Bhaskar joshi ,Head Teacher GPS Mateeladhura Tarikhet Almora

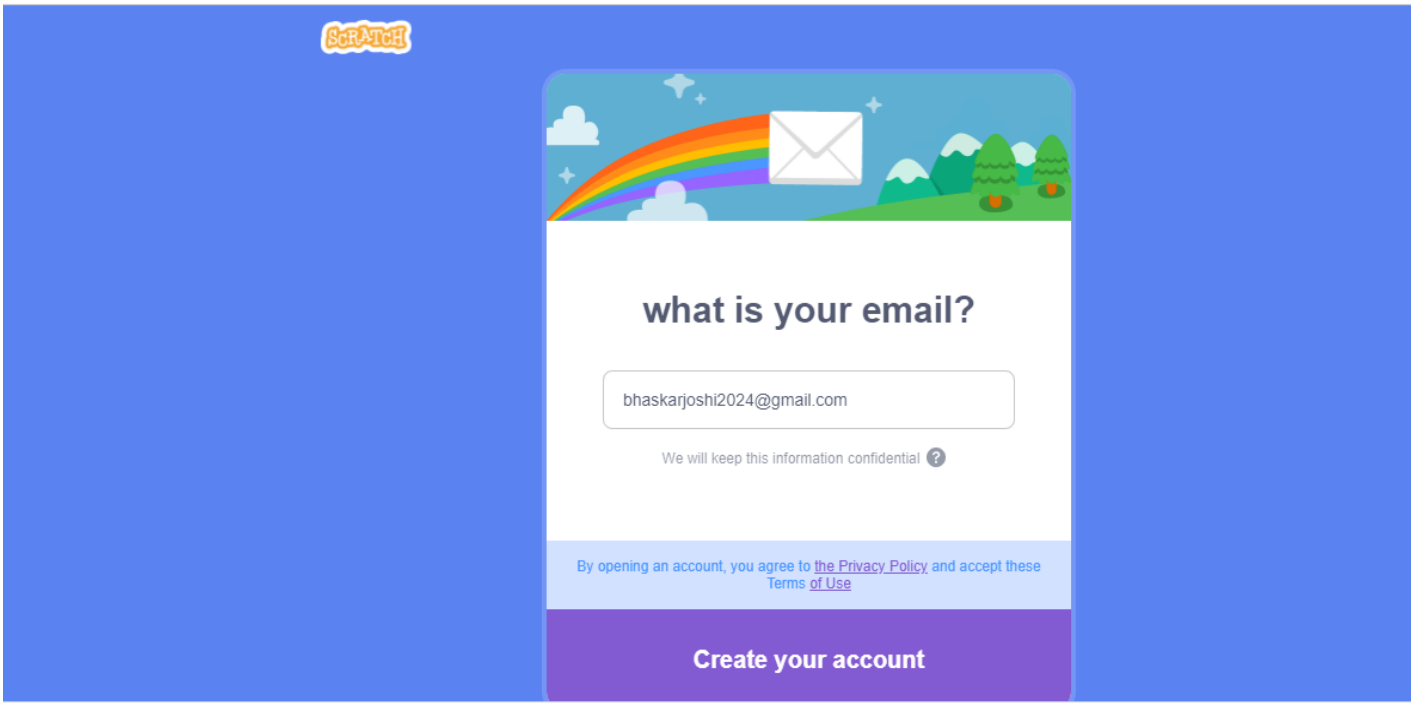
Engaging Young Minds: Teaching Scratch Programming for Creative Problem-Solving and Digital Literacy

