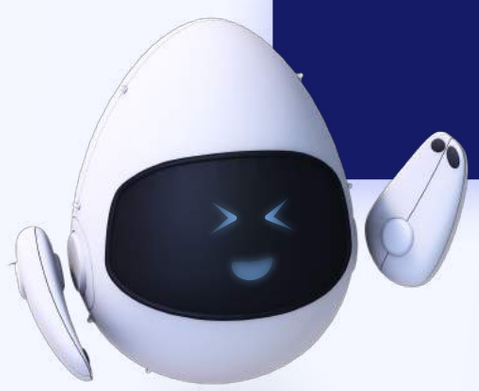
Guide For

Hatch Kid’s Blocks-Based Beginner Curriculum

Project 5:

3D Snake Game



Project 6:

3D Snake Game

Objective

In this guide we are going to learn to build the classic snake game in 3D on Hatch. You will learn to code to control the complete motion of an object in 3D space using your keyboard, learn to add features like score in your game and explore how to detect collisions between different objects.

Concepts covered:

Motion using coordinate values

Explore more about variables and Arrays and learn about boolean variables

Learn about Conditional statements

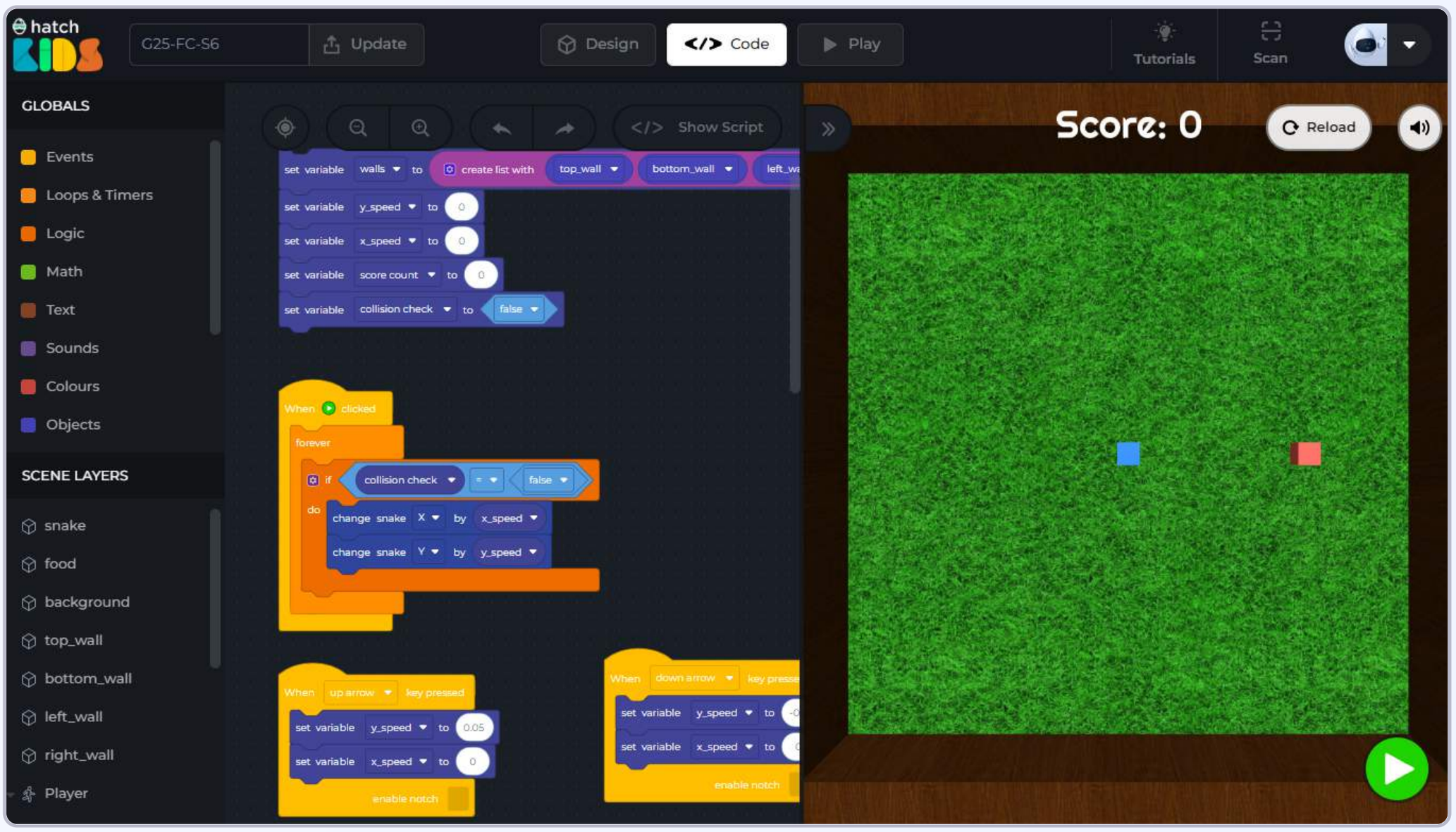
Learn to detect collision between different objects

Final Output Link:

https://kids.hatchxr.com/@XR4schools/G25-FC-S6

Student Template Link:

https://kids.hatchxr.com/@XR4schools/G25-FC-S6-template

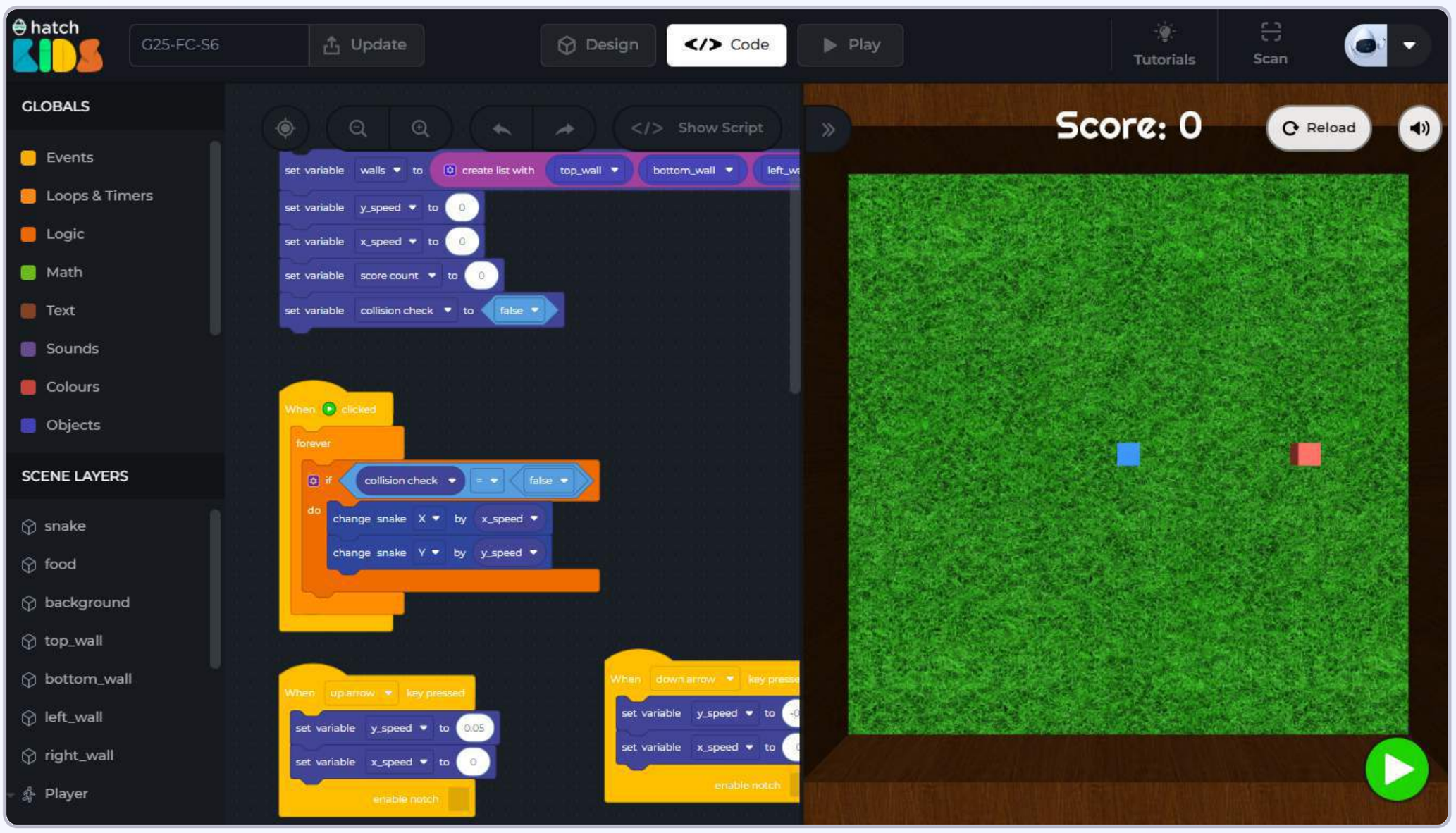


How it works?

Let’s first understand what we are going to be building in this session.

Open the completed project link: https://kids.hatchxr.com/@XR4schools/G25-FC-S6

You will a screen as shown below:



Click on the **“Green Play Button”** to run the code. Initially you will notice nothing happens, but then, as you press different keyboard buttons you will see:

1. Pressing the up / down / left / right arrow keys makes the blue box move in that direction
2. When the blue box hits any of the brown walls, the text says “game over” and the box stops moving
3. When the blue box hits the red box the red box moves to a new location
4. When the blue box hits the red box the score increases

We ar going to use all the concepts you learnt in the previous projects -- variables, lists, loops, keyboard events -- and learn some more new concepts, and build this game.

Let’s get started.

Objective No. 1: Understanding the template

Step 1: We will start by opening the template link of the project mentioned here.

Student Template Link: https://kids.hatchxr.com/@XR4schools/G25-FC-S6-template

Step 2: The link above will open up an empty project with no code in the hatch workspace, that looks as shown here.

You will also notice that there are brown colored barriers on the top, right, bottom and left the game space. They are named “top_wall”, “bottom_wall”, “left_wall”, “right_wall”. When same hits anyone of these walls, we will write the code to stop the game.

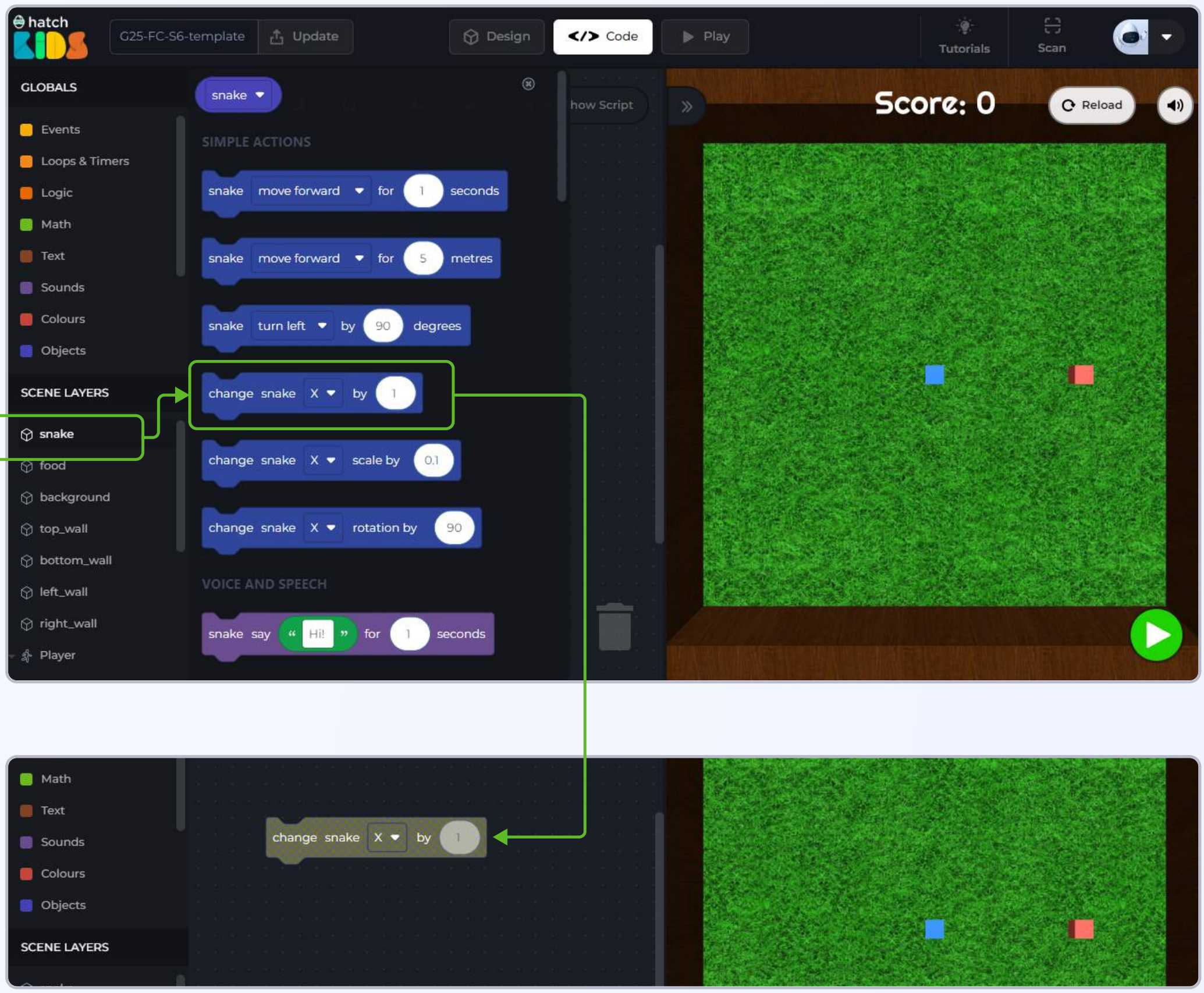
Also at the top of the game there is a text that says “Score: 0”. That is also an object by the name “Scoreboard” in the left panel. We are going to use it to display score of the game.

Objective No. 2: Move the snake box using keyboard buttons

Let’s start the game by moving the snake in the up down left and right direction by pressing the up down left and right arrow keys.

As we learnt in the previous projects, we can make the snake object move left/right by changing its x-position, And we can make the snake move up and down by changing its y-position.

Step 1: Click on the word **“snake”** in the left panel, and drag out the block that says **“change snake x by 1”**

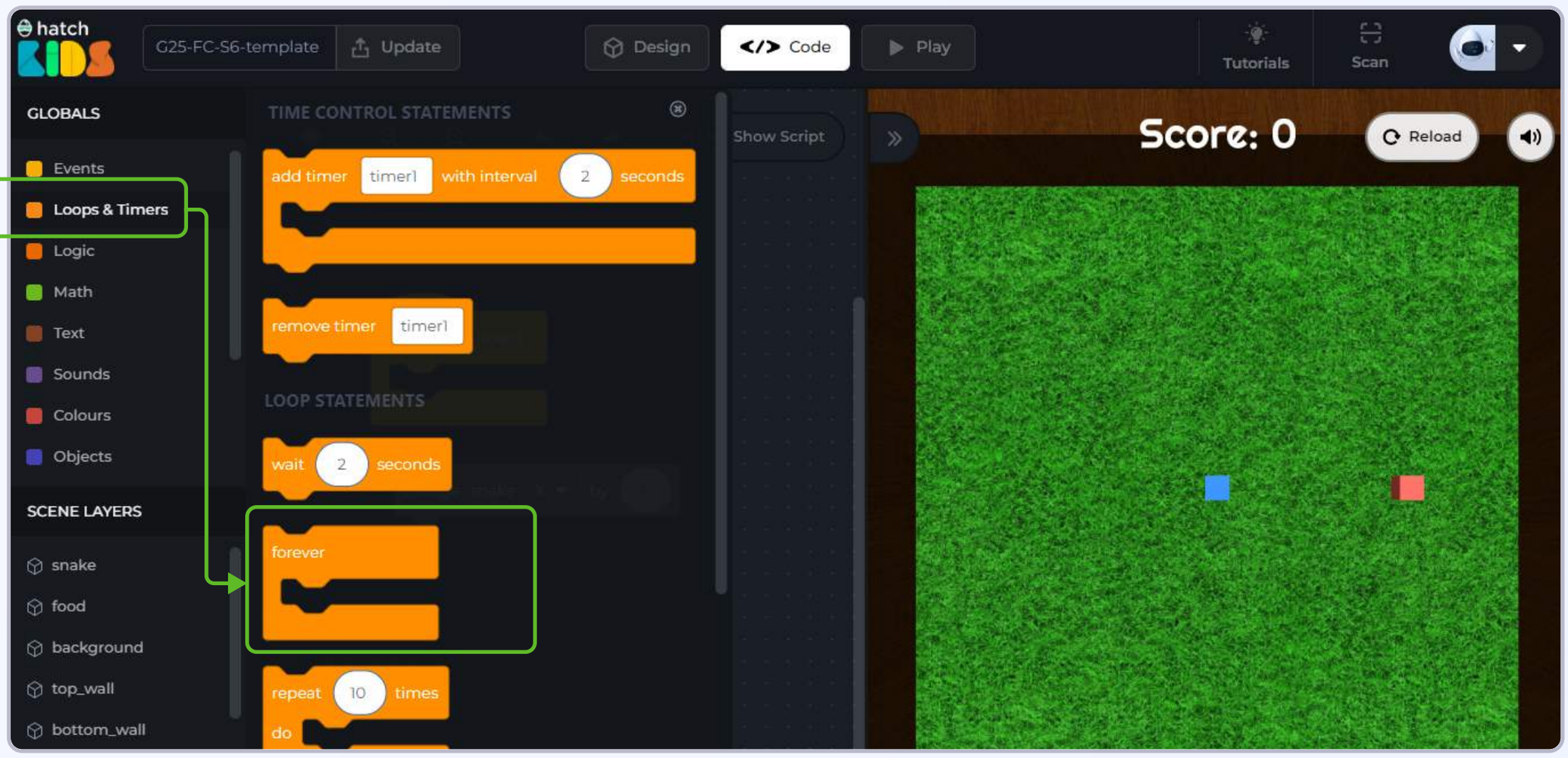


Step 2: We need an event block to define when should the position of the snake box change. Click on the **“events”** section in the left panel, and drag out the **“when green play button clicked” block**

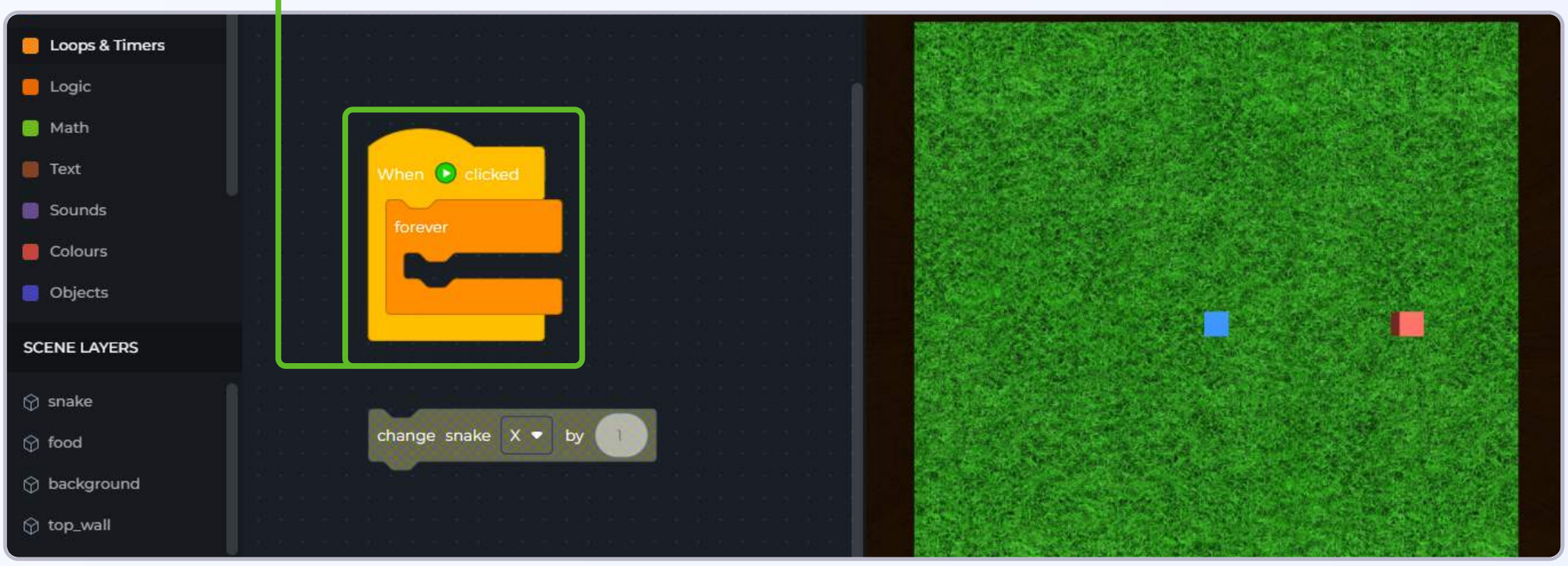
In the final output, you must have seen that the snake box keeps on moving forever, and we change the direction by pressing on different arrow keys.

In order to keep the snake object moving forever, we will need to use the “forever” block.

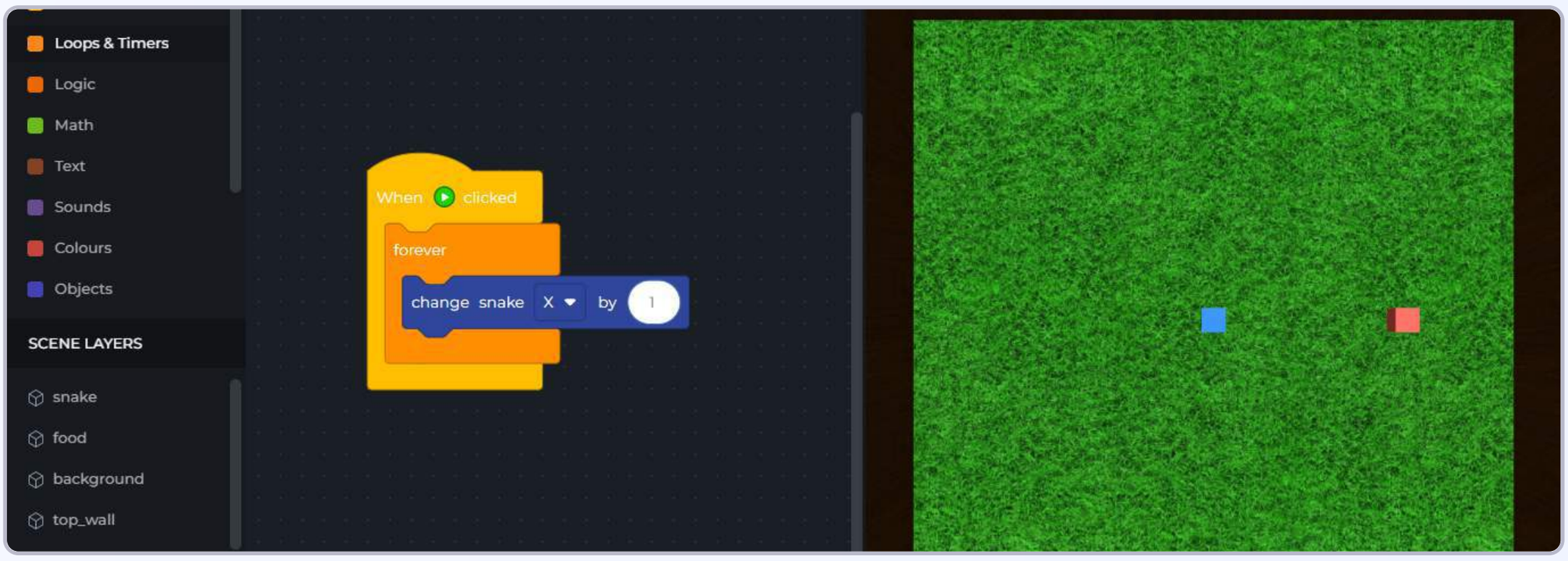
Step 3: Click on the **“loops & timers”** section in the left panel of the workspace, and drag out the block that says “**forever”**



Drag the “forever” block in the workspace, and attach it inside the “when green play button clicked” block



Step 4: Place the **“change snake x by 1”** block inside the **“forever”** block

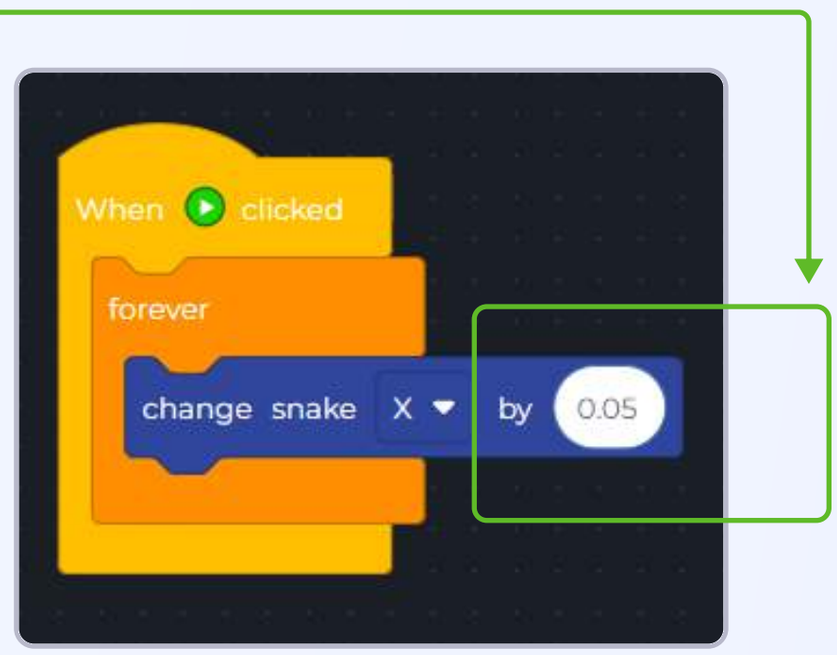


If you click on the green play button now, you will see the snake object move very fast in its right direction. And it will very quickly disappear from the scene.

Click on the reload button to reset the scene.

We need to slow down the speed with which the snake object is moving.

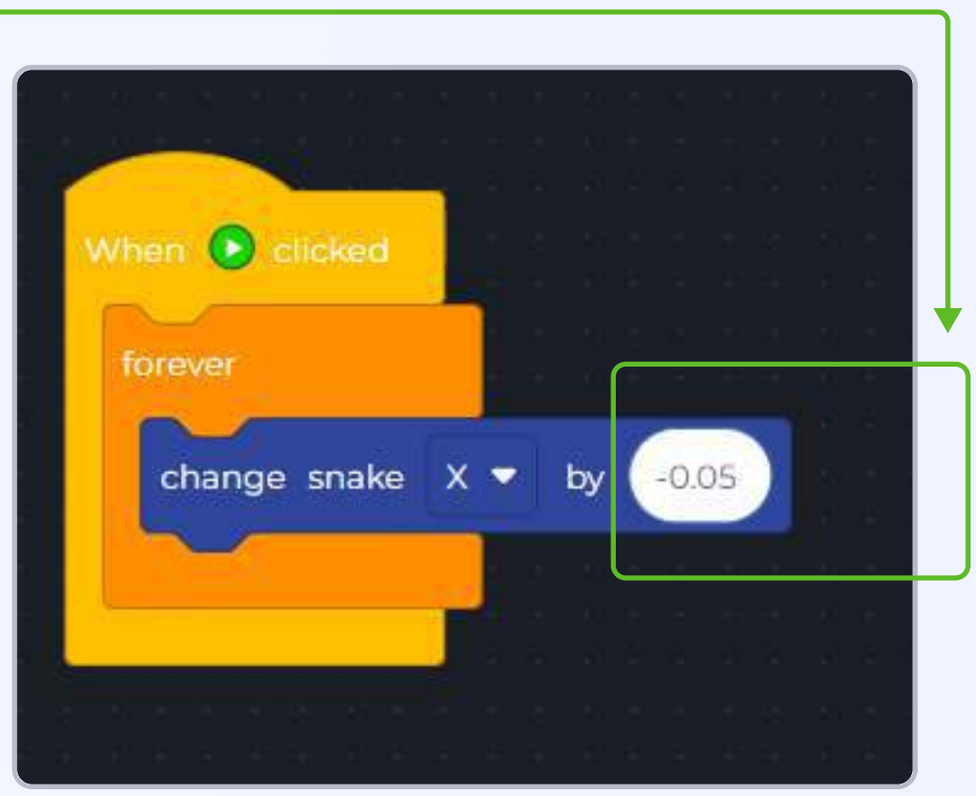
Step 6: To change the speed of the snake box, we can just reduce the value **inside the “change snake x by 1” block. Change the number “1” to “0.05”**



Click on the green play button now, and the snake will be moving in its right direction at a reasonable speed. (You can change the speed to any value that you want)

Click on reload and reset your game.

Step 7: Change the value inside the **“change snake x by 0.05”** block from **“0.05” to “-0.05”.**



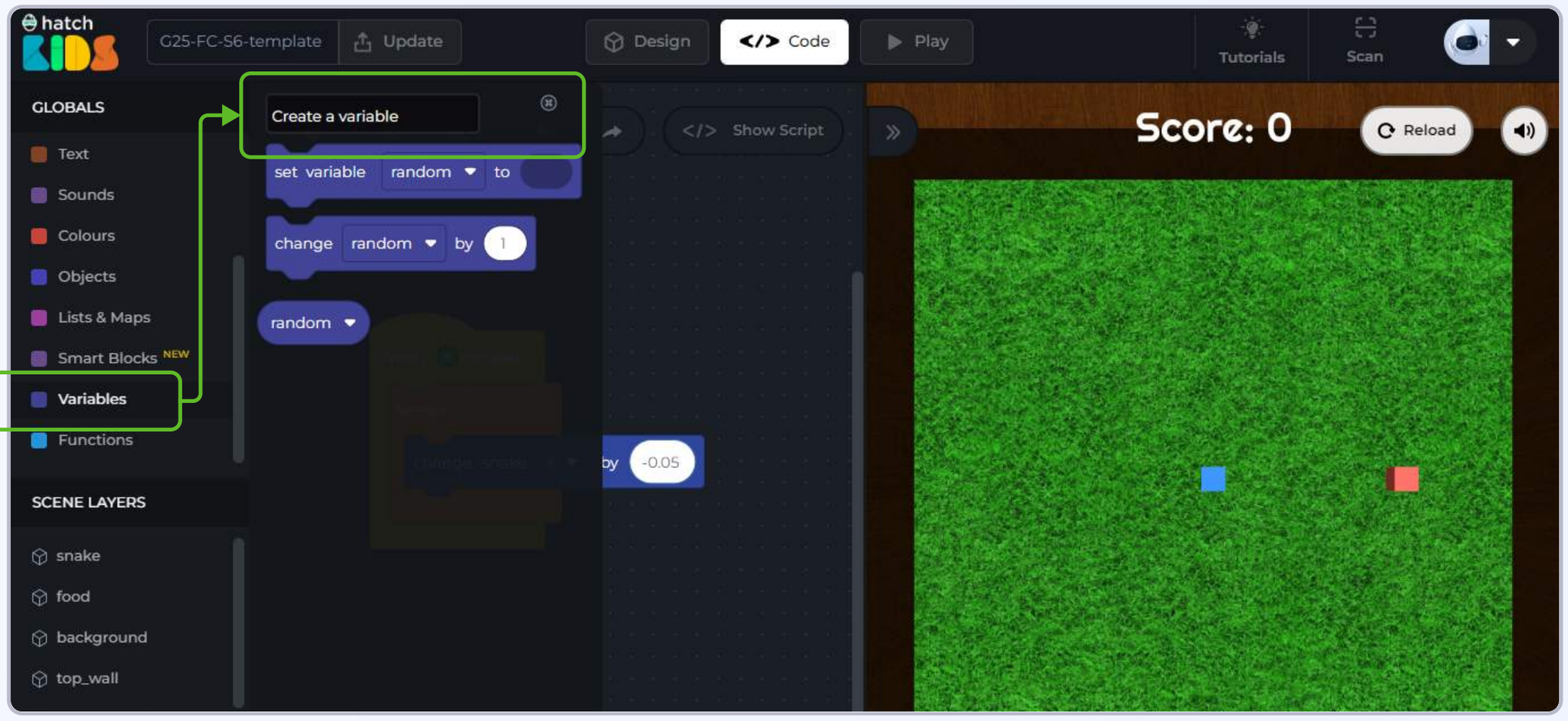
Now when you click on the green play button, you will see the snake moving in the left direction.

So we now know that setting the value inside the “change snake x” block to “0.05” makes it move in its right direction and “-0.05” makes it move in the left direction.

We can replace the number by a variable, and change the value of the variable between “0.05” and “-0.05” when you press the left or right arrow keys to make the snake move in different directions.

Let’s start by creating a variable.

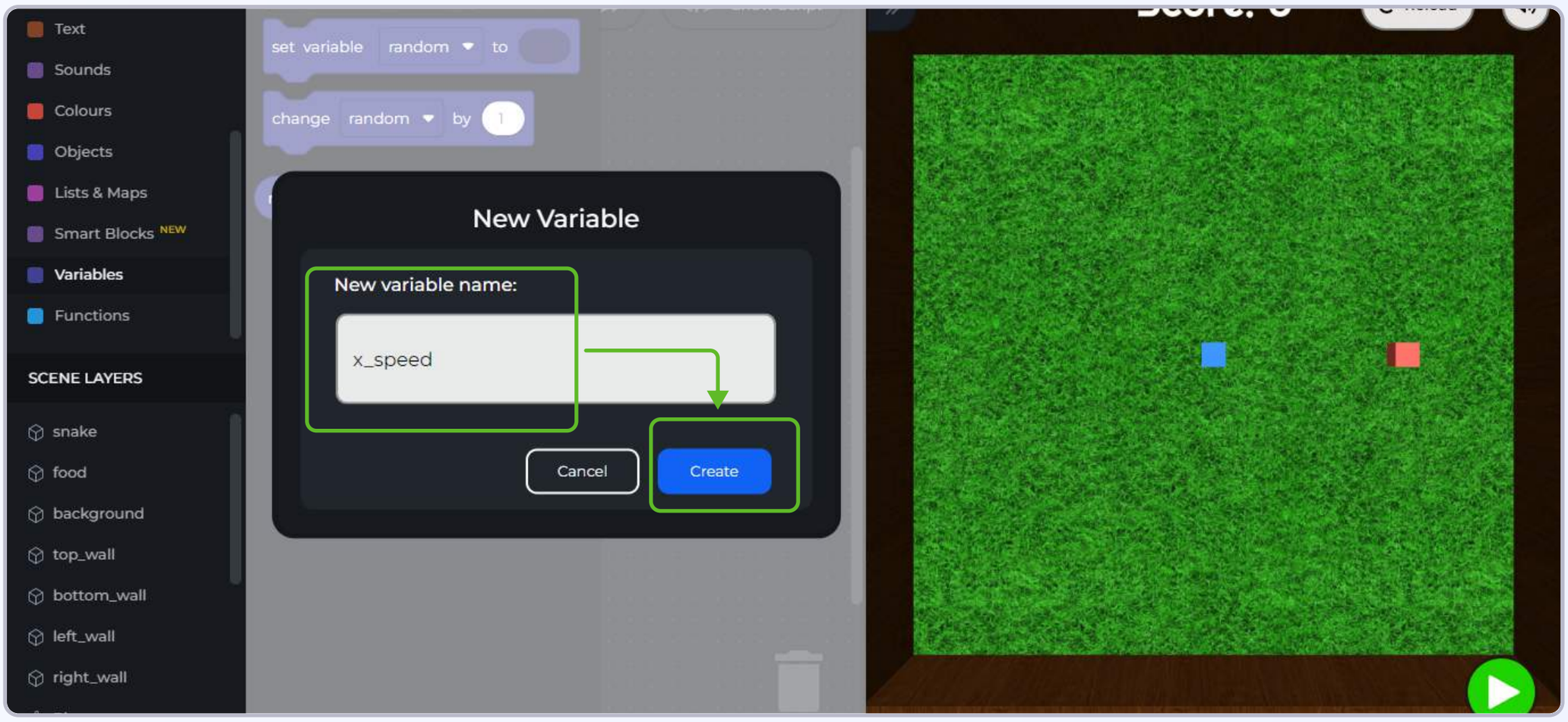
Step 8: Click on the **“variables”** section in the left panel, and click on the button that says **“create a variable”**



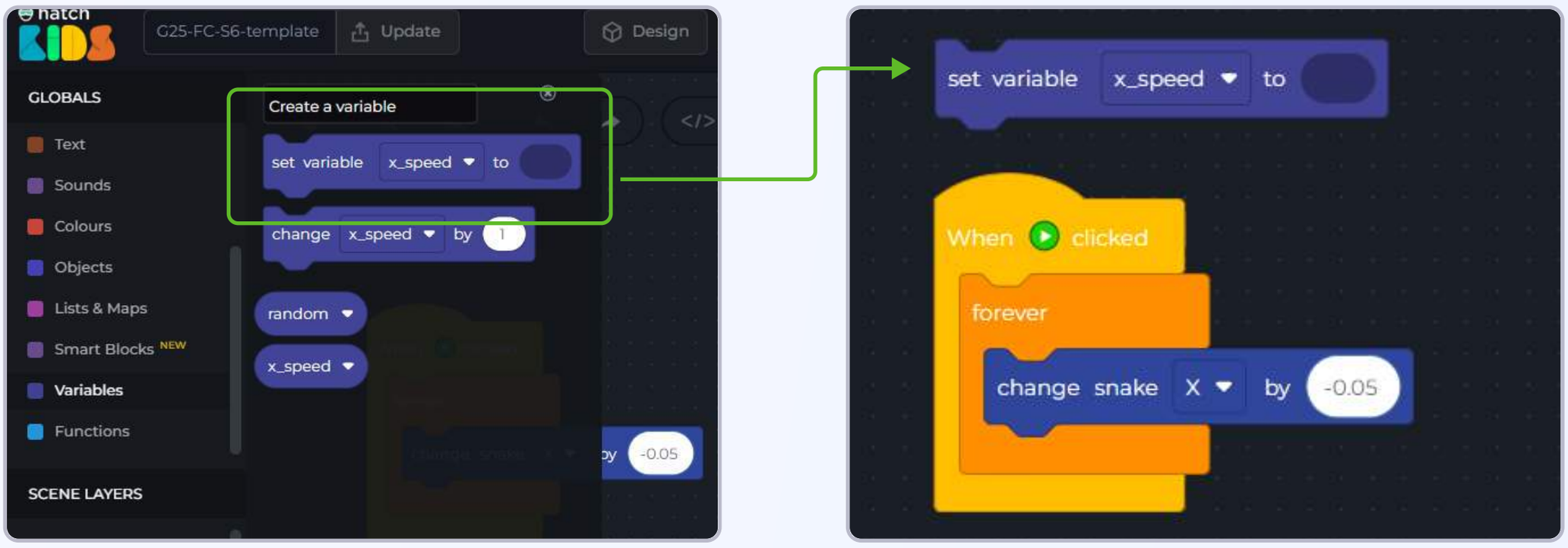
Step 9: A pop up box will appear on yor screen asking you to give a name to your variable.

