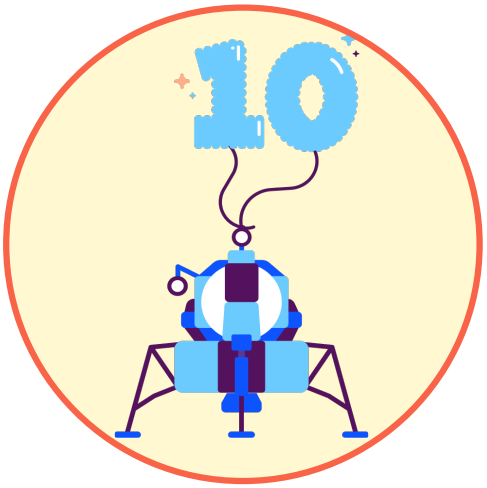
MOONHACK 2025



Puzzle Party

Create a jigsaw puzzle to celebrate Moonhack's 10th birthday!

INTRODUCTION

What you will need

HARDWARE

A computer capable of running Scratch 3

SOFTWARE

Scratch 3:
either online
rpf.io/scratchon
or offline
rpf.io/scratchoff

What you will learn

- How to create your **own blocks**
- How to use **abs block**
- Creating **variables** to control sprites

Project Links

Starter project

https://scratch.mit.edu/projects/1193332984/

Completed project

https://scratch.mit.edu/projects/1193333434/

Additional notes for educators

Read our blog that explores the creation of this project and the coding aspects behind it.

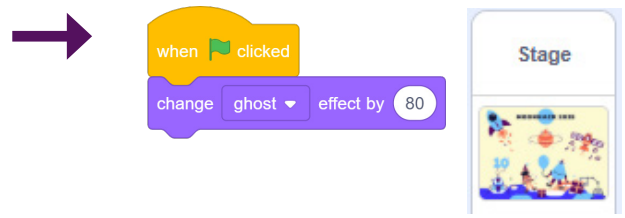
Moonhack Certificate

Finished the project? Download a certificate of completion here.

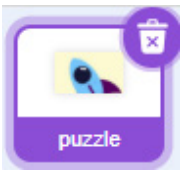
STEP 1 -CREATING PUZZLE PIECES

Start by opening the starter project - https://scratch.mit.edu/projects/1193332984/ Let's set the backdrop and then create our own block to create the puzzle pieces.

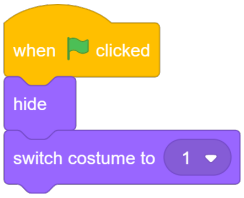
Select the backdrop. Add blocks so when the green flag is clicked the backdrop becomes transparent. This allows the user to see the image.



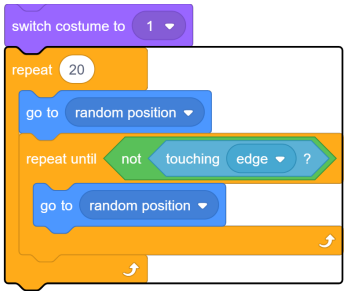
Select the puzzle sprite. If you look at its costumes you will see there are 20 pieces. Each piece is sitting in its finished location in the puzzle, not at the centre. This is important for later.



Start with the green flag event. From the looks menu add a hide block and a switch costume block. We don't want to see the original sprite, only the clones that we will create.

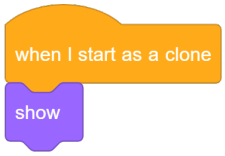


Add a repeat loop. Program the sprite to move to a random position until it is not touching the edge



After the repeat loop add a block to create a clone, change to the next costume, and for fun add in a sound that mimics the puzzle piece being put on the table.

Start a new algorithm using the control when I start as a clone. Add a show block.



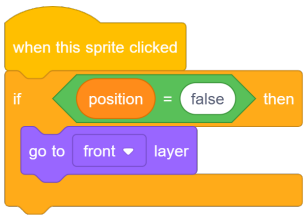
Test your code. Your backdrop should change to transparent, and 20 pieces of the puzzle should be placed in the middle of the screen.

STEP 2 -MOVING PUZZLE PIECES

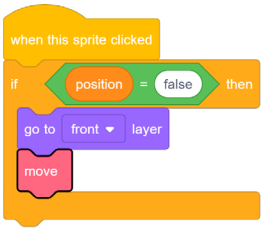
Users need to be able to move the pieces around the puzzle and have them 'click' into place when they reach the correct location. This will involve creating 2 of our own blocks.

