Nucleobases Matching Game

An example version of this code can be viewed and played here:
https://scratch.mit.edu/projects/438699096/editor/

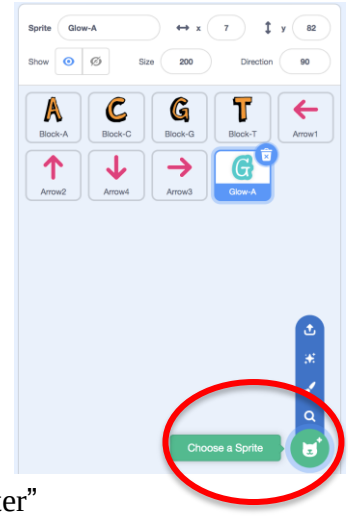
- Go to Scratch.mit.edi and Create a New Project

Create Sprites

This program uses nine sprites. Eight of these sprites do not require coding; they are displayed to establish the rules of the game.

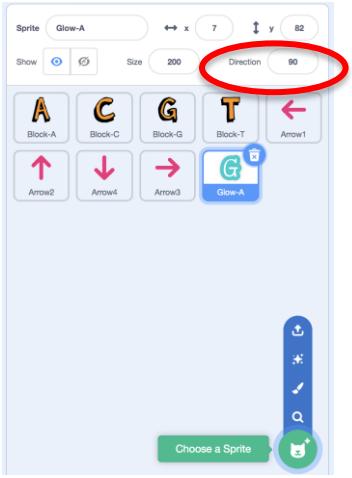
To create the display sprites (these are the sprites that we will not be programming):

- Press the Choose Sprite button at the bottom of the Sprites panel.



- In the search bar, type “letter”
- You will see an entire alphabet of Block letters. Make one sprite each for Block-A, Block-T, Block-C, and Block-G
- In the Choose Sprite search bar, type “arrow”
- Create four arrow sprites. To change the direction in which the arrow is pointing, you can rotate the sprites by selecting the sprite that you’d like to rotate and changing the value (in degrees) in the Direction field.

- You can also adjust the size of the sprites by changing the value in the Size field (in our example, the sprites are set to size 200).



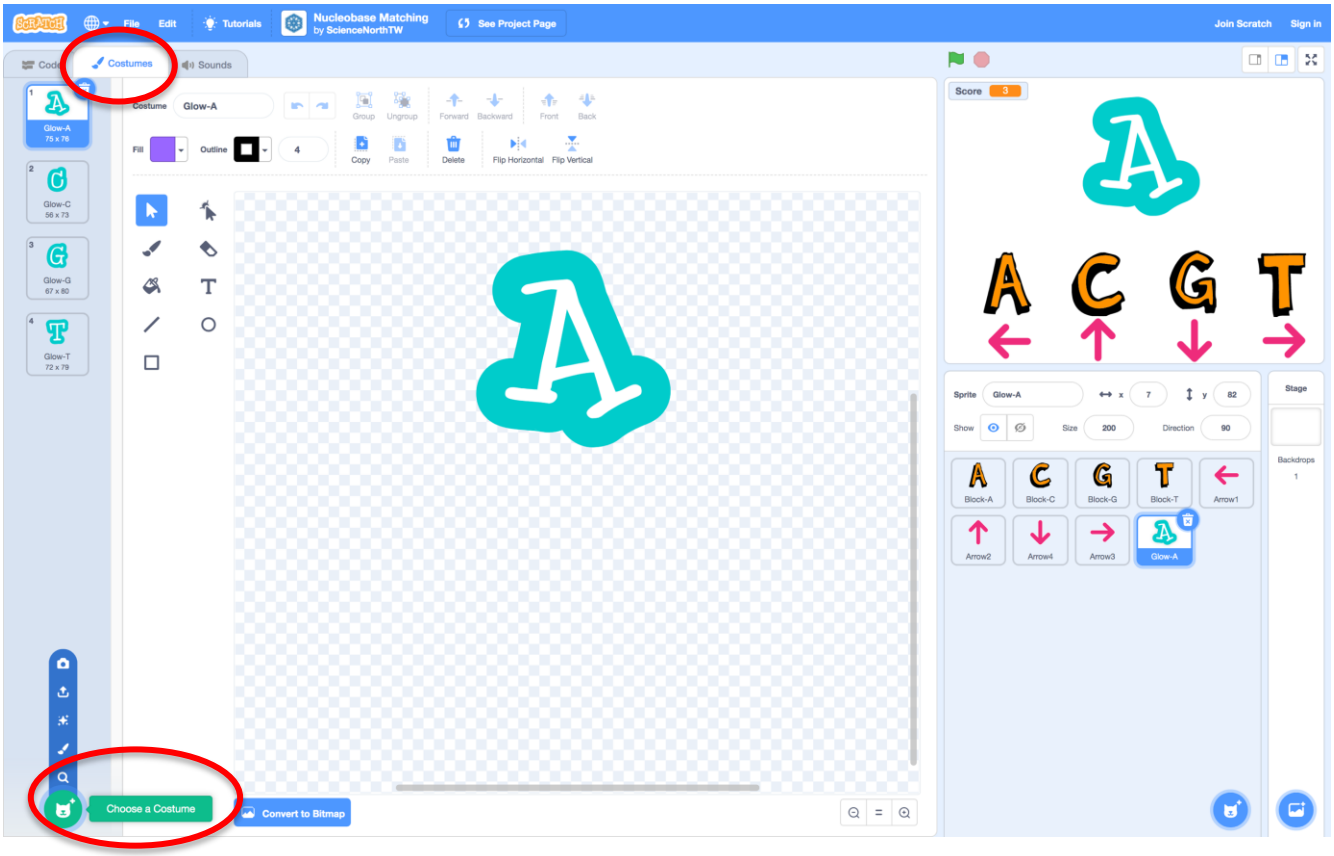
- Place the sprites manually on the canvas by dragging them into position.