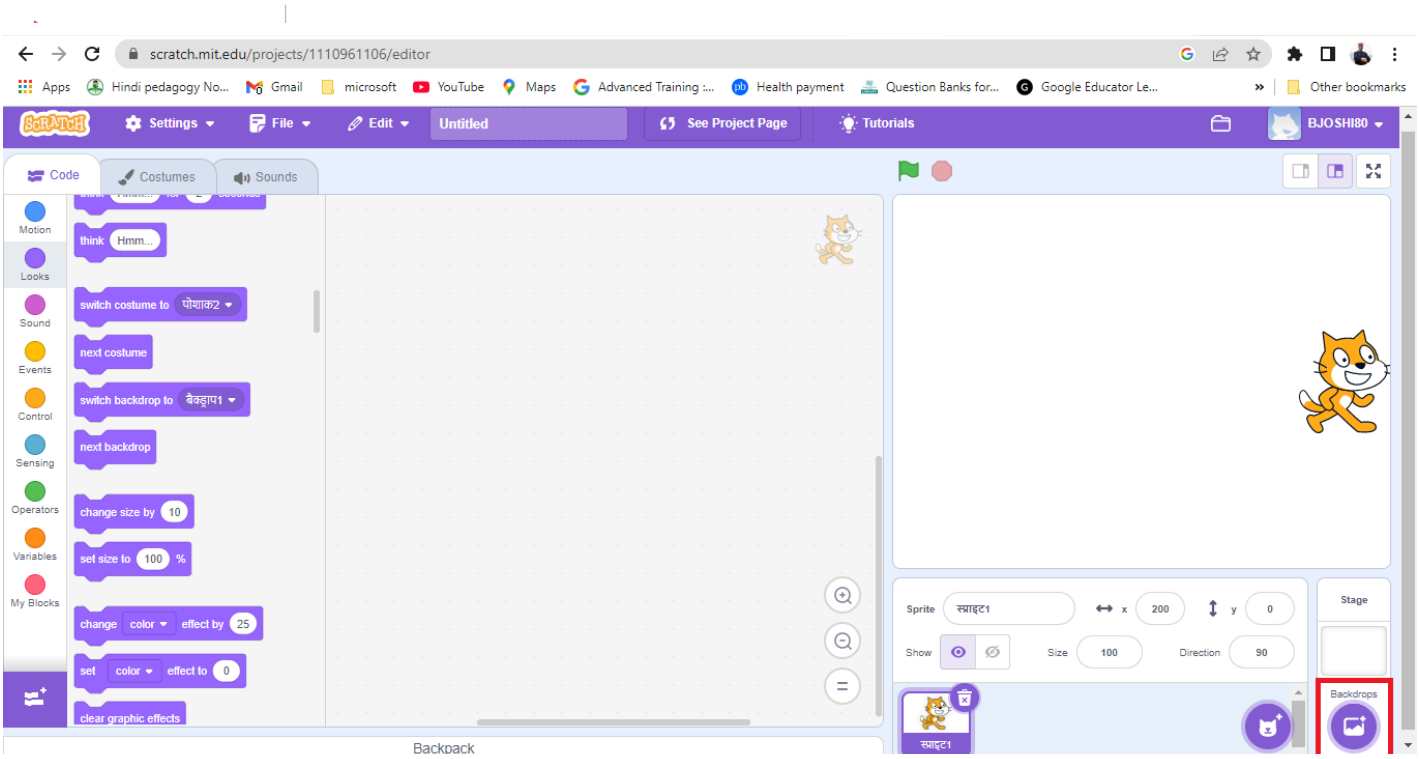
1. What is Scratch?

Scratch is a free, block-based programming language developed by MIT Media Lab, designed for young learners to create interactive stories, games, and animations. It uses a visual interface where students drag and drop blocks to create code, making it an ideal introduction to programming concepts without the need to write syntax. Scratch promotes creativity, problem-solving, and critical thinking, and is widely used in schools to teach the fundamentals of coding.

2. Creating a Scratch Account

1. Open a browser and go to the Scratch website: https://scratch.mit.edu.
2. Click on "Join Scratch" in the top-right corner.
3. Fill out the registration form with a username, password, and email address.
4. Verify the email address by clicking the confirmation link sent to the email.
5. Log in to access all Scratch features, including saving projects and sharing them online.





