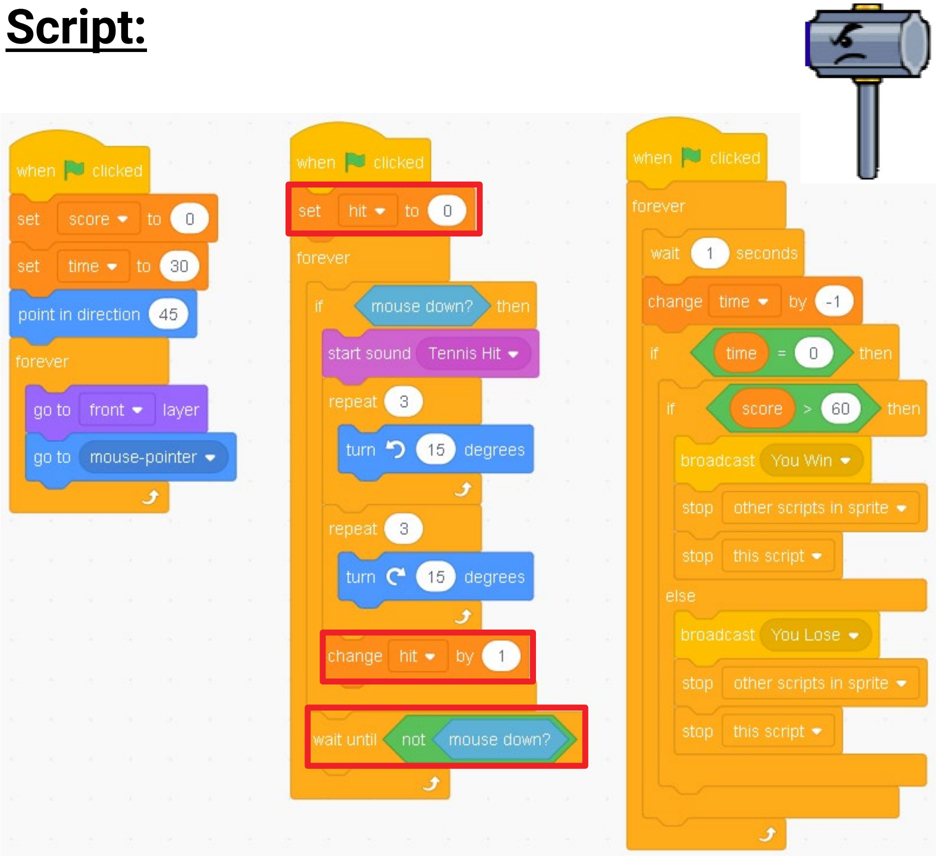
With every 1 to 4.5 seconds (randomly), the clones will spawn at all locations (different time interval for different holes).

And when the clone is created (spawned), it will go to front layer and show itself, followed by coming out and going back to the hole.

When you hit the clone by hammer, it will delete the clone and add 1 point to the score.

Review - Measure Accuracy (Trace hit count)

Script:



```
when 🏴 clicked
set score ▾ to 0
set time ▾ to 30
point in direction 45
forever
    go to front ▾ layer
    go to mouse-pointer ▾

when 🏴 clicked
set hit ▾ to 0
forever
    if mouse down? then
        start sound Tennis Hit ▾
        repeat 3
            turn ↺ 15 degrees
        repeat 3
            turn ↻ 15 degrees
        change hit ▾ by 1
    wait until not mouse down?

when 🏴 clicked
forever
    wait 1 seconds
    change time ▾ by -1
    if time = 0 then
        if score > 60 then
            broadcast You Win ▾
            stop other scripts in sprite ▾
            stop this script ▾
        else
            broadcast You Lose ▾
            stop other scripts in sprite ▾
            stop this script ▾
```

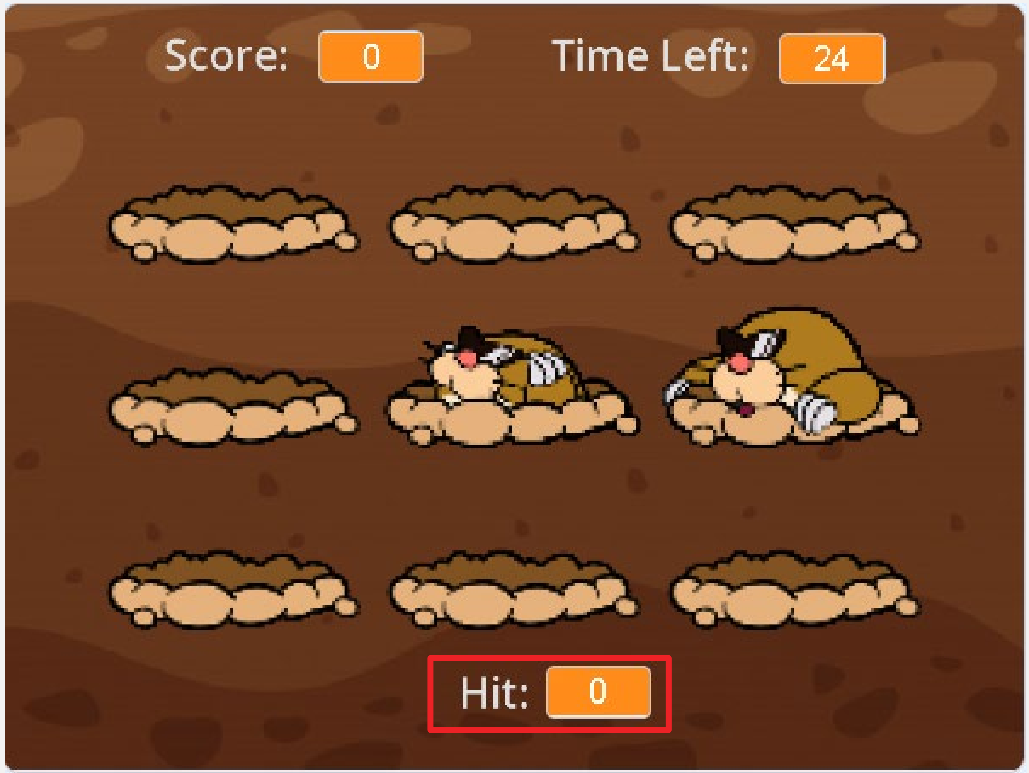
Let's set up a new variable to trace the hit count (how many time you click and smash), I will call it "hit" over here.

Then let's set "hit = 0" at the start.

Once we click the mouse, it will turn the hammer for 45 degree counter-clockwise, then turn it back to original point. Then we change hit by 1 (this is counted as 1 hit).

Then I want to make sure that player won't holding the mouse down forever, so I will set a wait until mouse is up (not mouse down means mouse up), then the hitting action will be enabled again.

Review - Show the hit variable



Now let’s show up the variable “hit” on the screen, and make a label (the “Hit: “ word) on the left side of the readout.

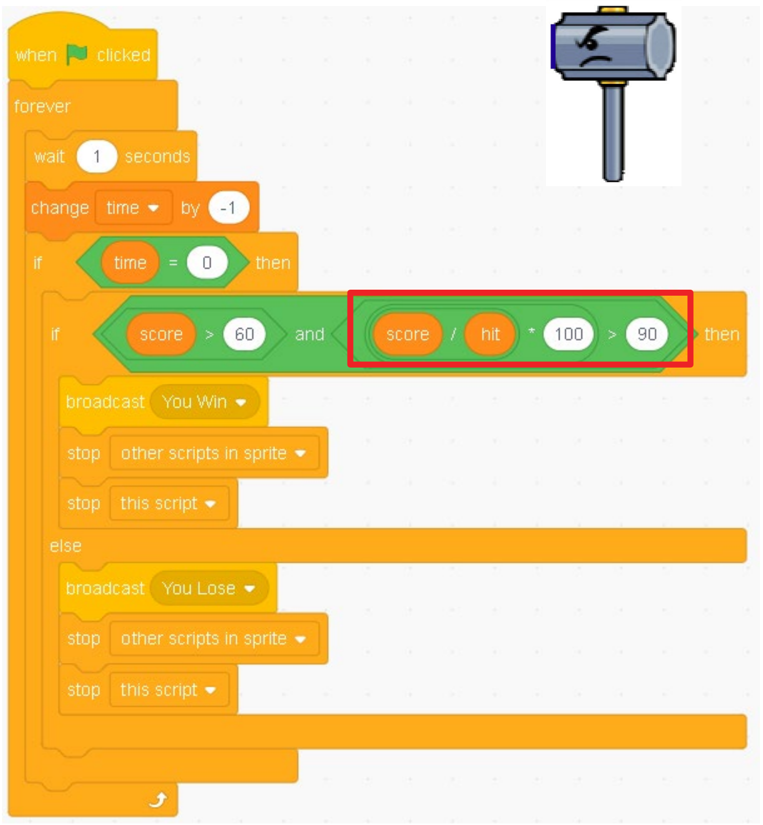
So that we can see how many time we hit and how many score we get.

So the more you hit but less score you get means lower accuracy.

The closer the count of hit and the score, the more accurate you are.

Review - Calculate the Accuracy (Formula)

Script:



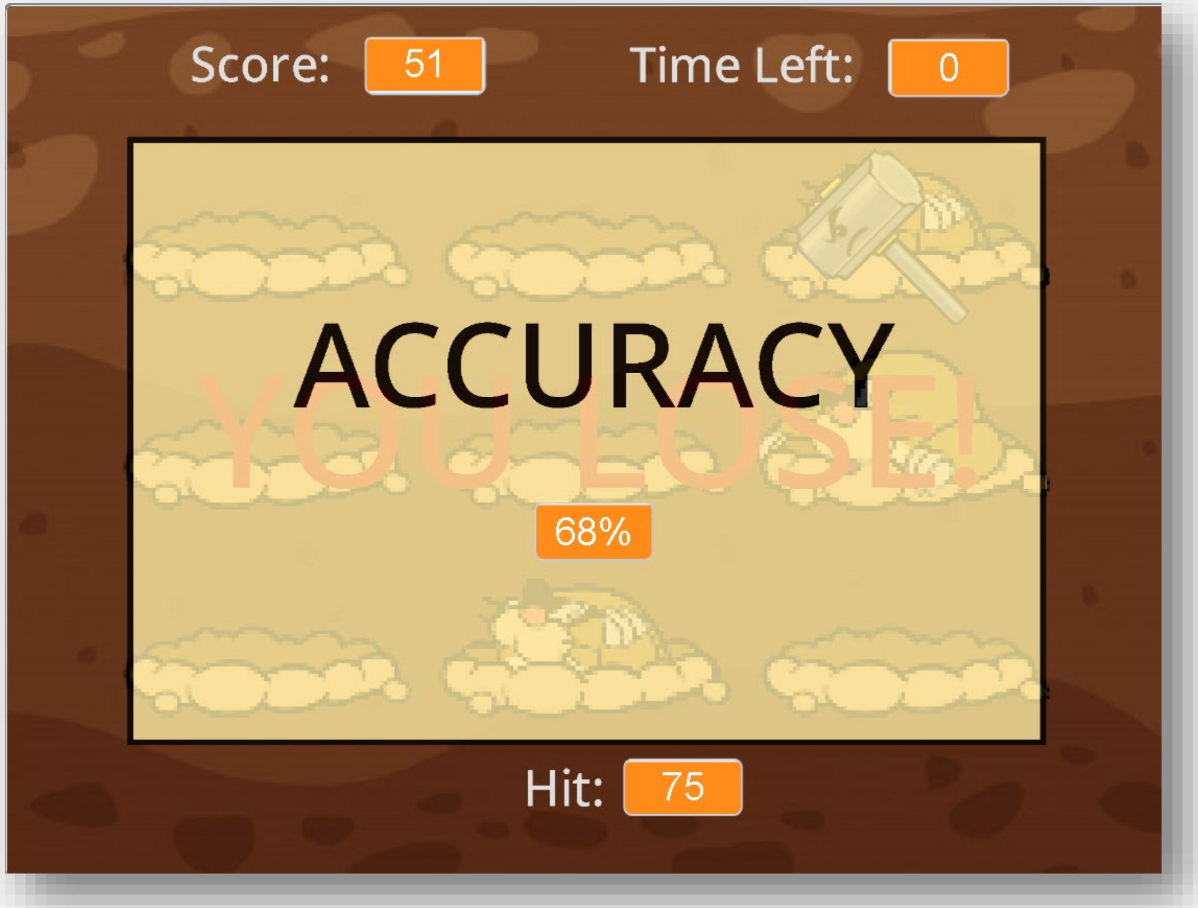
```
when 🏳 clicked
forever
    wait (1) seconds
    change [time ▾] by (-1)
    if <(time) = (0)> then
        if <<(score) > (60)> and <(((score) / (hit)) * (100)) > (90)>> then
            broadcast [You Win ▾]
            stop [other scripts in sprite ▾]
            stop [this script ▾]
        else
            broadcast [You Lose ▾]
            stop [other scripts in sprite ▾]
            stop [this script ▾]
        end
    end
```

$$Accuracy = \frac{Score}{Hit} \times 100\%$$

Let's use this formula to calculate our accuracy.