<u>Create your "Randomizer" Sprite</u> (this is the sprite that we will be programming to generate a random nucleobase on-screen)

- Press the Choose Sprite button at the bottom of the Sprites panel.
- In the search bar, type "letter"
- Select Glow-A

This sprite will use a Scratch function known as Costumes, which will allow the appearance of the sprite to change according to different conditions.

- Go to the Costumes Tab. You'll notice that Glow-A is the default costume for this sprite. We need to add three more costumes (C, G, and T).
- Click the Choose a Costume button at the bottom of the Costumes panel to add the remaining costumes.

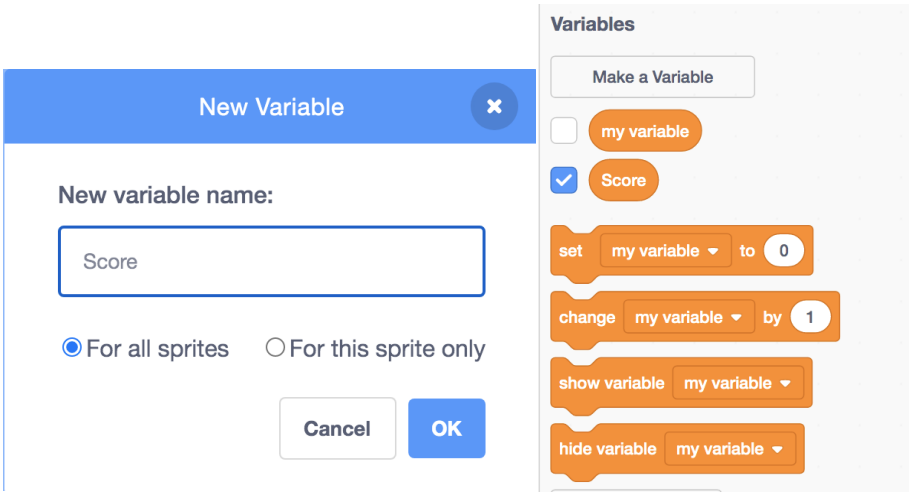


Create a Score Counter:

To create a score counter, we will need to create a variable to represent the score value.

To create a variable:

- Go to Code Tab
- Go to Variables (dark orange) menu
- Choose the Make a Variable button
- Name the Variable “Score” (without quotation marks)
- In this cases, it does not matter whether you choose “for all sprites” or “for this sprite only” as we are only programming for one sprite in this game.
- Once your variable is made, make sure that box next to the Score variable is selected



Create a Randomizer

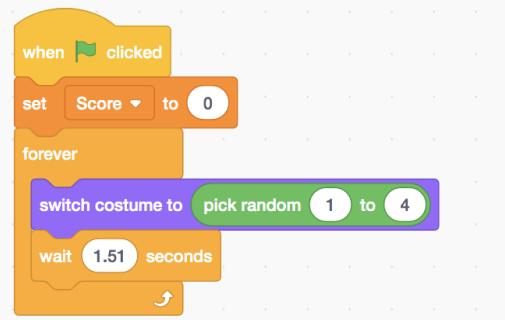
This program will set the score counter to zero at the start of the game and will show a random costume for our Randomizer sprite on-screen for a set amount of time before moving on to a new costume.