Start with the event when this sprite is clicked. Create a new variable called position. **Make sure you select 'this sprite only' when creating the variable. The position needs to be unique to each clone.

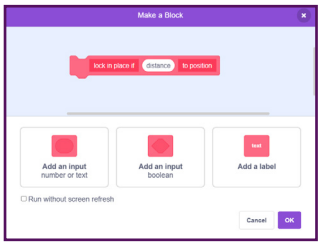
Create a branching statement that if the position is false the clone goes to the front layer. False indicates that it is not in the correct location and the selected piece will be positioned on top.

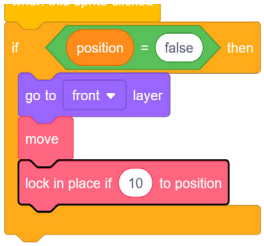


Make your own block and call it move. Add this to your algorithm. This block, when coded, will control how the pieces are moved on the screen.



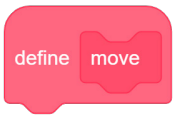
Make another block. Start with the text 'lock into place'. Add an input and call it distance. Then add a label called to position. This block will control the accurate location of the puzzle piece to lock it into place.



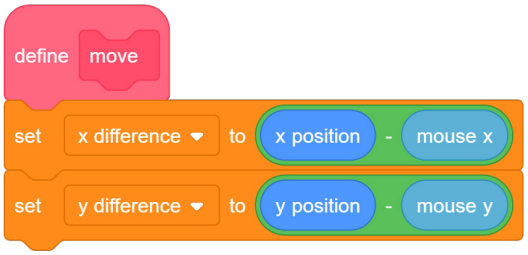


Go to full screen mode and test your project. We can't move the pieces yet. Time to define the move block. Each of the puzzle piece costumes are a sprite positioned on the screen, but its centre remains at coordinates 0, 0. When we control the sprite, we need to offset the x and y coordinates of the sprite by the relative position of the mouse. Let's code this!

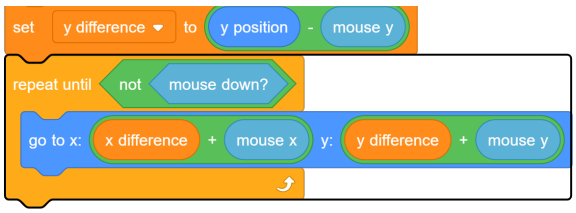
Go to your define move block. Create two new variables. One called x difference and one called y difference.



Add a set variable block for x difference. X difference will be the x position of the sprite (motion) minus the x position of the mouse (sensing). Create similar code for the y difference.

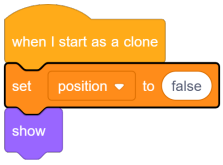


Now code how these variables are used. Add a repeat until loop with a condition of not mouse down. Inside the loop place a go to x where it is x difference (variable) + mouse x (sensing), and the same for the y coordinate.



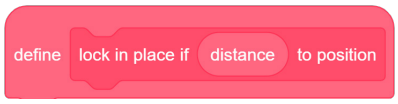
Go to full screen mode and test your project again. Did your pieces move? No - because for the sprite to move it's position has to be set to false. Let's add this in.

On the algorithm, when I start as a clone, add in set position to false.

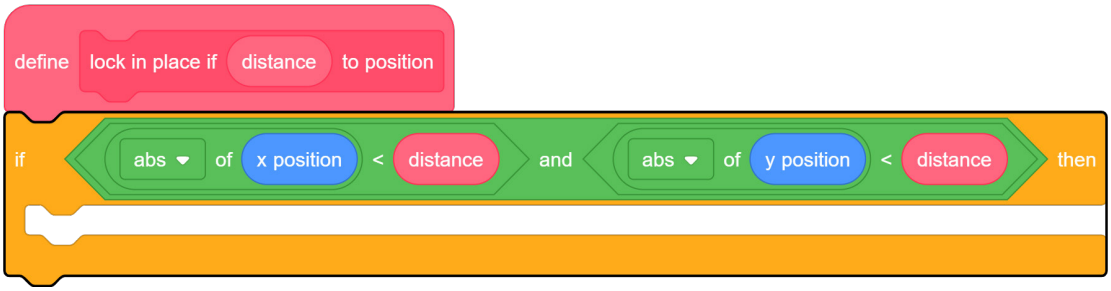


Go to full screen mode and test your project again. Now you should be able to use your mouse to move the puzzle pieces around the screen. Next, we need to define the lock into place block. This algorithm will use the absolute value block, which describes the distance of a number from 0. Our sprite is at coordinates 0, 0 which makes the calculations easy.