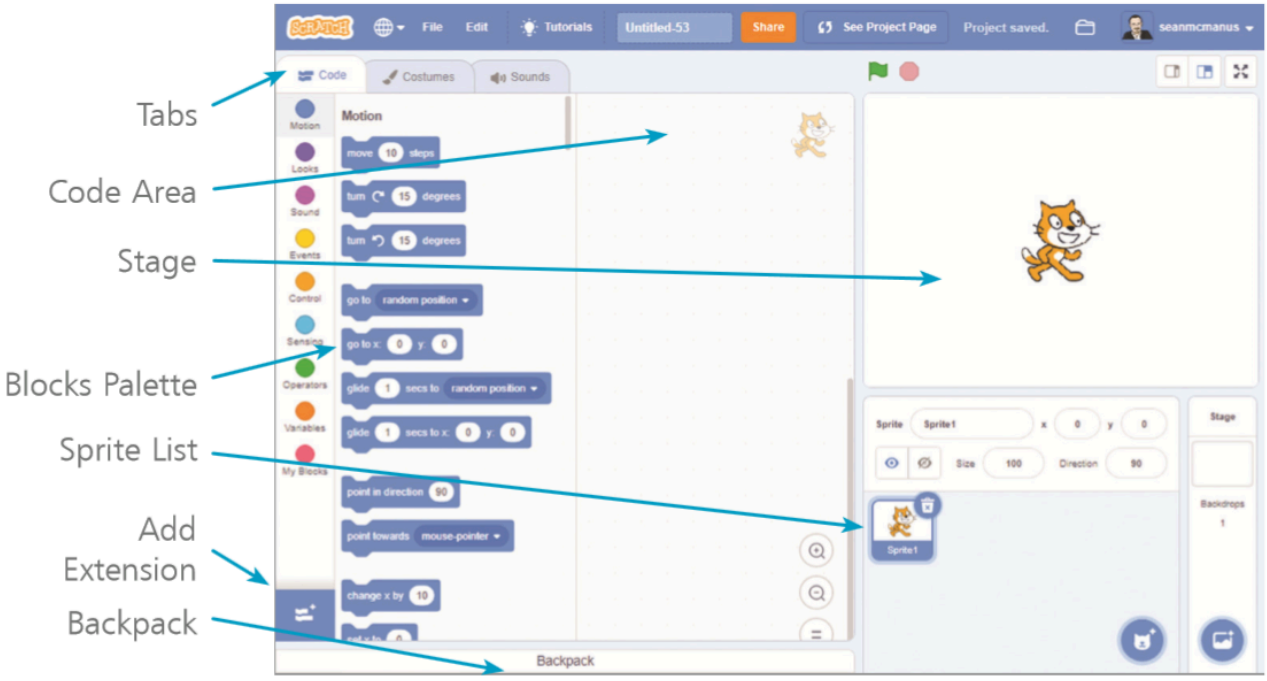
3. Using the Scratch Screen

The Scratch interface consists of several key sections that help you create animations and games effectively:

- **Stage**: This is the main area where your animations and games are displayed. When you open Scratch, you'll notice an orange cat in the center of the Stage. It serves as the default character, or "sprite," for your project.

- **Sprite List**: Sprites are characters or objects within your game or animation. For instance, a project might have multiple sprites like spaceships, aliens, or missiles. The Sprite List, located below the Stage, shows all the sprites in your project, allowing you to switch between them by clicking on their icons.
- **Blocks Palette**: Scratch uses blocks to give commands to the computer. These blocks, which snap together like puzzle pieces, are categorized by function. The Blocks Palette, positioned on the left, displays all the available blocks. By default, it shows the Motion blocks (colored in blue), which are used to move sprites. You can explore other categories, such as Looks or Sound, by selecting the respective buttons.
- **Code Area**: The Code Area, previously known as the Scripts Area, is where you create programs by assembling blocks into scripts. This section adjusts its size based on your screen resolution, offering more space to work on larger monitors.
- **Backpack**: Located at the bottom of the screen, the Backpack feature allows you to copy and transfer sprites and scripts between different projects. You can open it, drag items into it for storage, or retrieve items by dragging them into your current project.

The top of the Scratch interface includes tools for managing your project. You can rename your project by clicking the "Untitled" box or access your saved projects via the folder icon in the top-right corner. In the desktop version, the File menu lets you save your project manually, while the online version automatically saves updates.