- Go to Events menu (yellow) and choose the **When the green flag is clicked** block and drag it into the coding area. This is the block that we will use to start all of our programs for this game (so that they all start when the game starts)
- Go to Variables menu (dark orange) and choose the **Set my variable to 0** block. Connect it directly below the green flag block. Using the dropdown list on this block, change "my variable" to "Score".

Now we need to create the randomizer that will loop forever (this game doesn't have a set endpoint).

- Go to Control menu (light orange). Choose the **forever** loop block and connect it directly below the **Set Score to 0** block. Anything code that we place inside this block will loop forever.
- Go to the Looks (purple) menu. Choose the **switch costume to** block and place it inside the loop.
- Go to the Operators (green) menu. Choose the **pick random 1 to 10** block and drag it onto the black spot on the **switch costume to** block. The blocks should combine to create a purple and green block that reads **switch costume to pick random 1 to 10**.
- Since we have four costumes, change the values in the **pick random 1 to 10** block to instead read **pick random 1 to 4**.
- Go to Control (light orange) menu. Choose the **wait 1 second** block. Place this block directly below the **switch costume to pick random 1 to 10** block but still *inside* the forever loop. This block controls how long a letter will remain onscreen (in seconds)

before moving to a new randomized. One second may be too fast. This can be adjusted to personal taste to be faster or slower.



Program the Game

Now all that's left is to create programs that will recognize when a player presses the correct key when a specific letter costume is being displayed. We will have to create one program for each nucleobase (so four total).

- First we'll place a **when green flag is clicked** block to start our program.
- In the Control menu (light orange), choose a **forever** loop block and connect it directly to the start block. We will be building the next steps inside of this forever loop block.
- To create the first conditional statement, which executes the program if costume A is displayed:
    - In the Control menu (light orange) choose the **if [blank] then** block and nest it inside the forever loop.
    - In the Operators menu (green) chose the **[blank] = 50** block and drag it on top of the blank hexagon on the **if [blank] then** block.
    - In the Looks (purple) menu, choose the **costume number** block and drag it on top of the blank oval on the green **[blank] = 50** block. The if statement should now read **if costume number = 50 then**.
    - The costume numbers were automatically assigned in the Costumes Tab on Scratch as they were created. The number for costume A (if it was the first costume that you created) is 1. Edit **50** on the green block so that it is now **1**. (If your costume number for A is not 1, just make sure that the costume number in your program matches the costume number for A in your Costumes tab).

- To create the nested condition for when the player presses the correct key
    - In the Control menu (light orange), choose another **if [blank] then** block and nest it into the existing conditional block.