We are going to use this variable to control the speed of the snake in the x-direction.
So let's name the variable "x_speed"

Type in **"x_speed"** in the input field and click on the **"create"** button.



Step 10: In the variables panel, you will now see a block that says **"set variable x_speed to".**
Drag the **"set variable x_speed to"** block inside the workspace.

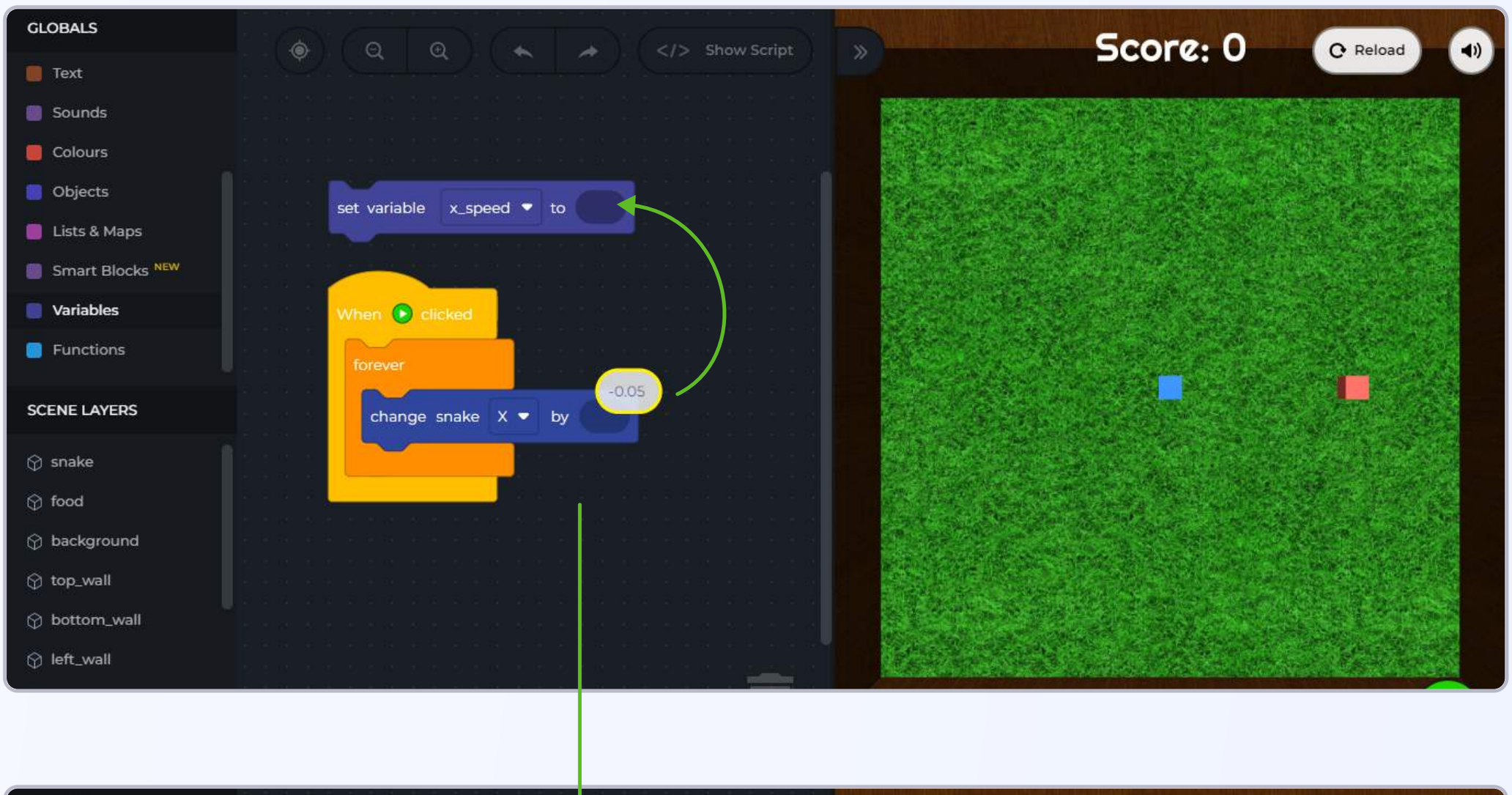


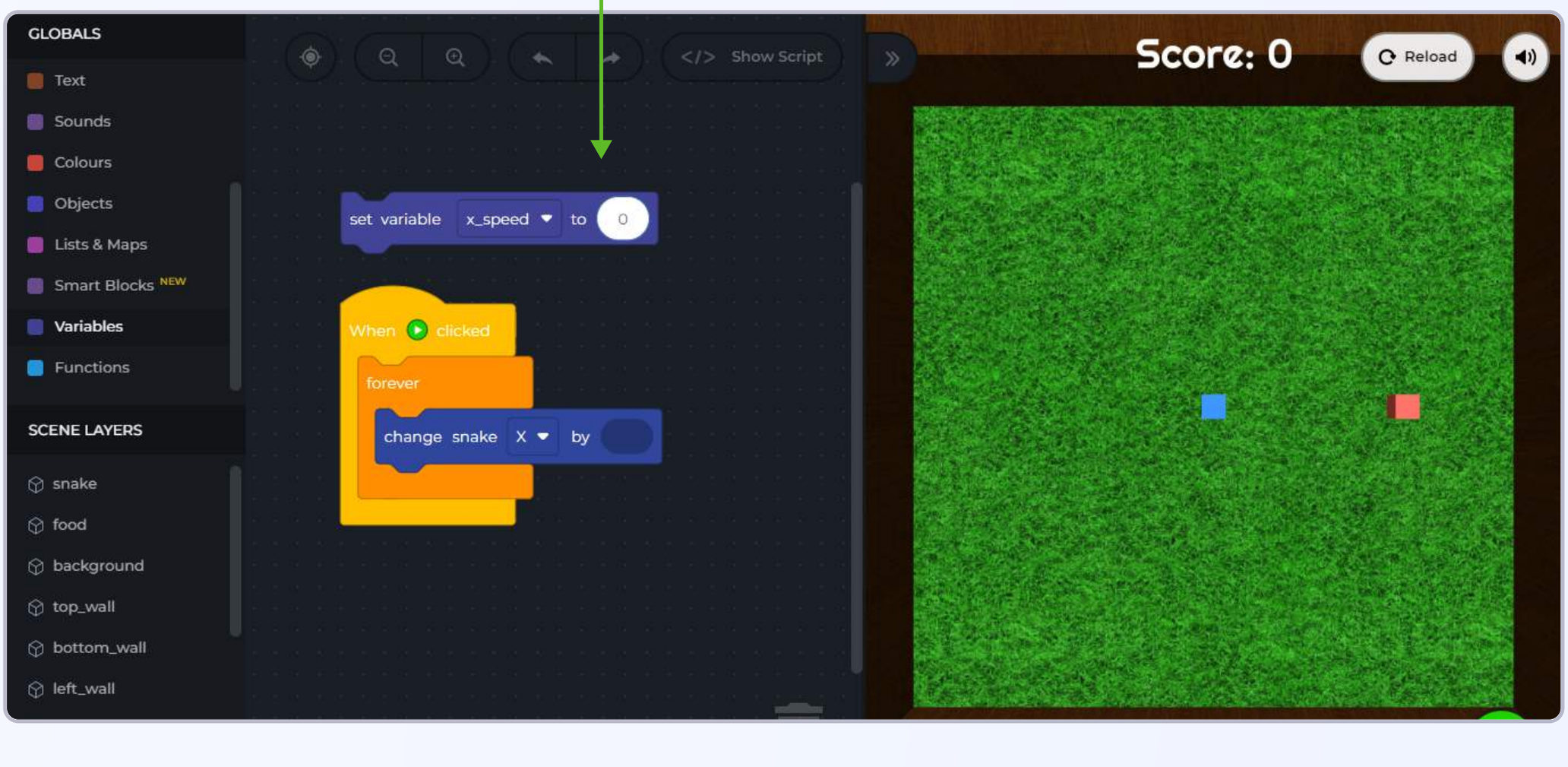
We need to give the variable "x_speed" a value at the beginning of the game. Let's give it a value "0".

Step 11: Drag out the number block from inside the **"change snake x"** block and attach it inside the **"set variable x_speed"** block. And change the number inside to **"0"**

We need to give the variable “x_speed” a value at the beginning of the game. Let’s give it a value “0”.

Step 12: Drag out the number block from inside the **“change snake x”** block and attach it inside the **“set variable x_speed”** block. And change the number inside to **“0”**





Inside the “change snake x by” block, we can now add the variable “x_speed”.

Step 13: Click on the **“variables”** section in the left panel, and drag out the block that just says **“x_speed”.** Attach this block inside the **“change snake x by”** block.

Click on the Green Play button, and you will see that the snake box doesn't move at all.

That’s because in the code we are saying that the x-position of the snake should change by the value of the variable **“x_speed” and the value of the variable x_speed = 0.**

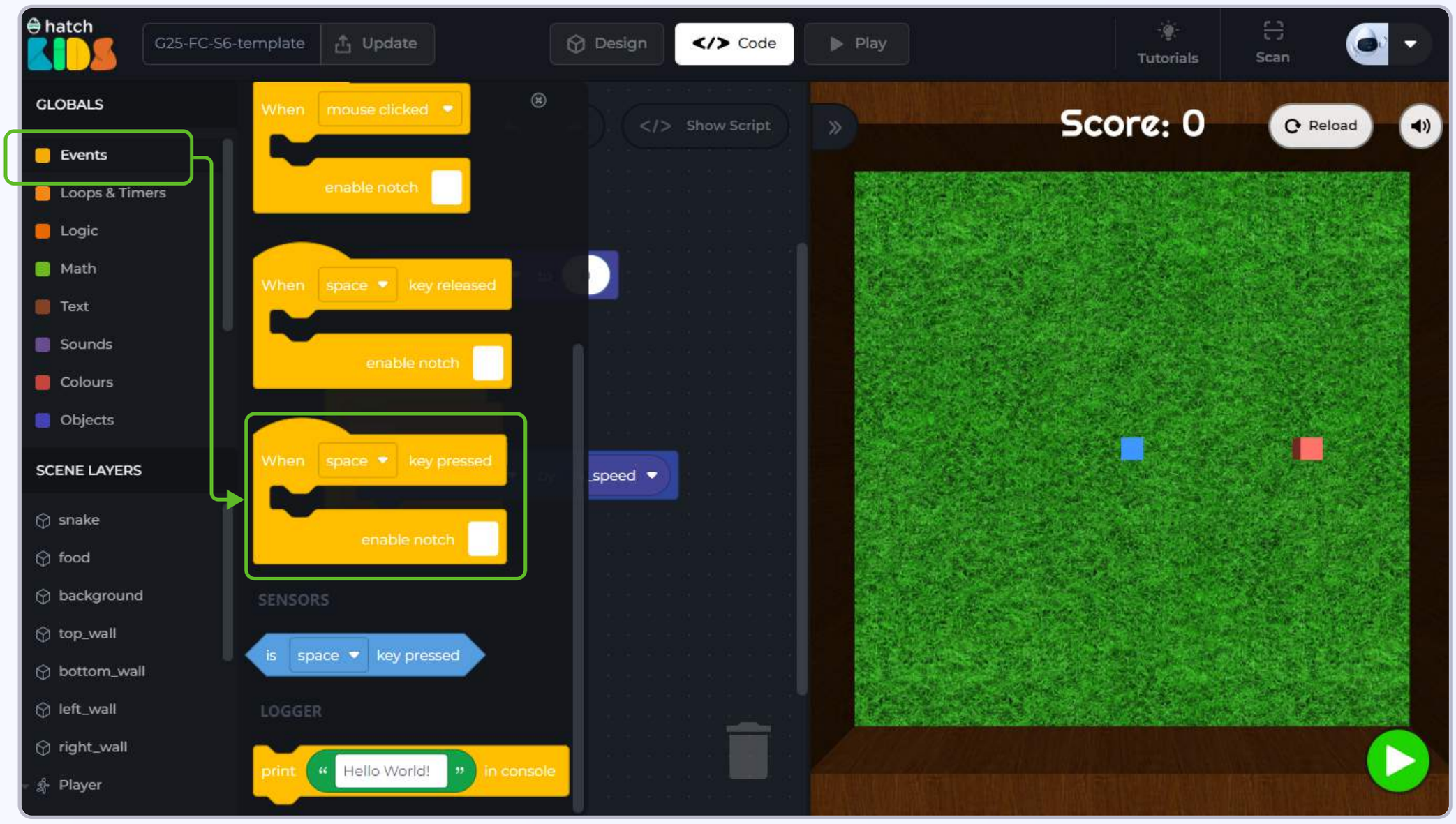
Click on the Reload button.

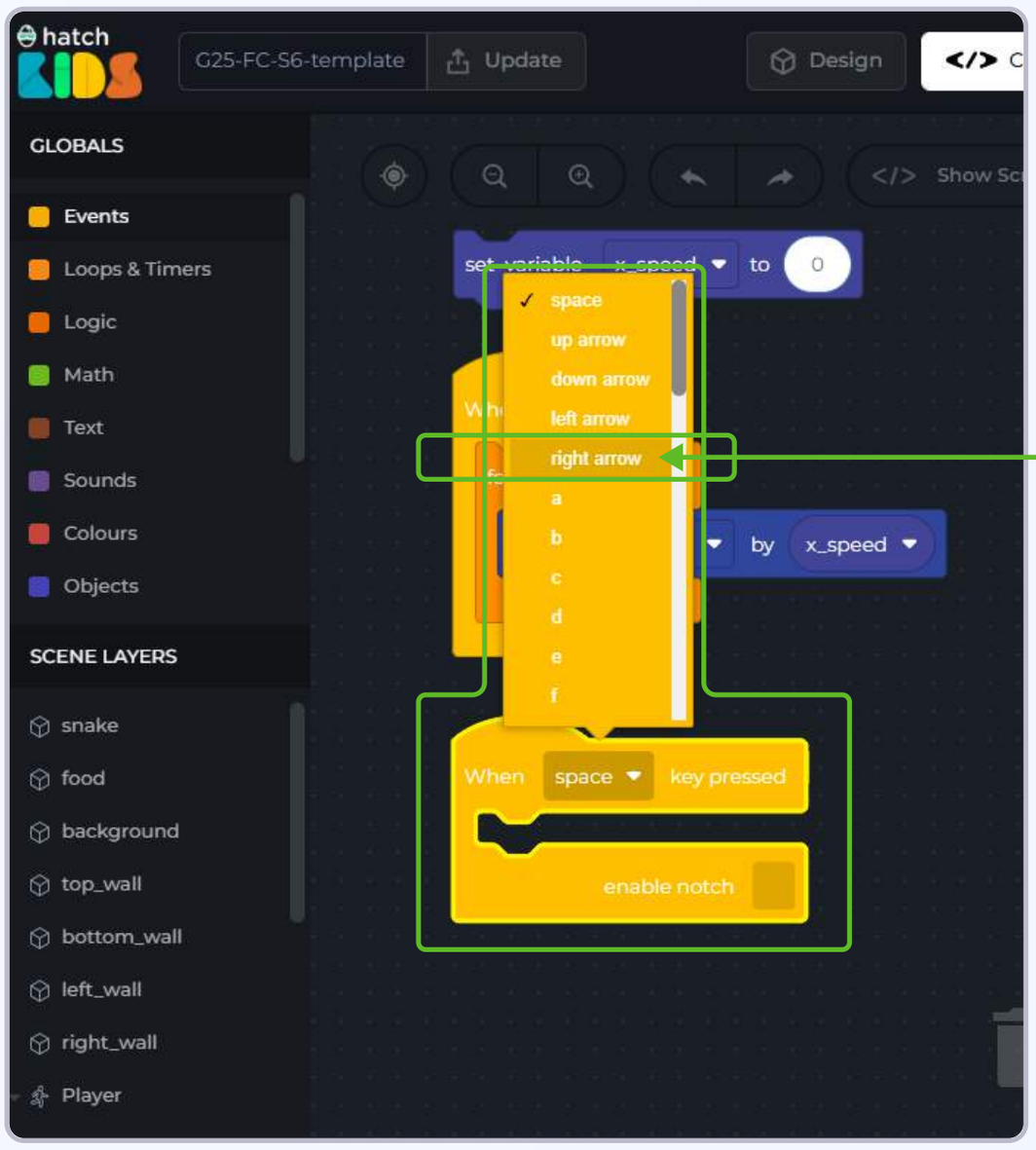
Let’s make the snake move right when you press the “right arrow key”.

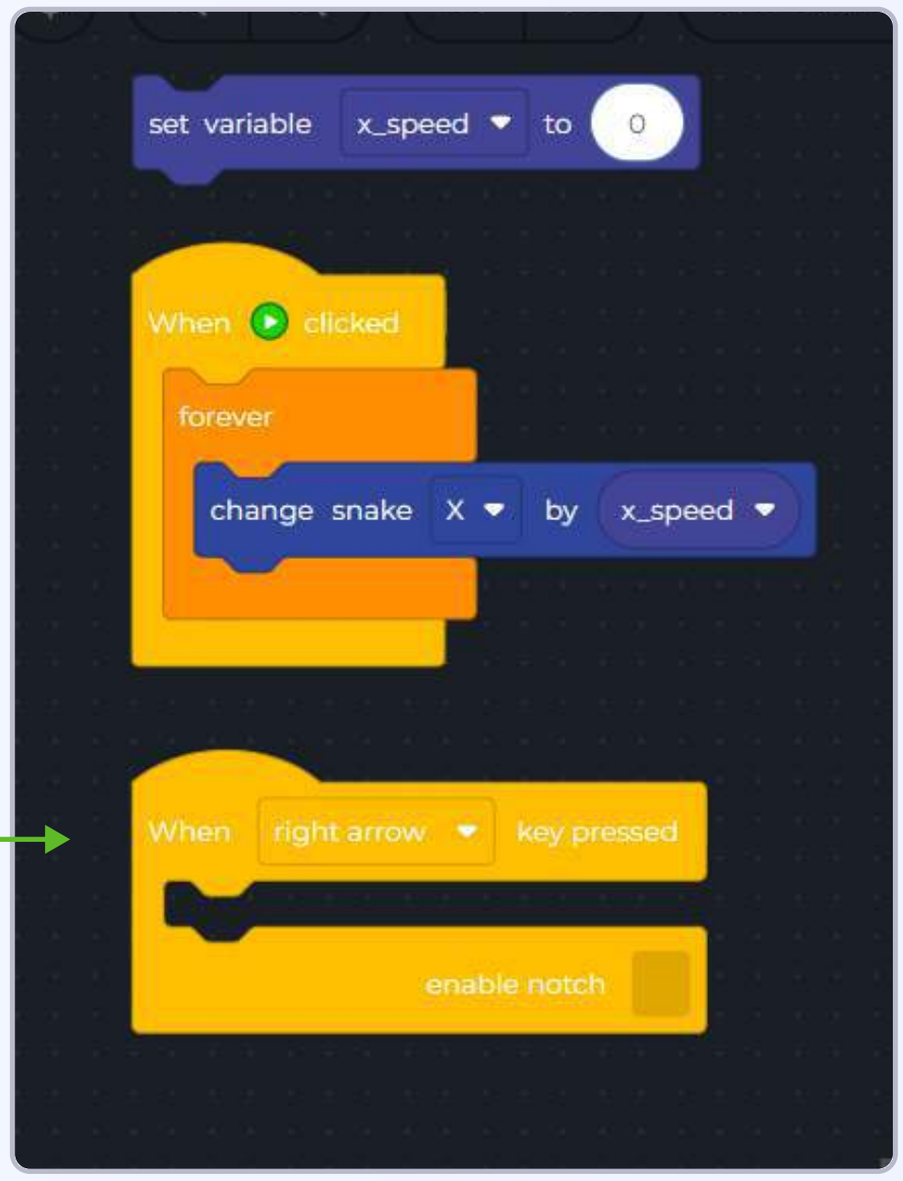
Step 14: Click on the **“events”** section in the left panel, and drag out the block that says **“when space key pressed”**

Click on the drop down option around the word **“space”** key, and select the option that says **“right arrow”.**

Your block should now say **“when right arrow key pressed”**





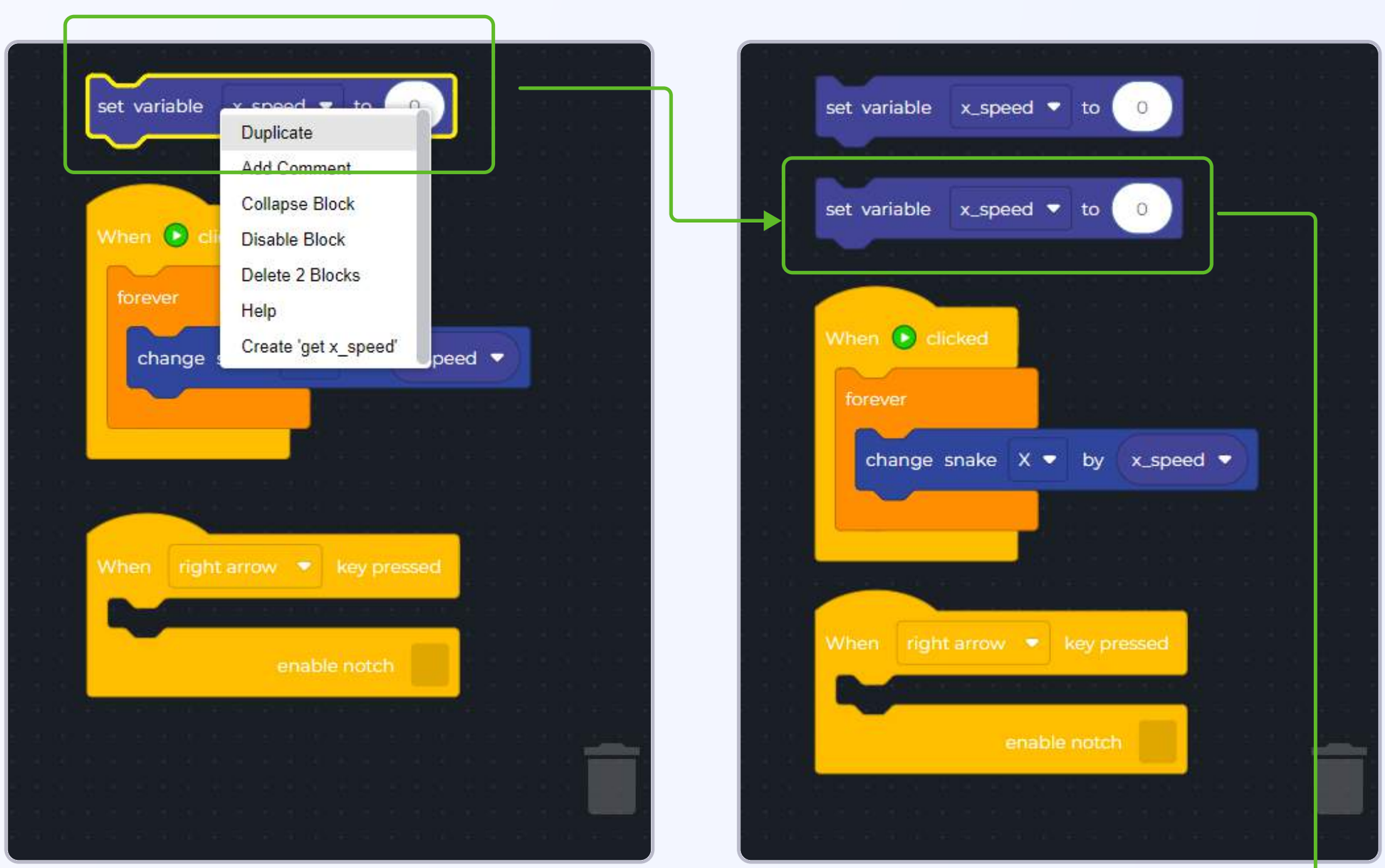


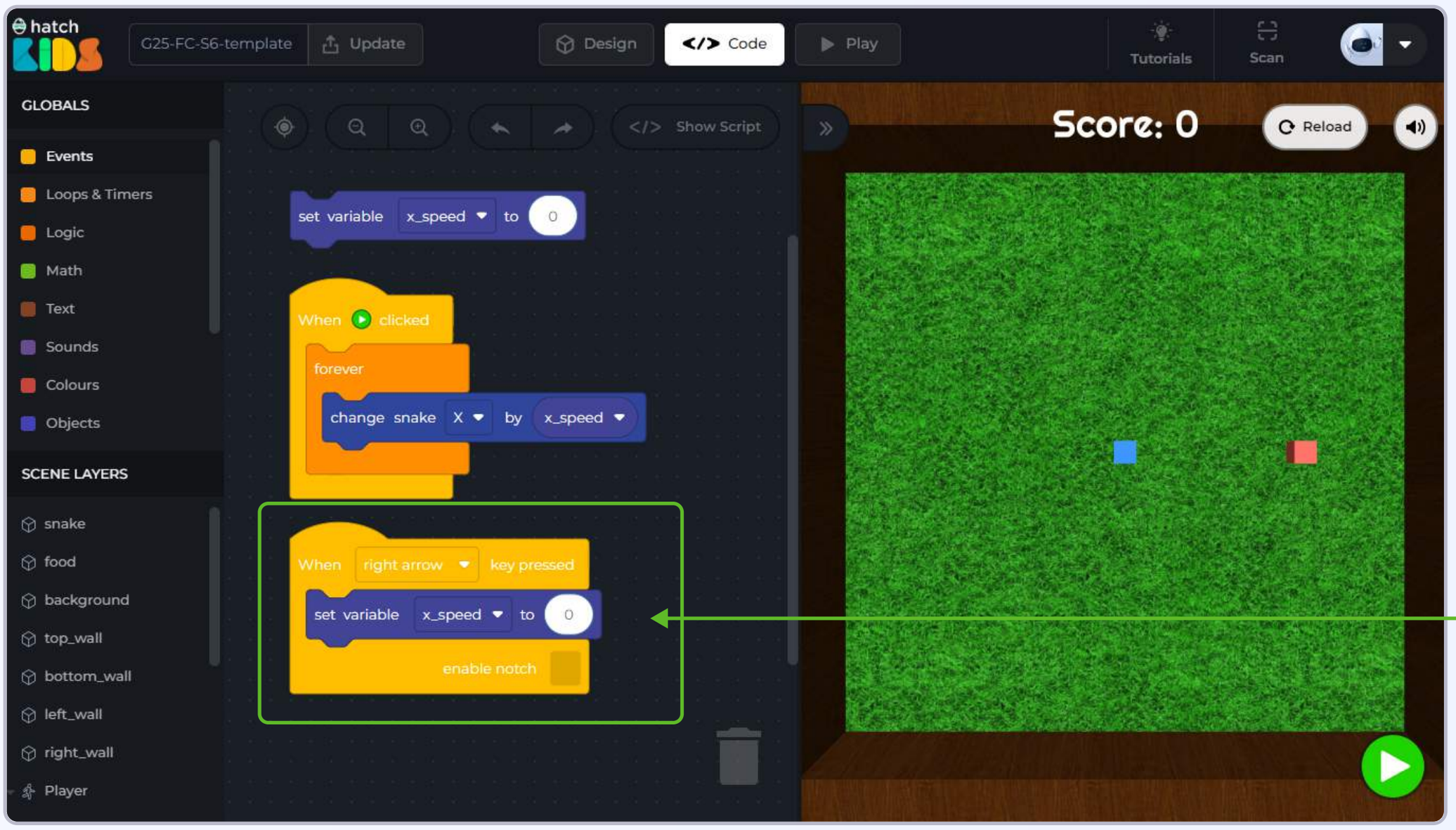
When the right arrow key is pressed, we want the snake to move in the right direction. We know to make the snake move in its right direction, the value of “x_speed” variable should be 0.05

Note: Any positive value of x_speed will make the snake move in the right direction, as to move it the right direction the x position of the snake should just increase. We are using 0.05 value here because that makes the snake move a good enough speed for everyone to be able to play the game

Step 15: Duplicate the **“set variable x_speed to 0”** block, and attach the **new “set variable” block** inside the **“when right arrow key pressed”** block.

Note: To duplicate a block, move your mouse over the block, and right click. A drop down menu will appear, select the option that says **“duplicate”**





Step 16: Once the **“set variable x_speed to 0”** block is placed inside the “**when right arrow key pressed”** block, **change the value “0” to “0.05”**

Now click on the green play button to run the code.

Initially the snake would not be moving, but then the moment you press the right arrow key, the snake starts moving towards its right side

Now let’s make the snake move left when the left arrow key is pressed.

We know to move the snake in its left direction, we can set the value of the variable **“x_speed” to -0.05**

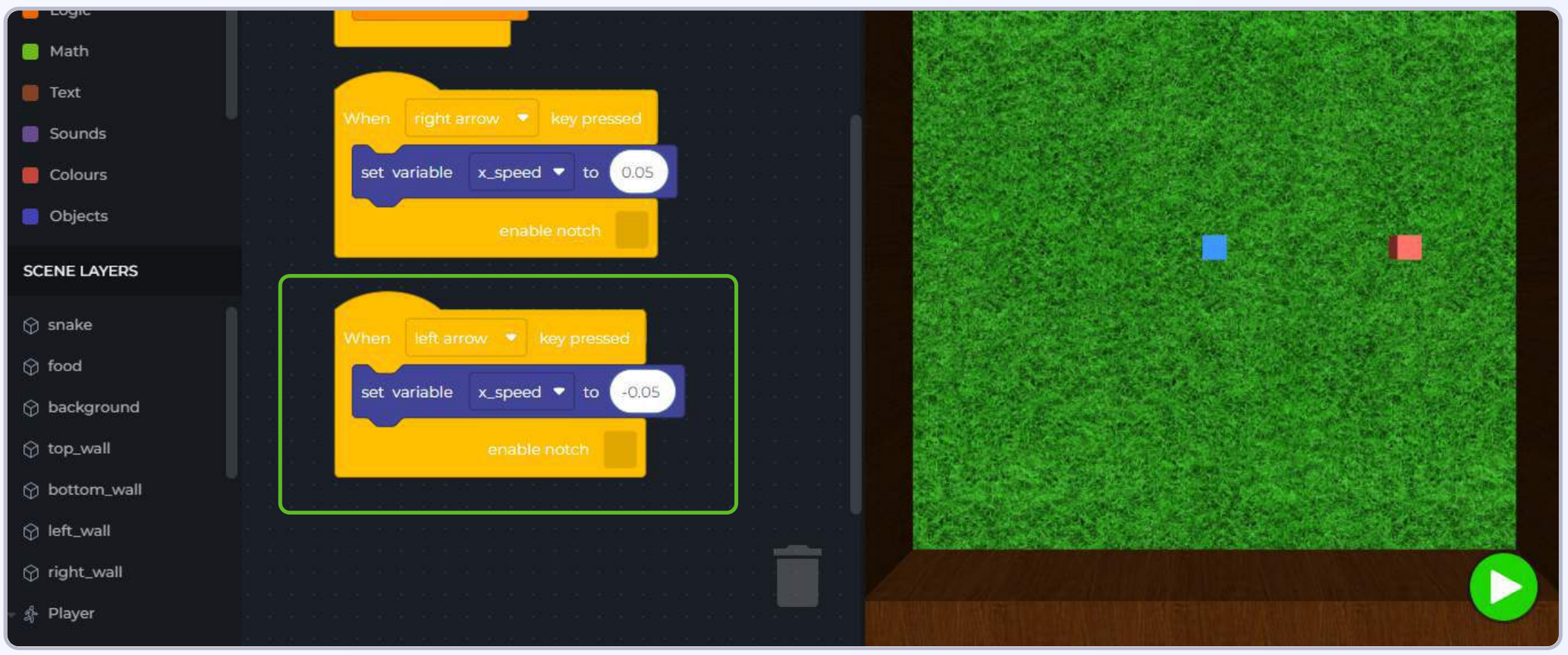
Let’s code this. Click on the reload button to stop the current code from running.

Step 17: Duplicate the **“when right arrow key pressed”** block.

Step 18: In the new **“when right arrow key pressed”** block, click on the drop down option beside the **“right arrow”**, and select the **“left arrow”** option from the menu.

Your new block would now read **“when left arrow key pressed -- set variable x_speed to 0.05”**

Step 19: Change the value **“0.05” to “-0.05”** inside the **“when left arrow key pressed”** block.



Click on the green play button to run the code. You will notice, you can move the snake in the left and right directions by just pressing the left and right arrow keys on the keyboard.