4. Using Scratch on Tablets

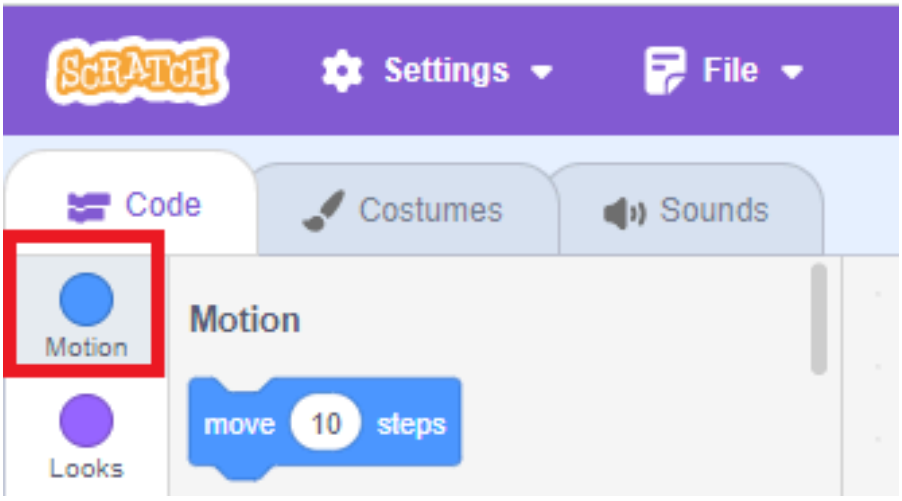
Scratch can also be used on tablets through the Scratch app (available on iOS and Android) or via a browser. Key features include:

- Touch-friendly interface optimized for smaller screens.
- Similar functionality to the desktop version, allowing students to code, animate sprites, and share projects.
- Offline access to projects when using the Scratch app.

5. Exploring the Blocks

Scratch programming is done using blocks. The blocks are categorized for easy understanding ,Here's how you can explore some basic blocks to understand their functionality:

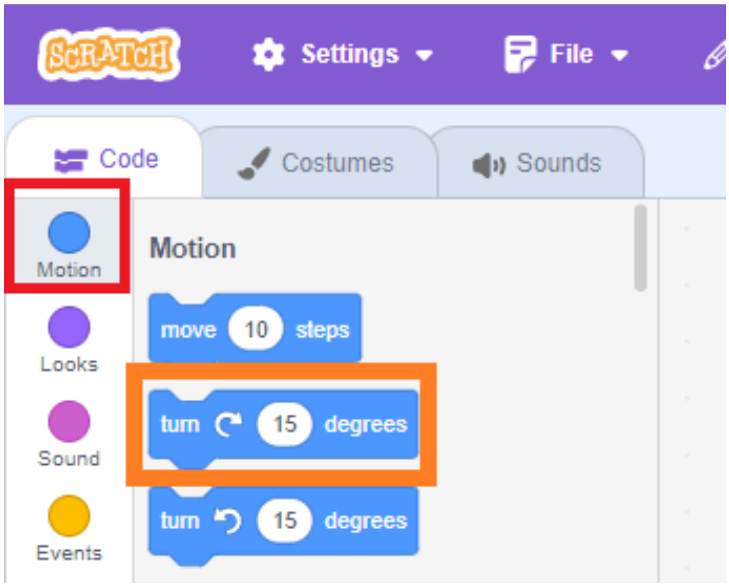
1. **Explore Motion Blocks**:
   - Click the blue **Motion** button on the left side of the Blocks Palette. This section is open by default when Scratch starts.
   - In the Blocks Palette, click the **move 10 steps** block. The cat sprite will move to the right, in the direction it is facing. Each click moves the sprite once. Note that this block changes the position of the sprite but doesn't animate its legs.



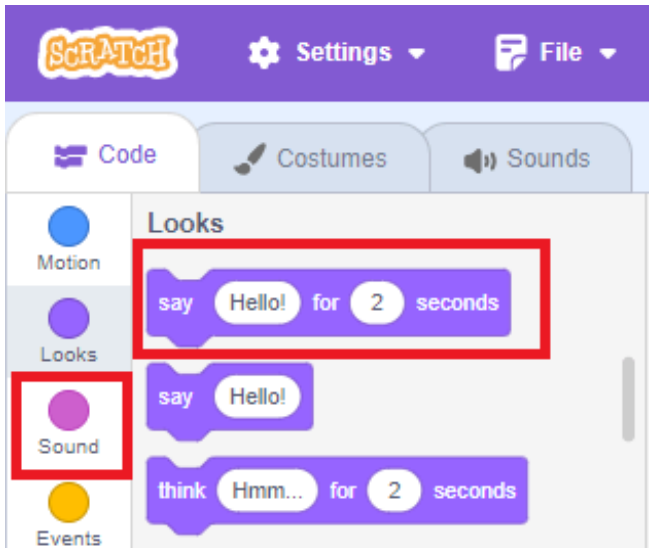
2. **Change Movement Distance**:
   - The number of steps determines how far the sprite moves. Click on the number `10` in the block and replace it with a different value, like `50`. When you click the block again, the cat will move

a greater distance. Whenever you see a white input area in a block, you can customize its value.