(Score / hit) * 100 > 90

Mission 2-2 – Showing Result

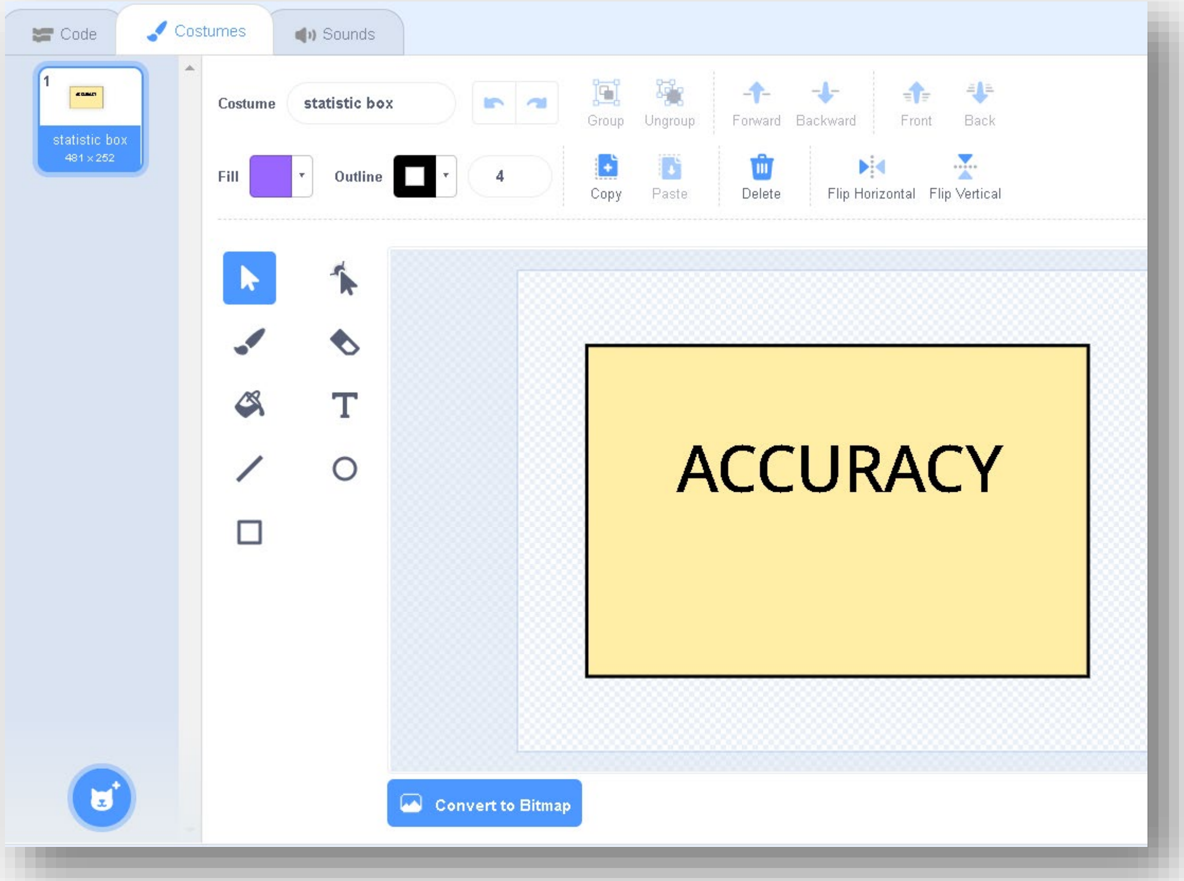


Let’s show the accuracy at the end of the game, no matter you win or lose.

Can you try to do a retry button at the end too?

Player can click the retry button and start the game again.

Solution for Mission 2-2



Now we want to show the accuracy once the game is ended.

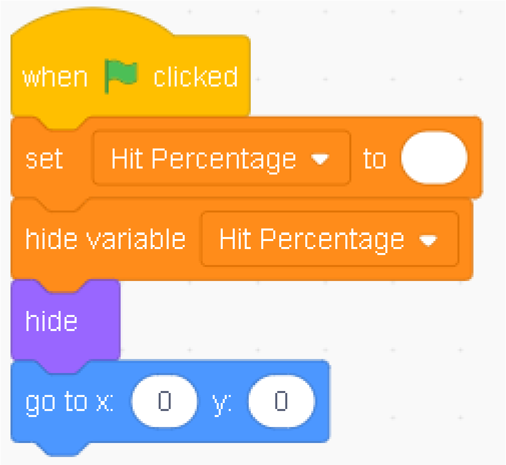
Let's draw a box to show the Accuracy.

Then we need to set a variable for the accuracy (by using the formula of calculating accuracy).

Solution for Mission 2-2

Script:

ACCURACY

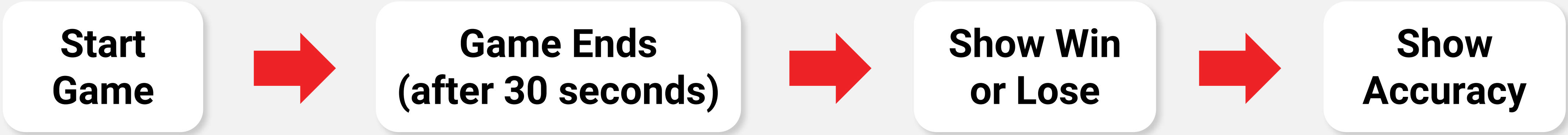


```
when 🏳 clicked
set Hit Percentage ▾ to ( )
hide variable Hit Percentage ▾
hide
go to x: 0 y: 0
```

First thing we want the accuracy box to do is set a new variable called "Hit percentage", and set it to "" (nothing) at start, and hide the variable.

We only want this box and the variable to show once our game ends.

Sequence of the Game



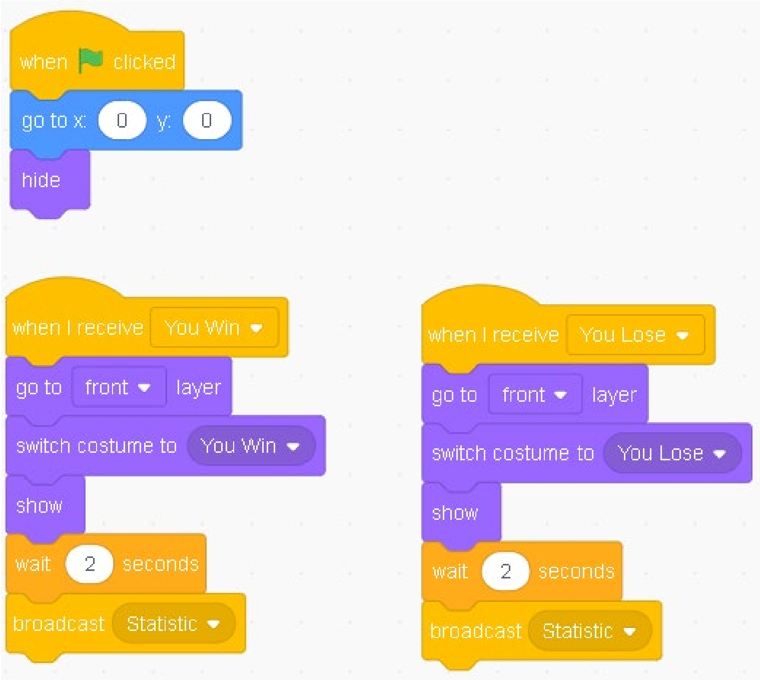
Start Game → Game Ends (after 30 seconds) → Show Win or Lose → Show Accuracy

This will be the sequence of our game, which will show the Win or Lose first once game ends, then only show the accuracy box.

Rescript the Win & Lose

Script:

YOU WIN!



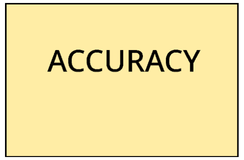
```
when 🏴 clicked
go to x: 0 y: 0
hide

when I receive You Win ▾
go to front ▾ layer
switch costume to You Win ▾
show
wait 2 seconds
broadcast Statistic ▾

when I receive You Lose ▾
go to front ▾ layer
switch costume to You Lose ▾
show
wait 2 seconds
broadcast Statistic ▾
```

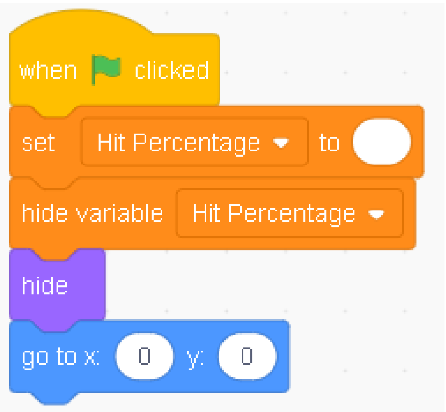
After showing “You Win” or “You Lose”, then we will show the accuracy box after 2 seconds.

I will make a broadcast called “statistic” to call out the Accuracy box.