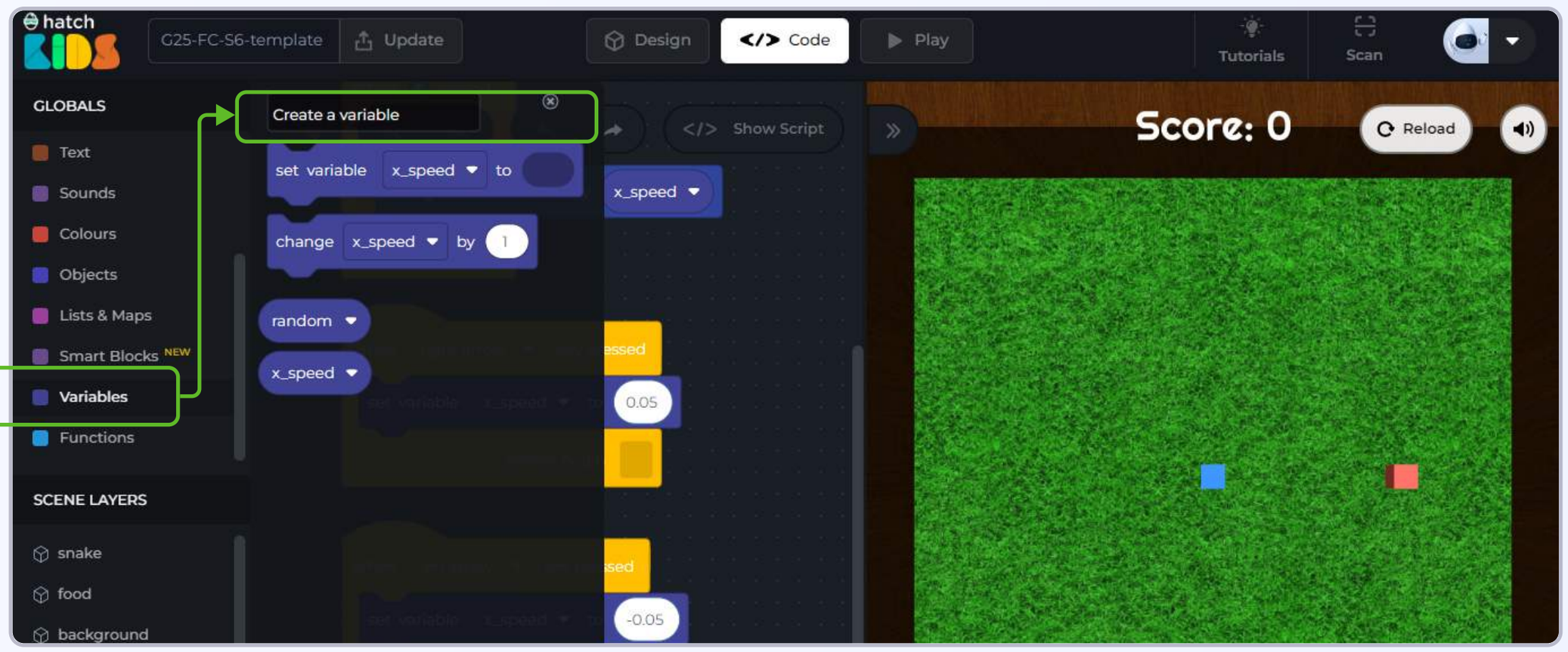
Click on the Reload button and reset the scene.

Let’s now make the snake move in up and down direction as well. We know that in order to move the snake box up or down we will need to change the value of its y-position.

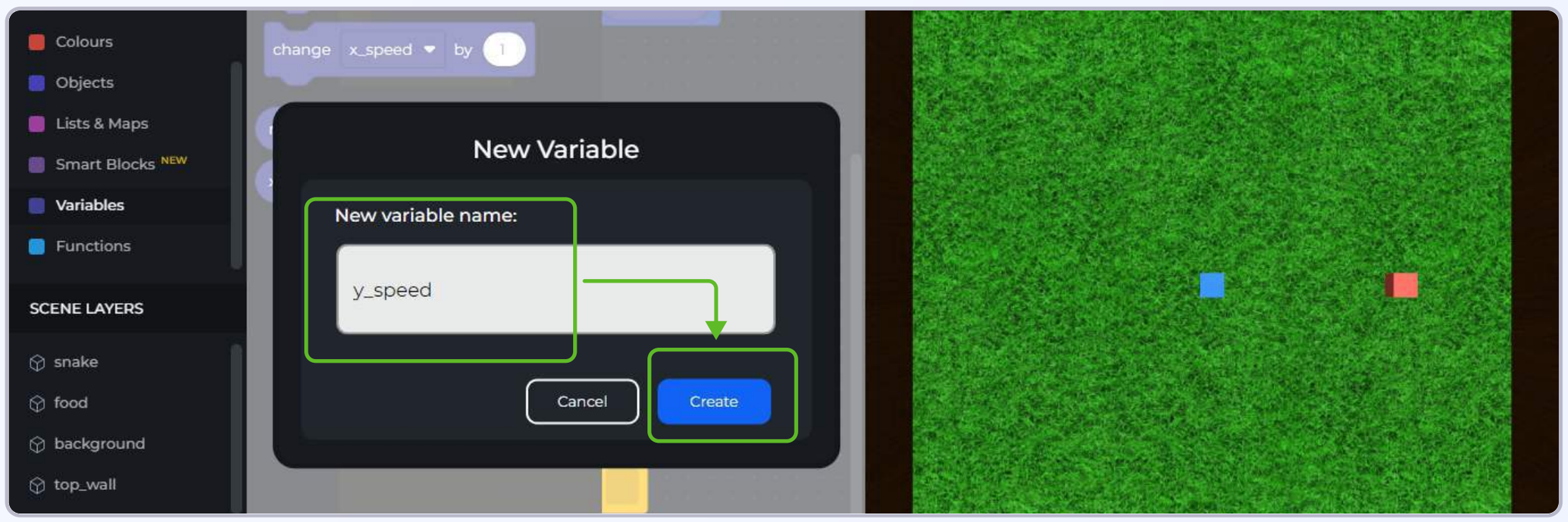
And just like we are controlling the x-position using the variable x_speed, similarly, we can create another variable “y_speed” with which we can control the snake’s up and down motion.

Let’s start by creating a variable with the name “y_speed”.

Step 20: Click on the **“variables”** section in the left panel, and click on the **“create a variable”** button

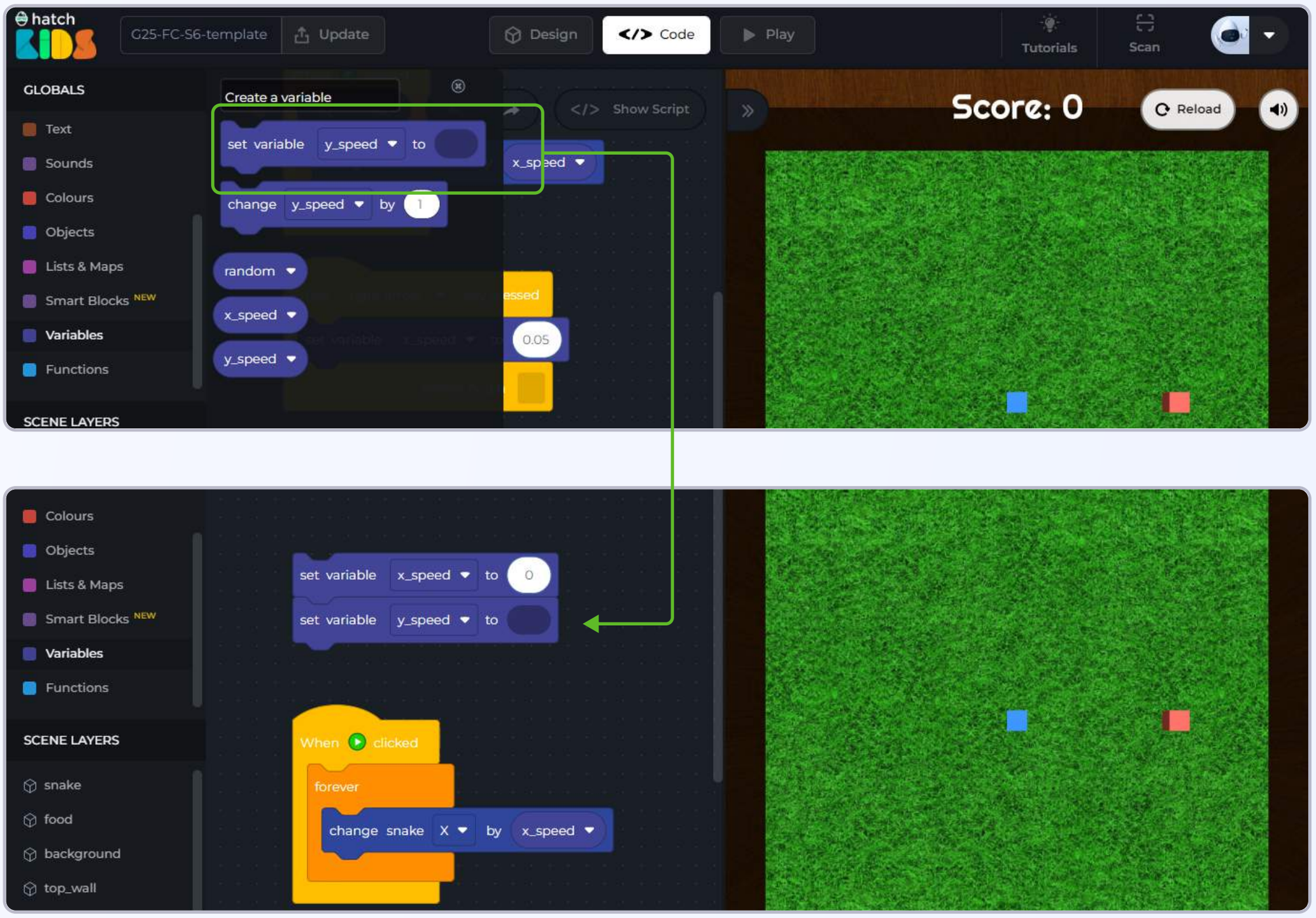


A window will appear, asking you to give a name to your variable. Type in **“y_speed”** and click on the **“create”** button.



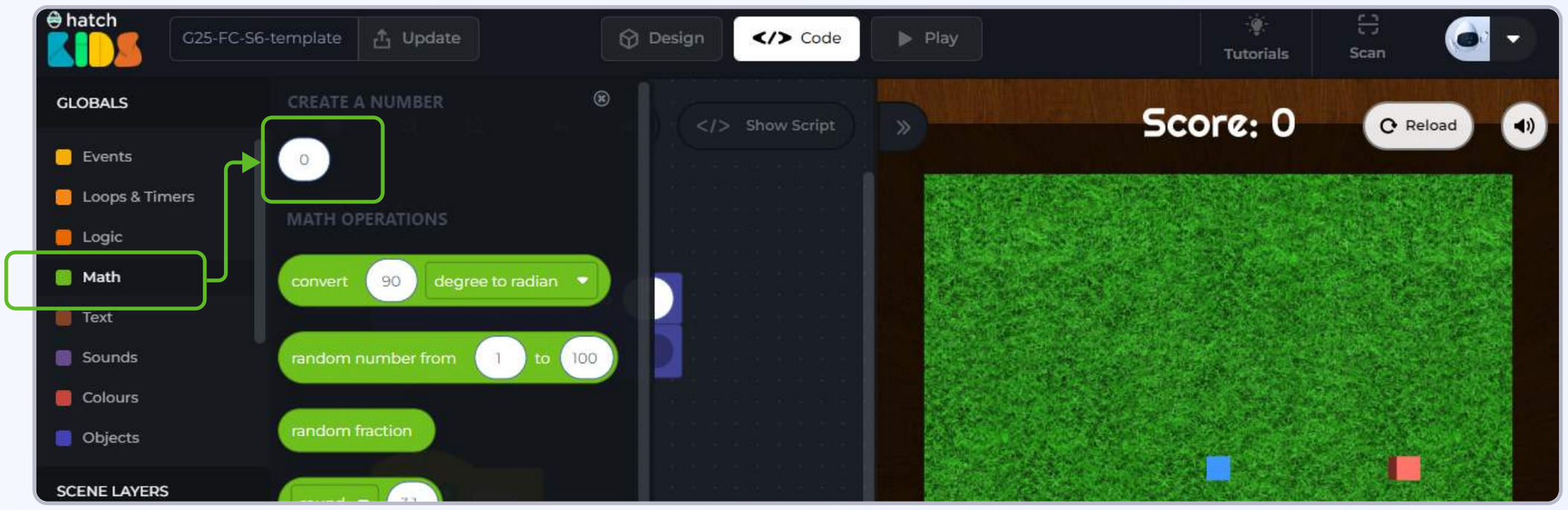
In the variables section you will now be able to see blocks related to “y_speed”

Step 21: Click and drag the **“set variable y_speed to”** block in the workspace, and place it below the **“set variable x_speed to 0”** block.

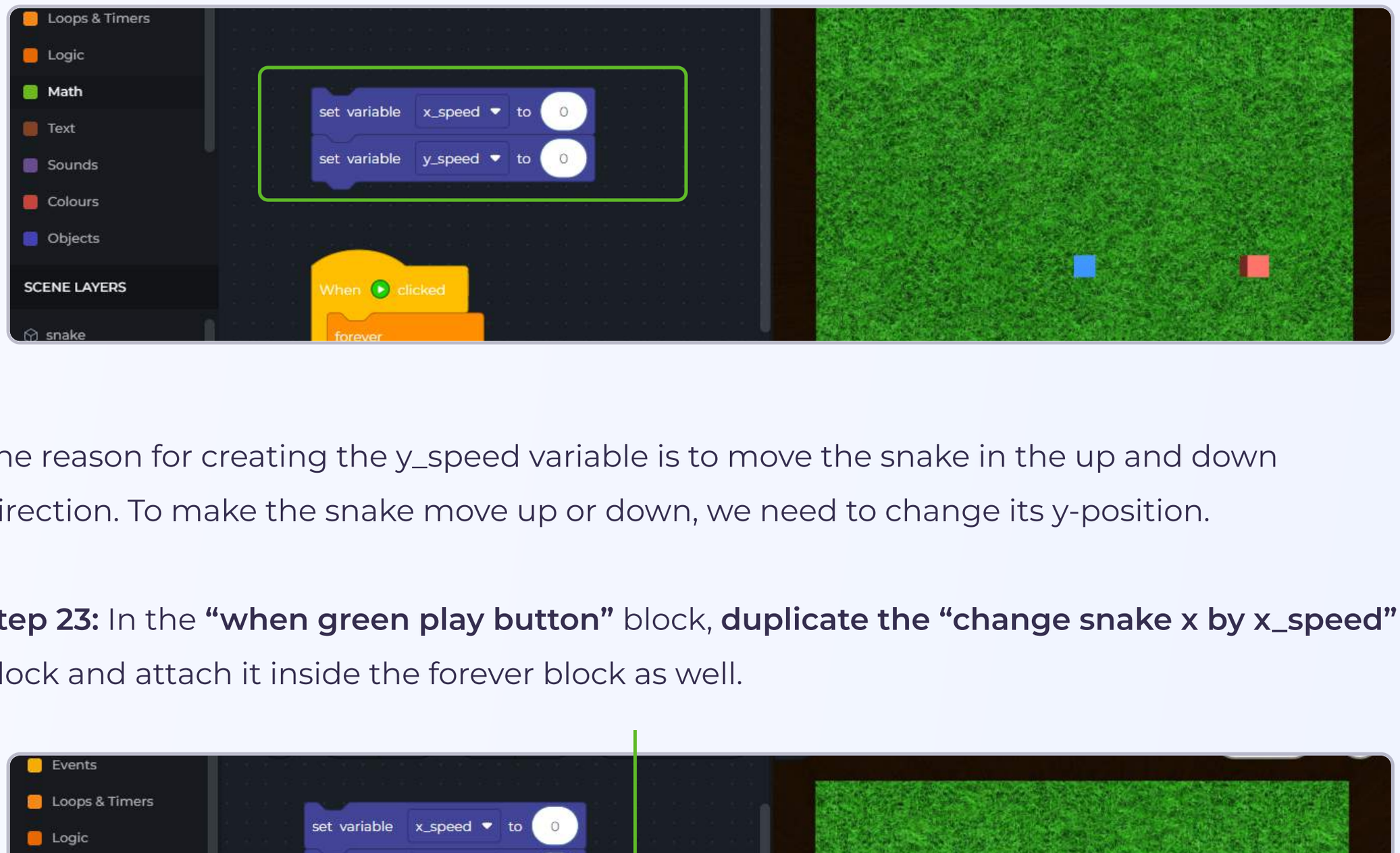


Just like we set the value of x_speed to 0 at the beginning of the game, we can also set the value of “y_speed” to 0.

Step 22: Click on the **“Math”** section in the left panel and you will see at the top of the list there is a number block that says “**0”**



Drag the “0” block and attach it inside the “set variable y_speed” block



The reason for creating the y_speed variable is to move the snake in the up and down direction. To make the snake move up or down, we need to change its y-position.

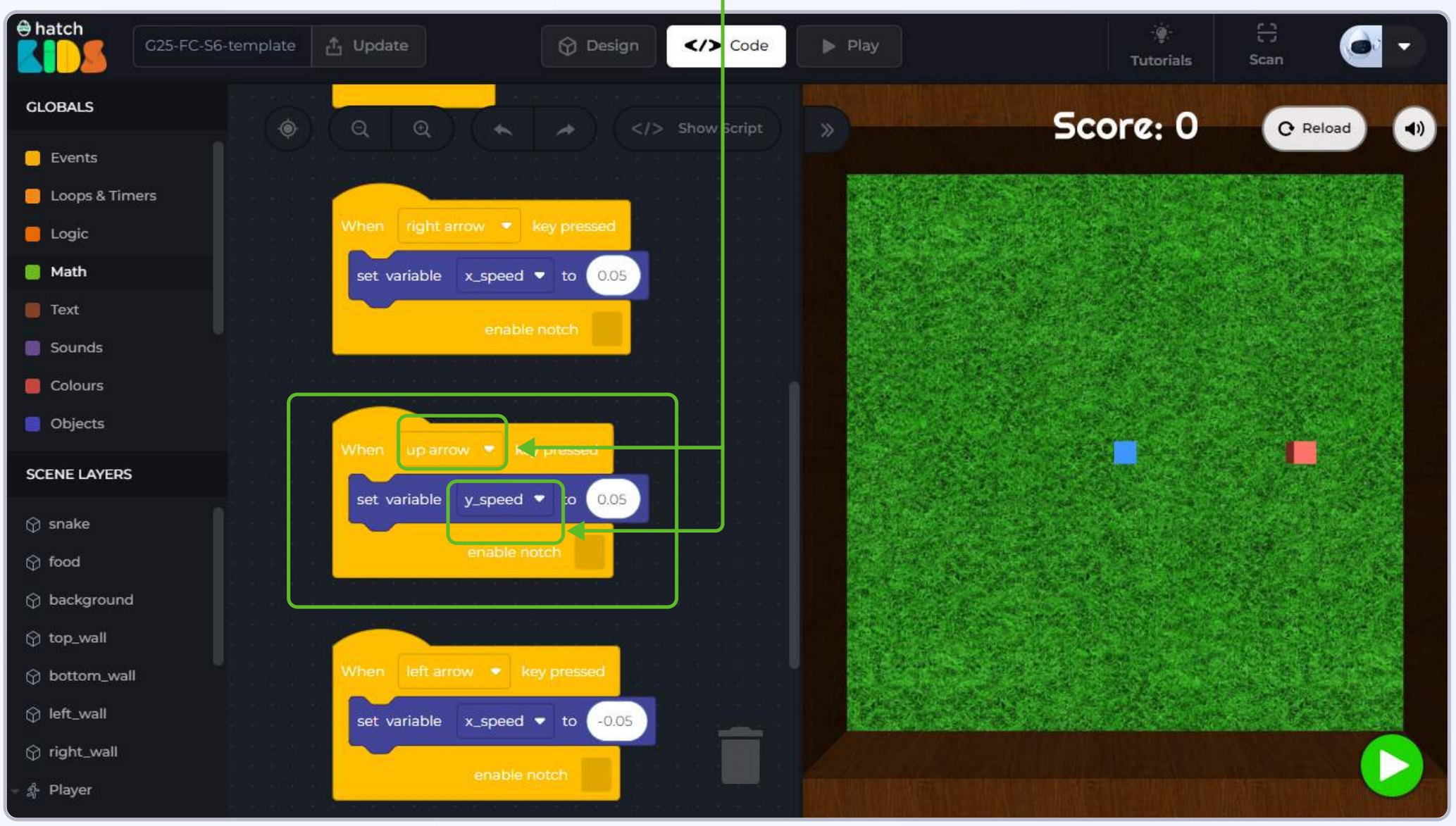
Step 23: In the **“when green play button”** block, **duplicate the “change snake x by x_speed”** block and attach it inside the forever block as well.

In the new **“change snake x by x_speed”** block, click on the **drop down** beside **“x”** and change it to **“y”**

Click on the **drop down** around **“x_speed”** and select the variable **“y_speed”.**

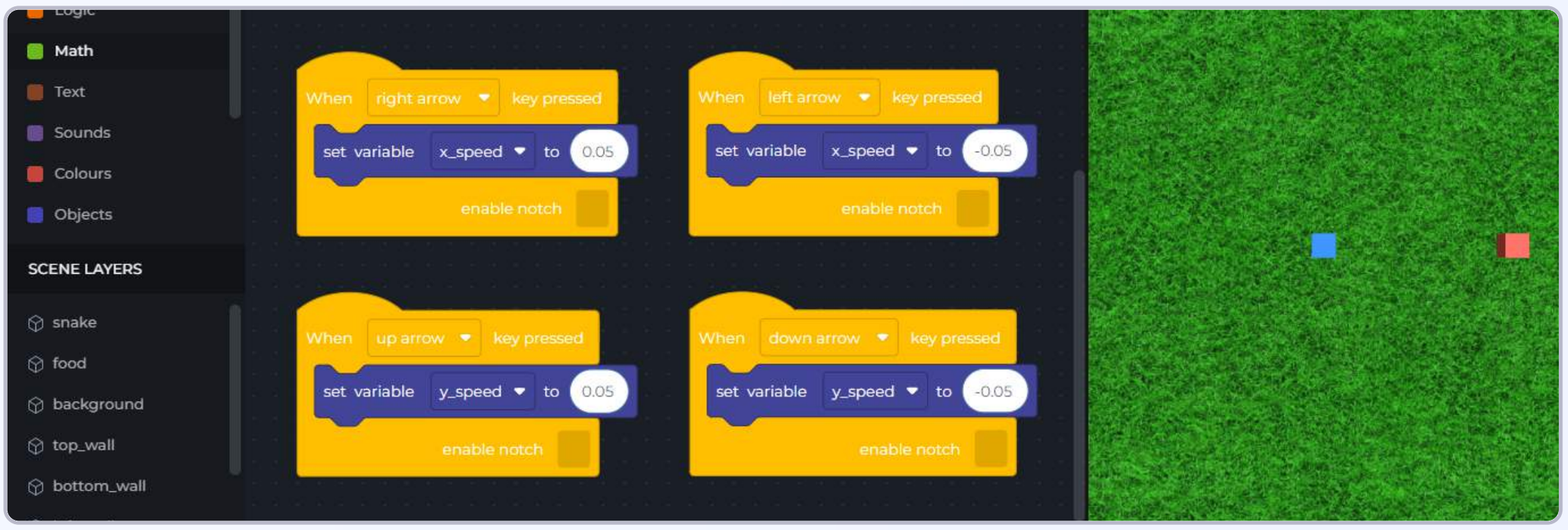
Just like we coded for x_speed to change its values as the left and right arrow keys are pressed. Similarly, to move the snake up, when the up arrow key is pressed, and down when the down arrow key is pressed, we need to add “when up arrow key pressed” & “when down arrow key pressed” blocks, and set y_speed to a positive value (0.05) for up arrow key, and a negative value (-0.05) for down arrow key

Step 24: Duplicate the **“when right arrow key pressed set variable x_speed to 0.05” block.** and click on the drop down options and change the “right arrow” to “up arrow” key. and change the “x_speed” variable to “y_speed”



Duplicate the **“when up arrow key”** block and **change** the **“up arrow”** option to **“down arrow”.**

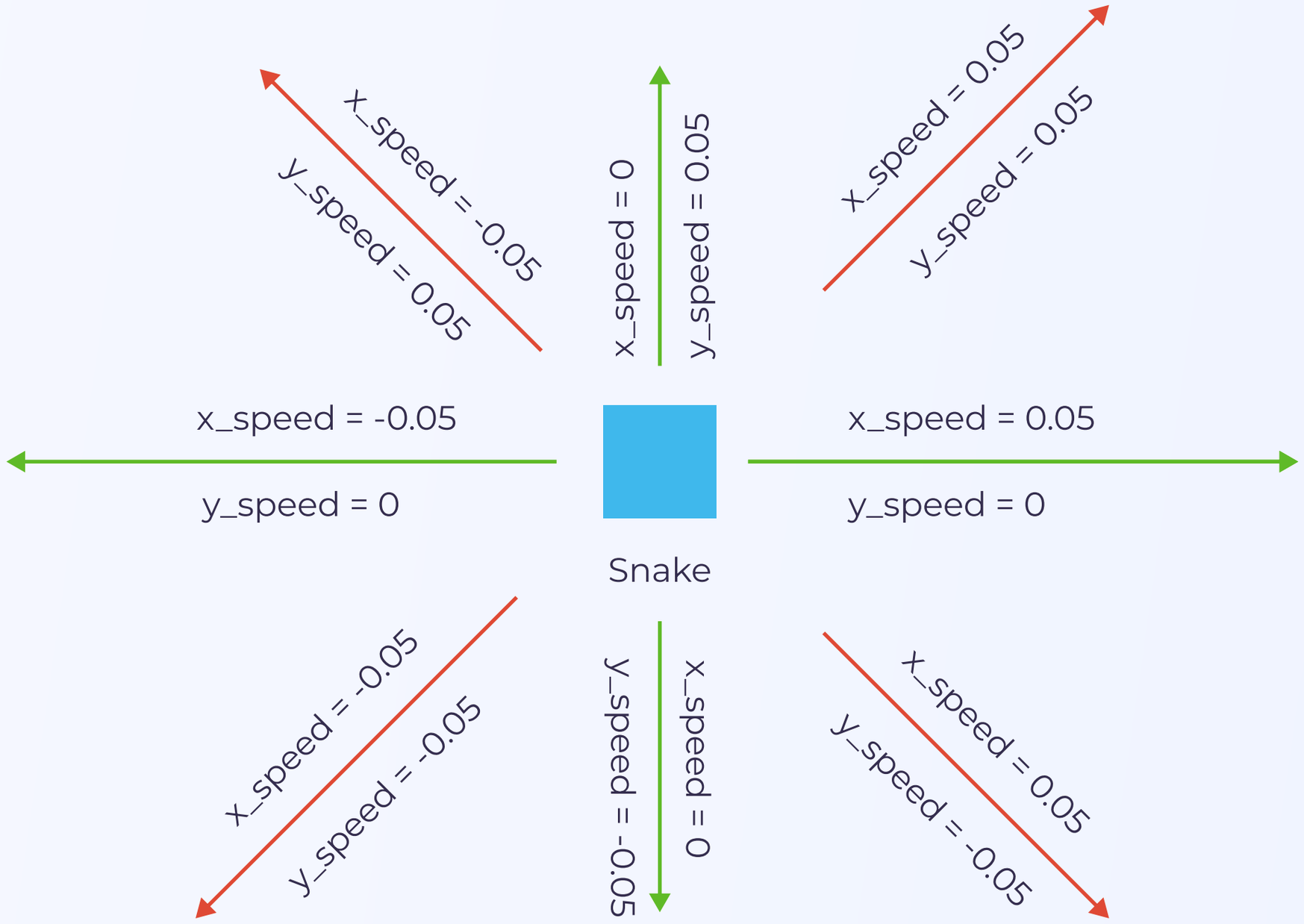
In the **“when down arrow key pressed set variable y_speed to 0.05”** block, **change the value of “0.05” to “-0.05”.**



Now when you click on the green play button, and press the up and down arrow keys, the snake will move up and down.

But you will also notice, that after pressing the up/down arrow key, if you press the left/right arrow keys, then instead of moving left/right, the snake box is moving diagonally.

Let’s understand why this happens



We have 2 variables:

- x_speed
- y_speed

- A --->> when x_speed = 0.05,, snake will move right
- B --->> when x_speed = -0.05, snake will move left
- C---->> when y_speed = 0.05, snake will move up
- D --->> when y_speed = -0.05, snake will move down

What happens when x_speed = 0.05, and y_speed = 0.05 ?

The snake will try to move right, and up. As a result, it would look like the snake is moving diagonally up. Similarly, you can see the other cases when the snake will move diagonally in the image above.

So, in order to just make snake move right, x_speed = 0.05, and also y_speed = 0.

y_speed = 0 will ensure that the snake is not moving up or down, and thn x_speed = 0.05 will make the snake move right.

Similarly,

- to move only left x_speed = -0.05 y_speed = 0
- to move only right x_speed = 0.05 y_speed = 0
- to move only up x_speed = 0, y_speed = 0.05
- to move only down x_speed = 0 y_speed = -0.05

So we have to update the code so that,

a) when **up arrow key is pressed**, we **set** the value of **x_speed to 0**, and **y_speed to 0.05**

b) when **down arrow key is pressed**, we **set** the value of **x_speed to 0**, and **y_speed to -0.05**

c) when **left arrow key is pressed**, **set x_speed to -0.05**, and **y-speed to 0**

and d) when **right arrow key is pressed**, **set x_speed to 0.05,** and **y_speed to 0**

Step 25: Duplicate the **“set variable x_speed to 0”** and **“set variable y_speed to 0”** blocks, and attach them in the **“when up / down / left / right arrow key pressed”** blocks as required based on the conditions mentioned in the statements a, b, c, and d above.

Your final output should look like:

Click on reload to reset your scene, and then click on the green play button to run your new code.

Press the up, down, left, and right arrow keys and the snake will move in the correct directions as intended, it wont move in any diagonal direction.

🎉🎉🎉🎉🎉 **CONGRATULATIONS!!! You just completed the first part of the game of making the snake move using your keyboard buttons** 🎉🎉🎉🎉

Objective No. 3: Detect collisions between snake and food

Now that you are able to move your snake, we need to code what happens when the snake reached the food.

In the final game, when the snake reaches the food:

1. the food moves to a new position
2. the score increases

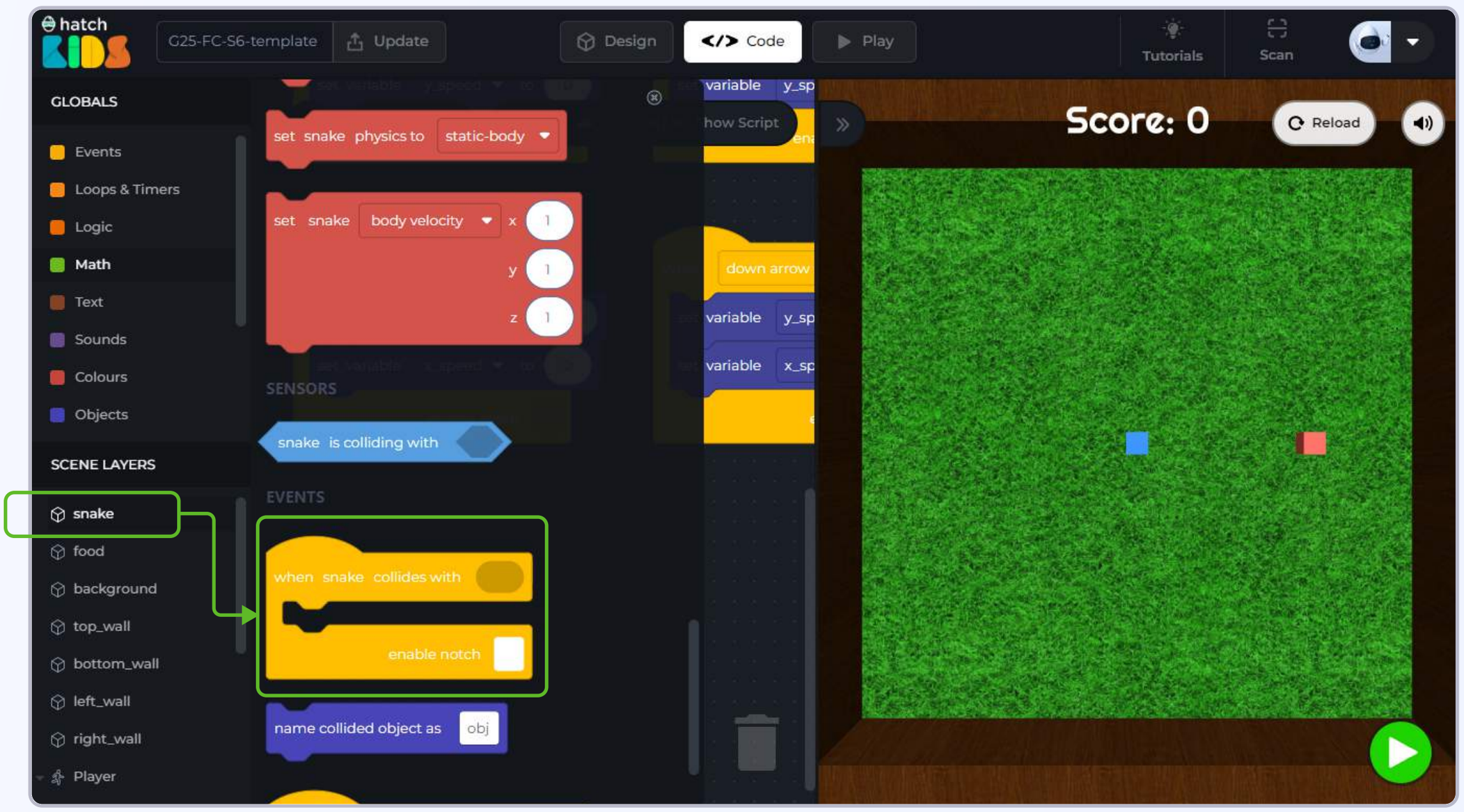
To code both of these features in the game, we need to first check if the snake has reached the food or not, that is, we need to check if the snake is colliding with the food or not.

To check if two objects are colliding or not, there is a special block in the hatch workspace, called “detect collision” block.

Let’s see how that block works

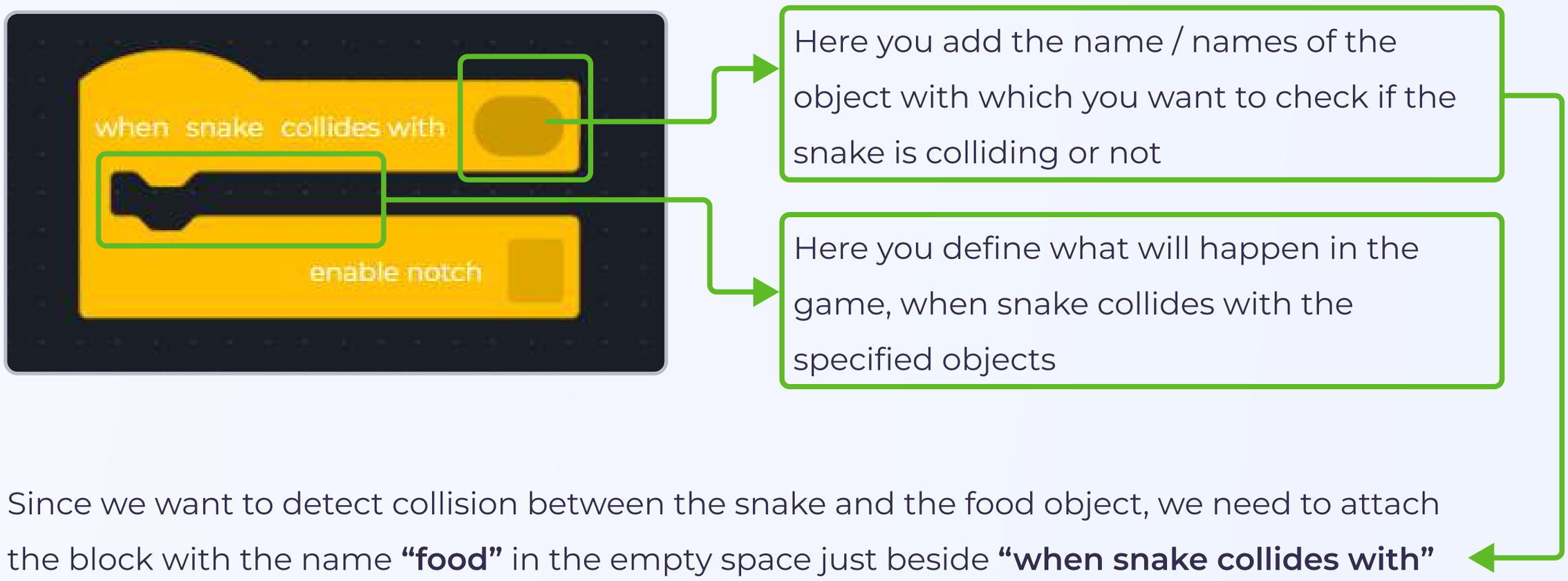
Step 1: Since we want to check if snake is colliding with the food or not, click on the name **“snake”** in the left panel, and scroll to the very bottom of the list of block that appears.