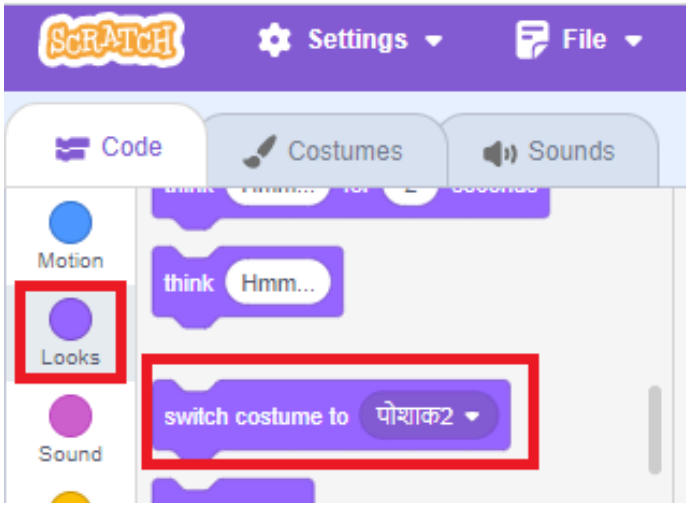
3. **Rotate the Sprite**:
   - Use the **turn clockwise 15 degrees** block to rotate the cat sprite. You can adjust the angle by changing the number in the block. After rotating, the sprite will face its new direction, and clicking the **move 10 steps** block will move it along this path.



4. **Repositioning the Sprite**:
   - If the cat reaches the edge of the Stage, drag it back by clicking and holding it with the mouse, moving it to a new spot, and releasing the button to place it.
5. **Experiment with Sound Blocks**:
   - Click the pink **Sound** button in the Blocks Palette.
   - Click the **play sound Meow until done** block to hear the cat's meow sound, provided your speakers are on.



6. **Try the Looks Blocks**:
    - Click the purple **Looks** button beside the Blocks Palette.
    - Use the **next costume** block to change the cat's appearance. You'll see its legs move as if it's running in place. Costumes are alternate appearances for sprites, and the cat sprite includes two costumes showing different leg positions.

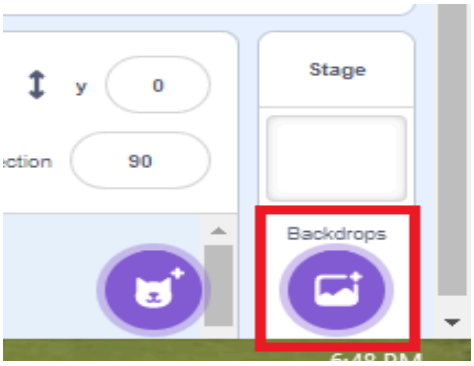


Tablet Users: If you're using a tablet, you can tap blocks instead of clicking them.Take a few minutes to experiment with these blocks and explore the different commands you can give to the cat sprite. More details about each block will be explained as you progress.

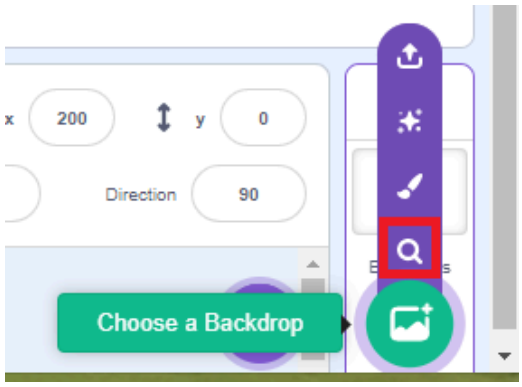
6. Changing the Backdrop

Before creating your first program, you can customize the Stage by selecting a more engaging background. In Scratch, the background is referred to as a **backdrop**. Follow these steps to change it:

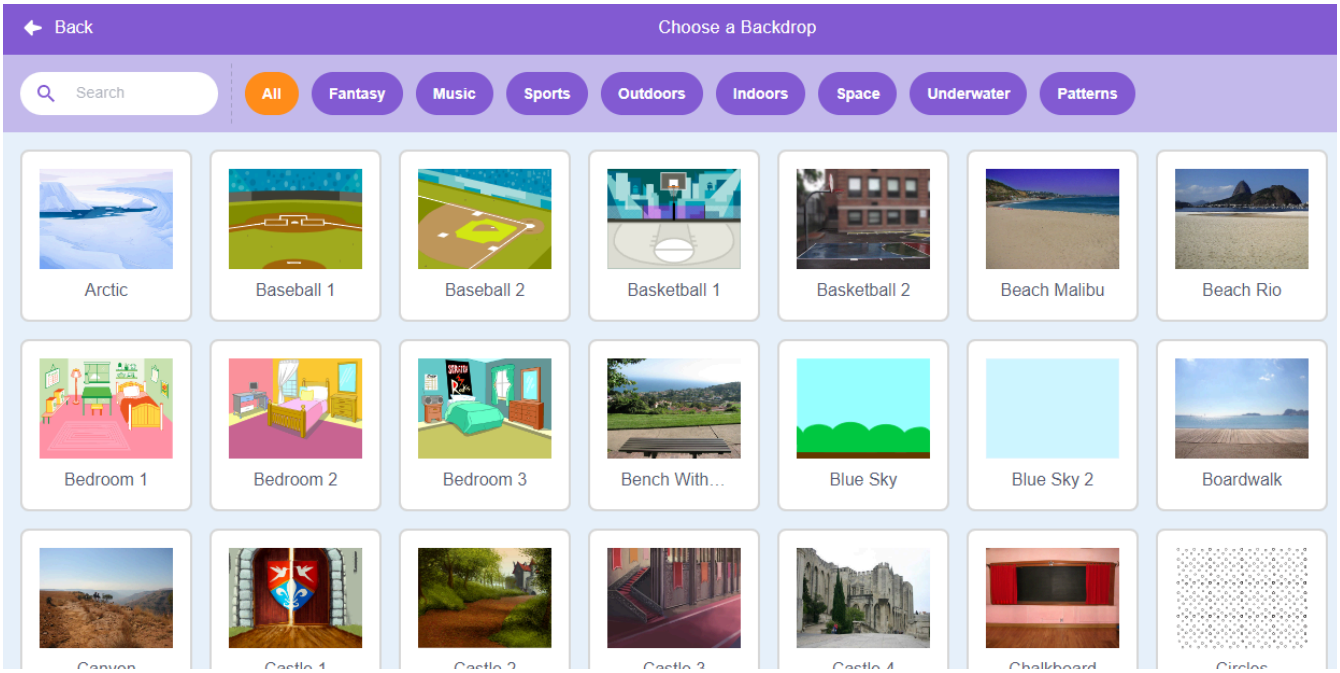
1. **Access the Backdrop Options**:
    - To the right of the Sprite List, you'll see the panel for the Stage. At the bottom of this panel, hover over the round button to reveal backdrop options.



- Click the **magnifying glass icon** to open the backdrop library. Alternatively, you can click the round button directly to access the library. Other options include painting your own backdrop, selecting a random backdrop, or uploading an image from your computer.



2. **Browse the Backdrop Library**:
   - Once the library opens, you'll see various backdrop categories displayed at the top of the screen. Click any category to view related designs.
   - Use the scrollbar on the right to explore more options.



3. **Choose a Backdrop**:
   - Navigate to the **Outdoors** category.
   - Scroll through the options and select the **Hill** image to set it as your new backdrop.

Now you've personalized your Stage, making it ready for your creative program!