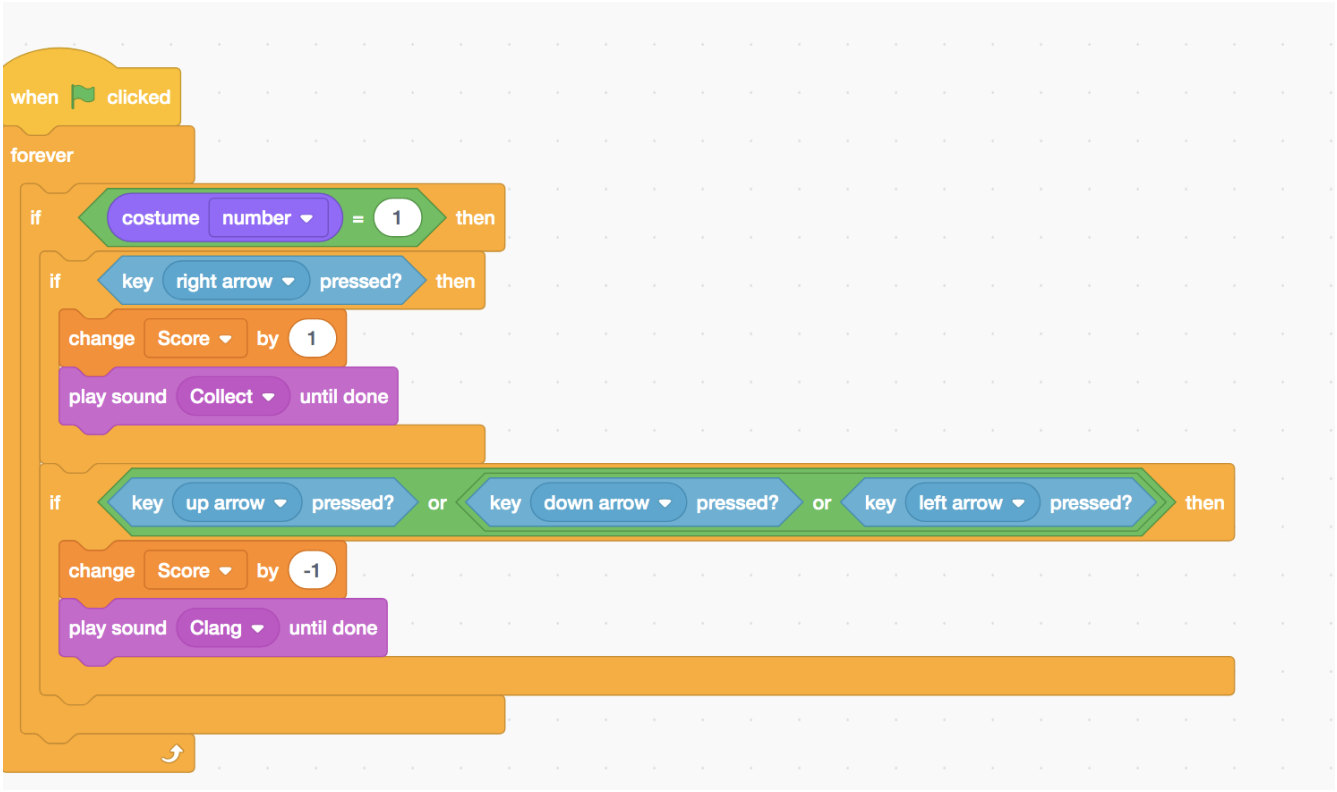
  - In the Sensing menu (light blue), choose the **key space pressed?** block and drag it on top of the blank hexagon on the **if [blank] then** block. The correct answer for when A is on screen is to press the right arrow (for T), so use the dropdown list on this block to select **right arrow**.
- Now we need to code what happens when the correct key is pressed
  - In the Variables menu (dark orange), choose the **change my variable by 1** block and place it inside the nested condition. Use the dropdown list on the block to select **Score**. This will ensure that the Score counter goes up by one point.
  - Optional: for flavour, you may choose to add a sound effect as feedback for getting the right answer. To do this, go to the Sound menu (pink) and choose the **play sound Clang until done** and place it directly below the **change Score by 1** block *inside* the same conditional statement. Use the dropdown list to choose the sound that you prefer.

- To create the nested condition for when the player presses the incorrect keys
  - In the Control menu (light orange), choose another **if [blank] then** block and nest it into the first conditional block (**if costume number = 1 then**).
  - In the Operators tab (green), choose the **[blank] or [blank]** block and drag it on top of the blank hexagon on the **if [blank] then** block *and then* grab a *second* **[blank] or [blank]** block and drag it onto the second blank space of the first **[blank] or [blank]** block. It will look like this:

  - Nesting these blocks allows for us to create the same scoring program for if the player presses any of the three incorrect keys.
  - In the Sensing menu (light blue), drag one **key space pressed?** block into each of the spaces on the green **[blank] or [blank] or [blank]** block. Use the dropdown list to change each of these so that they represent each of the incorrect answer keys (up arrow, down arrow, and left arrow).
- Now we need to code what happens when any of these incorrect keys is pressed
  - In the Variables menu (dark orange), choose the **change my variable by 1** block and place it inside the nested condition. Change the value to -1. Use the dropdown list on the block to select **Score**. This will ensure that the Score counter goes down by one point.
  - Optional: for flavour, you may choose to add a sound effect as feedback for getting the right answer. To do this, go to the Sound menu (pink) and choose the **play sound Clang until done** and place it directly below the **change Score by -**

1 block *inside* the same conditional statement. Use the dropdown list to choose the sound that you prefer.