You will see a yellow colored block that says**, “When snake collides with”**



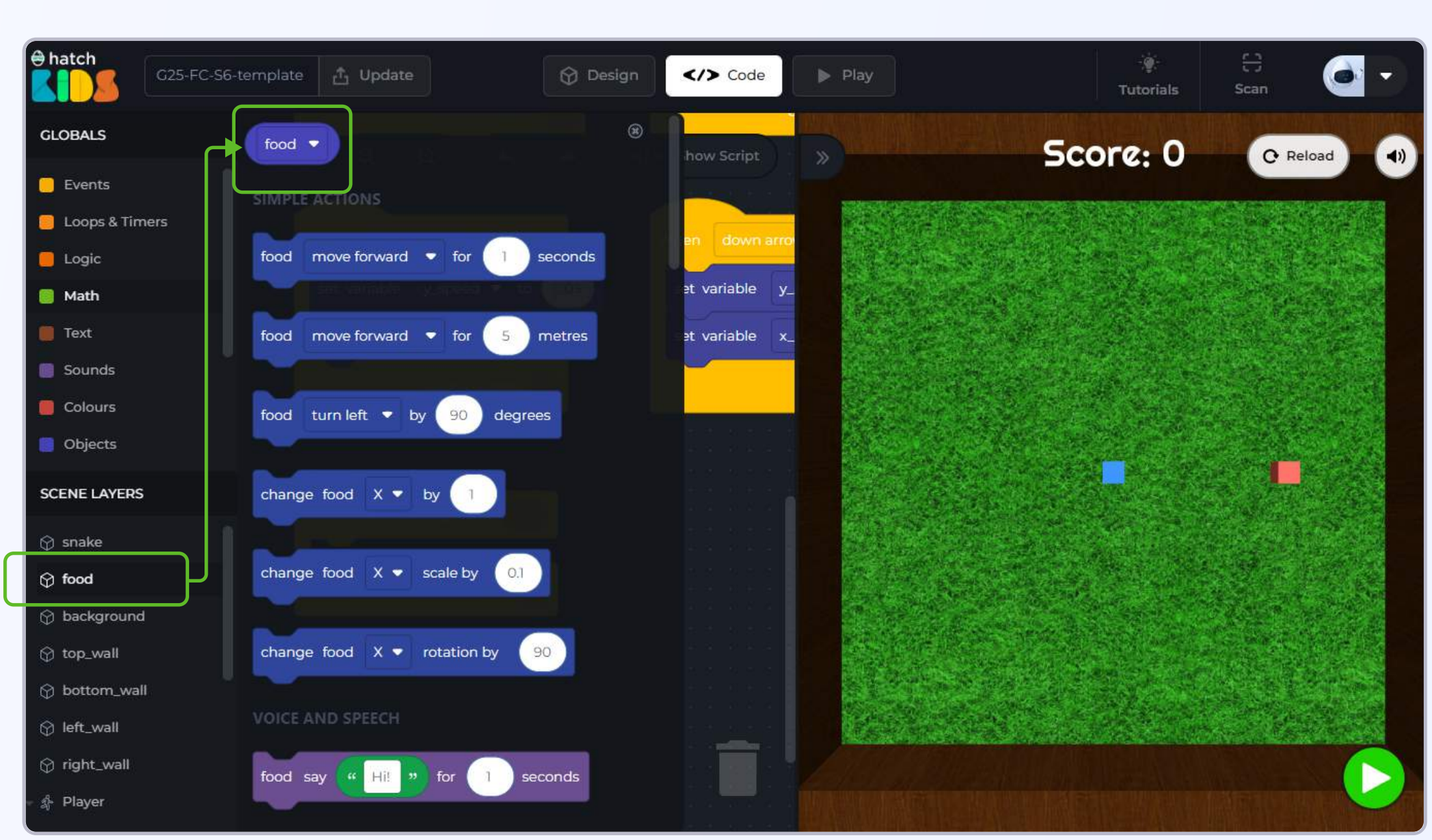
Drag out the **“when snake collides with”** block into the workspace.

Let’s understand the structure of this block.

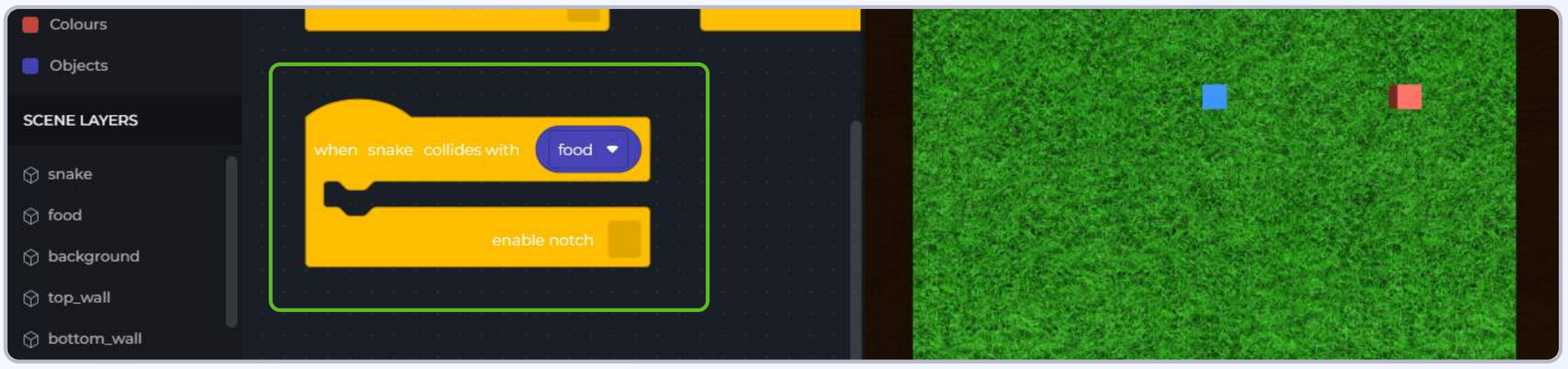


Since we want to detect collision between the snake and the food object, we need to attach the block with the name **“food”** in the empty space just beside **“when snake collides with”**

Step 2: Click on the name **“food”** in the left panel, and drag out the very first block from the list that just says **“food”.**



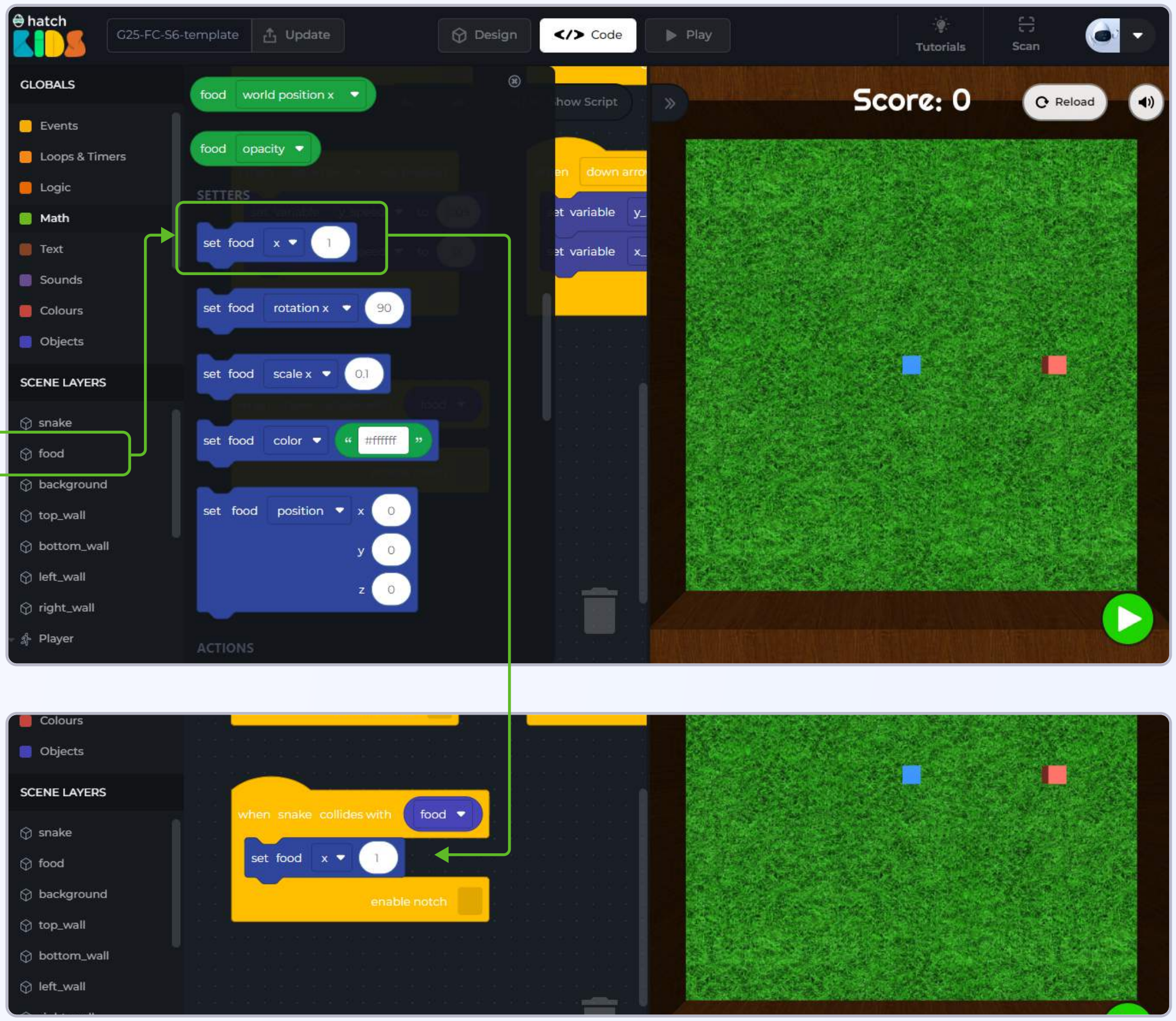
Drag out the **“food”** block, and attach it in the **“when snake collides with”** block as shown.



When snake collides with food, food moves to a new location. we need to code how will the food move to a new location.

We are going to use some new blocks for this activity.

Step 3: Click on the name **“food”** in the left panel, and scroll down the list of blocks. You will find a block called **“set food x” block.** Click and drag out the “set food x” block and attach itt inside the “when snake collides with food” block as shown



Reload the scene and Click on the green play button and to run this new code.

Press the up / down / left / right arrow keys to move the snake and make it go towards the food.

The moment the snake collides with the food, you will see the food move to a new position.

Now make the snake go towards the food again, the second time when the snake collides with the food, nothing changes in the game, the food stays where it is.

So, what's happening in the game?

The **"set food x"** block is used to modify the x-position of an object (food in this case)..

In the code, the block says, **"set food x 1". It means that, whenever this block will run, the x-position of the food object will become 1.**

So when we are playing the game for the first time and the snake collides with the food, the food object then moved so that its x-position value becomes 1.

After this, when the snake collides with the food the second or third time, the food doesn't move because the x-position of the food is already 1 now, and thus the code is trying to move the food to the location at which it is already present.

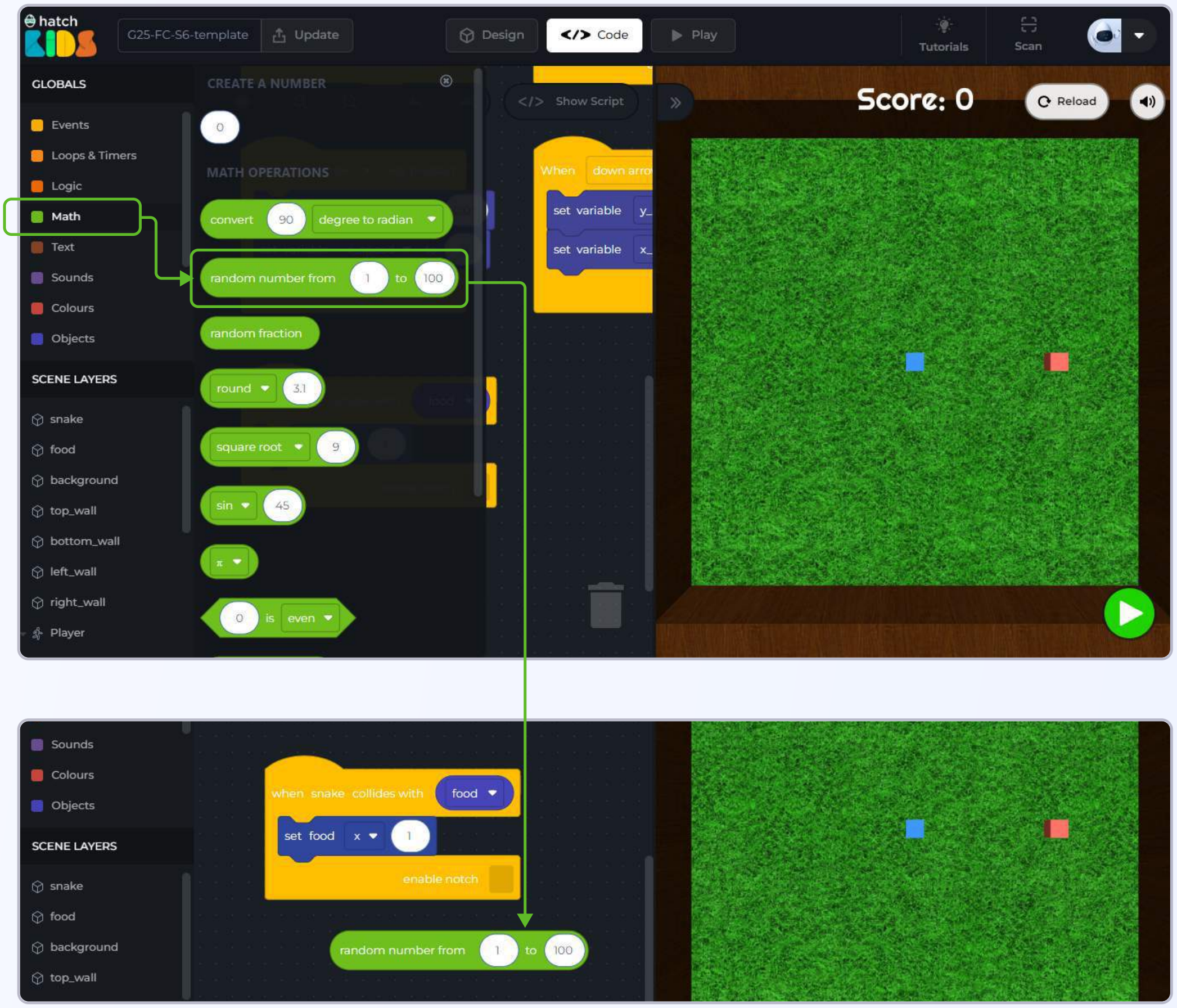
Hence, no change seems to happen in the game when snake hits the food second or third time.

So how do we ensure that the food keeps on appearing in random positions.

For this, we will use another new block called "random number".

Let's try it out. Click on the reload button to reset the scene first.

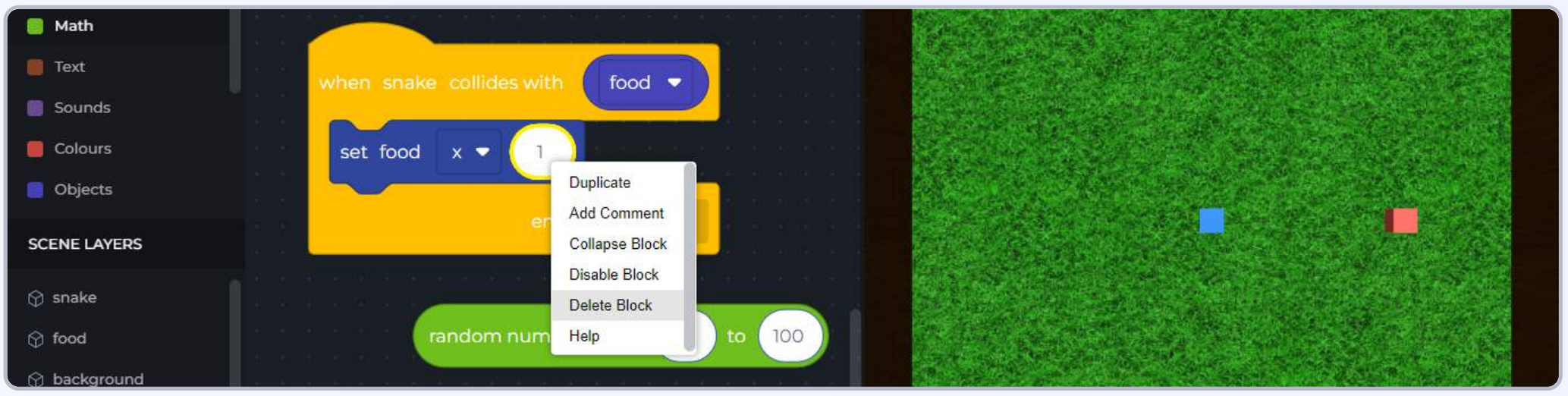
Step 4: Click on the **"Math"** section in the top half of the left panel of the workspace. In the list of blocks, you will see a block that says **"random number from 1 to 100".** Drag out that block into the workspace.

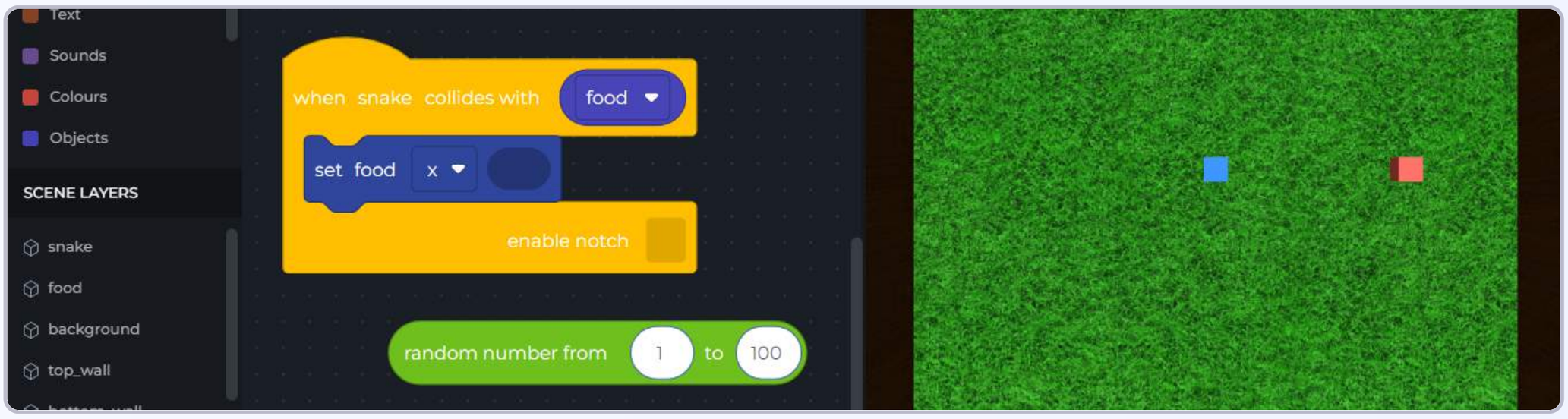


A “random number” block generates a random number in a given range. By default the block says “random number from 1 to 100”, so whenever the computer runs this block, it will generate a random number that lies between 1 and 100.

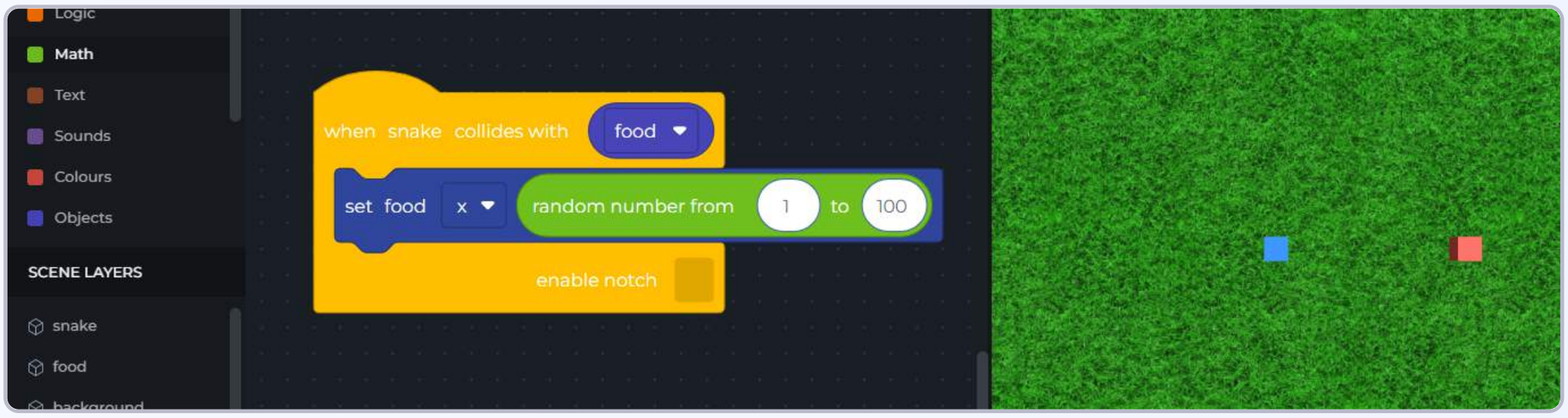
In our code we can say, whenever the snake collides with the food, set the x position of the food object to a random number” and this way, every time the snake hits the food, the food object will move to a new location.

Step 5: Right click on the number block inside the **“set food x 1”** and select the option **delete**.





Step 6: Attach the random number block inside the empty space of “set food x” block



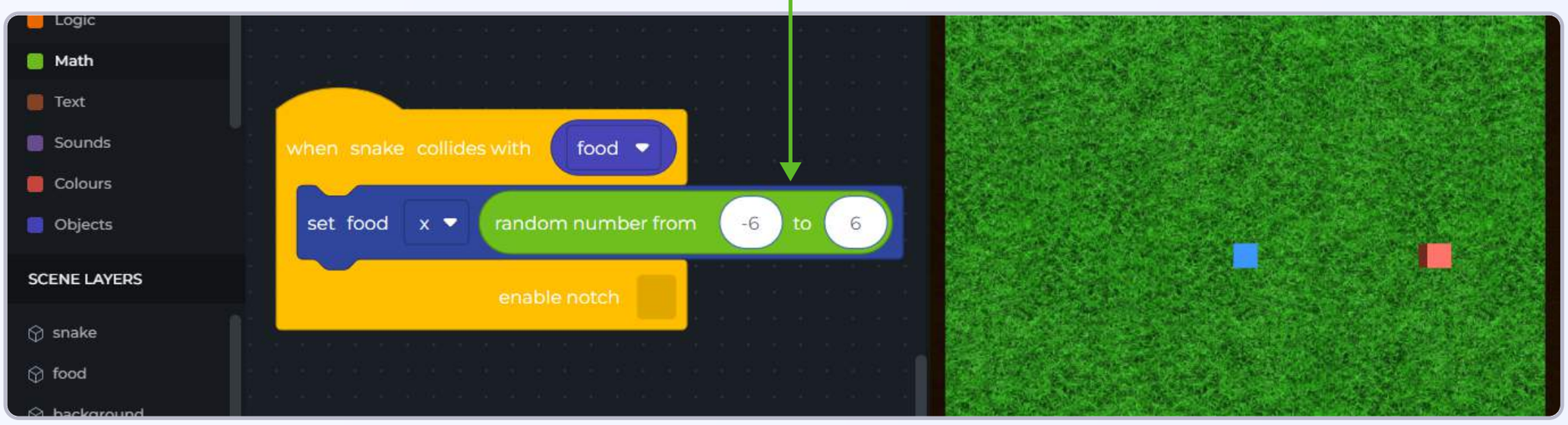
So the block now says,

“Anytime the snake hits the food, the computer will generate a random number between 1 and 100, and that will become the new x-position of the food object”.

The size of your game area is currently small. When we ask the computer to generate a random number between 1 to 100, and move the x-position of the food to that number, it means that sometimes the x-position of the food may become 99 or 100. And that will end up moving the food object out of the game arena.

In current situation, the game arena has a size of 12 units (from the center of the arena 6 units on left, 6 units on right, 6 units on up, and 6 units of space down)

Step 7: Change the values inside the random number block to say **“from -6 to 6“**



Click on the Green Play button and run the code.

Move the snake with your arrow keys, and now you will see that anytime the snake hits the food object, the food keeps moving to a new position.

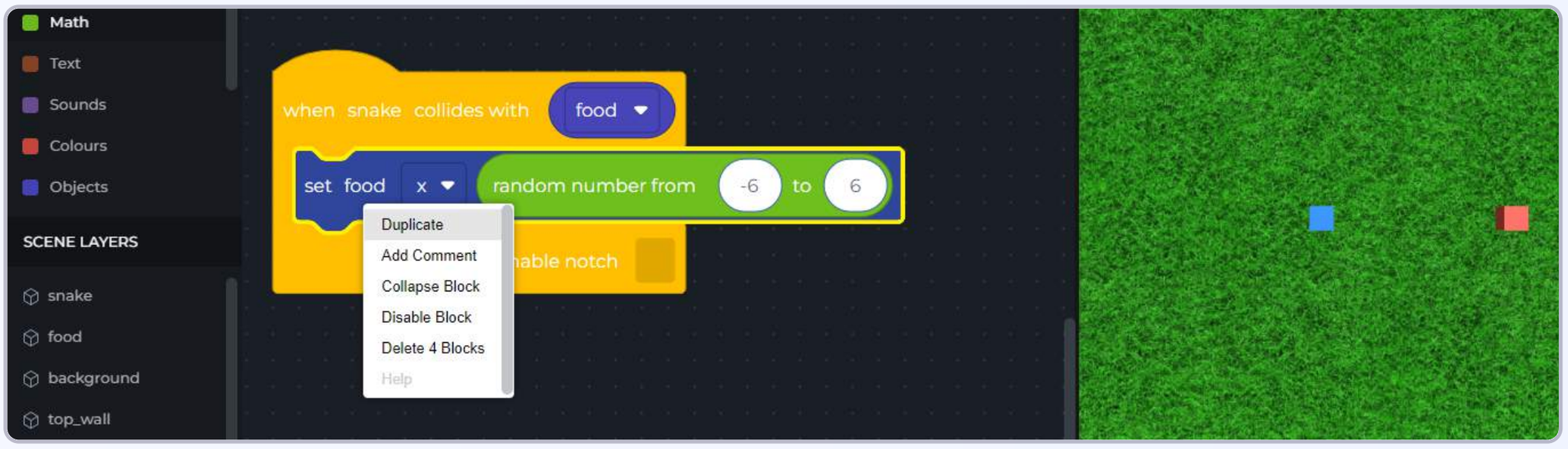
But there is one flaw here. You will notice, the food only moves to a random position in the left or right direction. The food is not moving up or down.

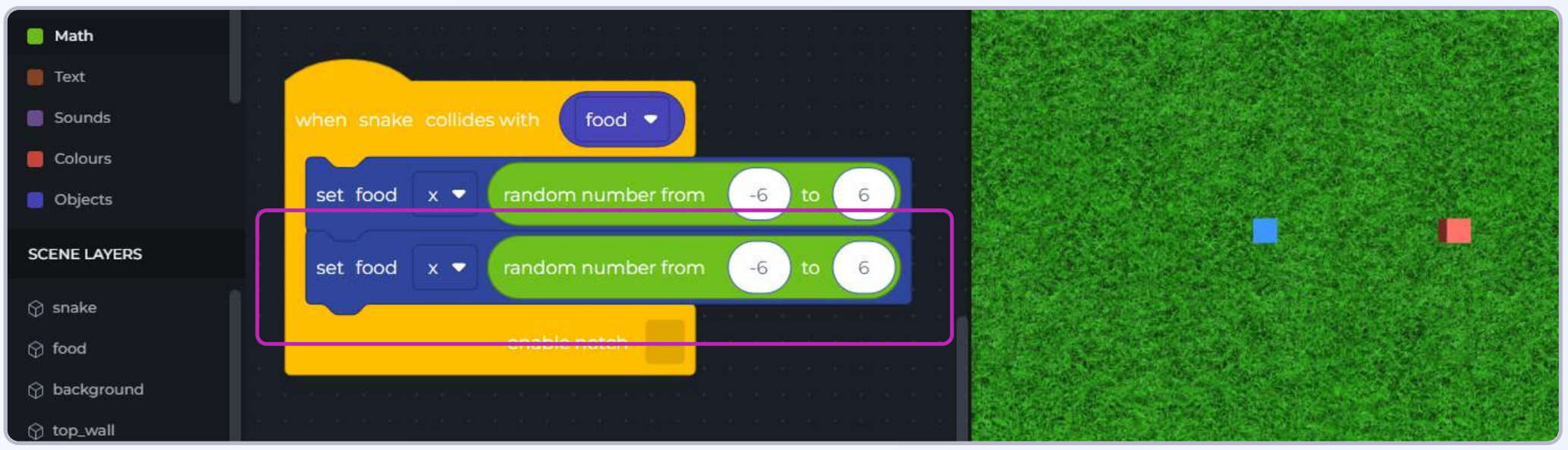
That's because in our code currently we are only changing the x-position of the food. The y-position of the food stays ths same throughout.

Just like we change the x-position of the food, let's change the y-position of the food as well.

Reload your game.

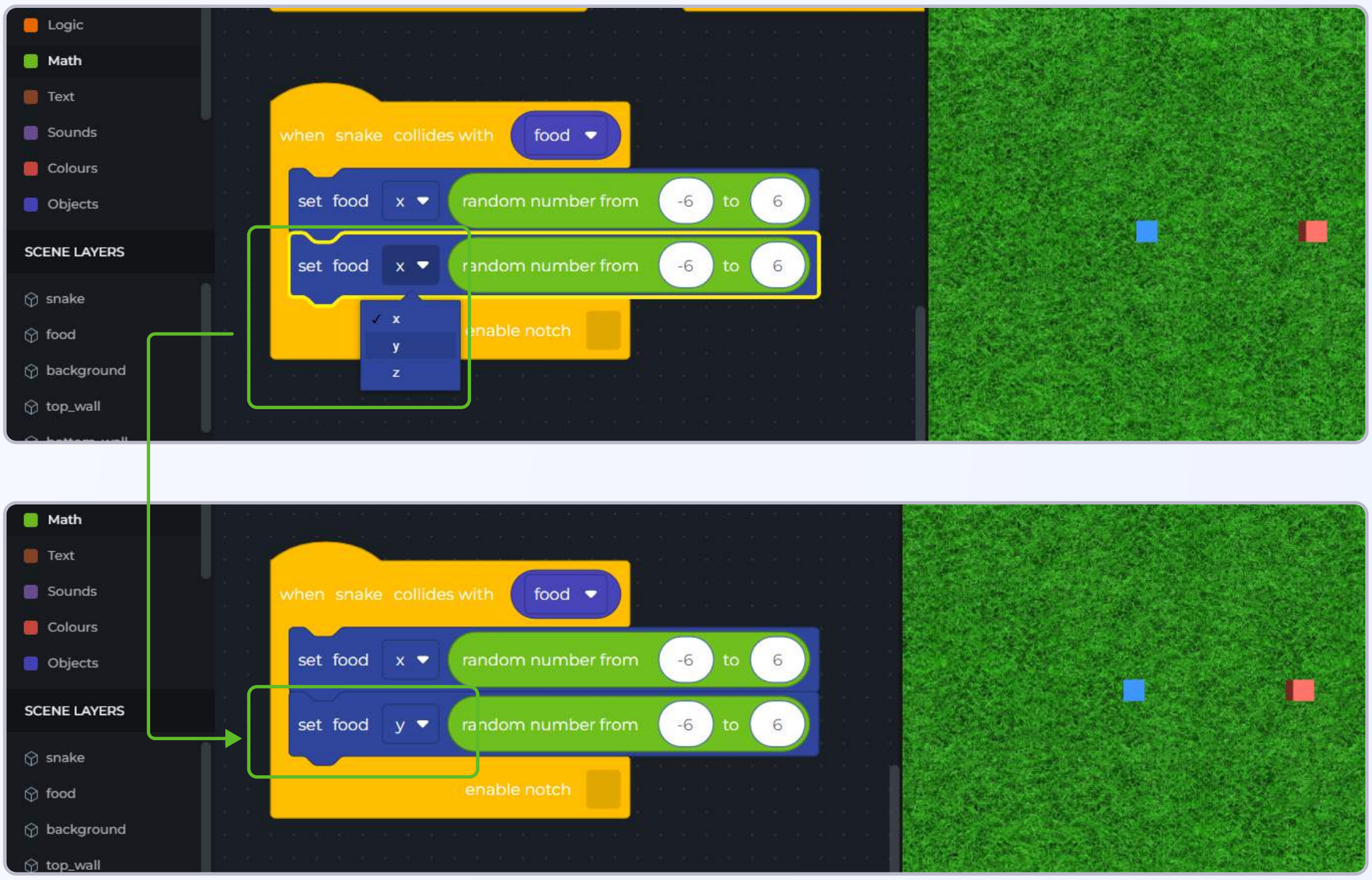
Step 8: Right click on the **"set food x"** block and **duplicate** it. Attach the new copy just below the first one as shown





Step 9: Click on the **drop down button** around the character **"x"** in **"set food x"** block, and select the option **"y".**

So your new block would now read as, **"set food y random number between -6 to 6"**



Now as per your code, anytime the snake hits the food, the x-position of the food will be set to a random nu,ber between -6 and 6. Also the y-position of the food will be set to a different random number between -6 and 6.

Click on the green play button and run the code.

Move the snake towards the food, and when the snake hits the food, you will see the food appear in a completely new position.

🎉🎉🎉🎉🎉 **CONGRATULATIONS!!! You just completed the second part of the game - detecting collisions between food and snake and move food to random positions** 🎉🎉🎉🎉

Click on reload to reset you scene.

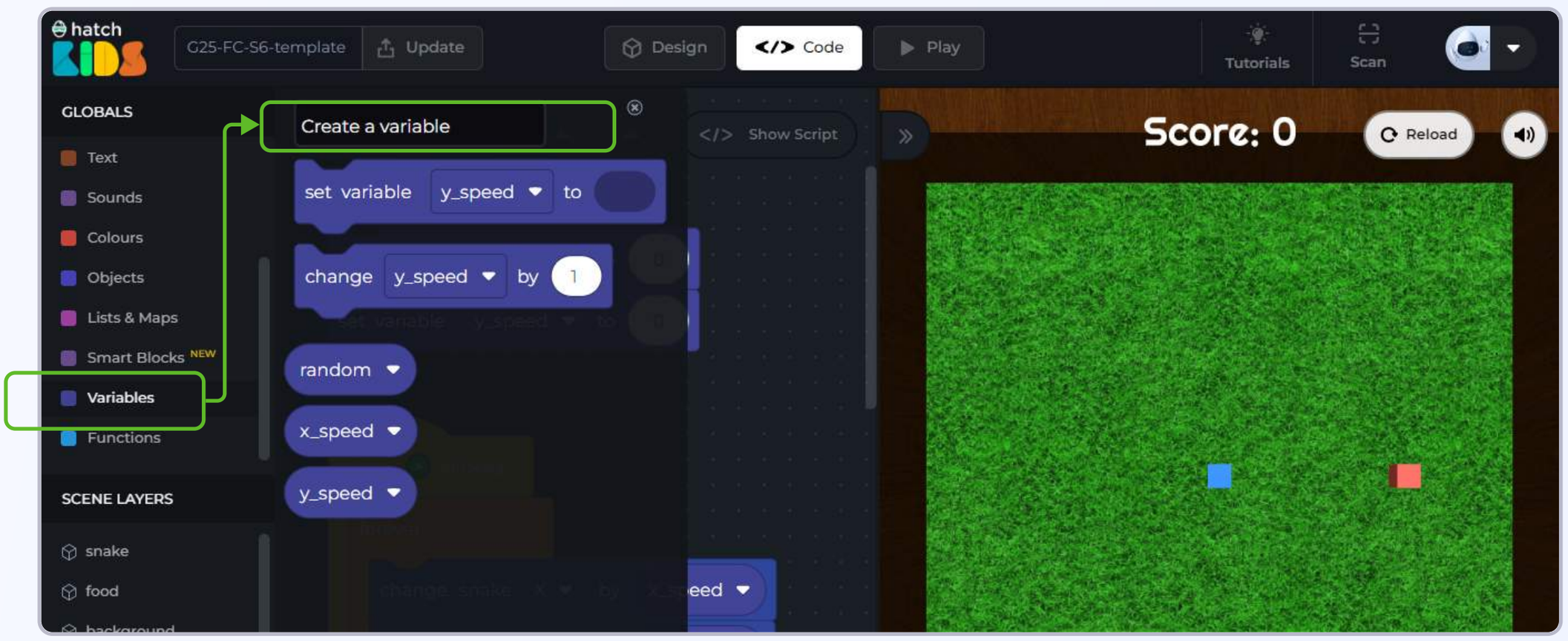
Let’s now add score to our game. Anytime the snake hits the food, the score can increase by 1.