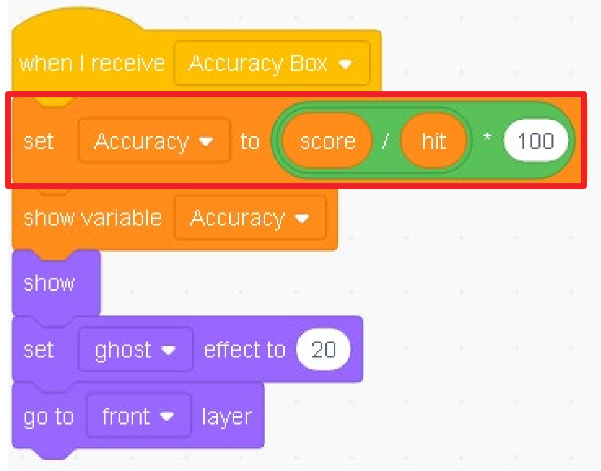
Accuracy Box

Script:





```
when 🏁 clicked
set [Hit Percentage ▾] to ( )
hide variable [Hit Percentage ▾]
hide
go to x: (0) y: (0)
```



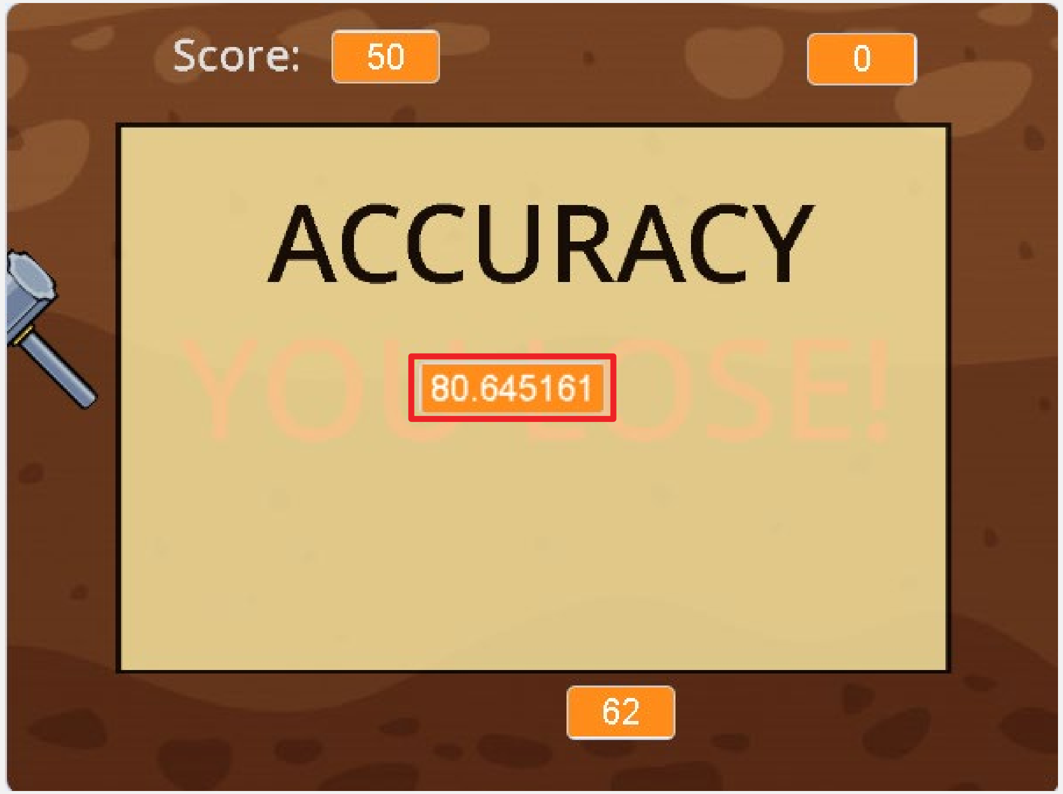
```
when I receive [Accuracy Box ▾]
set [Accuracy ▾] to (((score) / (hit)) * (100))
show variable [Accuracy ▾]
show
set [ghost ▾] effect to (20)
go to [front ▾] layer
```

And here we need to calculate the Hit Percentage by using the formula:

$$(Accuracy = Score \div hit \times 100\%)$$

Then I will show the variable and make my box to the front layer with transparent effect 20%.

Accuracy Box – Decimal Place



By using the formula to calculate, you will see decimal place for the result in Accuracy.

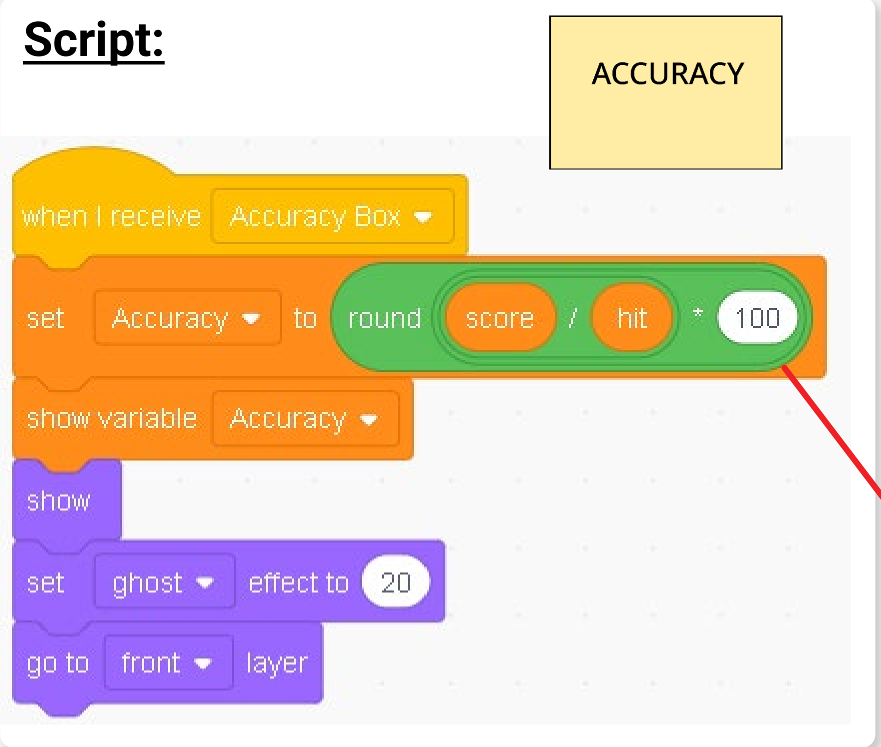
Now we want to round this up and add “%” symbol after it.

80.645161 → 81%

And we have to use some operators to transform this.

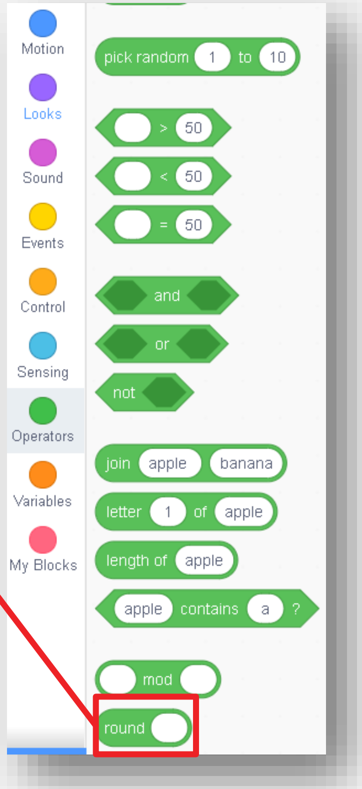
Accuracy Box – Rounding up the number

Script:



ACCURACY

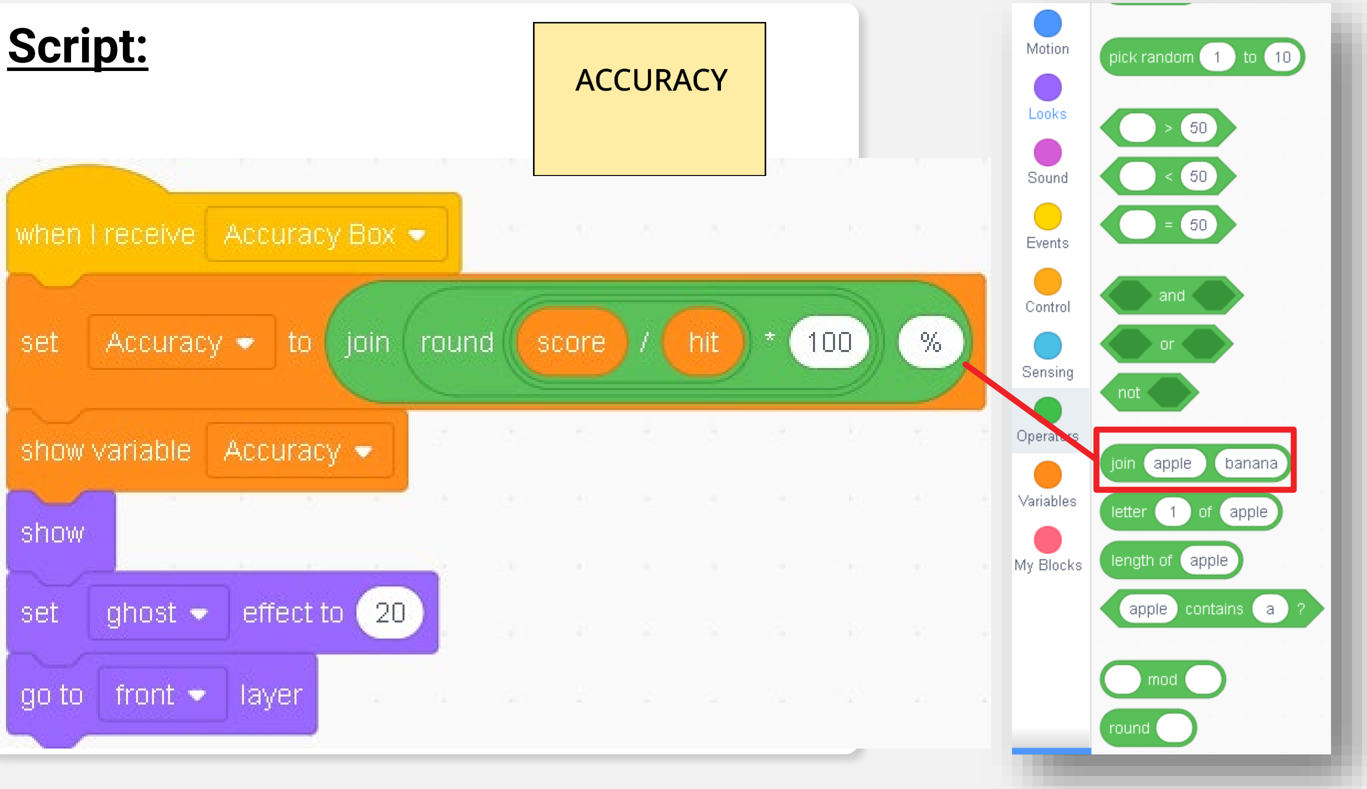
```
when I receive [Accuracy Box ▾]
set [Accuracy ▾] to (round (((score) / (hit)) * (100)))
show variable [Accuracy ▾]
show
set [ghost ▾] effect to (20)
go to [front ▾] layer
```



Let's round this off and make it to integer (number without decimal point).

Accuracy Box – Concatenate with “%”

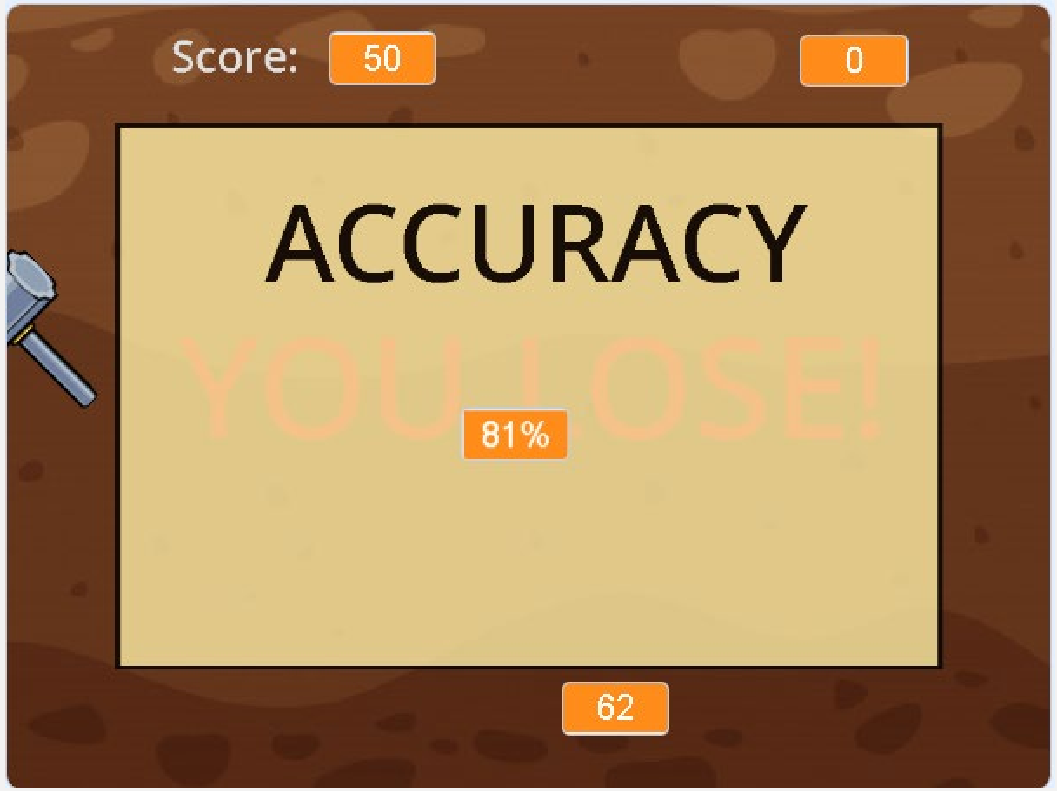
Script:



```
when I receive [Accuracy Box ▾]
set [Accuracy ▾] to (join (round ((score) / (hit)) * (100)) (%))
show variable [Accuracy ▾]
show
set [ghost ▾] effect to (20)
go to [front ▾] layer
```