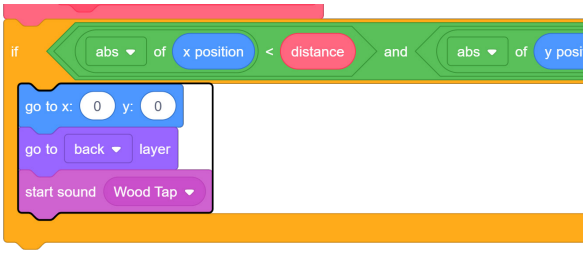
Let's define the block that will lock puzzle pieces in to place when they are moved to the correct location in the puzzle.



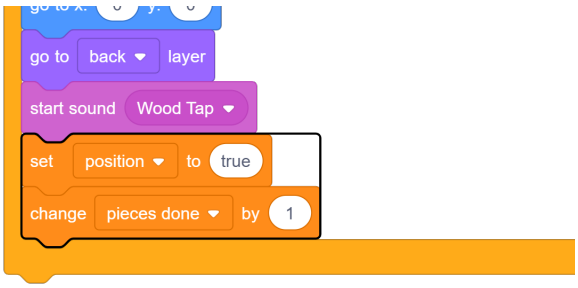
Start with an if then block. Inside it nest operators to create the condition of if the absolute value x position is less than distance (we made it 10) and the absolute value of y position is less than distance. **To get the 'distance' reporter, drag it from the define block.



then we want to lock the piece into place. Add blocks to go to 0, 0. This places the clone in the exact position. Send it to the back layer and add a sound so the user knows that it has been placed.



Add a block to set the position to true. This will 'lock' it into place. Create a new variable called pieces done. We need to track how many pieces are in 'place' to know when to finish the puzzle. Add a block to change pieces done by 1.

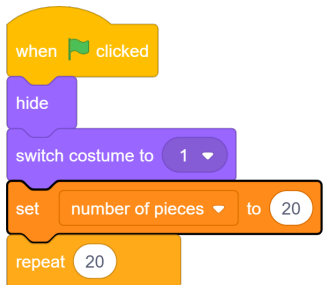


Go to full screen mode and test your project. If the pieces are in the correct location you should hear your sound and 'see' them lock into place.

STEP 3 - FINISHING THE PUZZLE

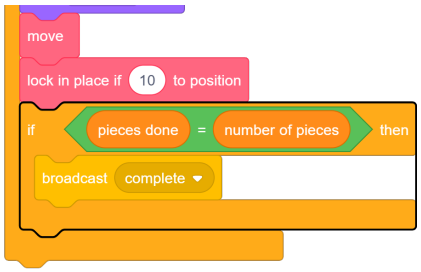
To finish the puzzle, we need to set 2 variables to track how many pieces there are in the puzzle and how many pieces are in place.

Create a new variable called number of pieces. In the algorithm, you already have that starts with the green flag event, add a block to set the number of pieces to 20.

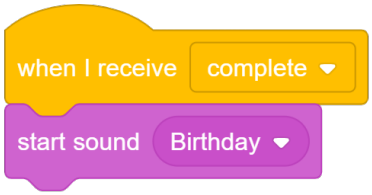


Add a variable block to set the pieces done to 0.

Go to the algorithm that starts with the event when this sprite is clicked. Let's add a condition that if the variable pieces done equals the variable number of pieces it will send a new broadcast of complete.



Start a new event with when I receive complete. Add the birthday sound. Now when 20 pieces are in place it will play Happy Birthday.



Your puzzle is now complete! Remember to go to full screen to run your project. Happy birthday Moonhack!

Challenges:

Gamify

Change the puzzle to be competitive. Perhaps this could be by adding a timer that records the quickest score. Or maybe the puzzle needs to be completed within a certain amount of time or the player loses.

Increase the difficulty

At the moment each piece of the puzzle is as it should be on the board. What if the puzzle pieces are rotated in random directions when placed on the board? The player would need to rotate the pieces before finding their home.

Create your own puzzle

The image used for the puzzle was uploaded to Pine Tools and split into pieces. It was then added to Canva to create an image for each puzzle piece. Can you create your own jigsaw puzzle pieces to create your own design? It's good to start with a 4 or 6 piece puzzle first to see how it all works and then work up to a puzzle with more pieces.

Congratulations! You're a Moonhack changemaker. You can try one of our other projects or try one of the challenges above.

Don't forget to talk to an adult about registering your participation and downloading your certificate at moonhack.com