Objective No. 4: Adding score to your game

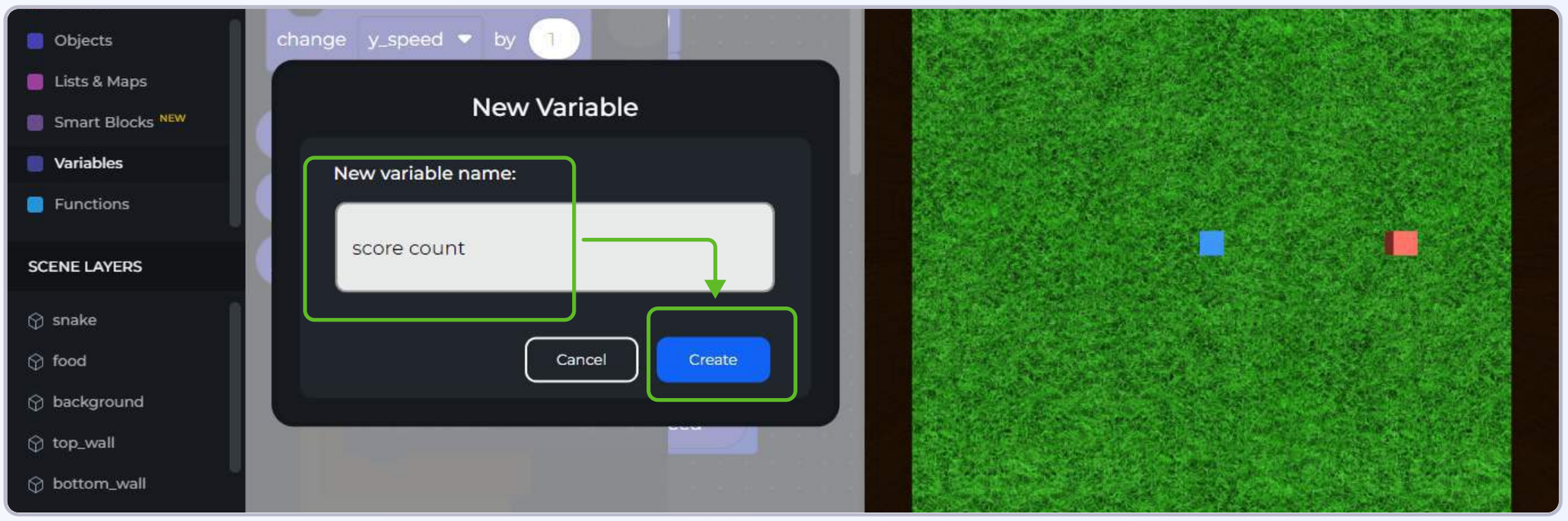
We are going to add score to the game such that anytime the snake eats the food, the value of score increases by 1.

To do this, we first need to create a variable that will store the value of your score. We can call this variable **“score count”**

Step 1: Click on **“variables”** in the left panel, and click on the **“create a variable”** button.

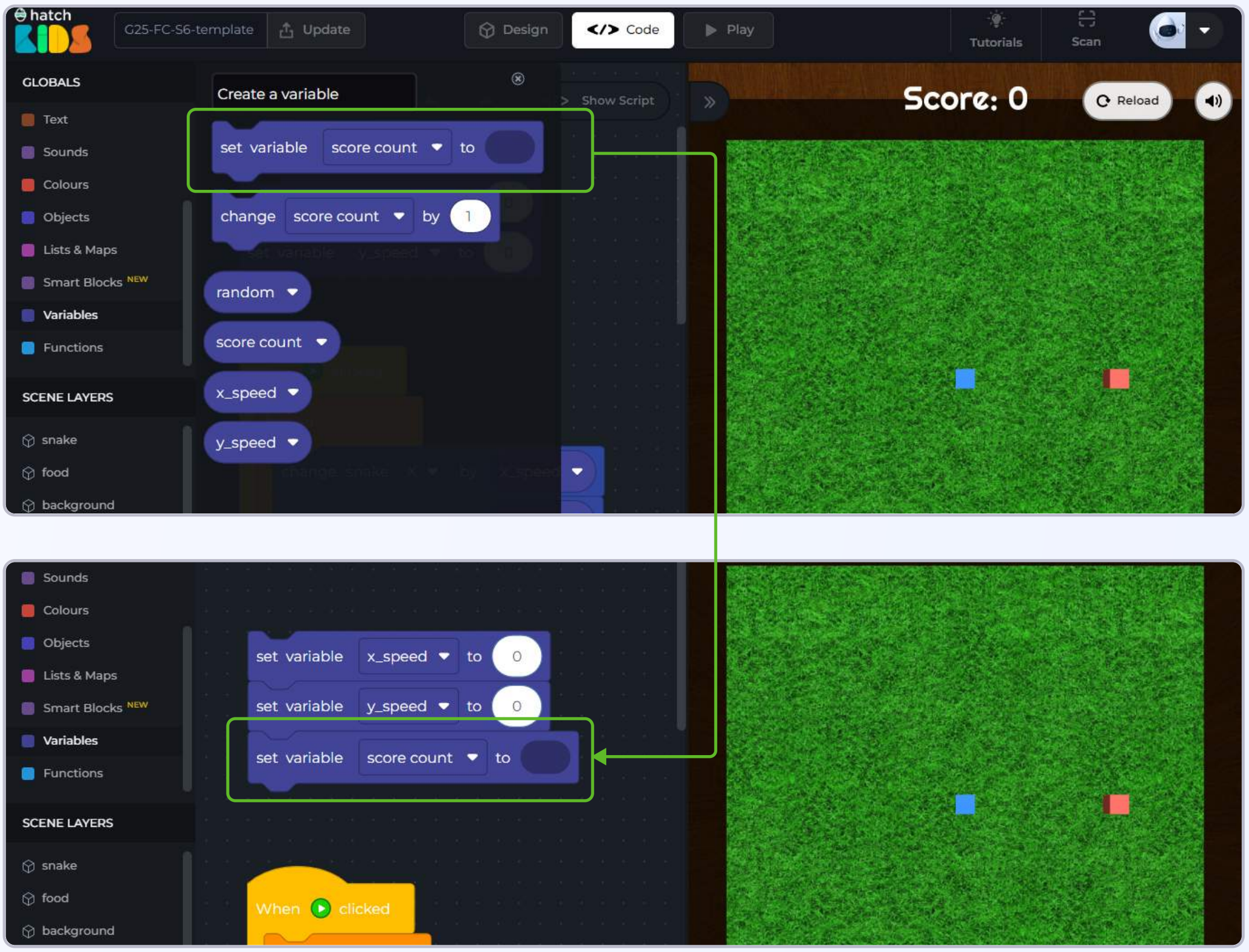


Step 2: A small window will appear asking you to give a name to your variable. Type in **“score count”** and click on the button create.



In the variables panel, you will now see block for the “score count” variable.

Step 3: When the game starts, the value of score should be “0”. Drag out the **“set variable score count to”** block and attach it below “set variable y_speed to 0” block as shown,

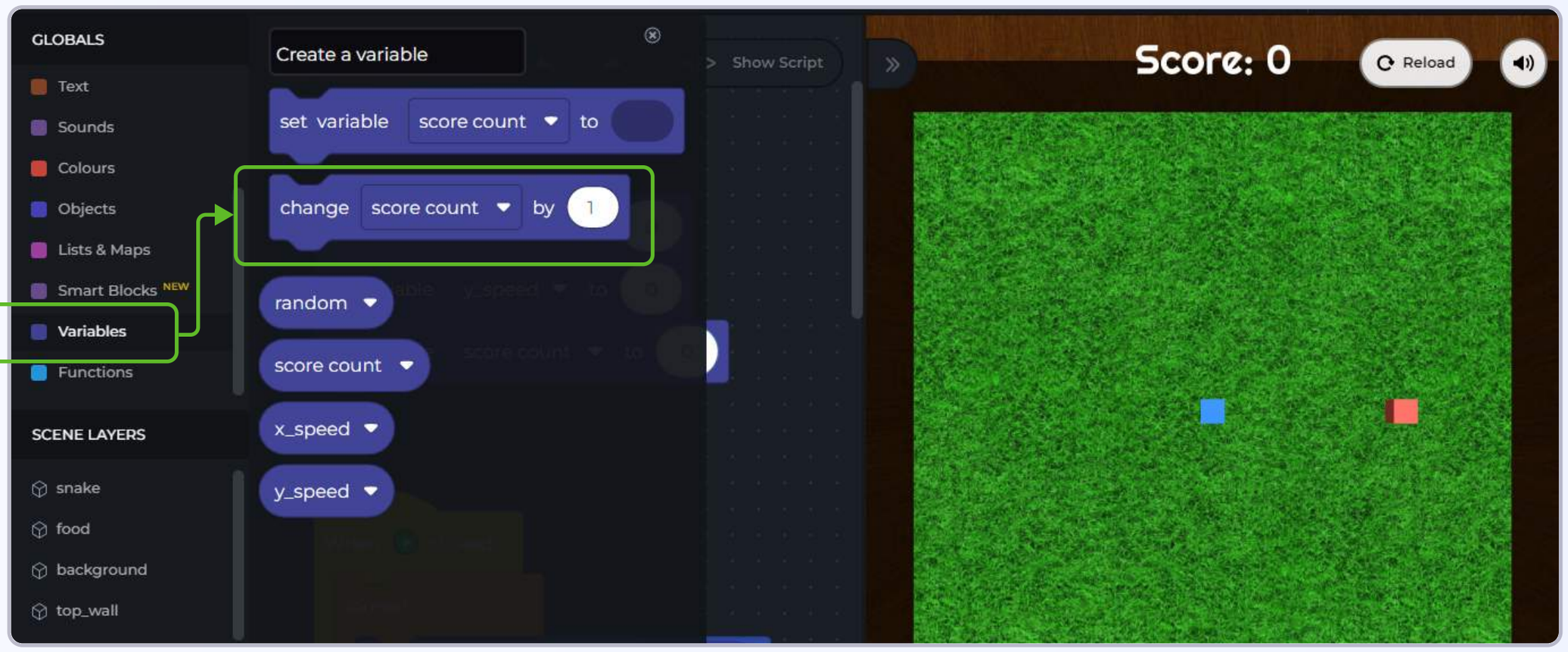


Step 4: Duplicate the number 0 block from “set variable y_speed to 0” and attach it inside the “set variable score count” block.

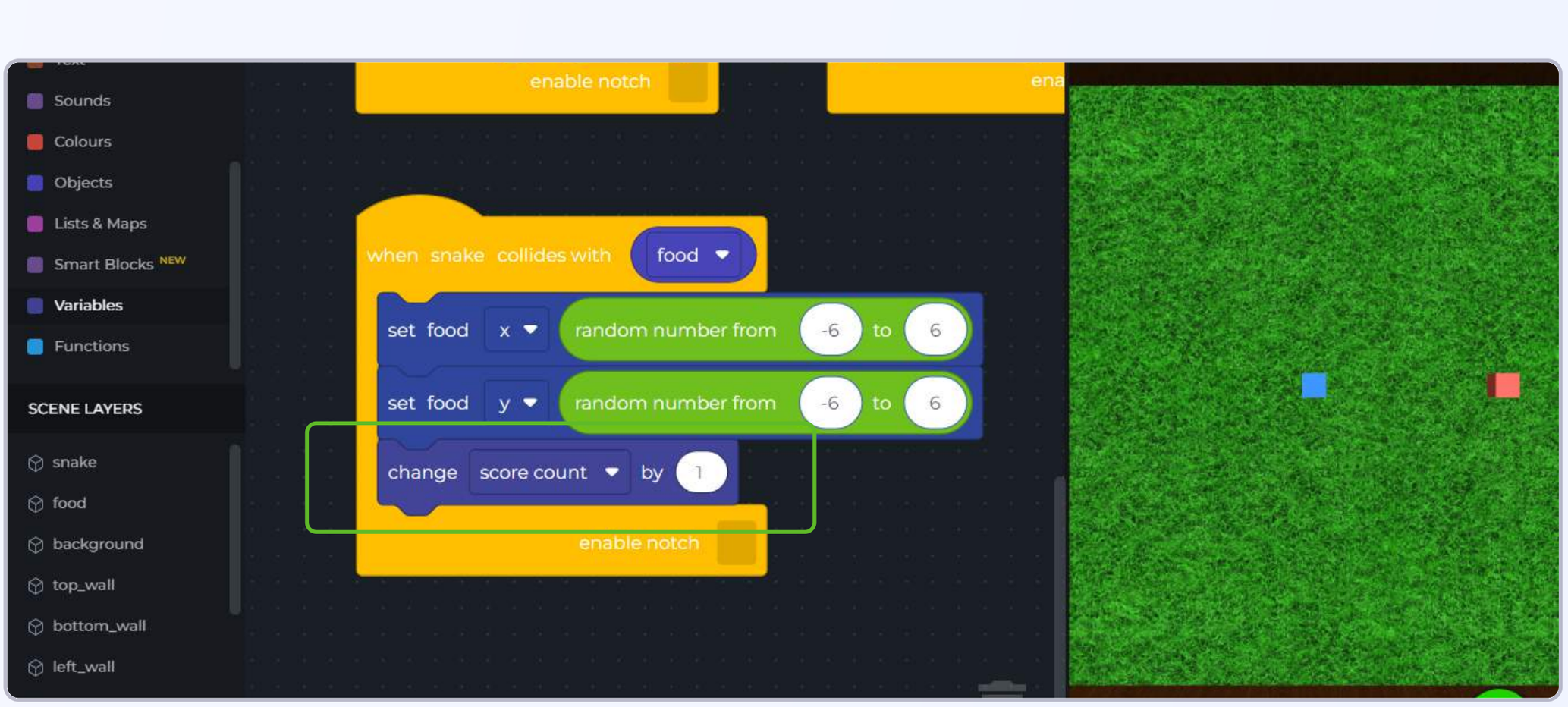
The new block should now read **“set variable score count to 0”**

The value of score count should increase every time the snake hits the food.

Step 5: In the variables section you will find a block called **“change score count by 1”**



Step 6: Drag out the **”change score count by 1”** block and attach it inside the **“when snake collides with food”** block

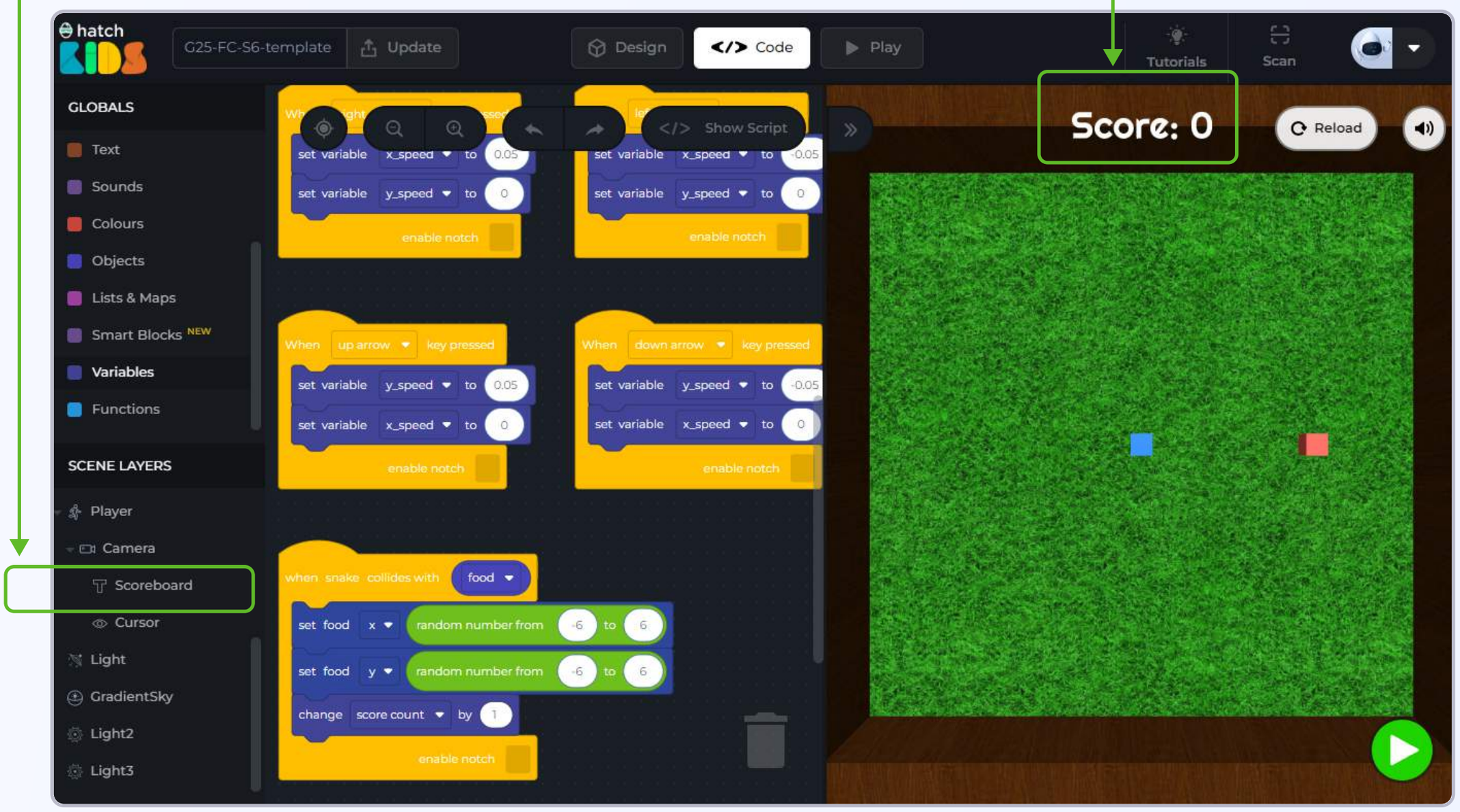


If you run the code right now, you wont be able to see the score anywhere in the game. That’s because even though the variable score count is saving your score value, we have not added the code to display the score in the game.

Let’s display the score in the game. Click on the reload button in the game and reset the scene.

In the game you will see that there is a text that says **“Score: 0”**

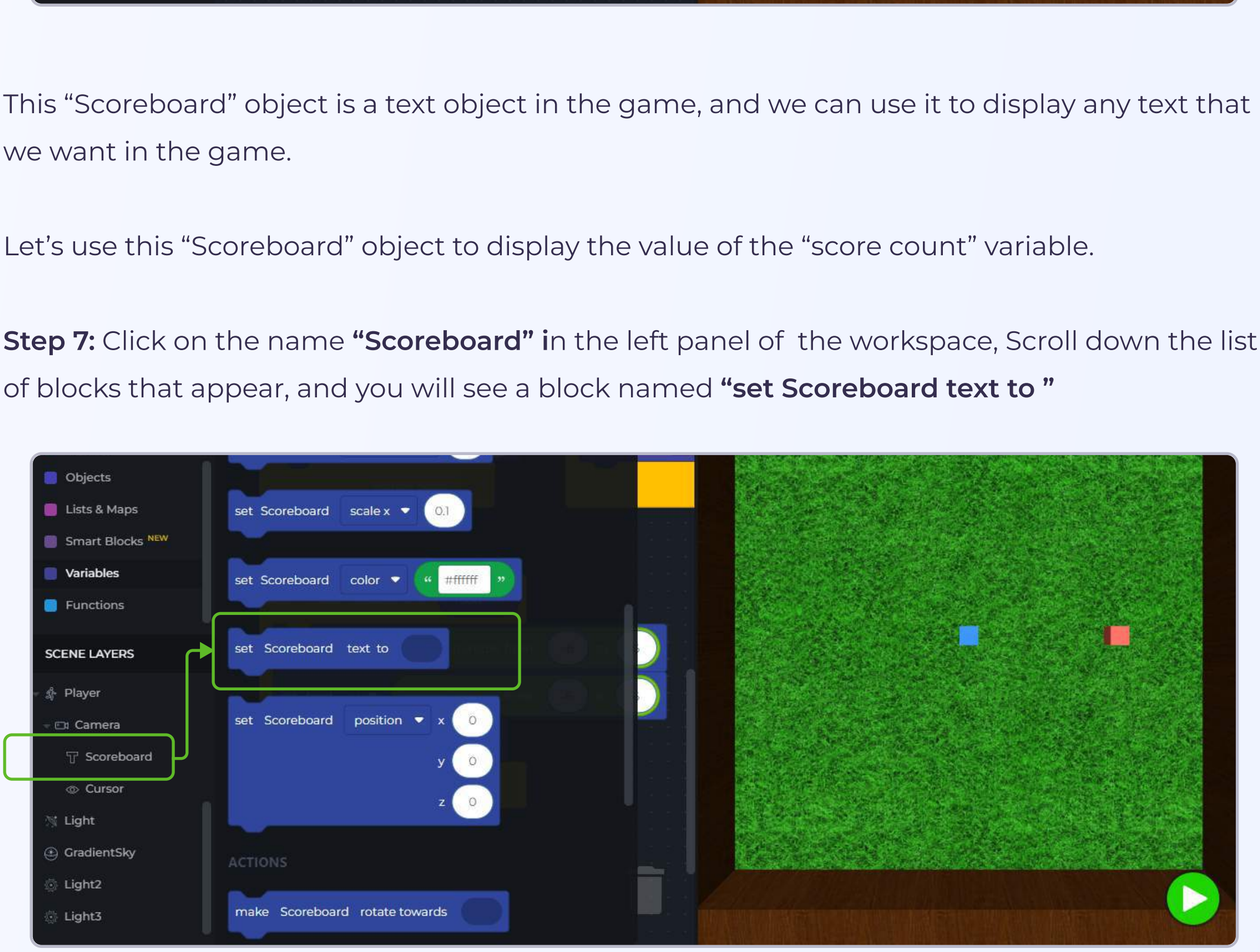
It is an object by the name **“Scoreboard”** in the game, just like the snake and food objects.



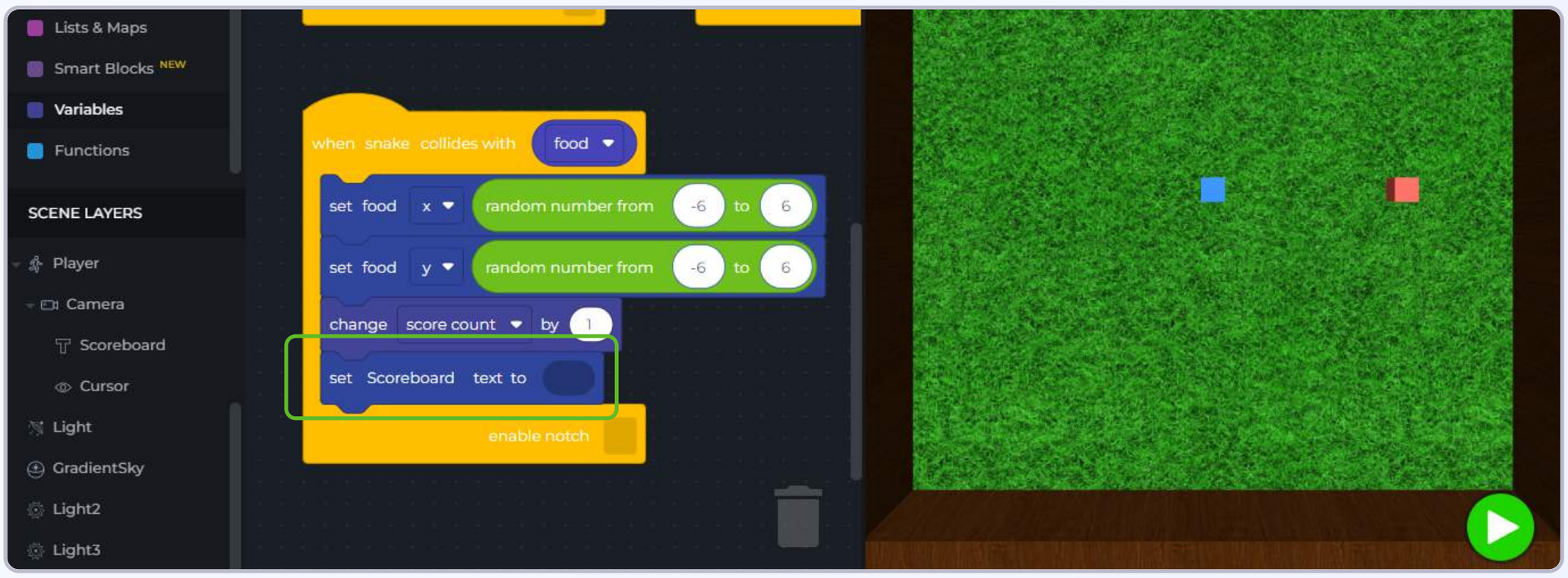
This “Scoreboard” object is a text object in the game, and we can use it to display any text that we want in the game.

Let’s use this “Scoreboard” object to display the value of the “score count” variable.

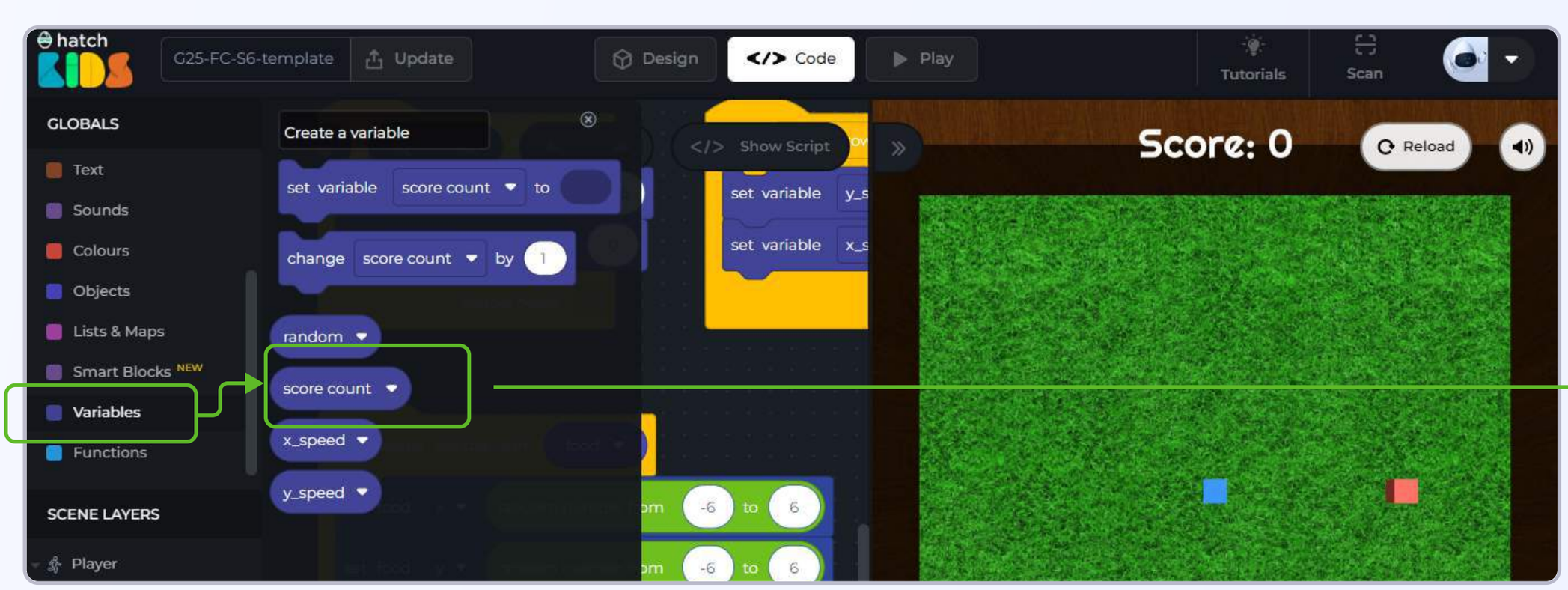
Step 7: Click on the name **“Scoreboard” i**n the left panel of the workspace, Scroll down the list of blocks that appear, and you will see a block named **“set Scoreboard text to ”**



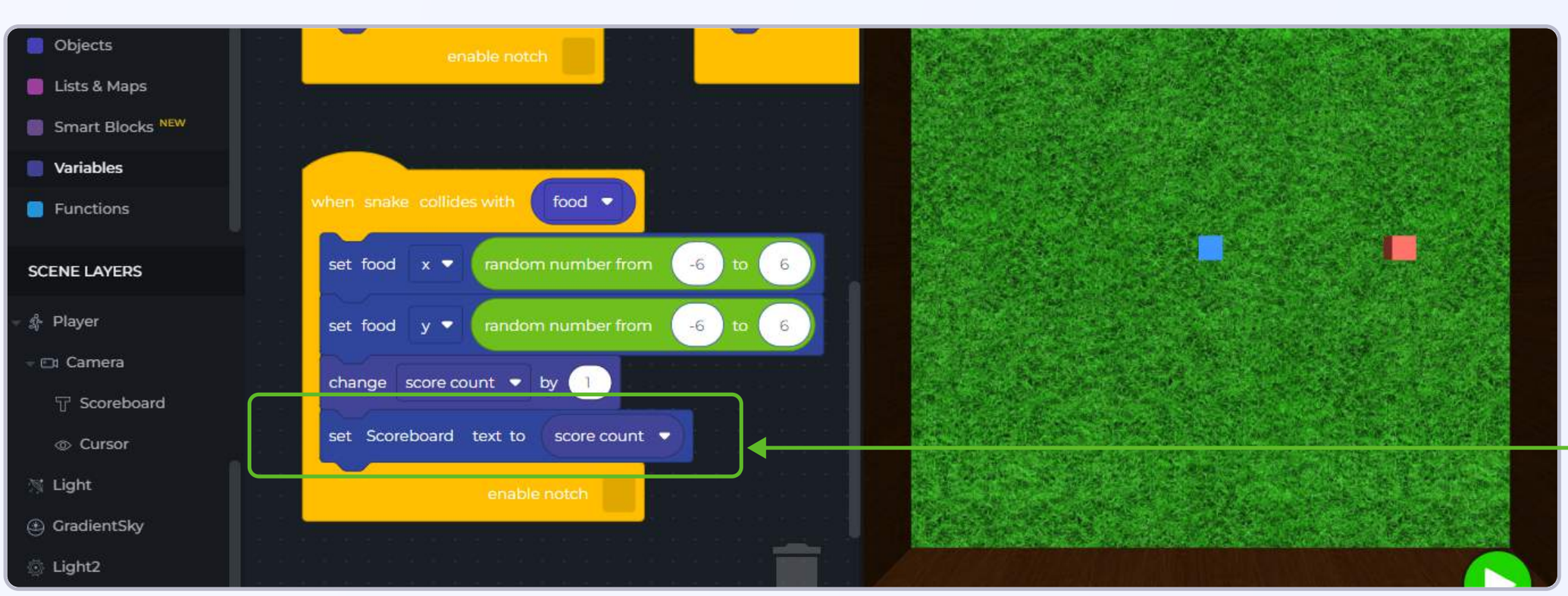
Step 8: Drag out the **“set Scoreboard text to”** block and place it inside the **“when snake collides with food”** block



Step 9: Click on the **“variables”** section in the left panel and drag out the block that says **“score count”**

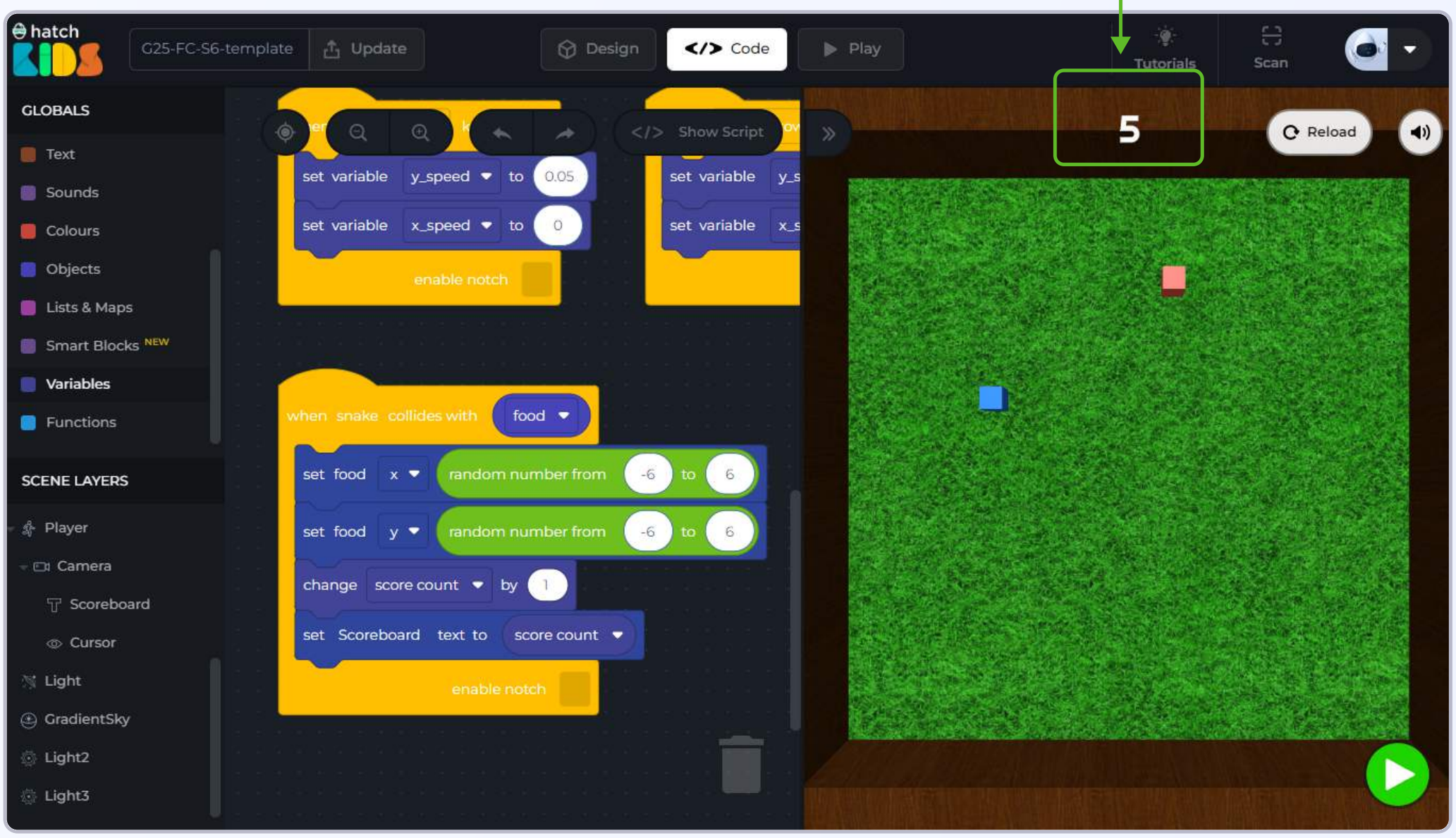


Step 10: Place the **“score count”** block inside the **“set Scoreboard text to”** block



Click on the green play button and play the game.

You will see that the text that was earlier saying “Score: 0”, now displays a number that increases every time snake hits the food.



Click on reload to stop the code and let’s understand whats happening here.

The “set Scoreboard text to” block can be used to display any text in the game.

When we added the “score count“ block inside the “set Scoreboard text to” block, the final block reads as “set Scoreboard text to score count”.

The code is telling out computer to set the text in the game to the value if the variable “score count”, and as the vaue of score count increases, you can see it increasing in the text as well.