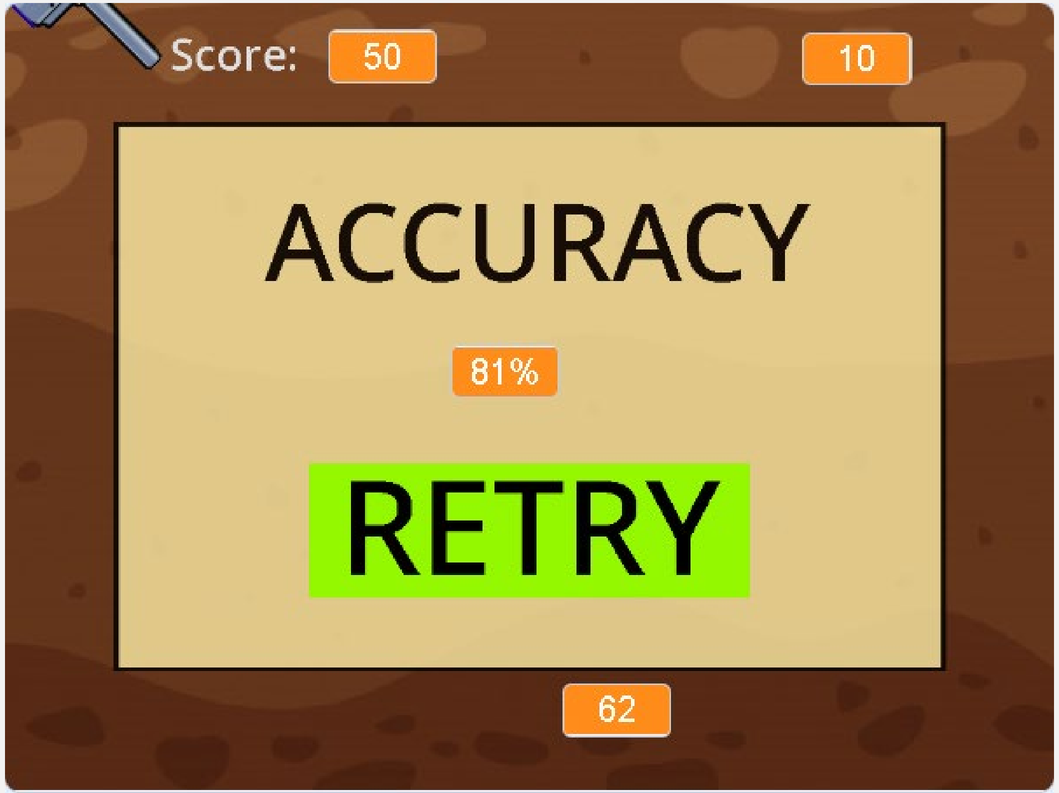
Let’s use the join “__” “__” to join the accuracy with a “%” symbol.

Accuracy Box



You will get this outcome when you run the game.

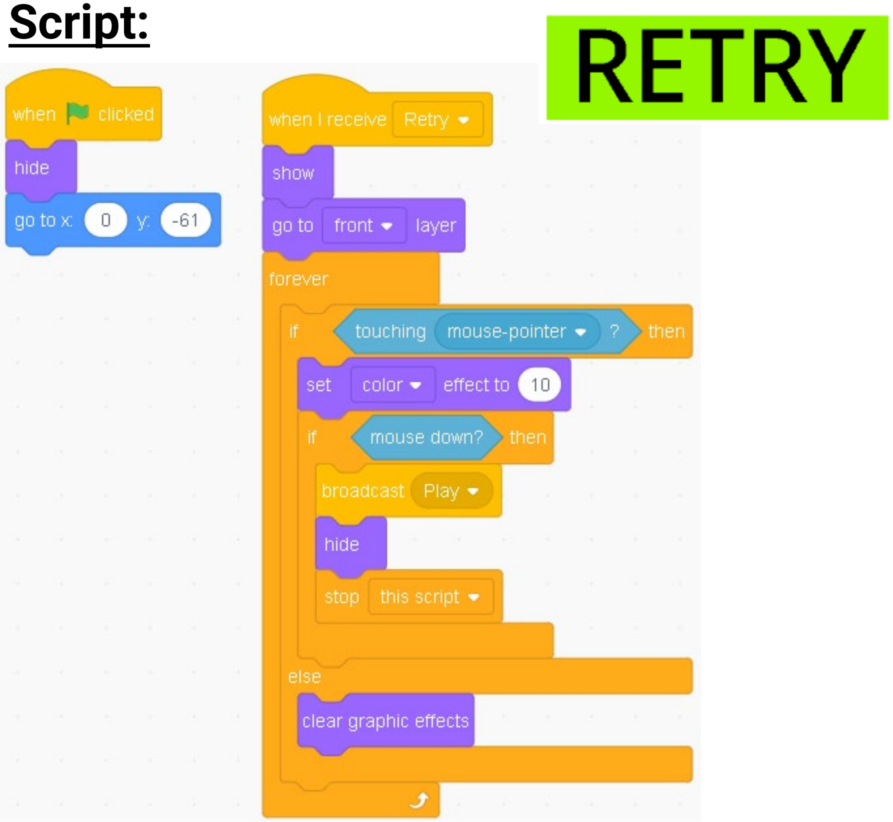
Retry Button



We also want to add a "Retry" button so that when player click this button, it's going to restart the game.

Retry Button - Script

Script:



```
when 🏳 clicked
hide
go to x: 0 y: -61

when I receive Retry ▾
show
go to front ▾ layer
forever
  if <touching mouse-pointer ▾ ?> then
    set color ▾ effect to 10
    if <mouse down?> then
      broadcast Play ▾
      hide
      stop this script ▾
  else
    clear graphic effects
```

Just create the Retry button using paint like how we created the "play" button for the menu page.

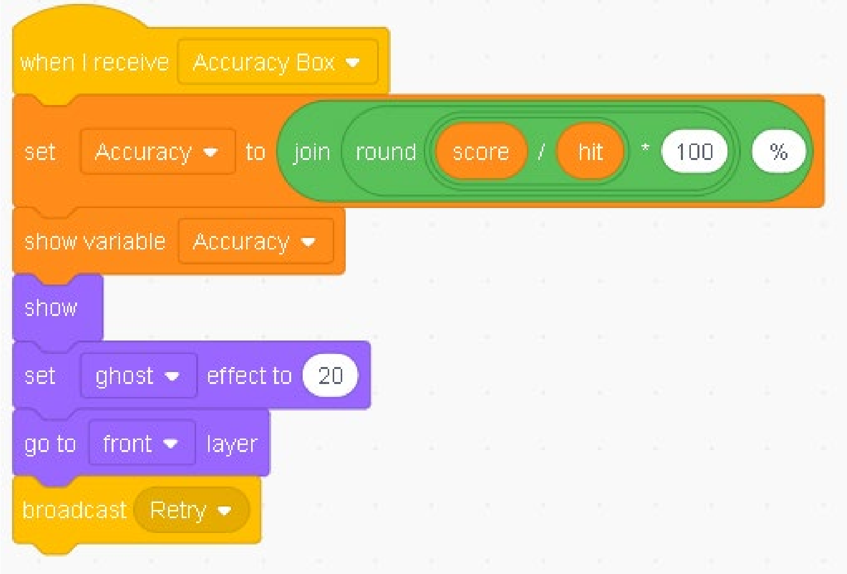
The script is similar to "Play" button, but it has to has a command to make it go to front.

Then we will broadcast "Retry" after the accuracy box came out to call out the "Retry" button.

Accuracy Box – Broadcast Retry

Script:

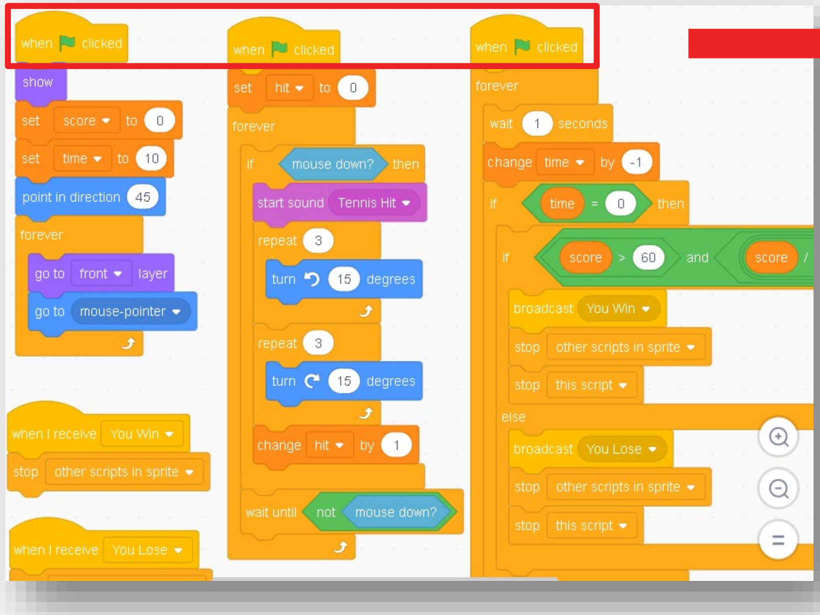
ACCURACY



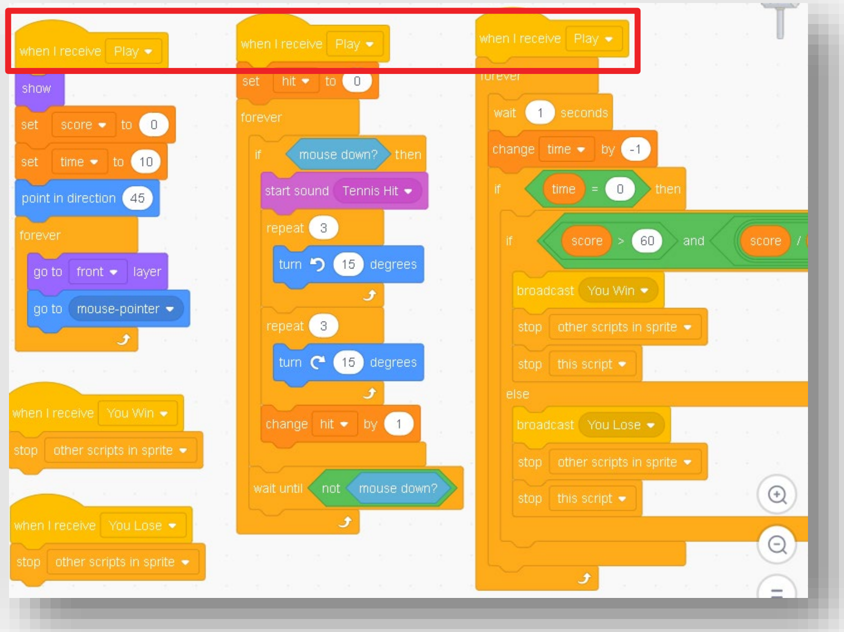
```
when I receive [Accuracy Box ▾]
set [Accuracy ▾] to (join (round ((score) / (hit) * (100))) (%))
show variable [Accuracy ▾]
show
set [ghost ▾] effect to (20)
go to [front ▾] layer
broadcast (Retry ▾)
```

Broadcast "Retry" at the end of the accuracy box.