The completed program should look like this:

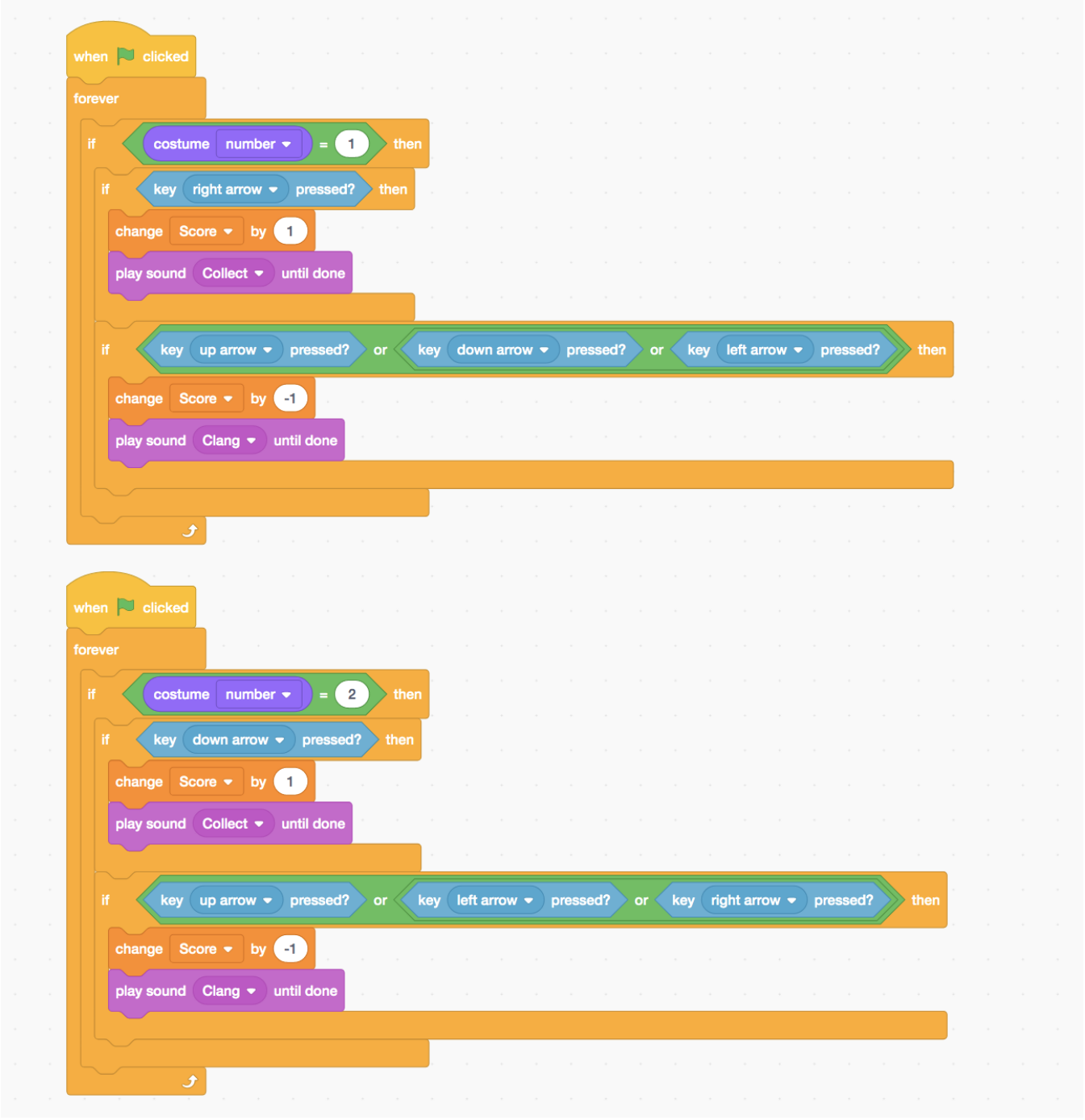


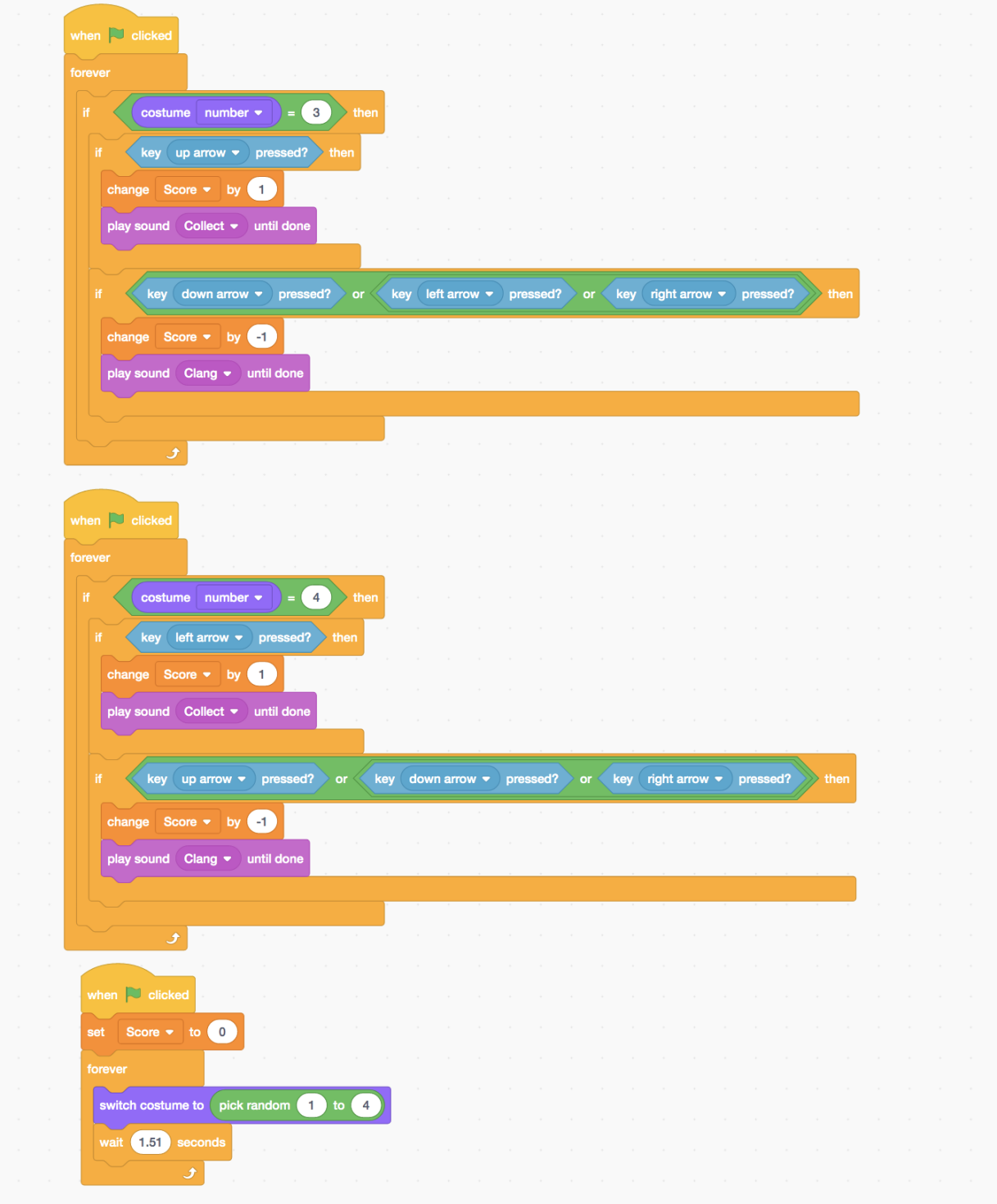
Create a program for each costume

Now we'll have to create the same program again for the remaining letter costumes (T, G, and C). The programs are all identical except for the costume number value, and which keys are attributed to correct or incorrect answers.

Hint: Since we are reusing large pieces of our program to code the remaining costumes, you can duplicate existing code (Right Click > Duplicate) rather than rebuilding the code three more times. Then all you will have to do is adjust the code parameters in each copy of the code.

The final program will look something like this:


when clicked
forever
if costume number = 1 then
if key right arrow pressed? then
change Score by 1
play sound Collect until done
if key up arrow pressed? or key down arrow pressed? or key left arrow pressed? then
change Score by -1
play sound Clang until done
when clicked
forever
if costume number = 2 then
if key down arrow pressed? then
change Score by 1
play sound Collect until done
if key up arrow pressed? or key left arrow pressed? or key right arrow pressed? then
change Score by -1
play sound Clang until done



Encourage students to playtest their game and to make changes, such as adjusting the challenge of the game by changing the value in the wait block or the score values for different conditions, or adjusting the aesthetics of the game by changing the sounds played or the look of the game.