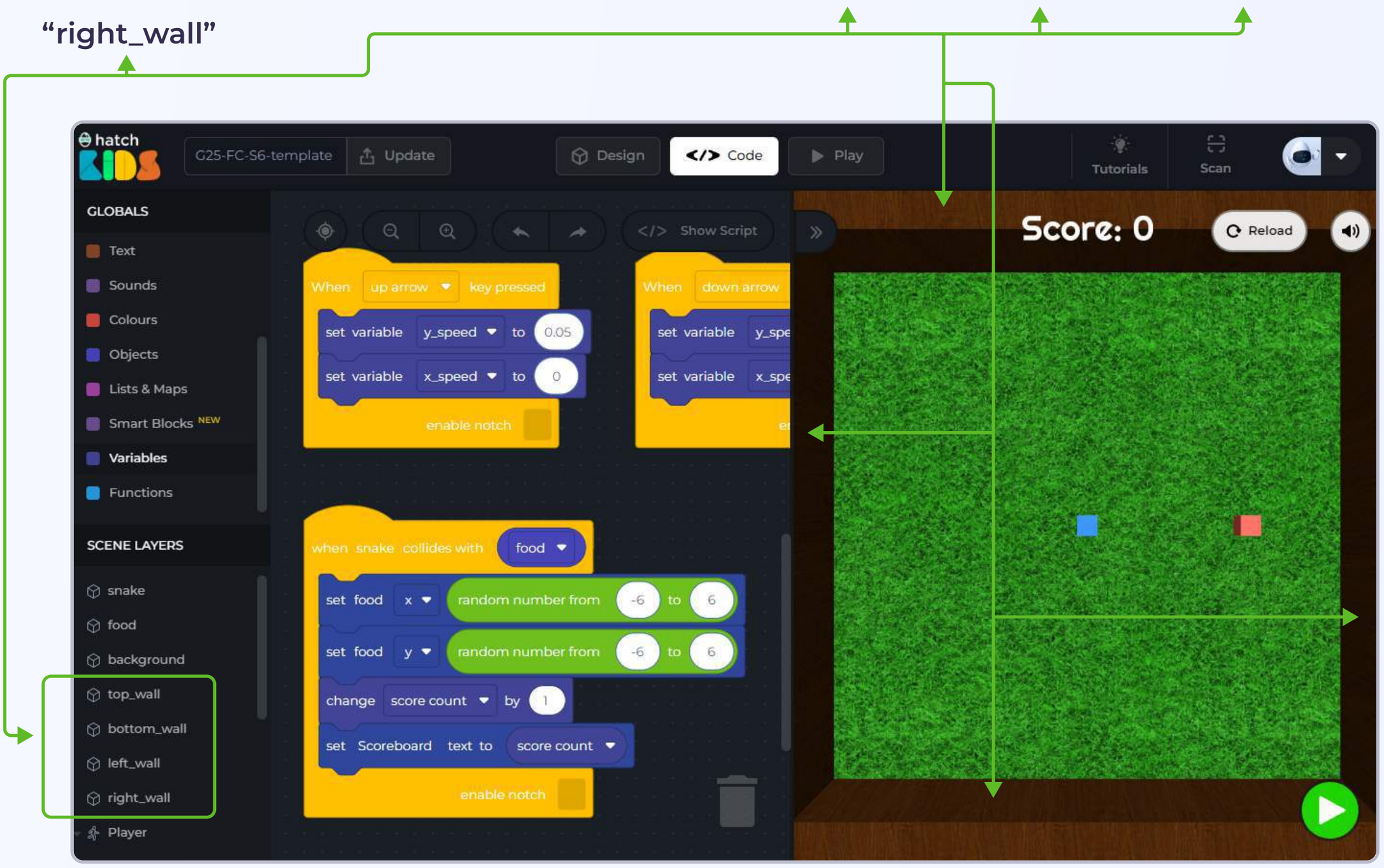
🎉🎉🎉🎉🎉 **CONGRATULATIONS!!! You just completed the third part of the game - counting and displaying score** 🎉🎉🎉🎉

The last part of the game is to add the logic so that the game stops when the snake hits the walls in the game.

Objective No. 5: Adding the game over logic

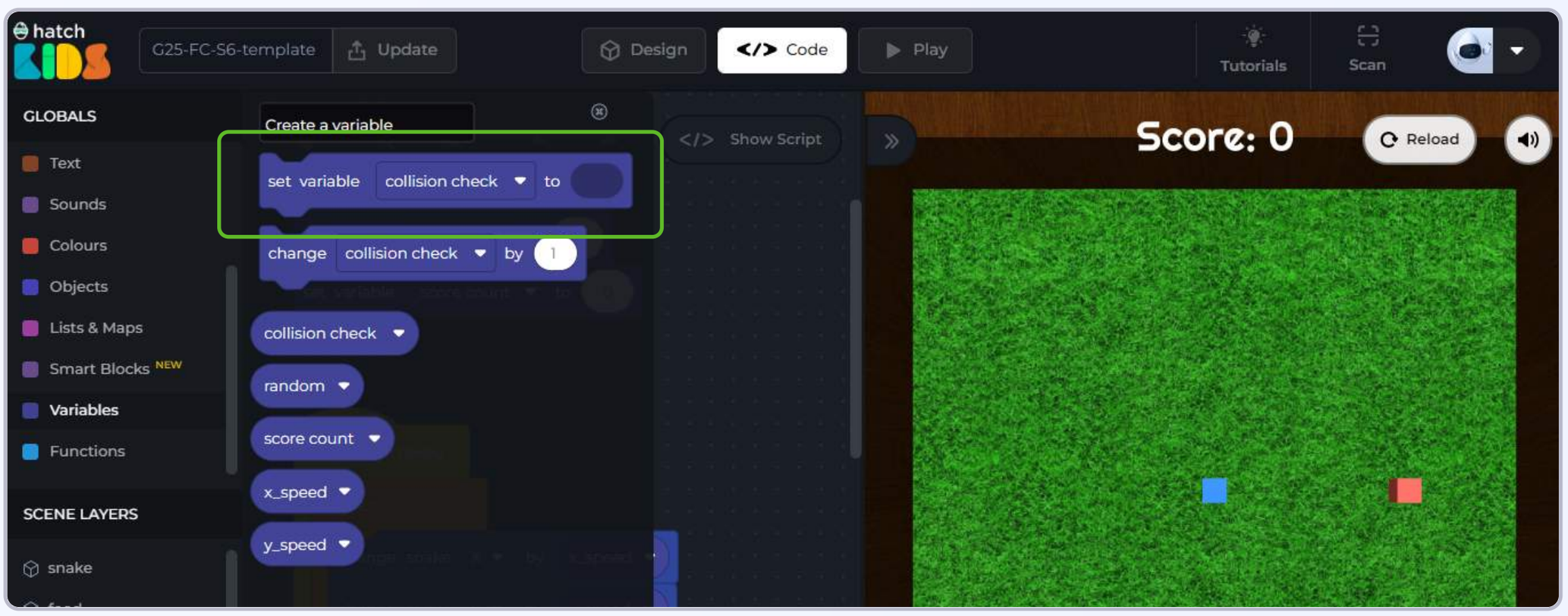
If you look closely in the game, there are 4 walls surrounding the snake in the game.

In the left panel as well you can see their names as **“top_wall”**, **“bottom_wall”, “left_wall”** and **“right_wall”**

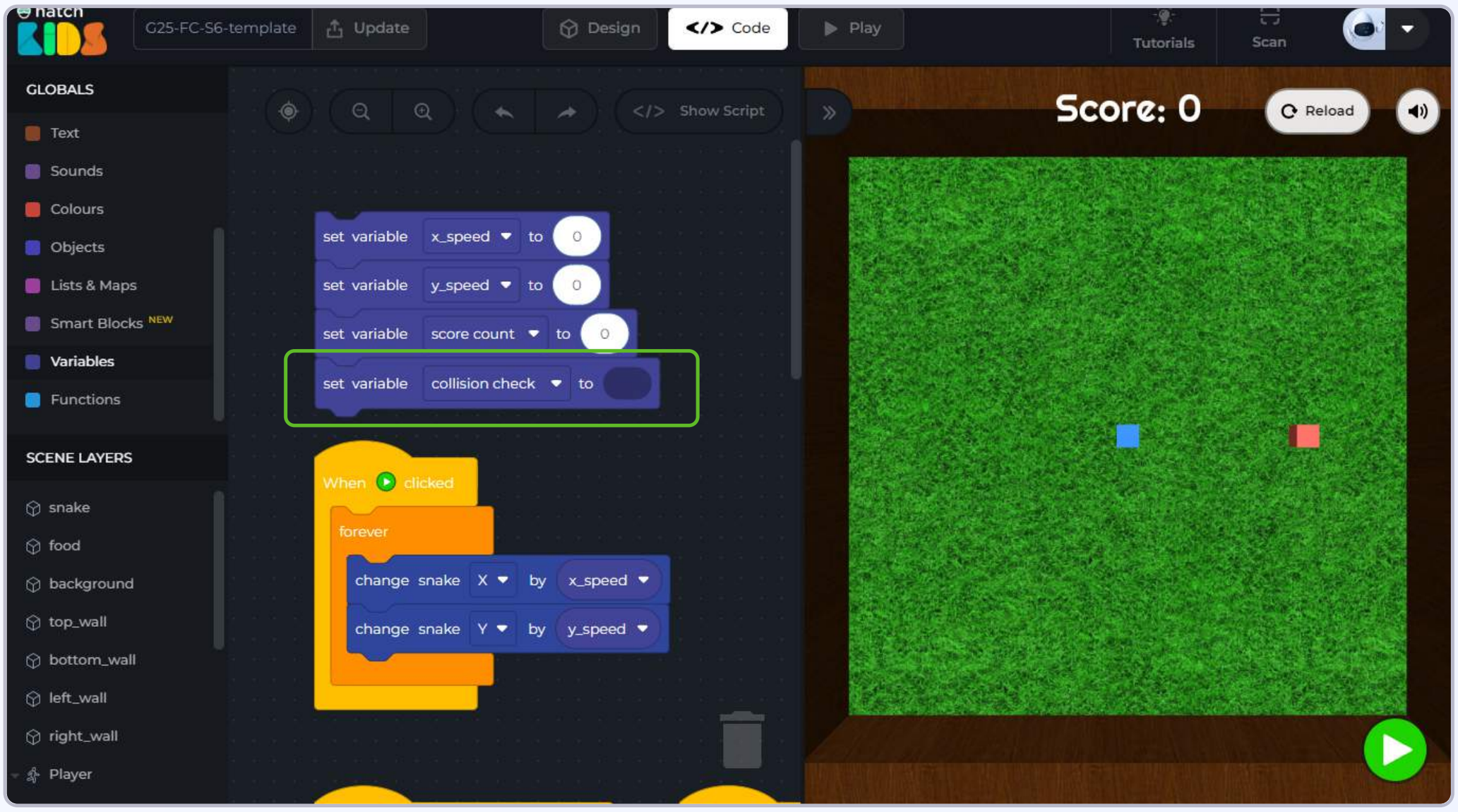


We have to write the code such that, when snake hits any one of these 4 walls, the game is over, meaning that, you wont be able to move the snake using your keyboard buttons after the game is over.

Step 1: We will start by creating a variable called **“collision check”.** Follow the usual steps of creating a variable, and when it asks you for a name for the variable, write “collision check”.

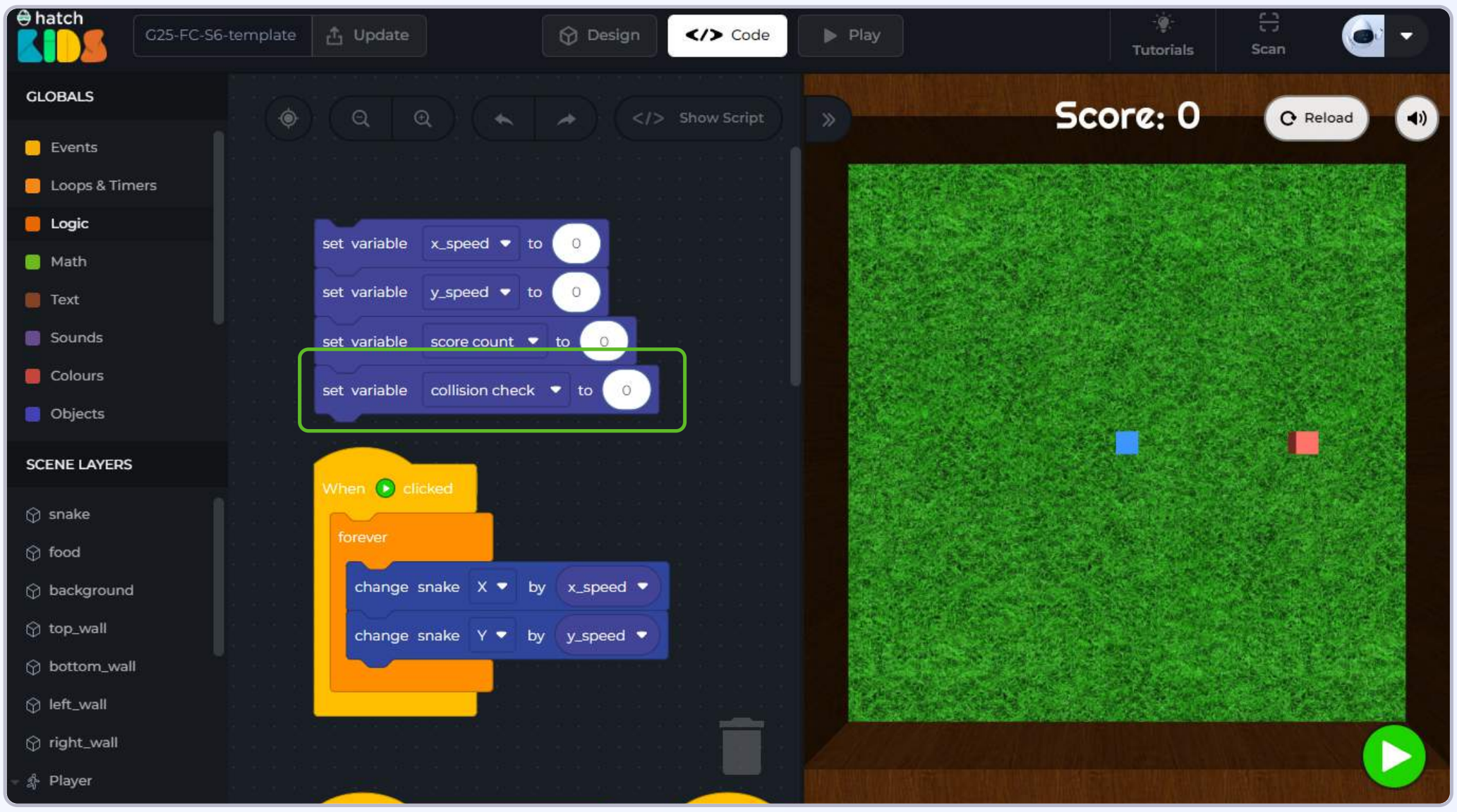


Step 2: Click and drag out the block that says **“set variable collision check to”**, and place it inside the workspace as shown



We need to give the variable “collision check” a value.

Step 3: Duplicate the number “0” block from above and place it in the “set variable collision check to” block. So we are saying, at the beginning of the game, the variable “collision check = 0”



Let’s understand the logic that we are going to implement for “game over”.

We have created a variable called “collision check” and at the start of the game
collision check = 0

When snake hits any one of the walls in the game, we can set collision check = 1.

So if, collision check = 1 at any point in the game, it means that the game is over, and that means we will not be able to move the snake with our keyboard buttons anymore.

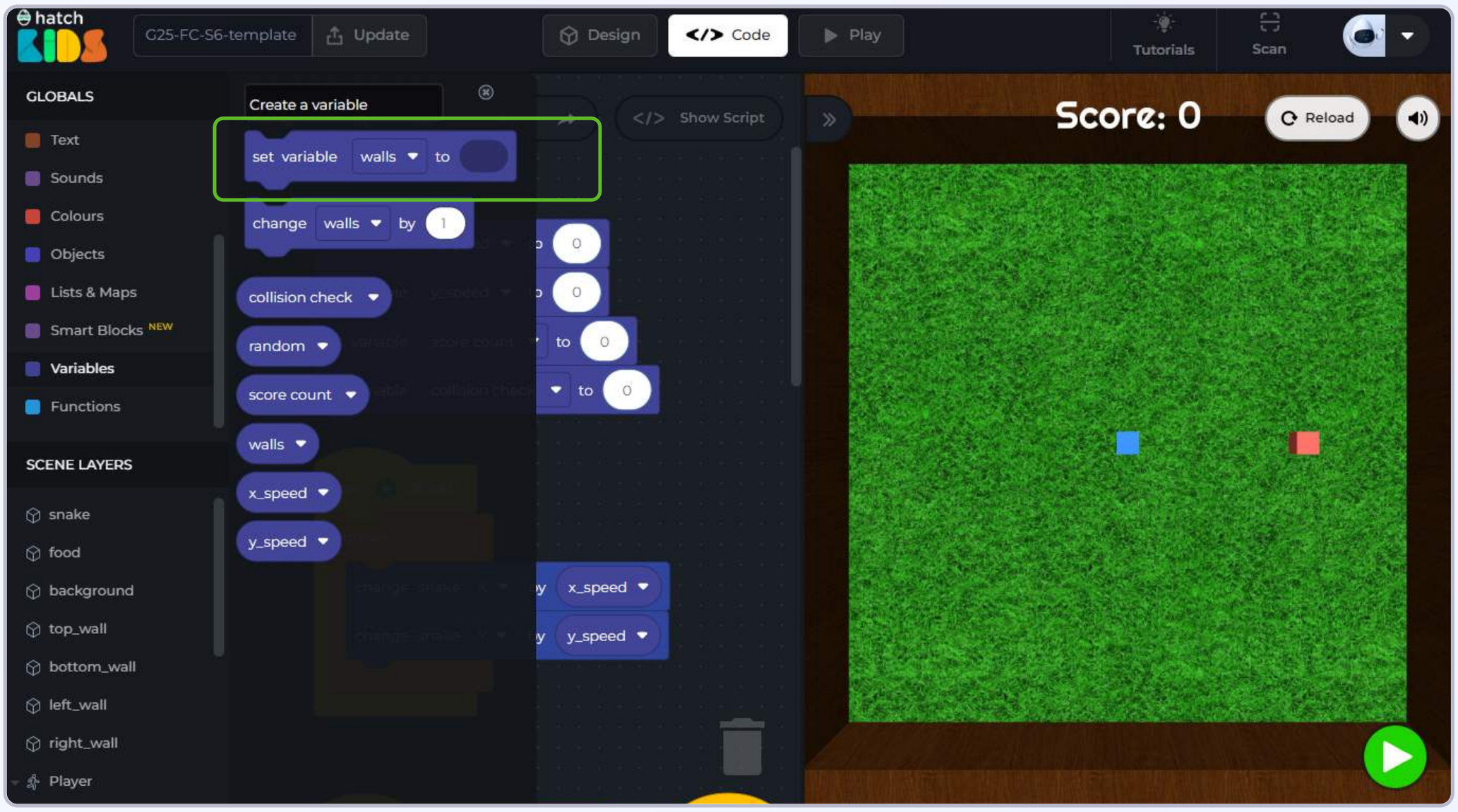
The code for moving the snake is written inside the “when green play button” block, where we are saying, that snake should forever move in x-direction with the value of x_speed variable, and in the y-direction with the value of y_speed variable.

So to implement the game over logic, all we need to do is add an if-condition that checks the value of collision check variable. If the collision check variable = 0, then snake will keep moving otherwise snake will not move.

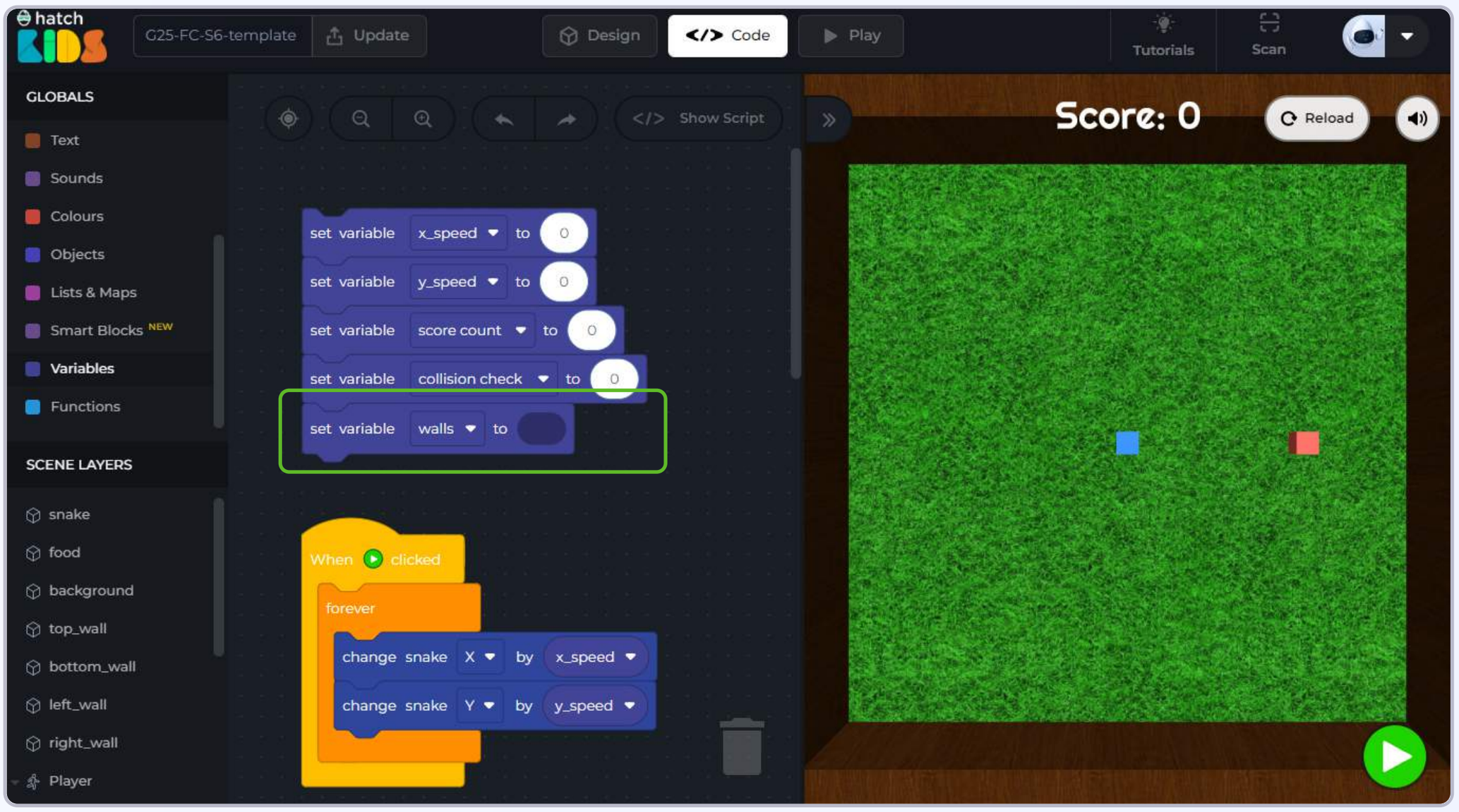
Let’s implement this logic.

We can start by **creating a variable** called **walls** and **storing a list of all the wall objects** inside it.

Step 4: Click on the variables section in the left panel and create a variable by the name “walls”.

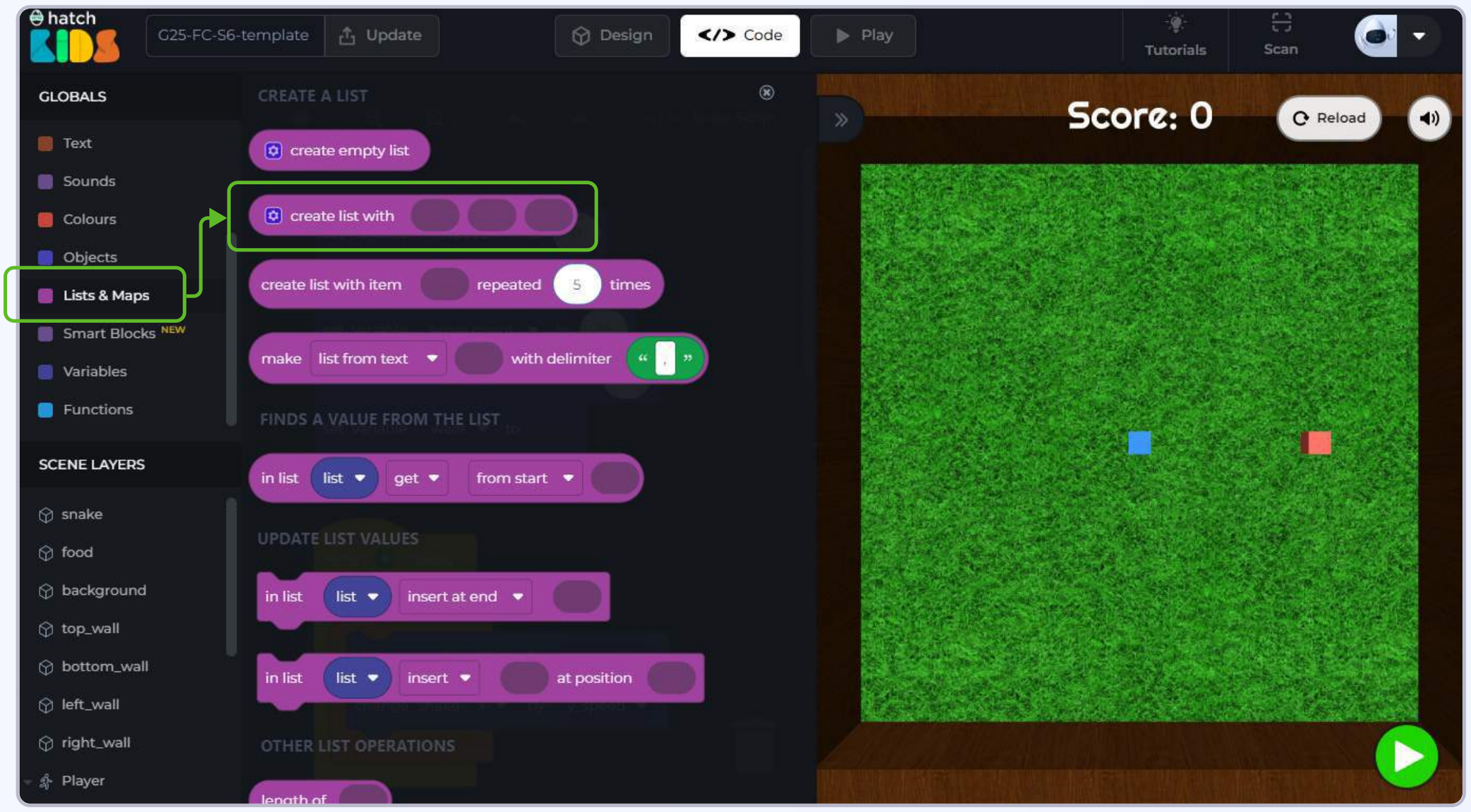


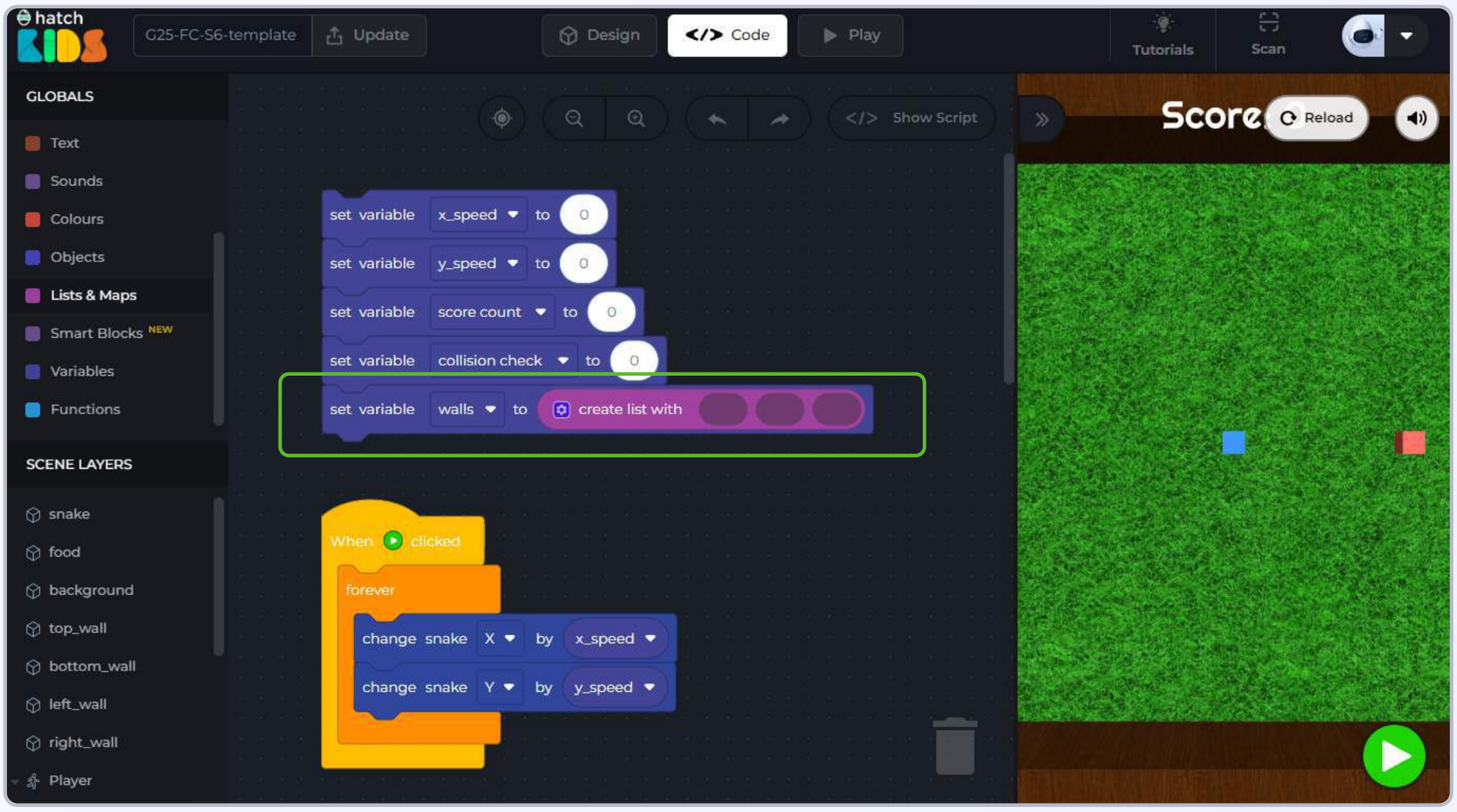
Step 5: Click and drag the “set variable walls to” block inside the workspace and place it as shown



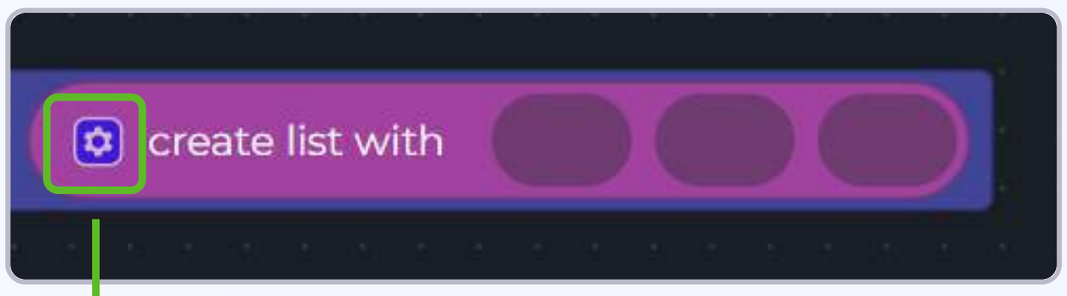
We are going to store a list of all 4 wall objects --- top_wall, bottom_wall, left_wall, and right_wall --- in the variable walls.

Step 6: Click on **“List & Maps”** category in the left panel, and drag out the block that says **“create list with”**, and place it inside the **“set variable walls to”** block.





As you can see the “create list with” block has 3 empty spaces. This is where you can add the values that you want to store in the list (in our case we will store the names of all the four walls).

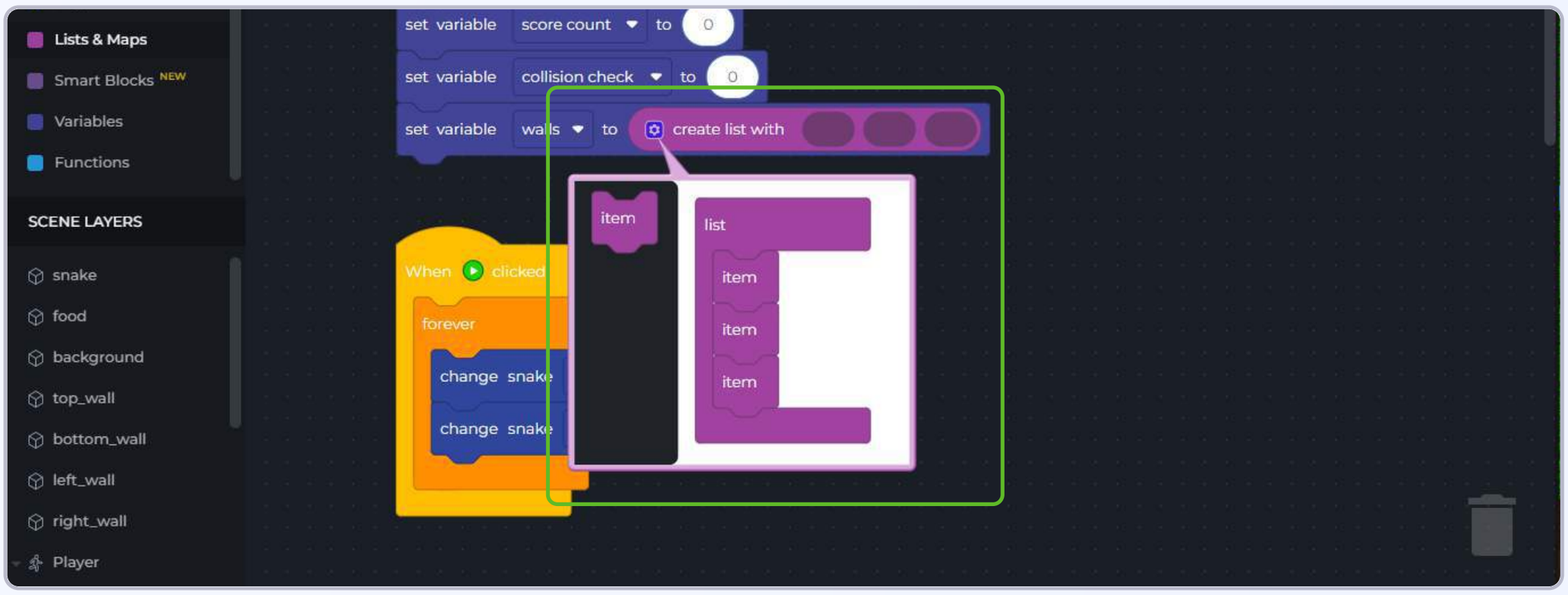


Since a list can store multiple values, you can modify the number of empty spaces available in the block

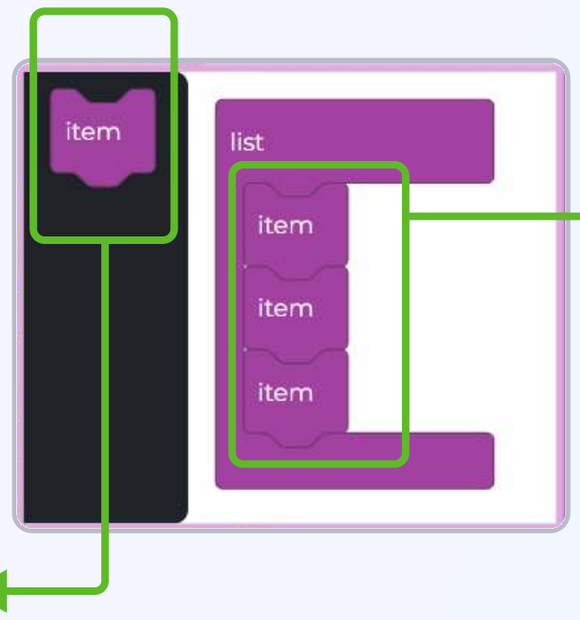
Click on this button to modify the number of empty spaces available in the list block

We want to add the names of the eight shapes in the list. So we need eight empty slots.