Change all “Flag” to “Play” for starting blocks



```
when 🏁 clicked
show
set score to 0
set time to 10
point in direction 45
forever
    go to front layer
    go to mouse-pointer

when I receive You Win
stop other scripts in sprite

when I receive You Lose

when 🏁 clicked
set hit to 0
forever
    if mouse down? then
        start sound Tennis Hit
        repeat 3
            turn ↺ 15 degrees
        repeat 3
            turn ↻ 15 degrees
        change hit by 1
        wait until not mouse down?

when 🏁 clicked
forever
    wait 1 seconds
    change time by -1
    if time = 0 then
        if score > 60 and score /
            broadcast You Win
            stop other scripts in sprite
            stop this script
        else
            broadcast You Lose
            stop other scripts in sprite
            stop this script
```

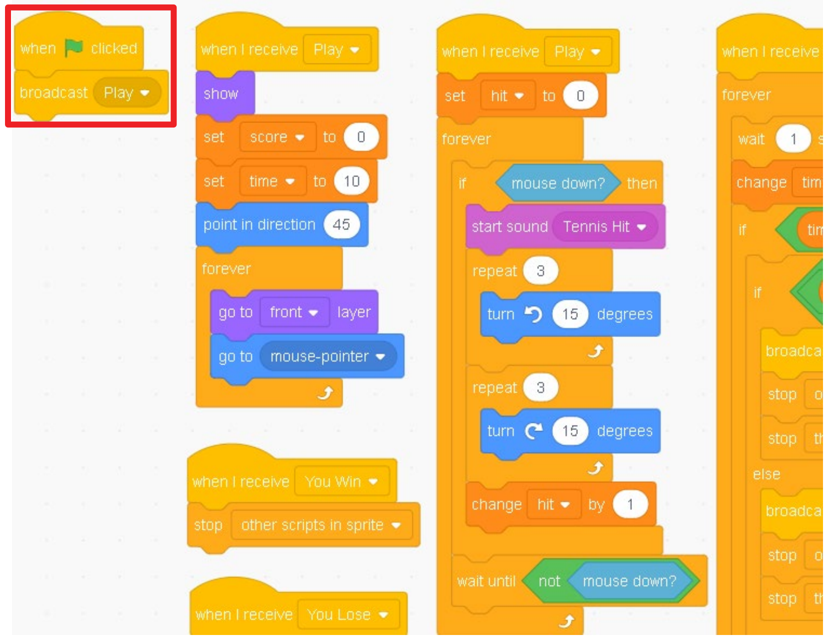


```
when I receive Play
show
set score to 0
set time to 10
point in direction 45
forever
    go to front layer
    go to mouse-pointer

when I receive You Win
stop other scripts in sprite

when I receive You Lose
stop other scripts in sprite

when I receive Play
set hit to 0
forever
    if mouse down? then
        start sound Tennis Hit
        repeat 3
            turn ↺ 15 degrees
        repeat 3
            turn ↻ 15 degrees
        change hit by 1
        wait until not mouse down?

when I receive Play
forever
    wait 1 seconds
    change time by -1
    if time = 0 then
        if score > 60 and score /
            broadcast You Win
            stop other scripts in sprite
            stop this script
        else
            broadcast You Lose
            stop other scripts in sprite
            stop this script
```

All of your starting blocks that are programmed with “When Flag clicked” should be changed to “When received Play”, including for your Hammer, Holes, Moles, Win & Lose broadcast, title, accuracy box and retry button.

Start with broadcast “Play”

Script:



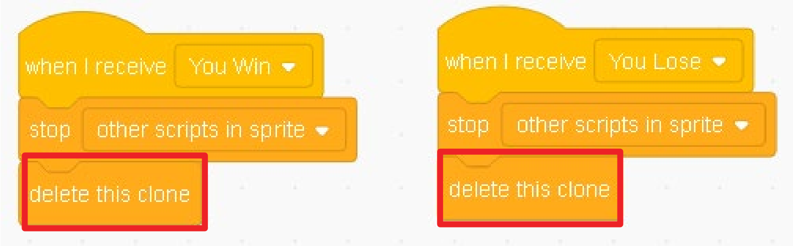
```
when 🏴 clicked
broadcast Play ▾

when I receive Play ▾
show
set score ▾ to 0
set time ▾ to 10
point in direction 45
forever
    go to front ▾ layer
    go to mouse-pointer ▾

when I receive You Win ▾
stop other scripts in sprite ▾

when I receive You Lose ▾

when I receive Play ▾
set hit ▾ to 0
forever
    if mouse down? then
        start sound Tennis Hit ▾
        repeat 3
            turn ↺ 15 degrees
        repeat 3
            turn ↻ 15 degrees
        change hit ▾ by 1
    wait until not mouse down?

when I receive
forever
    wait 1 s
    change tim
    if tim
        if
            broadca
            stop o
            stop th
        else
            broadca
            stop o
            stop th
```

Remember to add 1 “When Flag Clicked”, then broadcast “Play”.

So everything can also restart once you clicked the “Flag”.

Delete the clones once game ended

Script:

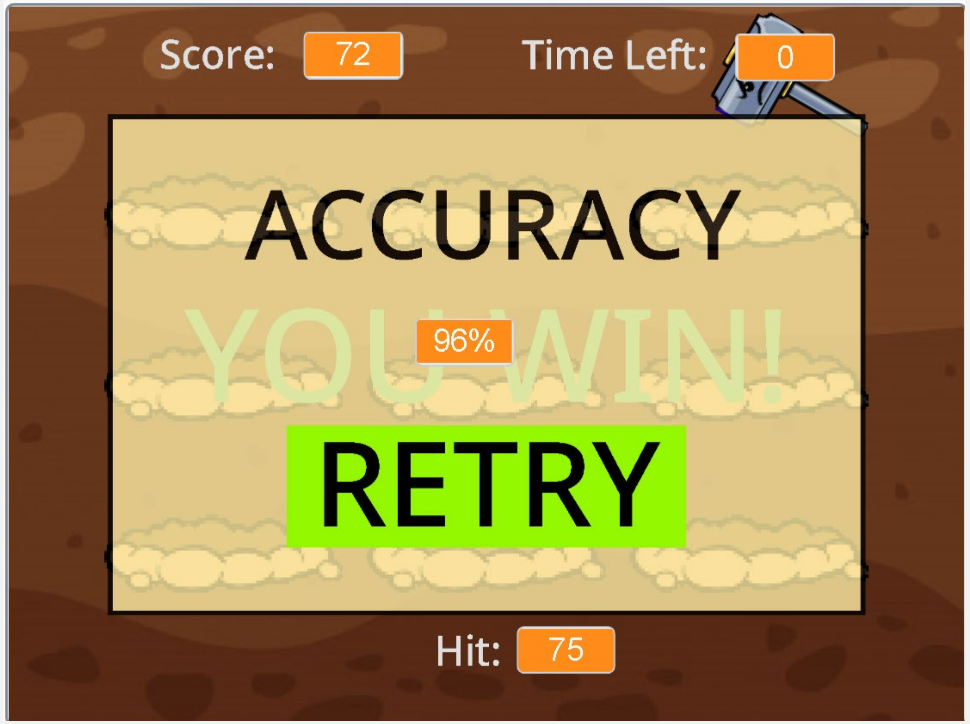


```
when I receive [You Win ▾]
stop [other scripts in sprite ▾]
delete this clone
```

```
when I receive [You Lose ▾]
stop [other scripts in sprite ▾]
delete this clone
```

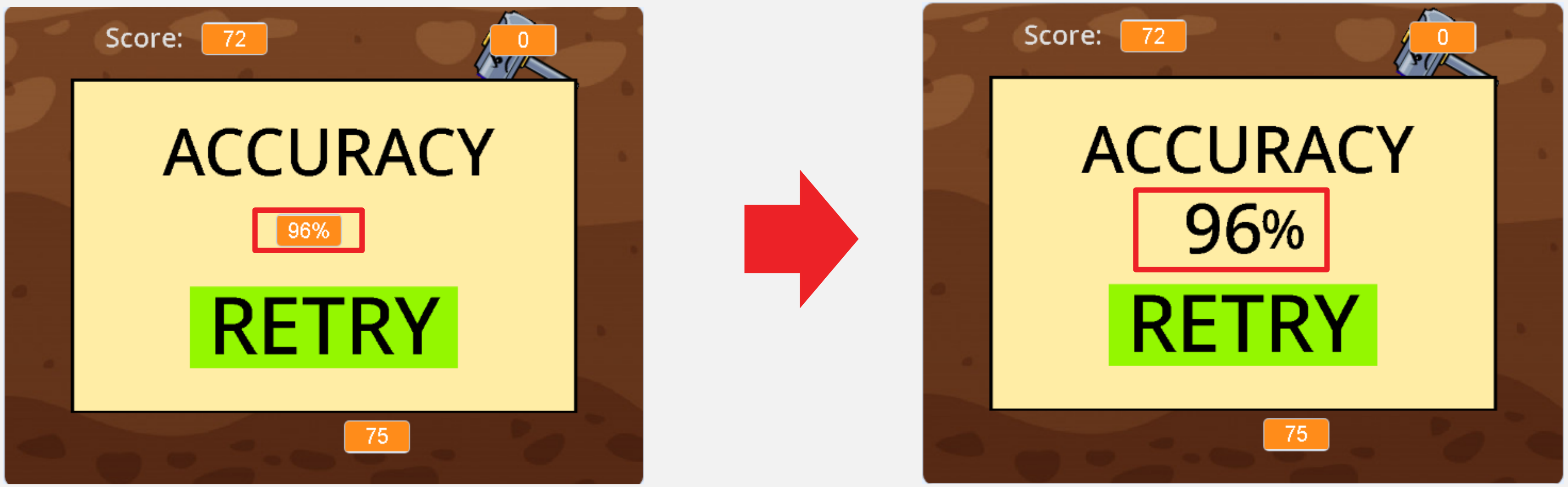
As we are going to re-run the game once we click "Retry", then we should delete all the clones once we received "You Win" or "You Lose", otherwise you will create too many moles for every time you retry the game, and it will get you into trouble of PC lagging.