Step 6: Click on the “ ” button inside the **“create list with block”**



When you click on the  button, a pop up appears on your screen that looks like this.

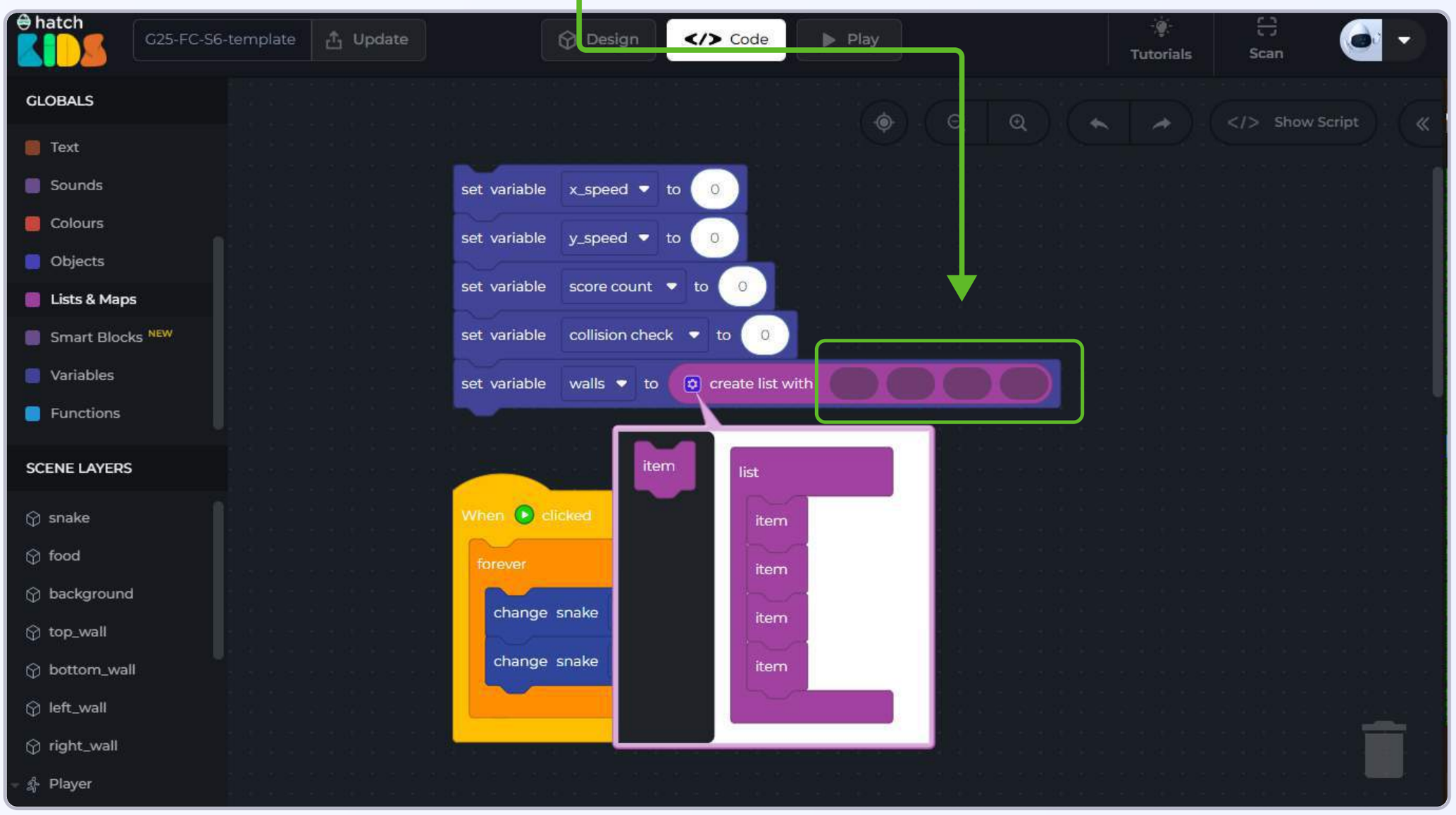


This space shows the number of items that can be added in the list, in the image as you can see it shows 3 items, and that's why you see the 3 empty spaces in the create list with block

To add more items (empty spaces) in the list, click on this block and drag it to the right side, and attach it with the other item blocks that you see

As you can see, the moment you add the new item block in the pop up box, a new empty space appears in the **"create list with"** block.

We need 4 empty spaces. so drag abd place one more item blocks in the white pop-up box.

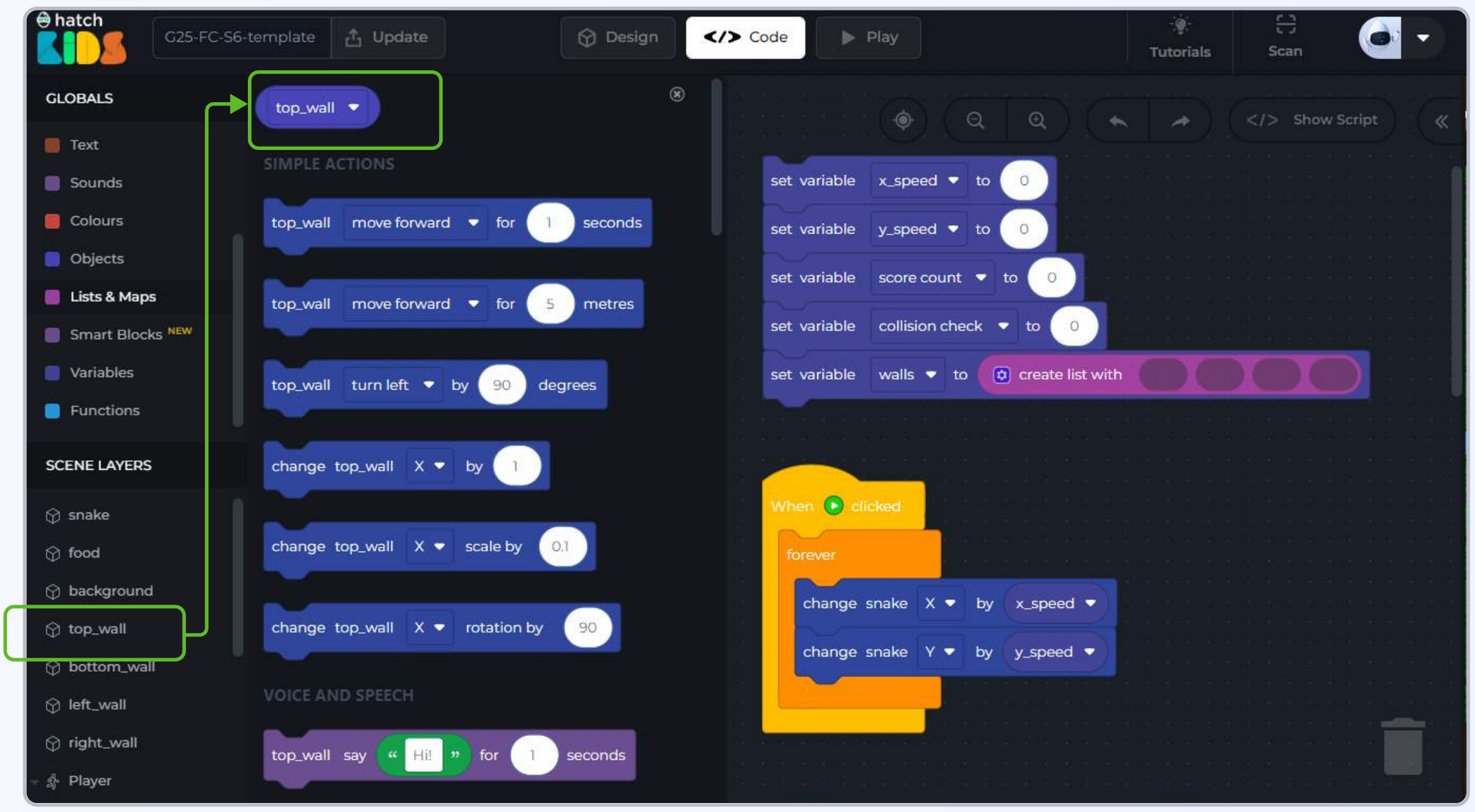


Step 7: Modify the **"create list with"** block to have **four empty spaces.**

Step 8: Let's fill those 4 empty spaces with the names of the objects in the project.

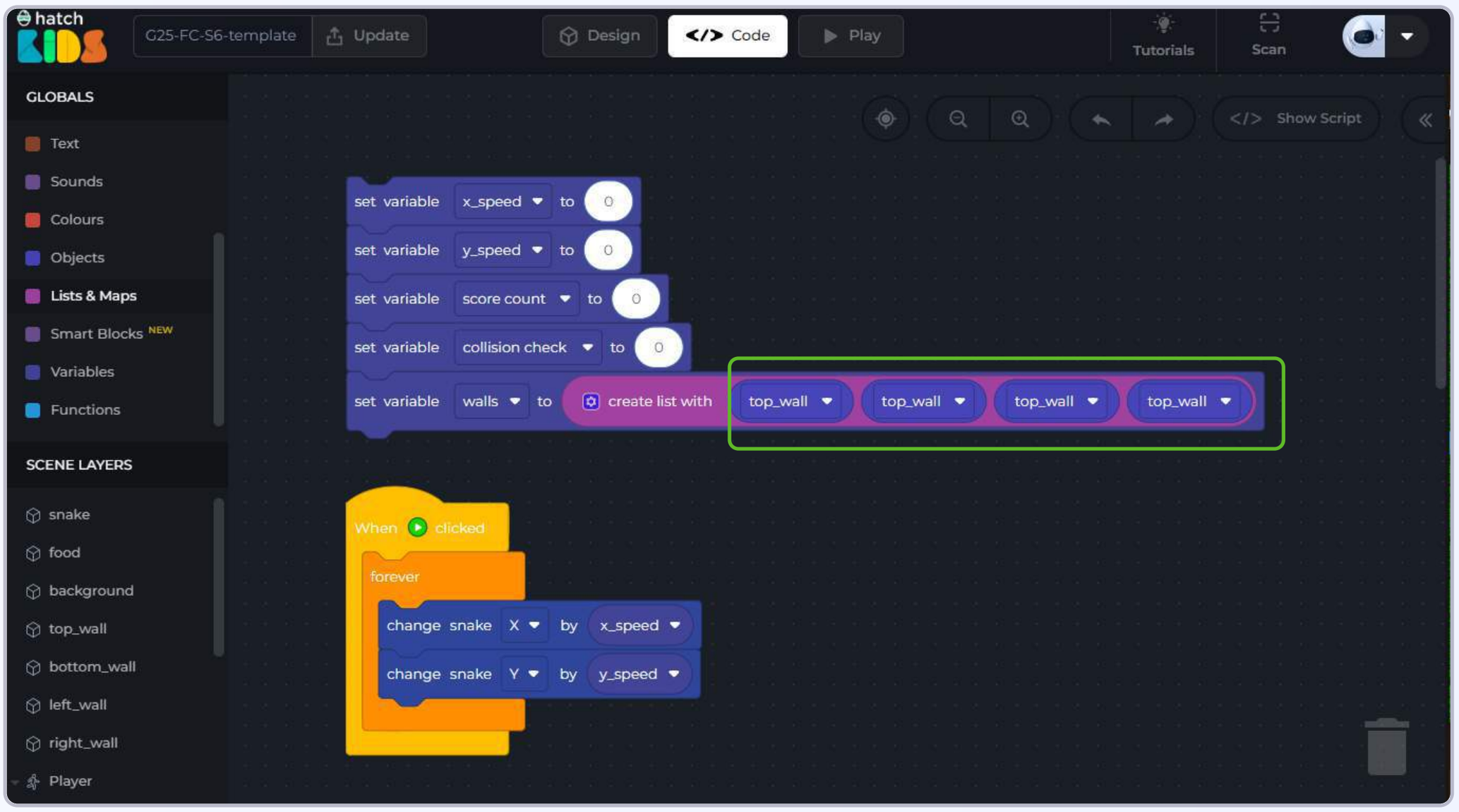
Let's start with **"top_wall"**

Click on the name **"top_wall"** in the **left panel,** and **drag out the very first block** that just contains its name.



Step 9: Drag out the block that says **"top_wall"** and attach it in the first empty space inside the **"create list with" block.**

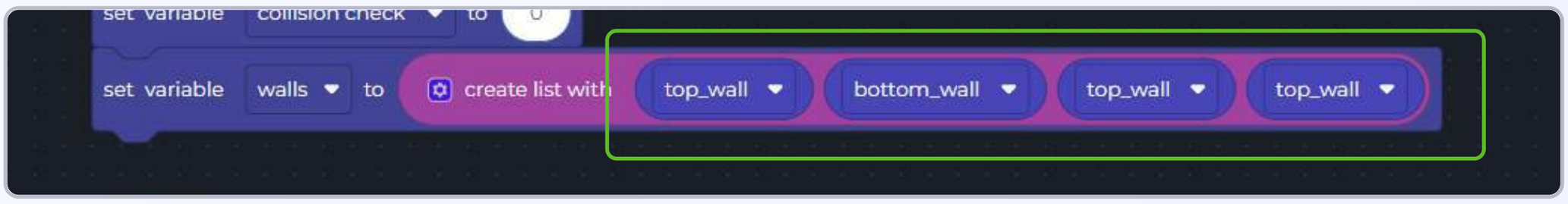
Step 10: Duplicate the **“top_wall”** block 3 times, and attach them in the empty spaces of **“create list with”** block.



Step 10: Click on the drop-down menu option on the second “top_wall” block, and select the option “bottom_wall”

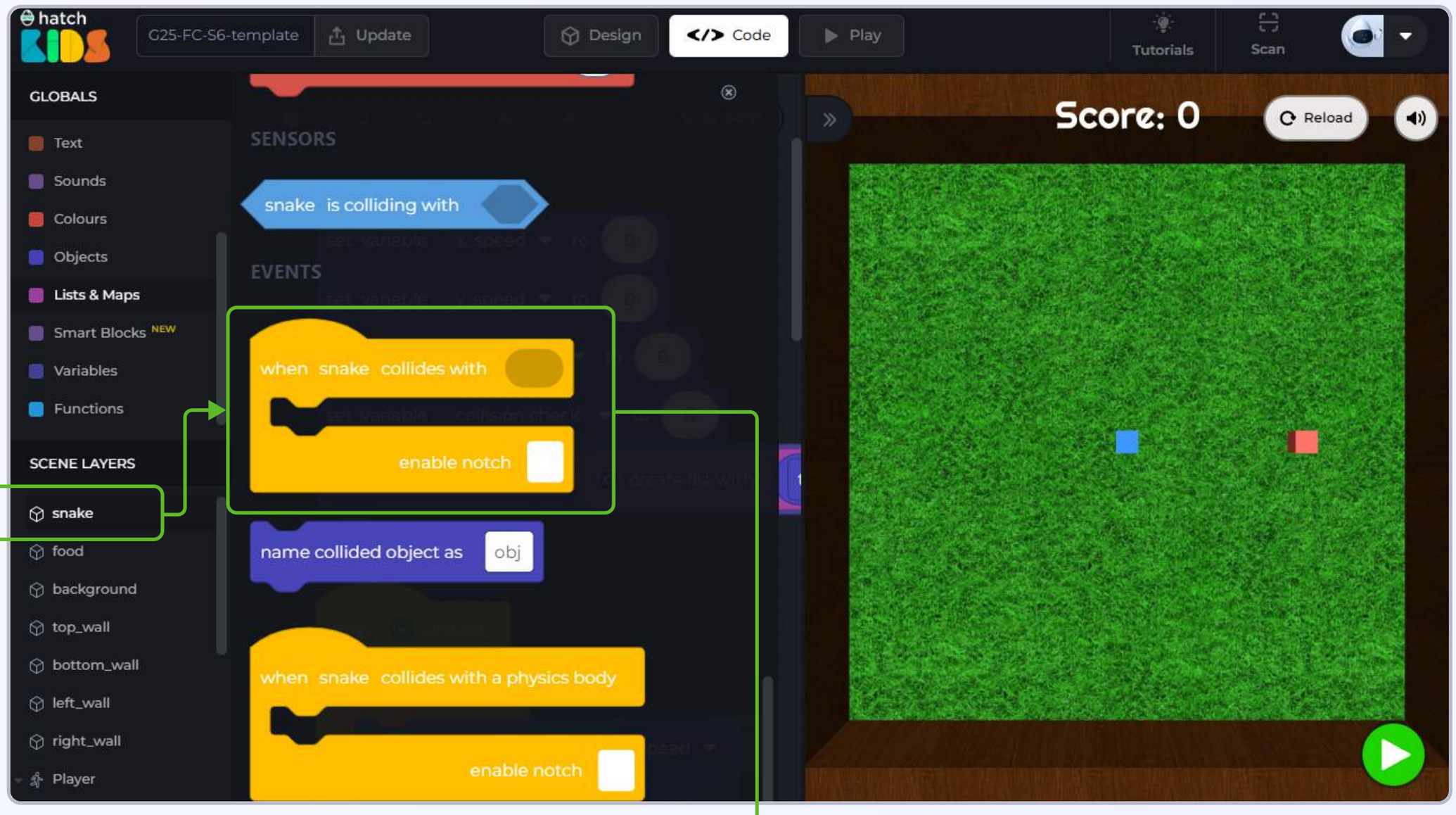
Similarly in the other two “top_wall” blocks, select the names “left_wall” and “right wall” from the drop down menu options.

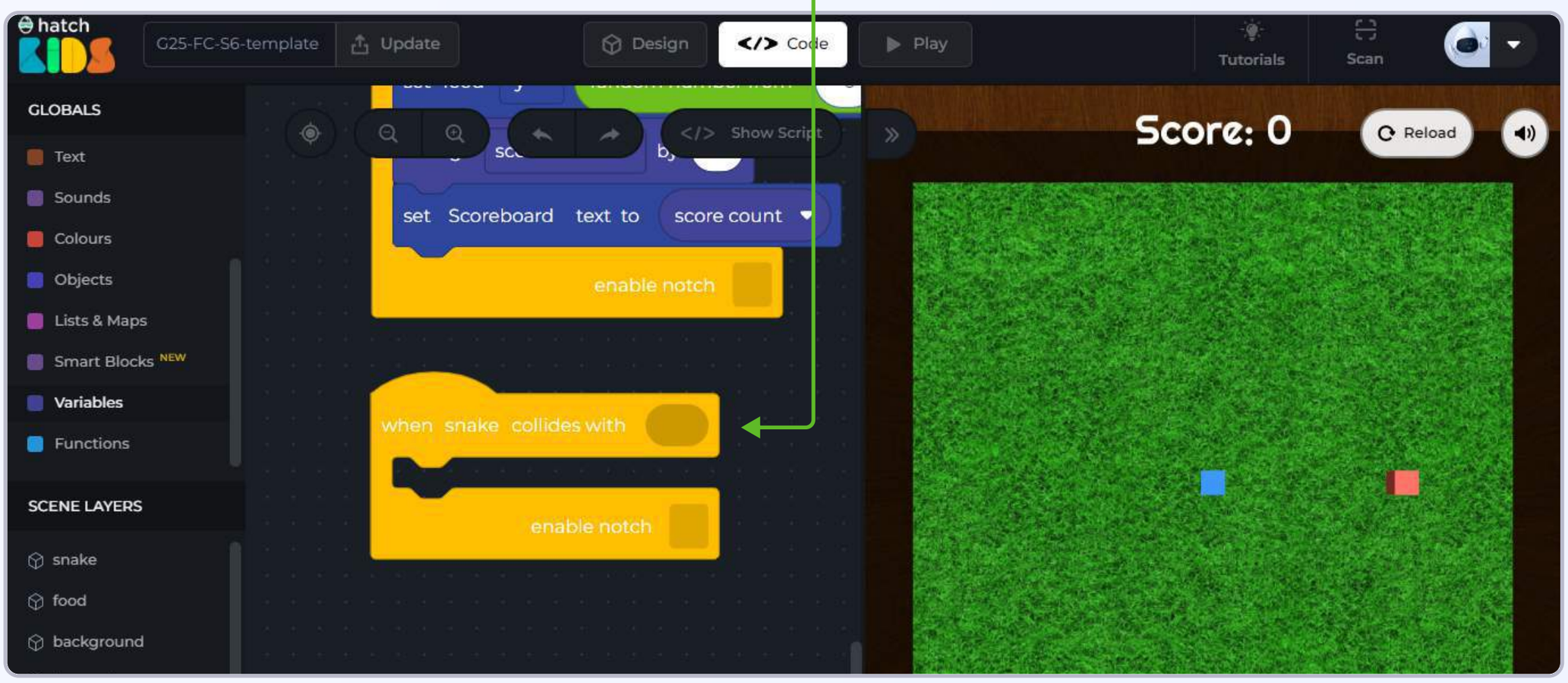
So your final list should be:



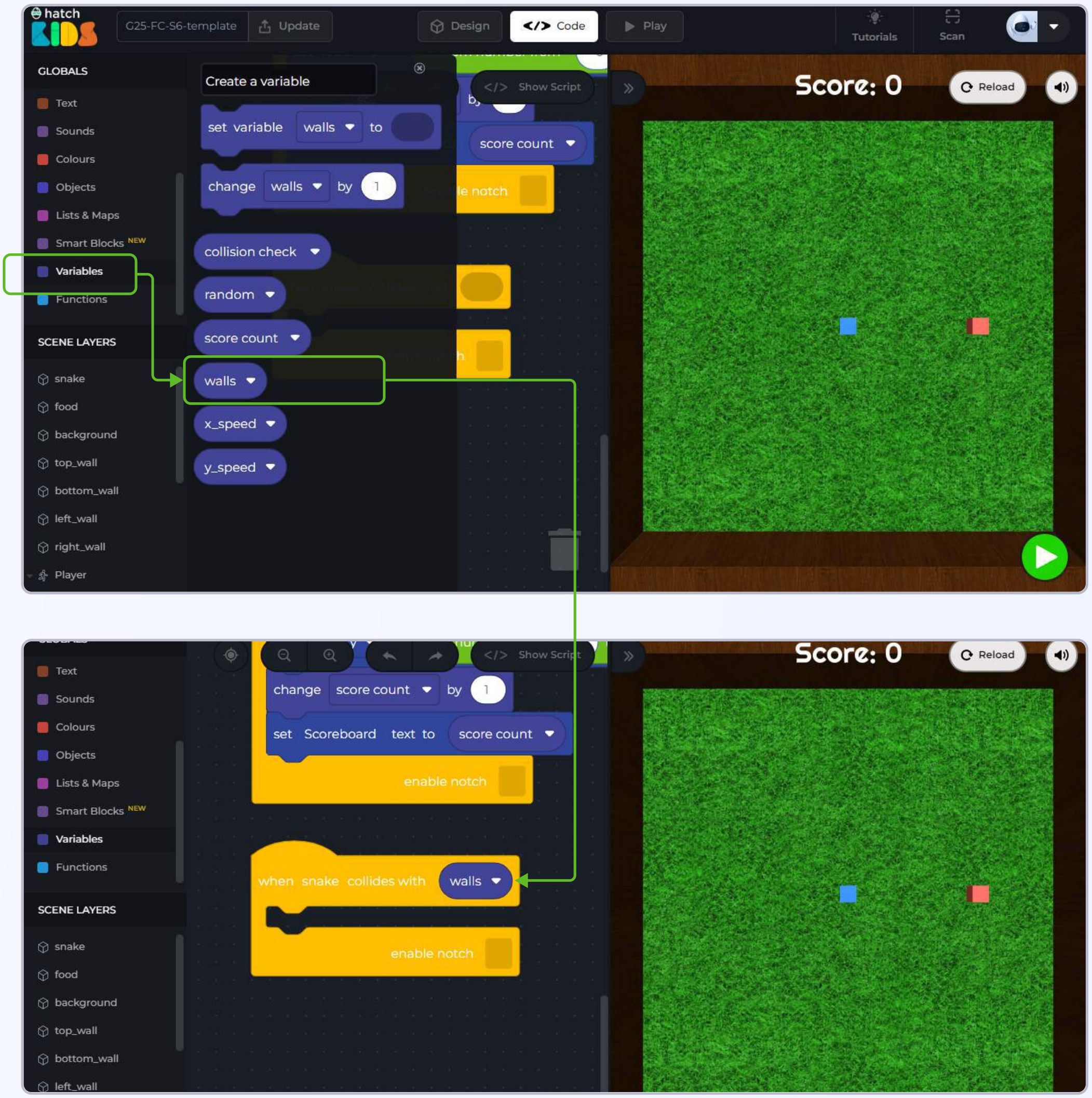
When the “snake” object hits any one of the object mentioned in the “walls” list, then the value of “collision check” variable will become one.

Step 11: Click on the name **“snake”** in the left panel, and drag out the block that says **“when snake collides with”**.





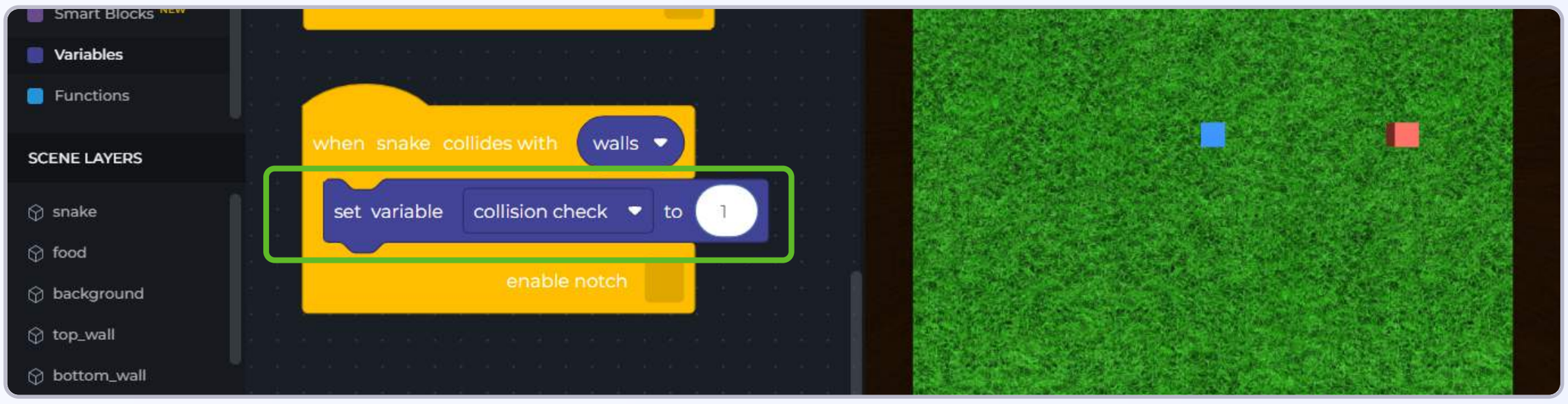
Step 12: Click on the **“variables”** category in the left panel, and drag out the **“walls”** block inside the workspace, and place it in the **“when snake collides with”** block as shown



When snake collides with the wall, the variable “collision check = 1”.

Step 13: Duplicate the block that says **“set variable collision check to 0”** and place the duplicate block inside the **“when snake collides with walls” block.**

Change the number “0” to “1”.

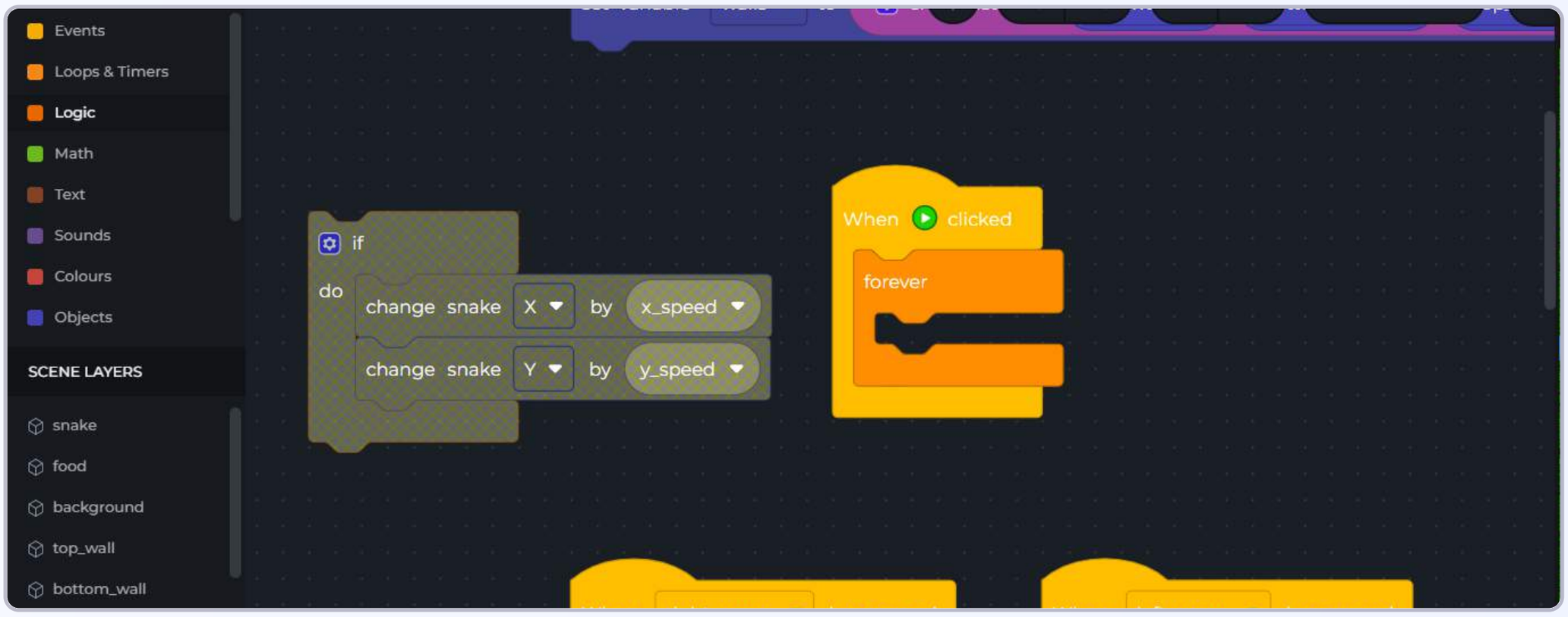


Now we need to modify the code inside the forever block, so that the snake only moves when the value of **“collision check = 0”**

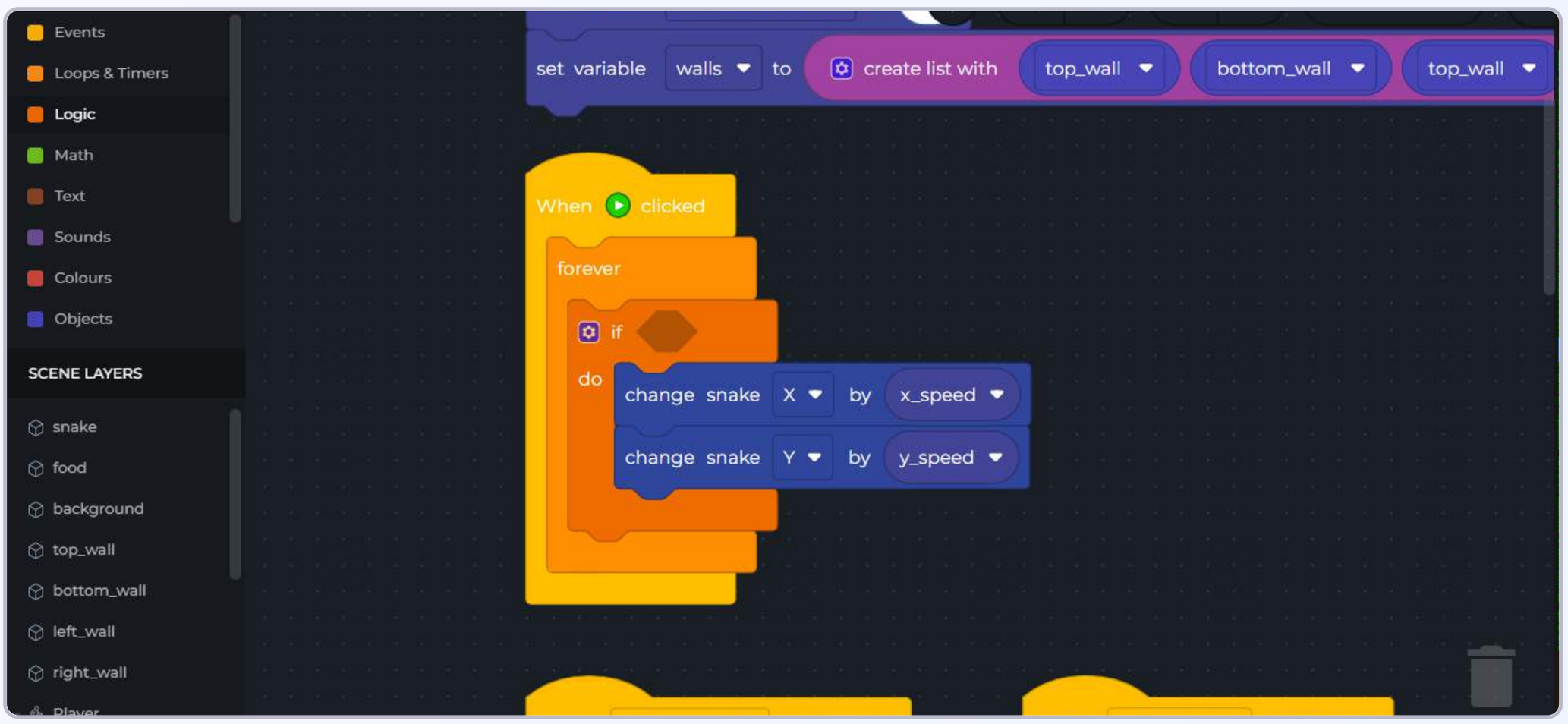
We do this by adding an if-condition block.

Step 14: Click on the **“Logic”** category in the left panel, and drag out the block that says **“if___ do___”**

Step 15: From inside the **“forever”** block, drag out the **“change snake x by x_speed”, and “change snake y by y_speed”** blocks and place them inside the **“if__ do__”** block



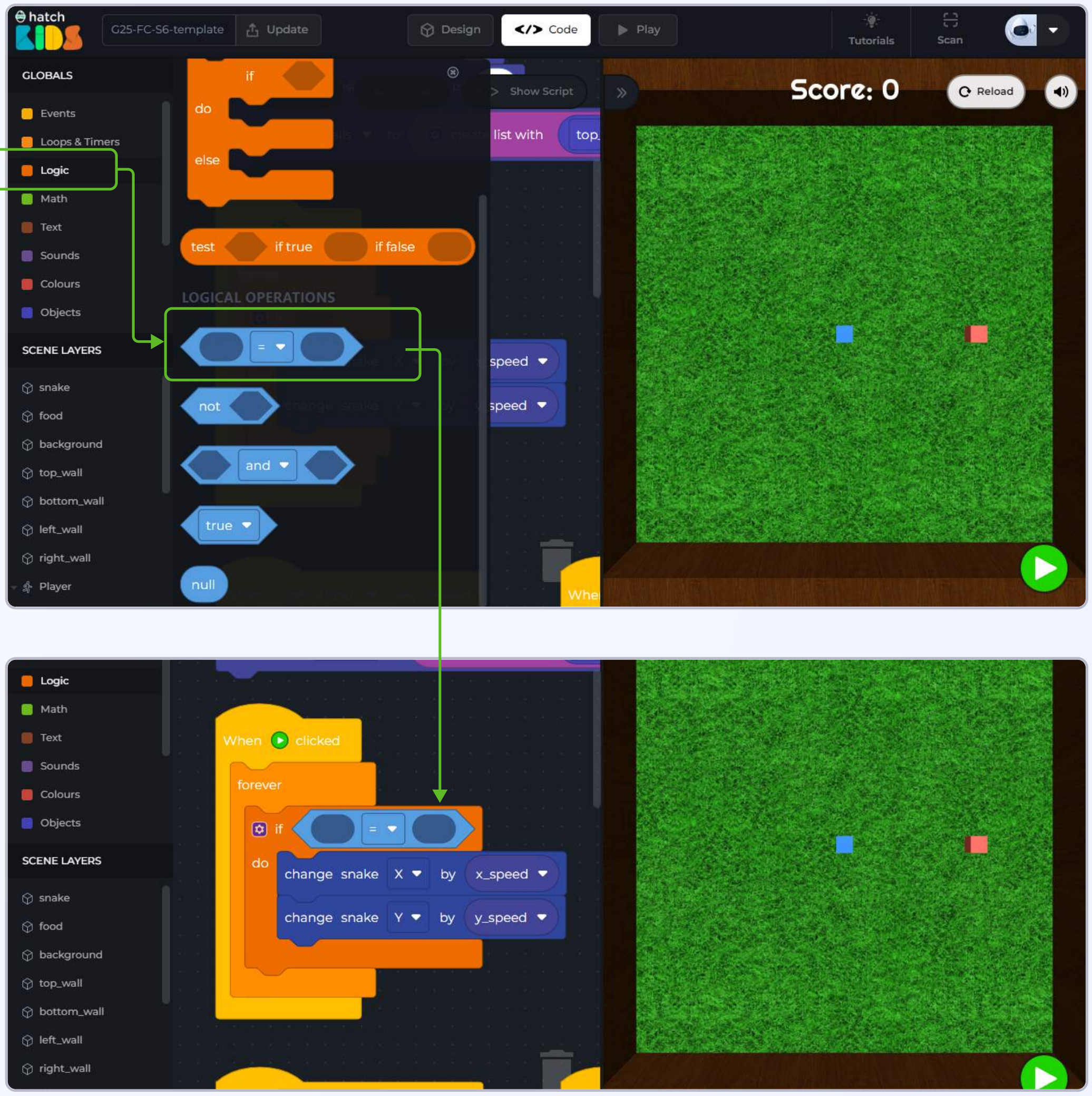
Step 16: Drag the complete **“if-block”** and place it inside the **“forever”** block



We want to implement the logic so that snake only moves if “collision check = 0”.