Rerun your game and try to win the game



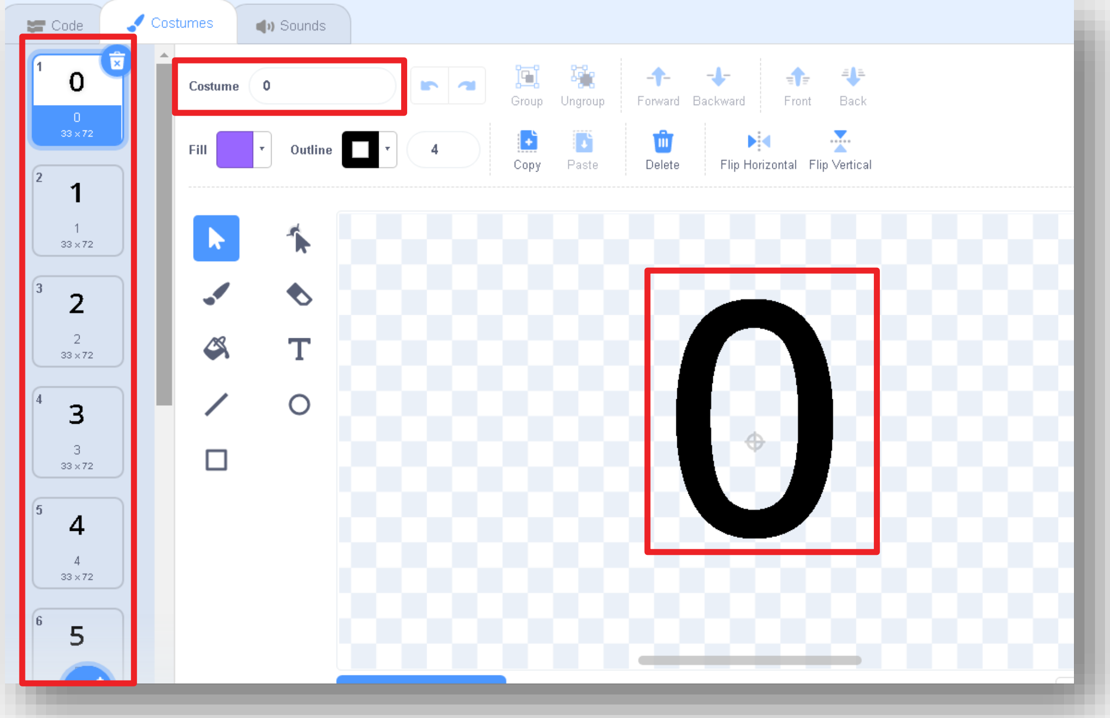
Once every "When Flag Click" has been changed to "When Receive Play", then you can rerun the game and try to win the game.

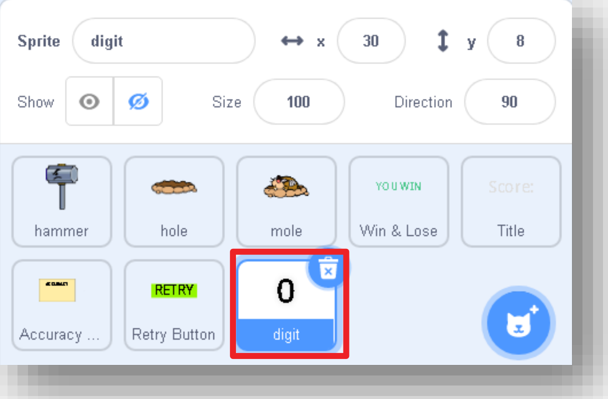
Use our own digit to show the variable



How do we use our customized digit for the accuracy or for the score?

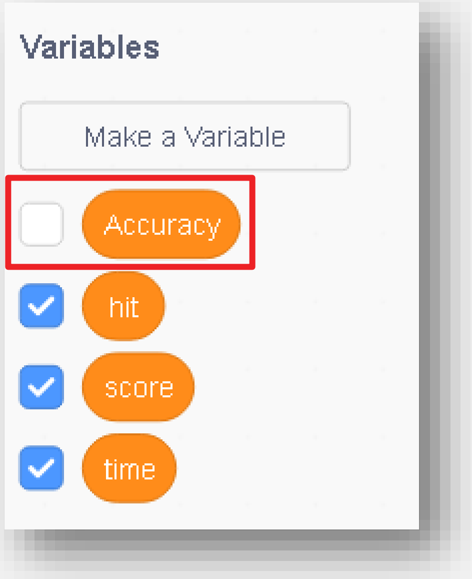
Add new Sprite - Digit

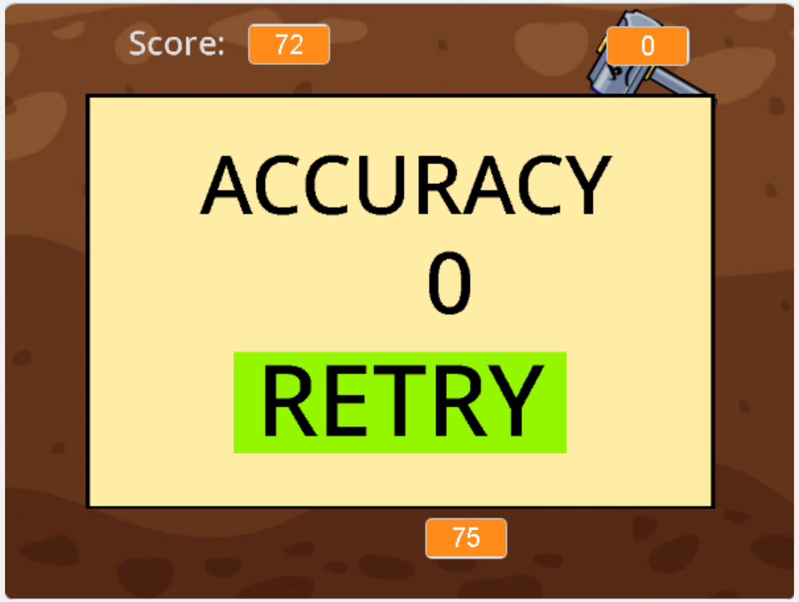




Create a new sprite with paint and add in the costume from 0-9 and a "%" symbol. Then rename the costume to its own name.

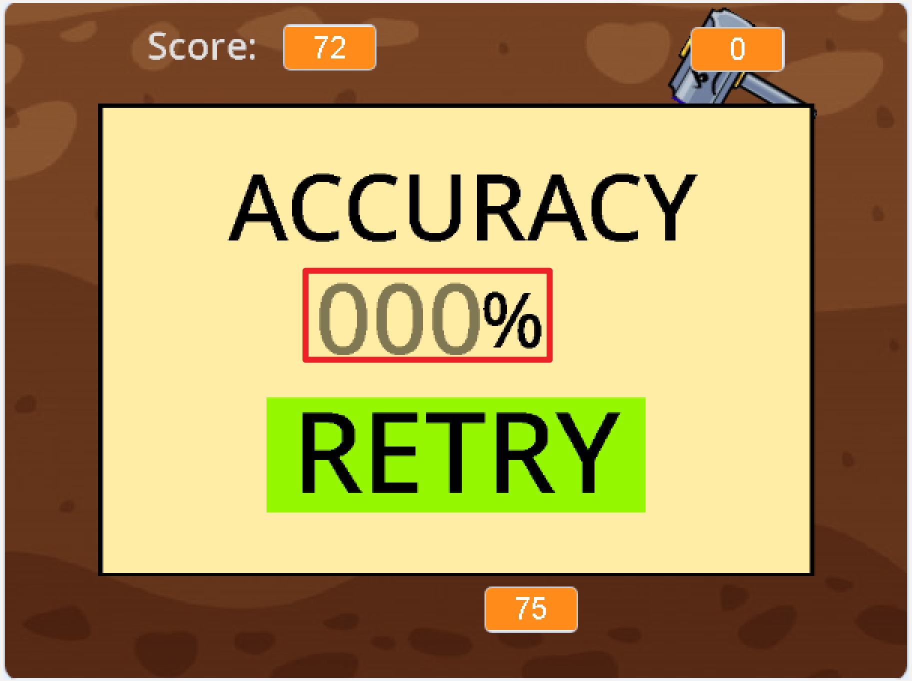
Use our own digit to show the variable





Uncheck the accuracy variable to hide it, then adjust the position of the digit slightly to the right side in the middle.

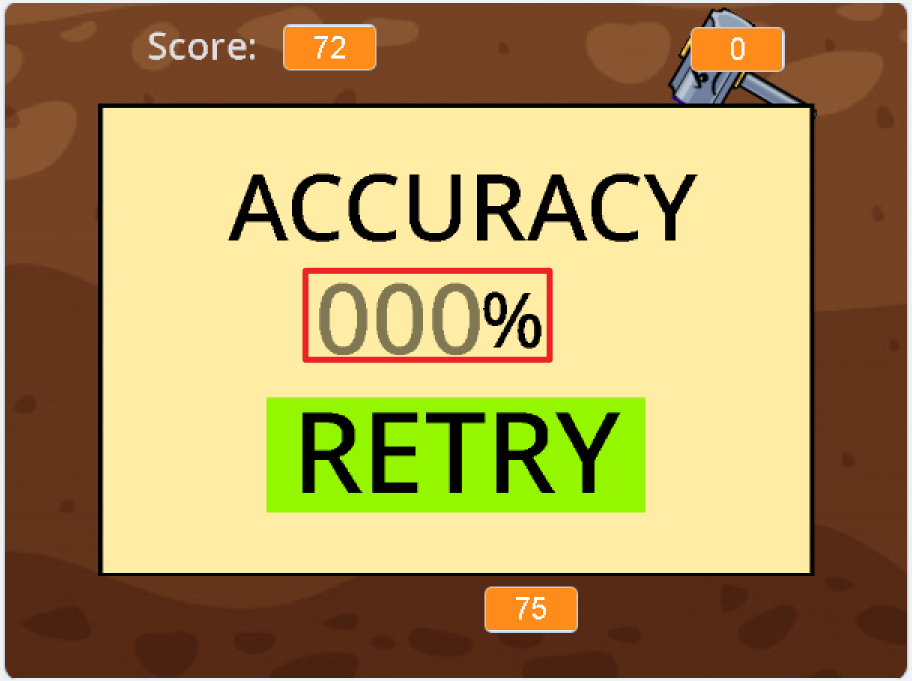
Use our own digit to show the variable



Our digit display should be like left picture, so it has four position (left to right):

1st digit = (-60,8)
2nd digit = (-30,8)
3rd digit = (0,8)
4th digit (%) = (30,8)

Use our own digit to show the variable



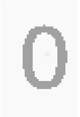
Now we want to know when we should display 2 digits, 3 digits or 4 digits.

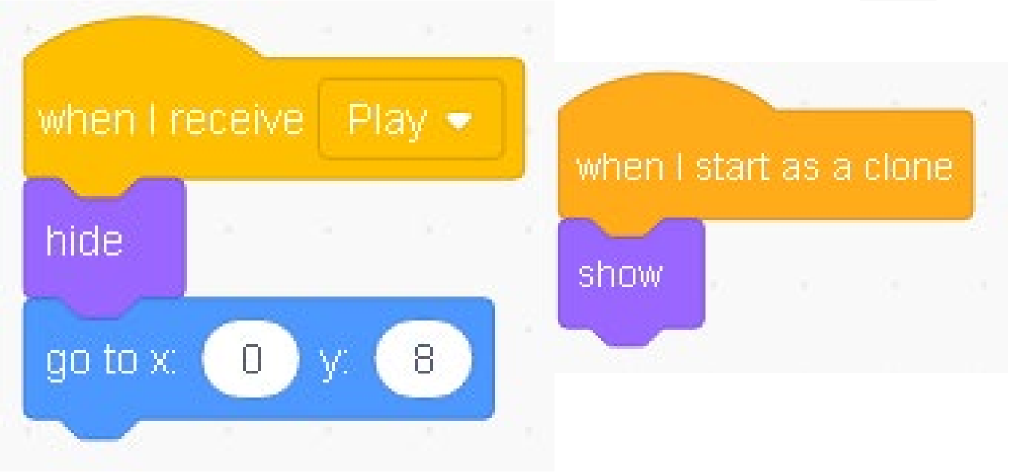
The possible value we will get in our accuracy is:

0% ~ 9% - 2 digits (“0” & “%”)
10% ~ 99% - 3 digits (“1” & “0” & “%”)
100% - 4 digits (“1” & “0” & “0” & “%”)

Script - Digit

Script:





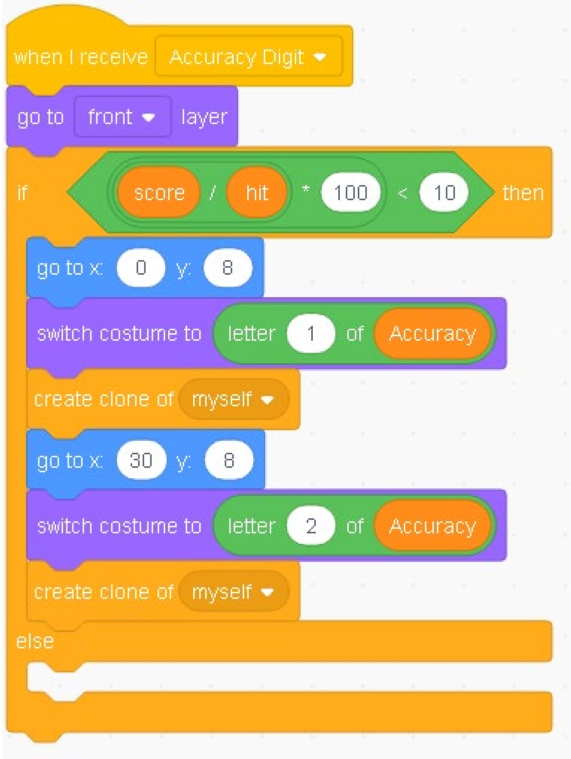
```
when I receive [Play ▾]
hide
go to x: (0) y: (8)

when I start as a clone
show
```

Of course digit will be hidden at the start (when received play).

And it will only show when the clone is created.

Script – Digit If-statement

Script:



```
when I receive [Accuracy Digit ▾]
go to [front ▾] layer
if <((score) / (hit)) * (100) < (10)> then
    go to x: (0) y: (8)
    switch costume to (letter (1) of (Accuracy))
    create clone of [myself ▾]
    go to x: (30) y: (8)
    switch costume to (letter (2) of (Accuracy))
    create clone of [myself ▾]
else
```

I will create a new broadcast called "Accuracy Digit", which will be broadcasted after the accuracy box is created.

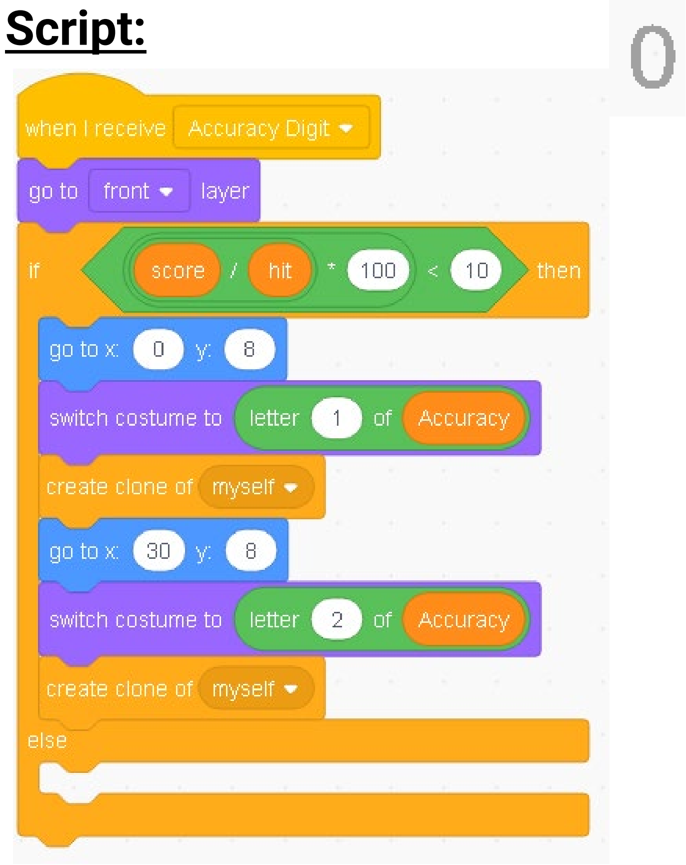
Then we need to check how many digit will be used for showing the accuracy.

In previous slide, we know that it will only have 3 possibilities:

2 digits when accuracy < 10%
3 digits when accuracy = 10% - 99%
4 digits when accuracy = 100%

Script – Digit If-statement

Script:



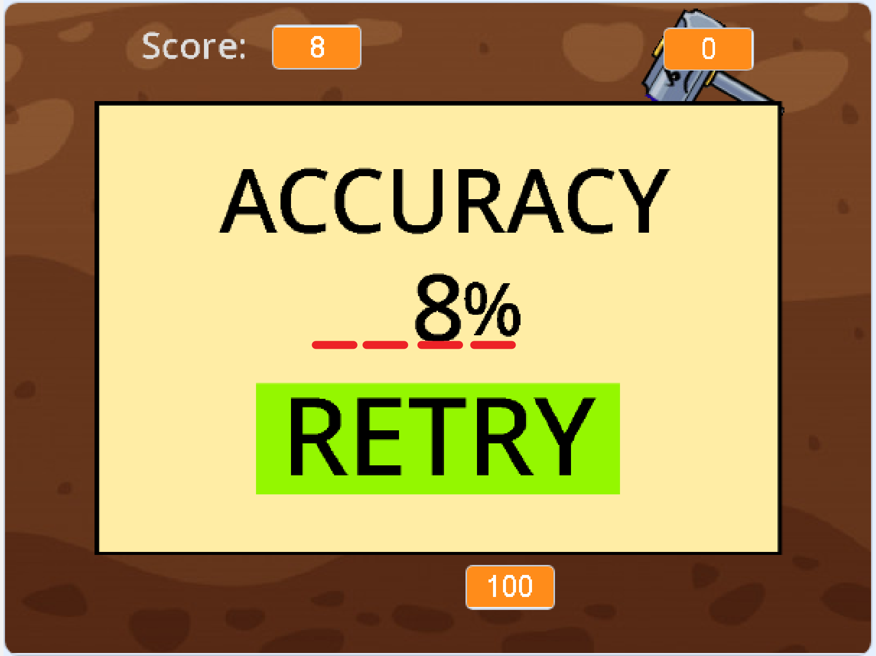
```
when I receive [Accuracy Digit ▾]
go to [front ▾] layer
if <(((score) / (hit)) * (100)) < (10)> then
    go to x: (0) y: (8)
    switch costume to (letter (1) of (Accuracy))
    create clone of [myself ▾]
    go to x: (30) y: (8)
    switch costume to (letter (2) of (Accuracy))
    create clone of [myself ▾]
else

end
```

Now we should give the script to do first checking, if accuracy is <10:

Then I will only show 2 digits at (0,8) and (30, 8).

Before I create 1st clone, I will switch my costume to the first letter of accuracy. Then switch to 2nd letter of my accuracy, then create clone.

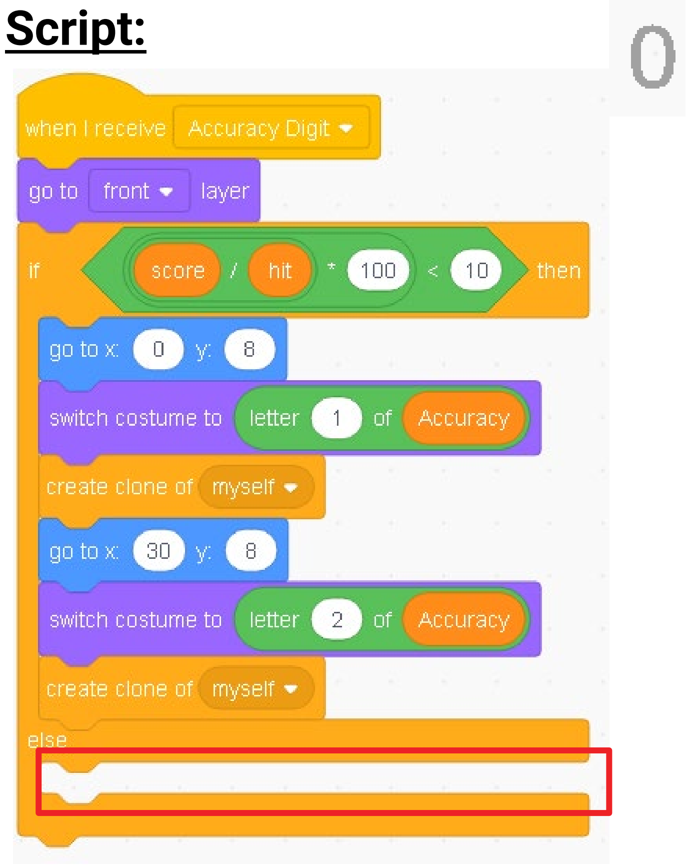
Script – Digit If-statement explained



So if the accuracy checking shows it is lower than 10% (e.g, 8%), it will only create 2 clones for showing first letter “8” and 2nd letter “%”.

Script – Digit If-statement

Script:



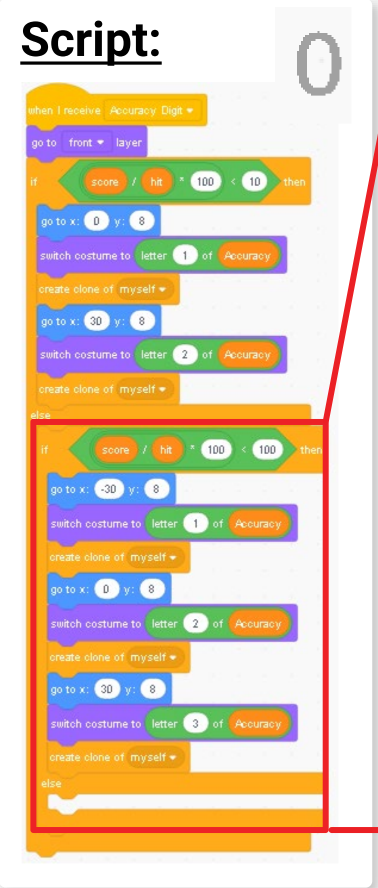
```
when I receive [Accuracy Digit ▾]
go to [front ▾] layer
if <(((score) / (hit)) * (100)) < (10)> then
    go to x: (0) y: (8)
    switch costume to (letter (1) of (Accuracy))
    create clone of [myself ▾]
    go to x: (30) y: (8)
    switch costume to (letter (2) of (Accuracy))
    create clone of [myself ▾]
else

end
```

If the score is more than 9%, which is 10% or more, it will skip and go to else to do the 2nd checking.

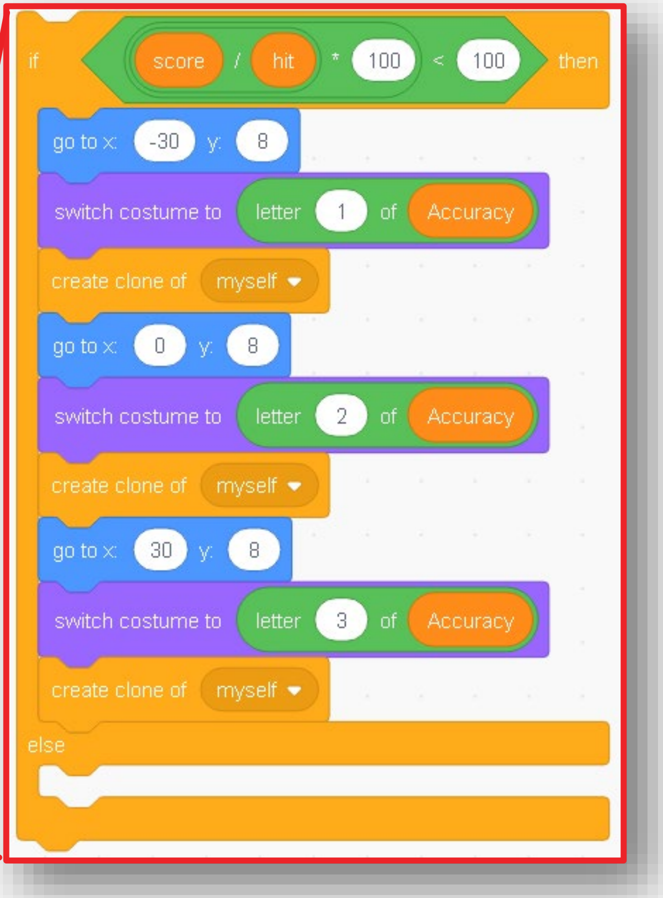
So we will insert a condition to check if the accuracy is less than 100% to the red black area.