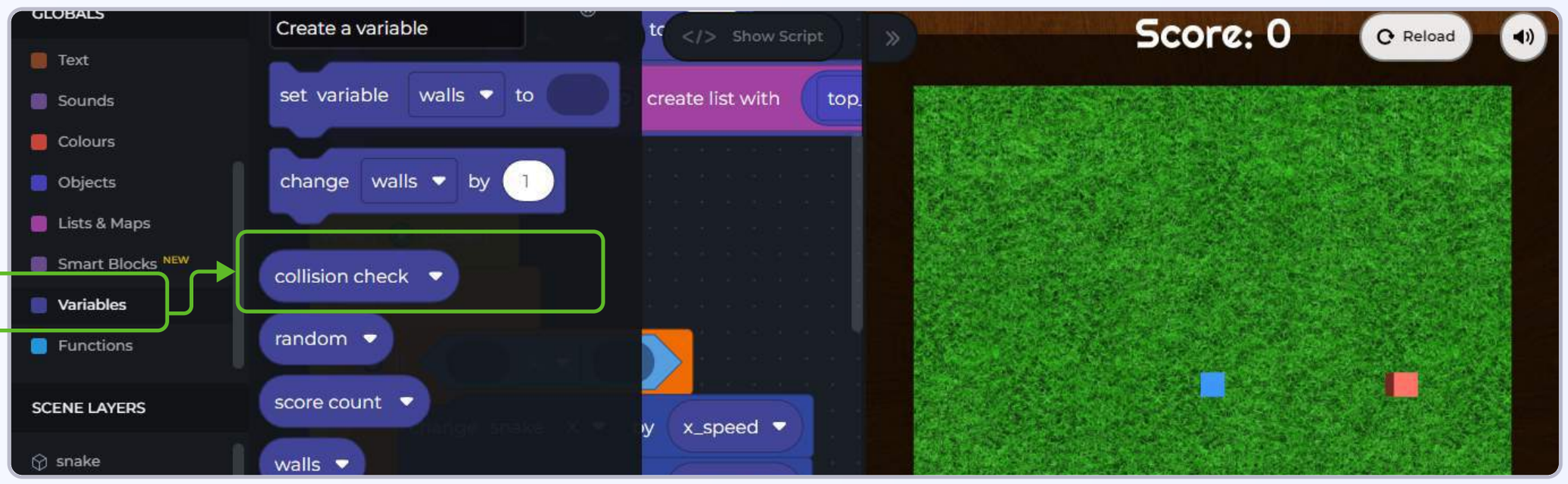
Step 17: Click on the **“Logic”** category in the left panel, and bring out the  block

Place the  block inside the if-block as shown

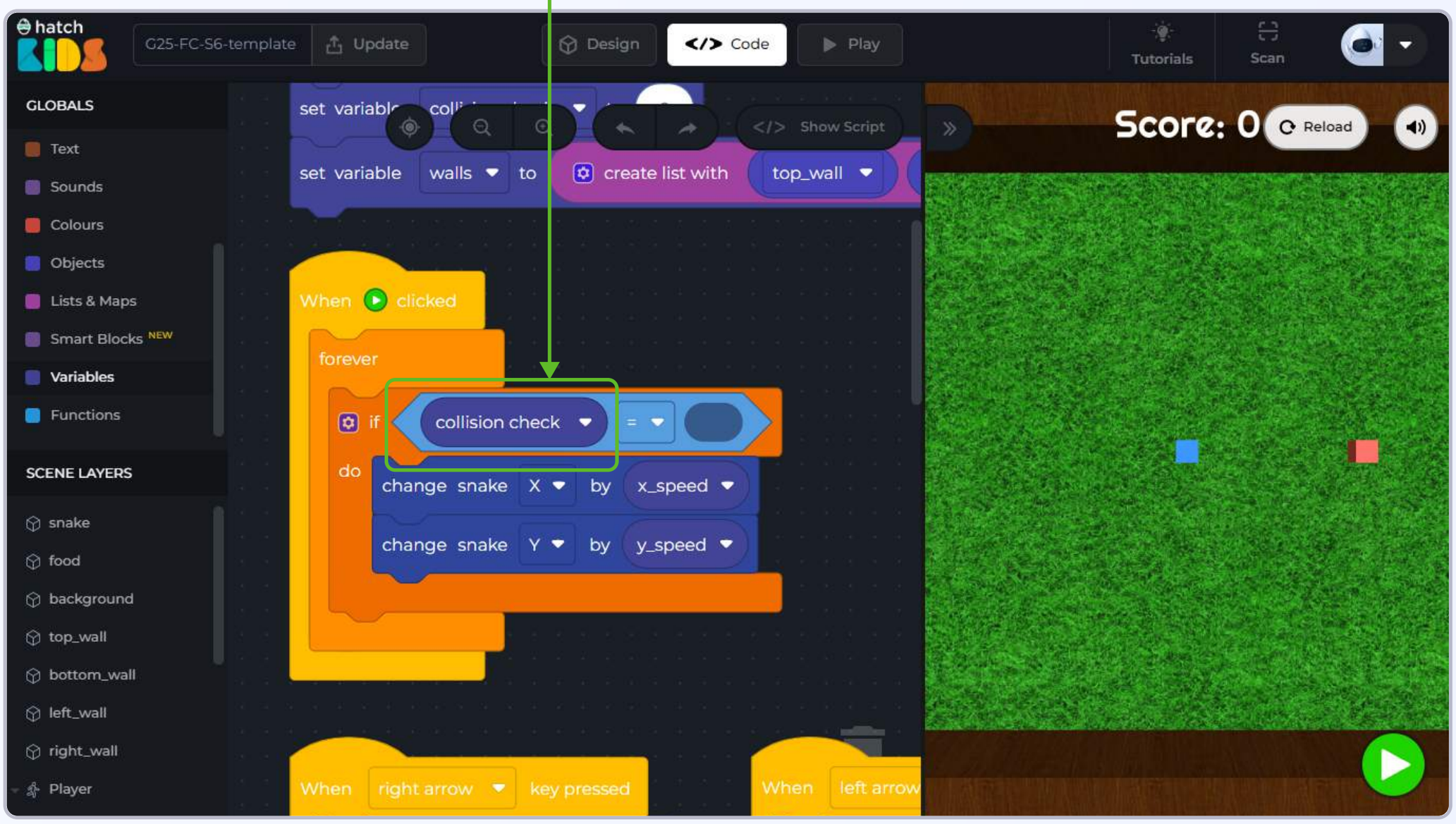


Inside the if-condition we need to code “collision check = 0”.

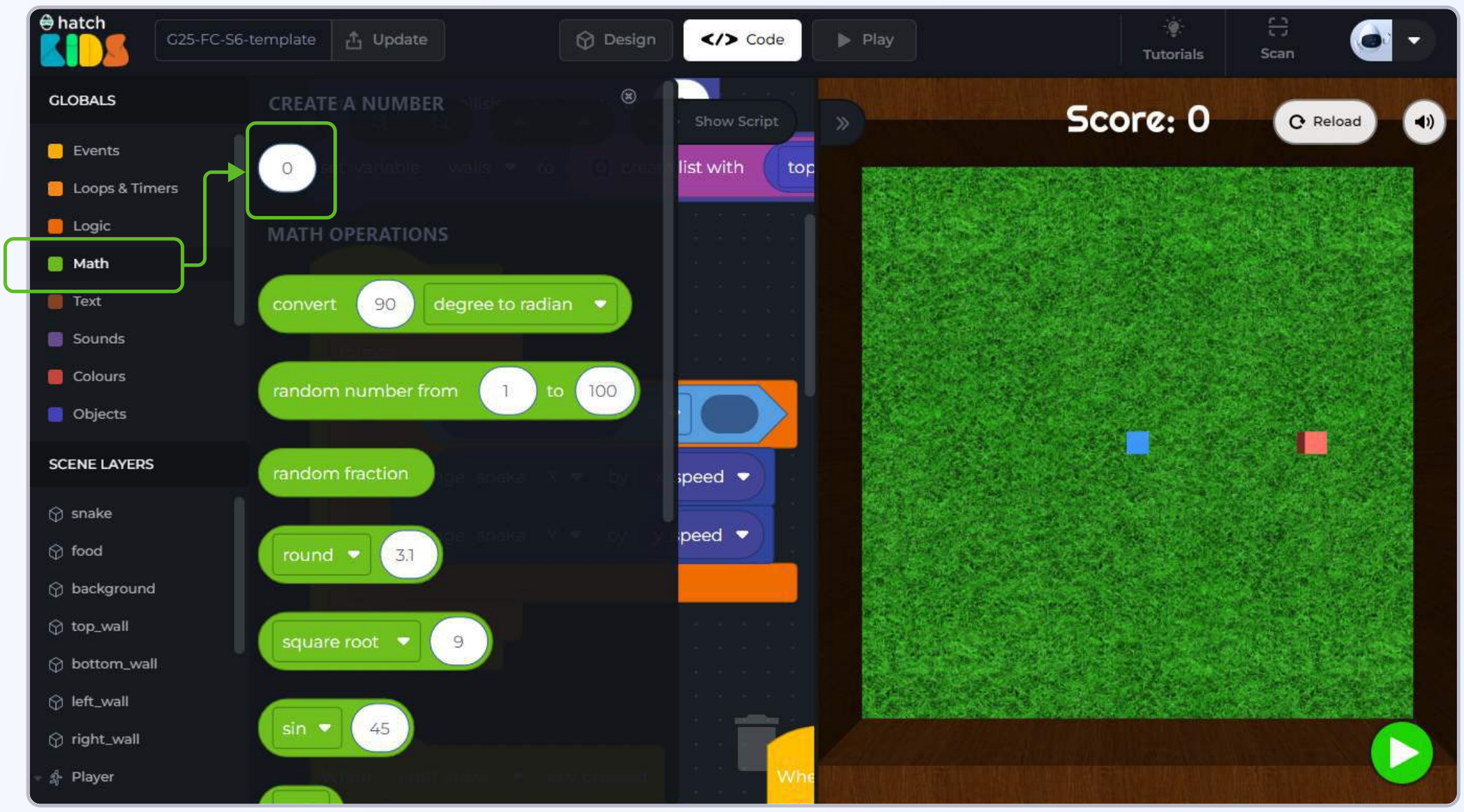
Step 18: Click on the **“variables”** category in the left panel, and drag out the **“collision check”** block

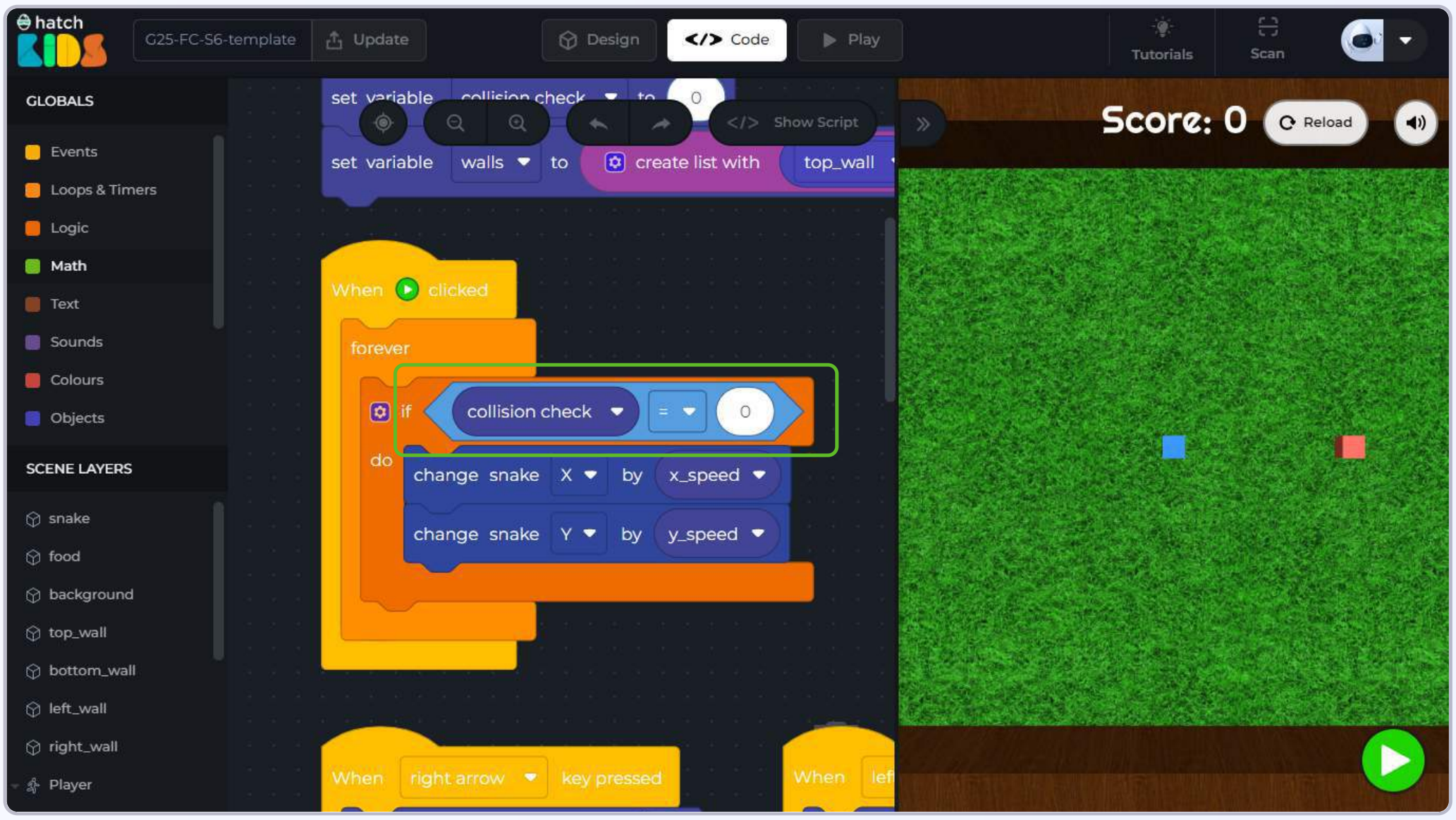


Drag out the “collision check” block and place it inside the if-block as shown



Step 19: Click on the **“Math”** category in the left panel, and drag out the **number “0” block** and place it inside the if-block as shown





After placing the **number “0” block** inside the if-block, read the final combination.

It says,
When green play button is clicked --->> Forever --->> If collision check = 0 --->> move snake

So when game starts, as you press the arrow keys, your snake will move and will keep on moving.

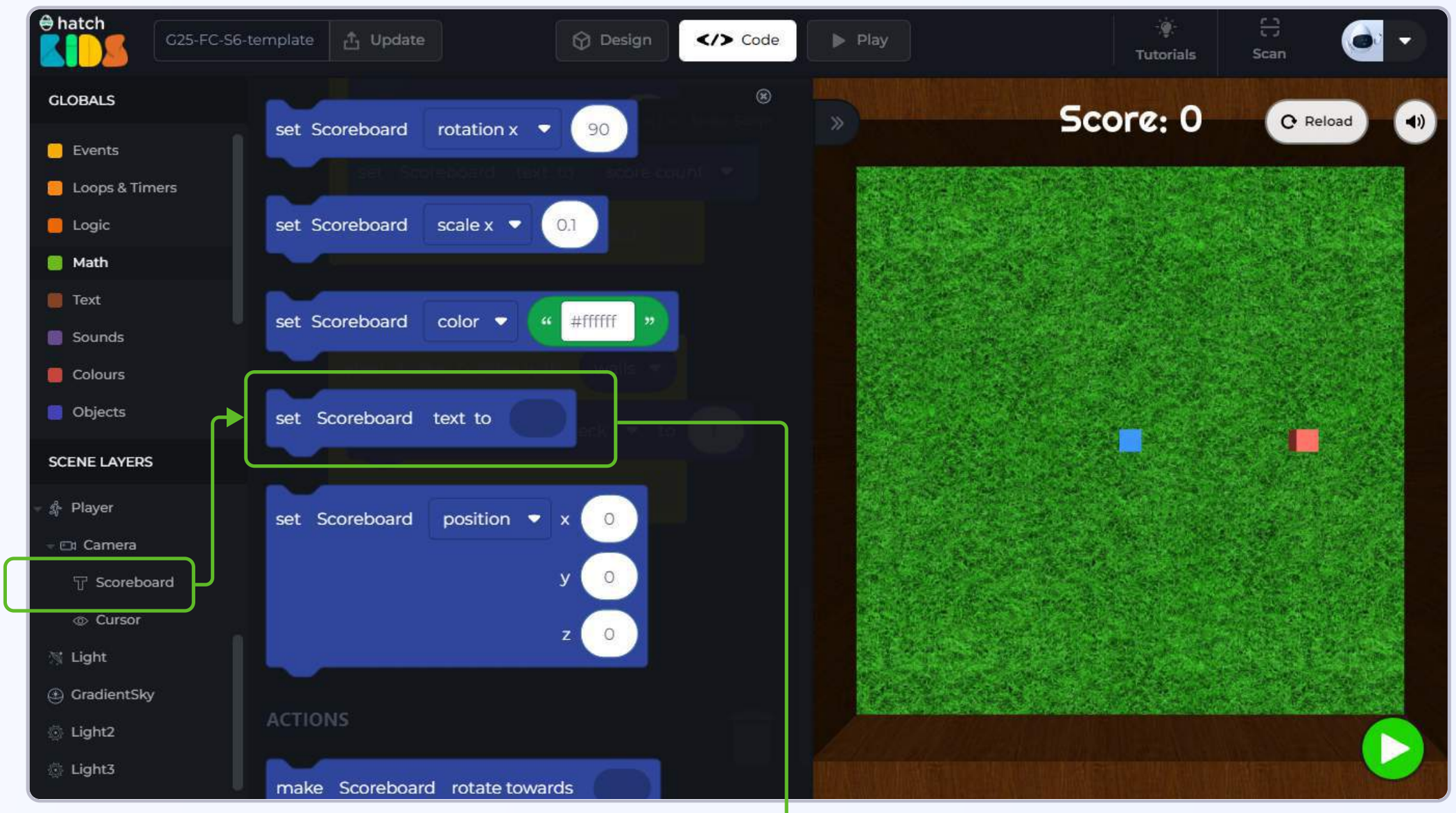
But when the snake hits a wall, we are changing the value of “collision check” to “1”.

So, since “collision check ≠ 0” anymore, computer will not be able to move the snae object.
And to play the game again you will have to reload the game and then play.

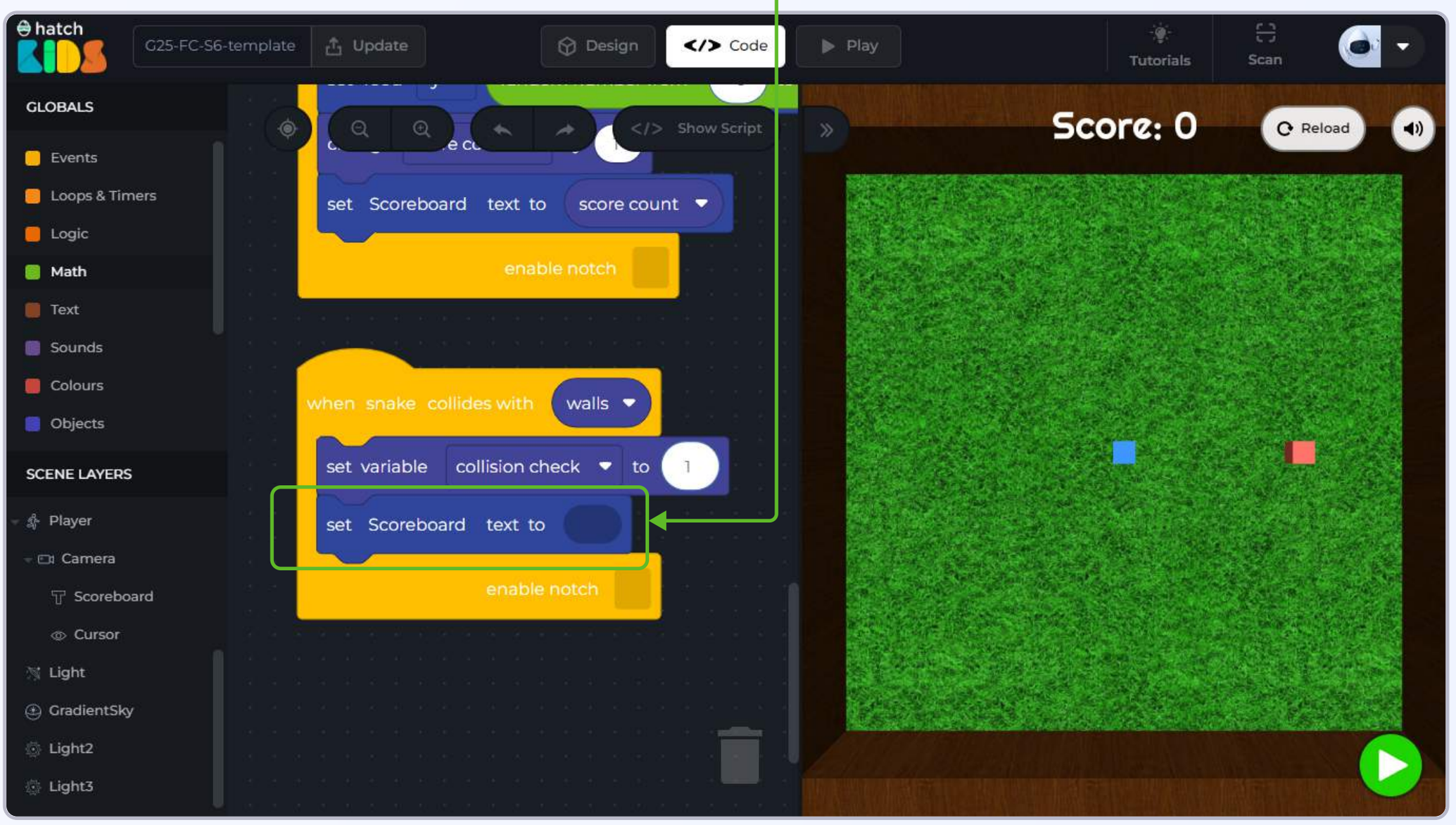
Let’s add one last block and our game is ready.

When the game is over, let’s display the text “game over” on screen so that the player knows that they have lost the game.

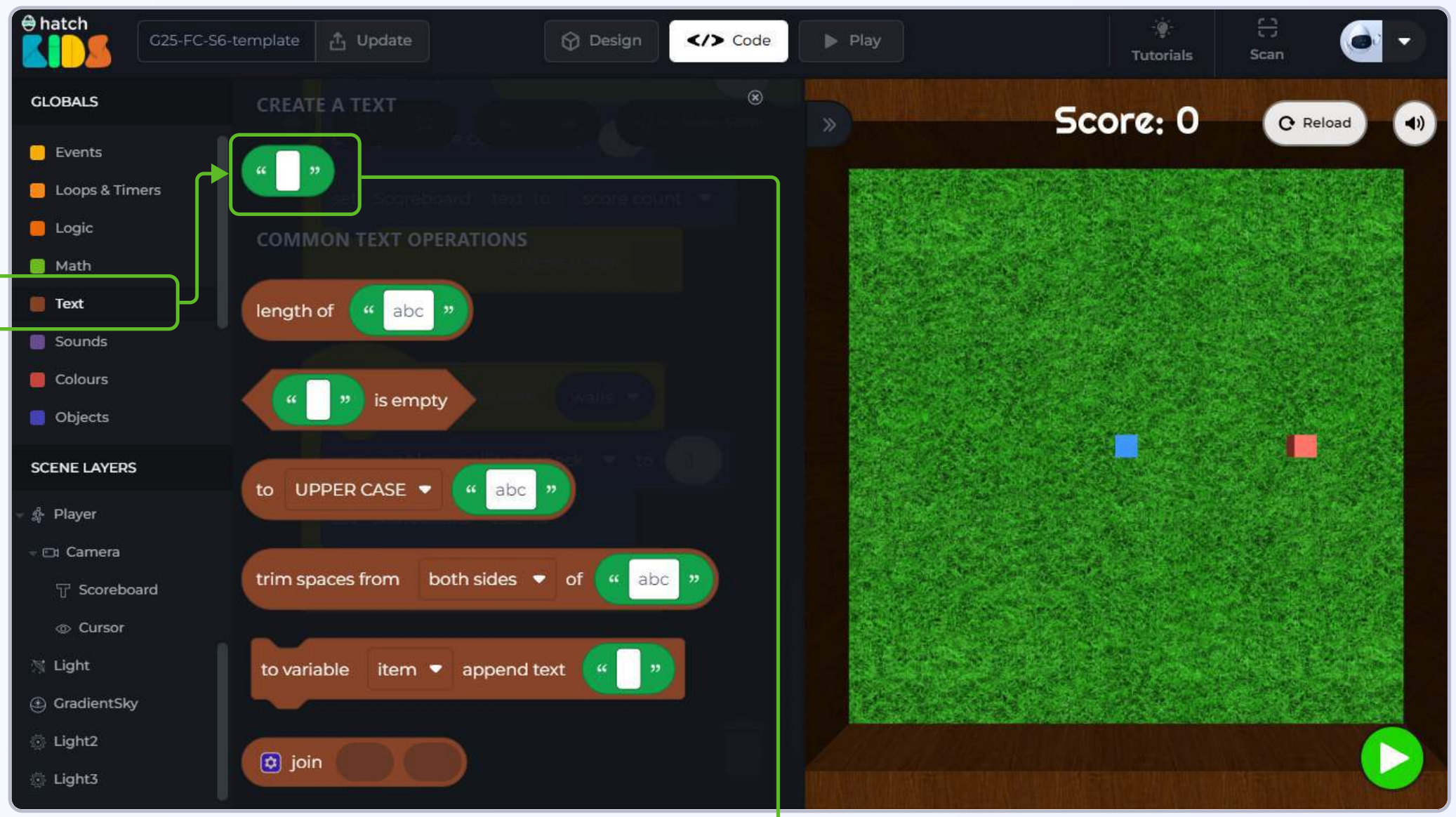
Step 20: Click on the name **“Scoreboard”** in the left panel. A list of blocks will appear. Scroll and look for the block that says **“set Scoreboard text to”**



Step 21: Drag the **“set Scoreboard text to”** block and attach it inside the **“when snake collides with walls”** block as shown

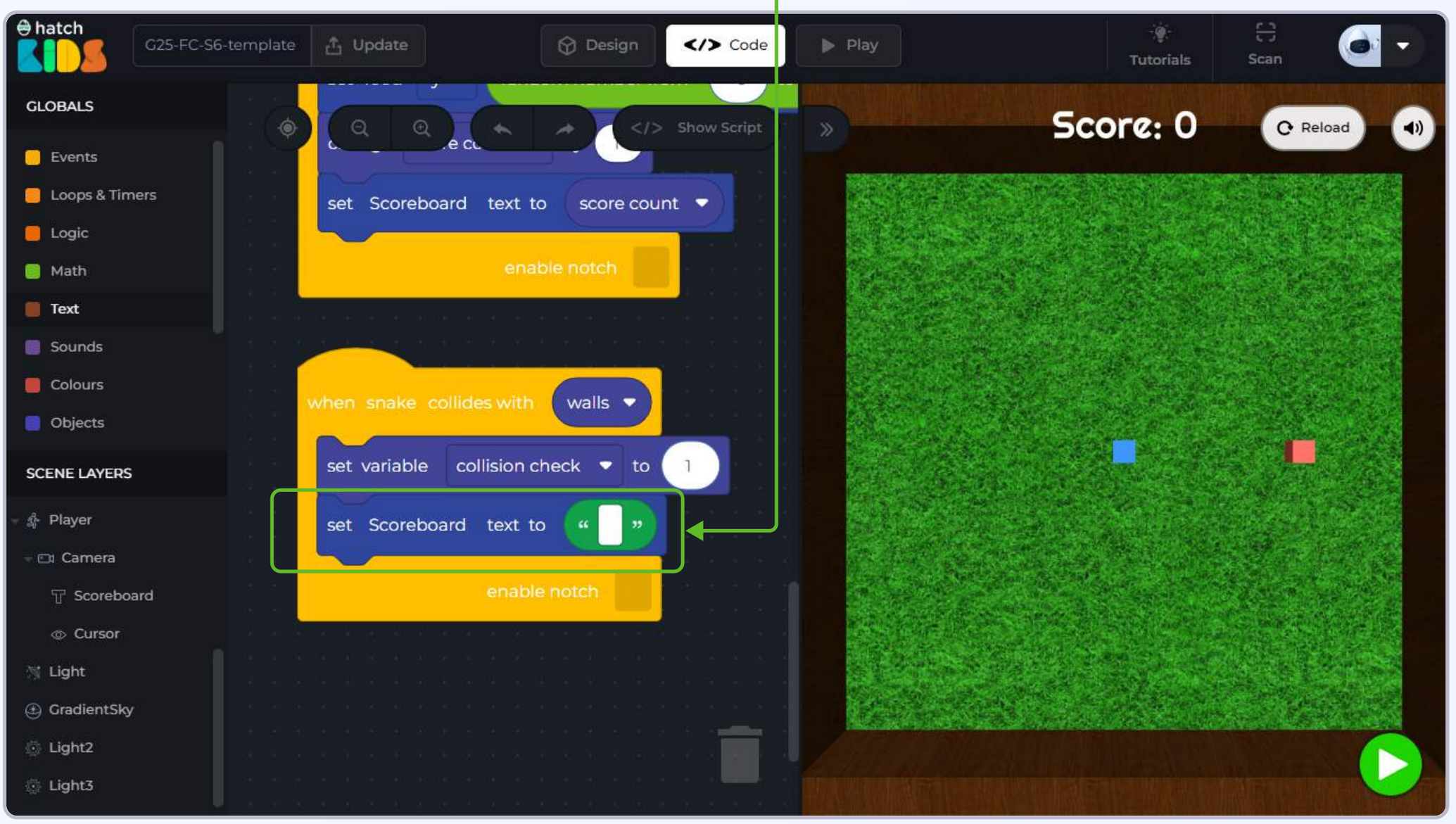


Step 22: Click on the **“Text”** category in the top half of the left panel. A list of blocks will appear. Drag the very first block from the top of the list

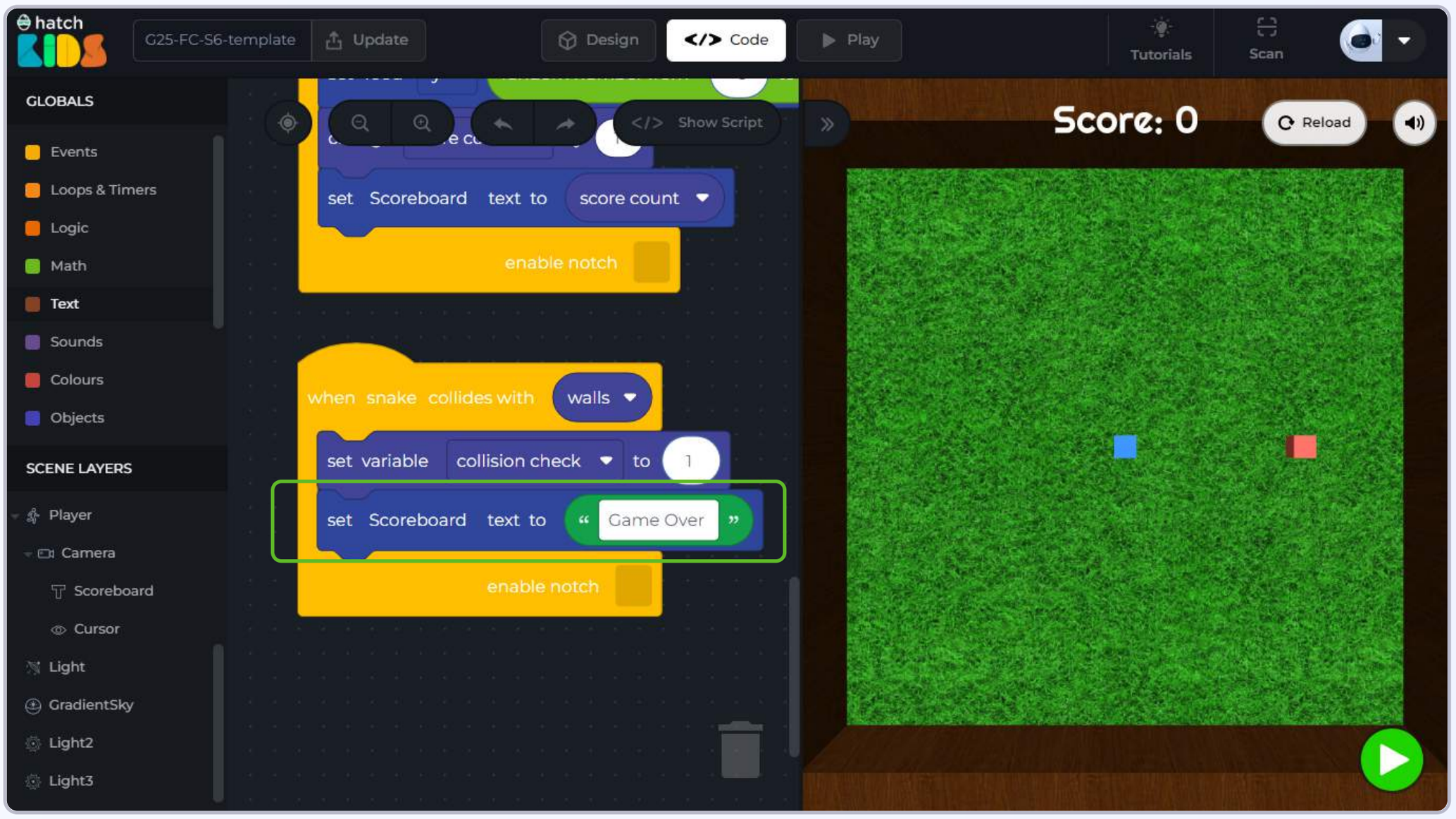


This is an empty text block. You can write any text inside this block and use for various things while coding,

Step 23: Place this block inside the “set Scoreboard text to” block as shown



Step 24: Click inside the text block that you added, and typer “Game Over”



Now when you run the code, and move the snake, after the snake hits any one of the walls, you will see that you are not able to move the snake using the keyboard buttons.

And the text that was displaying your score earlier, will be saying “Game Over”.

This completes the snake game project.

You can now give this project a name, and publish it and share it with your friends and let them play.

🎉🎉🎉 **END OF GUIDE** 🎉🎉🎉