Script – Digit If-statement

Script:



```
when I receive Accuracy Digit
go to front layer
if score / hit * 100 < 10 then
    go to x: 0 y: 8
    switch costume to letter 1 of Accuracy
    create clone of myself
    go to x: 30 y: 8
    switch costume to letter 2 of Accuracy
    create clone of myself
else
    if score / hit * 100 < 100 then
        go to x: -30 y: 8
        switch costume to letter 1 of Accuracy
        create clone of myself
        go to x: 0 y: 8
        switch costume to letter 2 of Accuracy
        create clone of myself
        go to x: 30 y: 8
        switch costume to letter 3 of Accuracy
        create clone of myself
    else
```



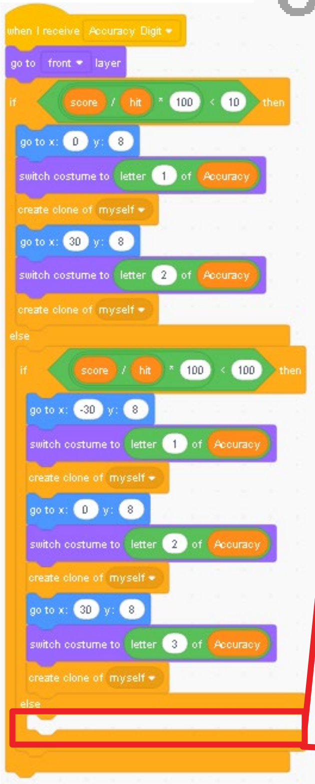
```
if score / hit * 100 < 100 then
    go to x: -30 y: 8
    switch costume to letter 1 of Accuracy
    create clone of myself
    go to x: 0 y: 8
    switch costume to letter 2 of Accuracy
    create clone of myself
    go to x: 30 y: 8
    switch costume to letter 3 of Accuracy
    create clone of myself
else
```

So I will add a condition to check if the accuracy is less than 100%.

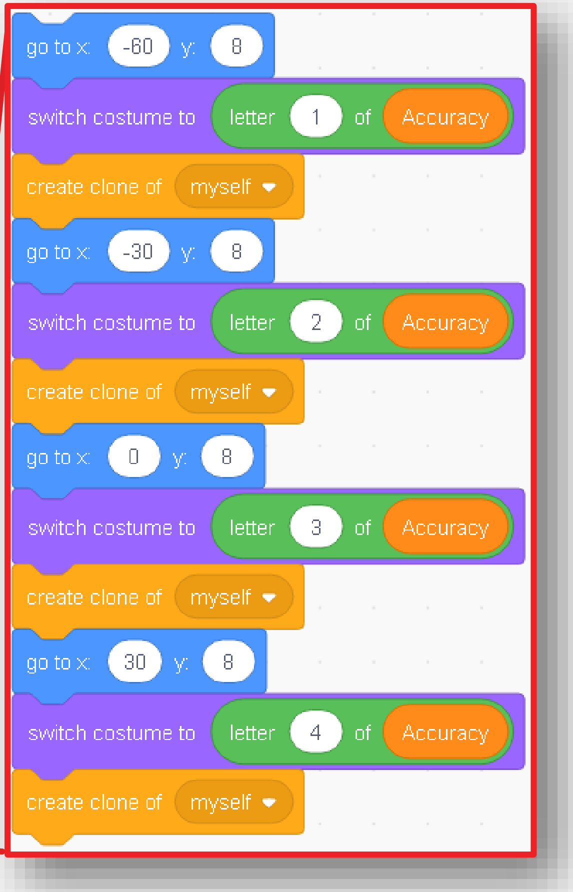
If yes, it should be fall under 10% - 99%, then we will create 3 clones for showing the accuracy with our digits.

Script – Digit If-statement

Script:



```
when I receive Accuracy Digit ▾
go to front ▾ layer
if ((score / hit) * 100) < 10 then
    go to x: 0 y: 8
    switch costume to (letter 1 of Accuracy)
    create clone of myself ▾
    go to x: 30 y: 8
    switch costume to (letter 2 of Accuracy)
    create clone of myself ▾
else
    if ((score / hit) * 100) < 100 then
        go to x: -30 y: 8
        switch costume to (letter 1 of Accuracy)
        create clone of myself ▾
        go to x: 0 y: 8
        switch costume to (letter 2 of Accuracy)
        create clone of myself ▾
        go to x: 30 y: 8
        switch costume to (letter 3 of Accuracy)
        create clone of myself ▾
    else
```

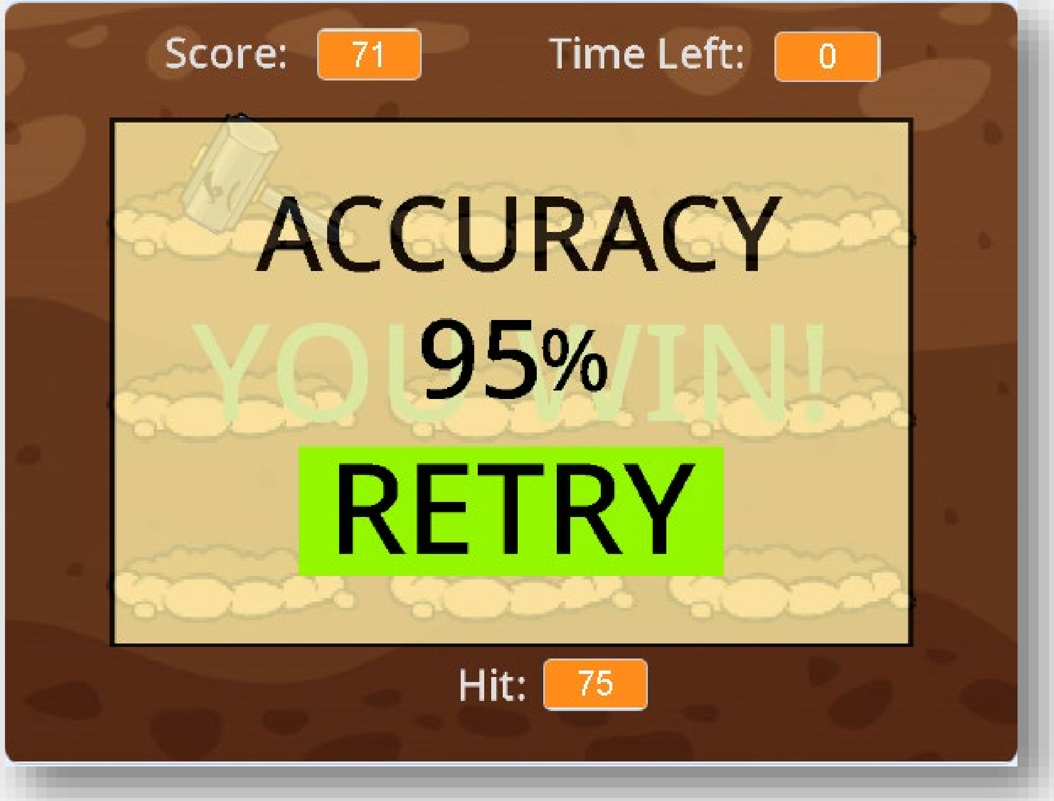


```
go to x: -60 y: 8
switch costume to (letter 1 of Accuracy)
create clone of myself ▾
go to x: -30 y: 8
switch costume to (letter 2 of Accuracy)
create clone of myself ▾
go to x: 0 y: 8
switch costume to (letter 3 of Accuracy)
create clone of myself ▾
go to x: 30 y: 8
switch costume to (letter 4 of Accuracy)
create clone of myself ▾
```

If the 2nd condition doesn't fulfil, then it must be 100%, and it will execute the command blocks in the else statement.

Therefore we will put in these commands to show 4 digits on the screen.

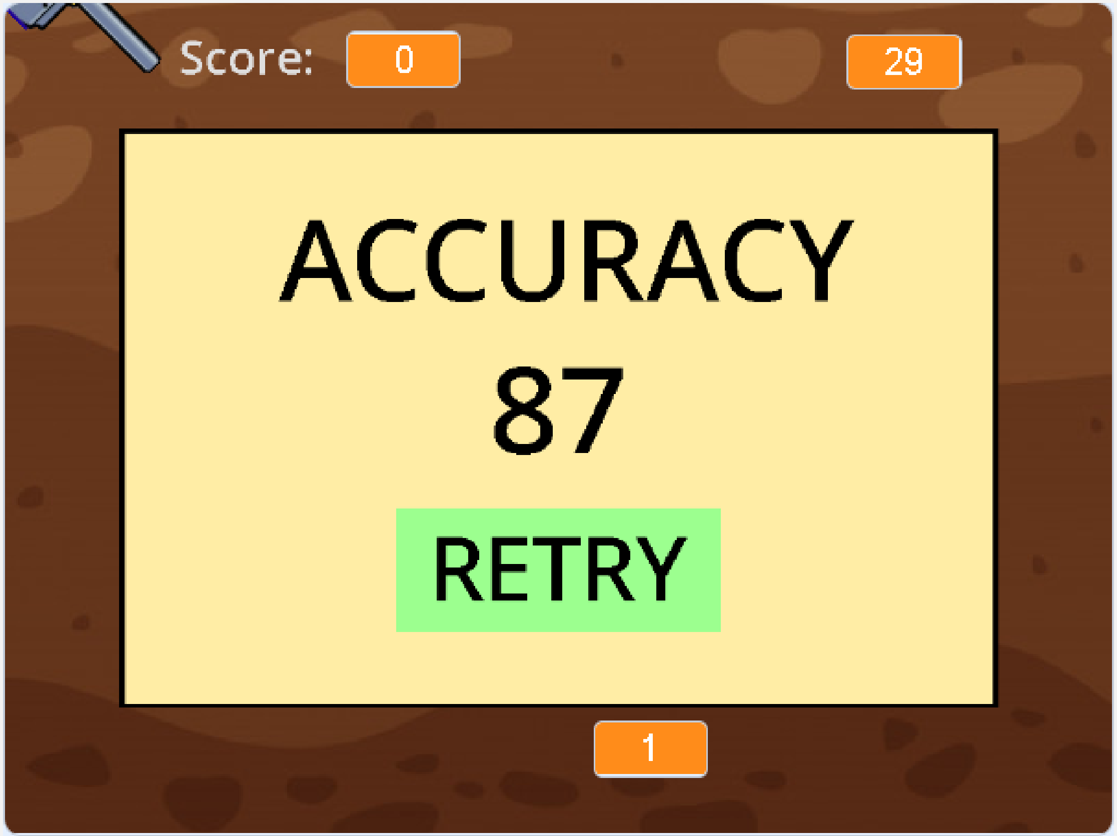
Play the Game



Now you should have a accuracy box which shows your accuracy with customized digits once the game ends.

ASSIGNMENT *for*

Lesson 2-3



L2-3 – Mission

Try to set your Accuracy to digit instead of just using “show variable”.

Can you show your Score at the end of the game too?

You can direct message your teacher and ask your question through Slack Robotene Community or arrange a One-to-One Consultation with your teacher.

Q&A Any Questions?

Thank you ;)