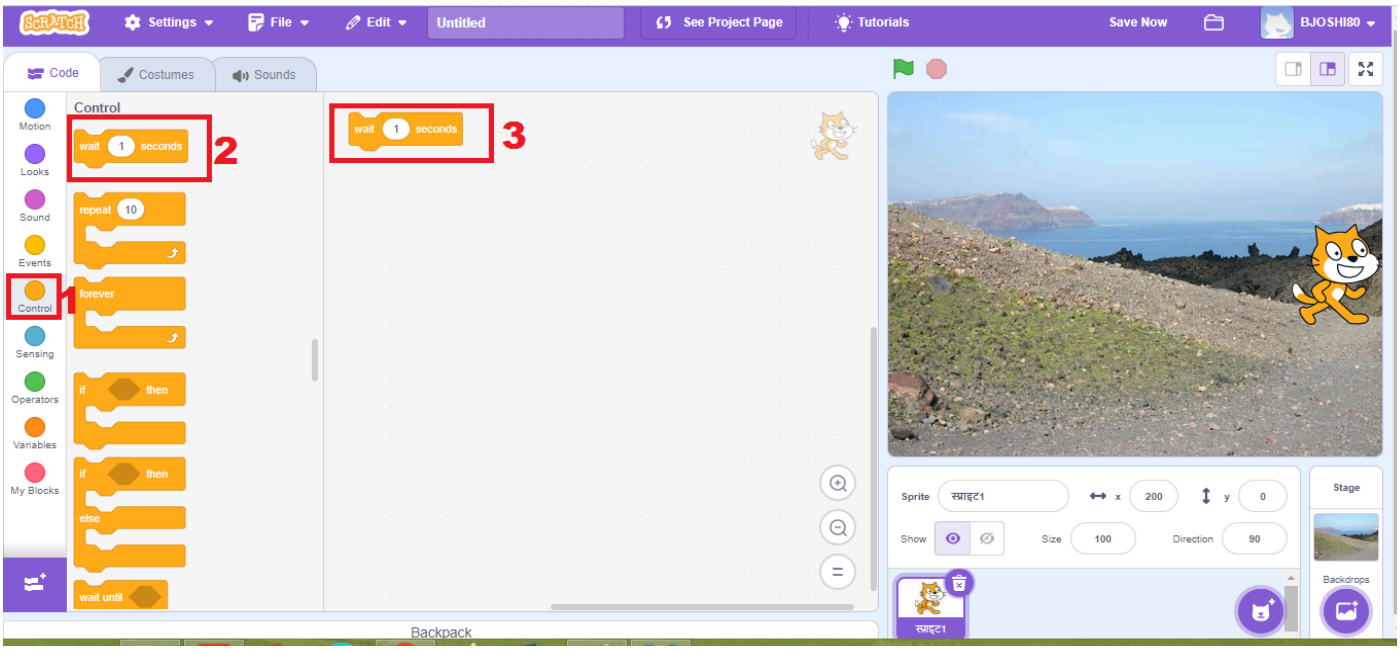
7. Creating Your First Program

When building programs in Scratch, it's important to ensure that blocks snap together properly. If the blocks are not close enough to snap together, they won't work as a single script.

- **Moving and Deleting Blocks**:
  - If you click and drag a block that is connected to others above it in the Code Area, it will detach, taking all the blocks underneath it as well.
  - You can rearrange blocks within the Code Area freely, but if you drag a block into the Blocks Palette, it will be deleted.

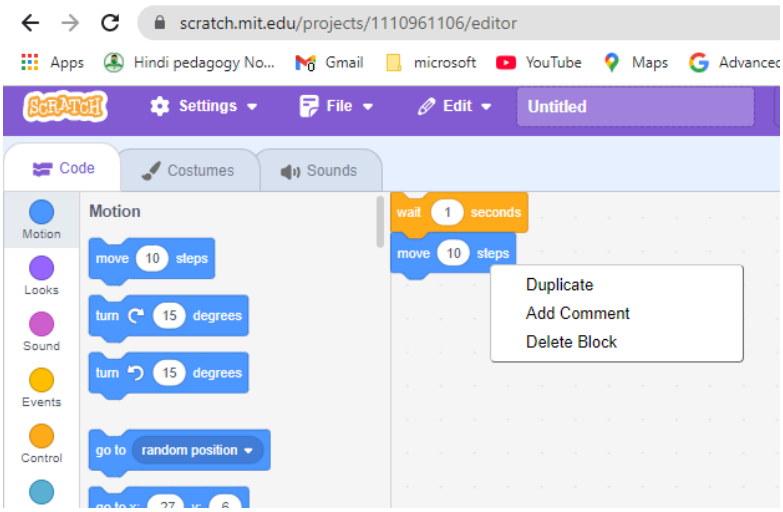
Steps to Create Your First Program:

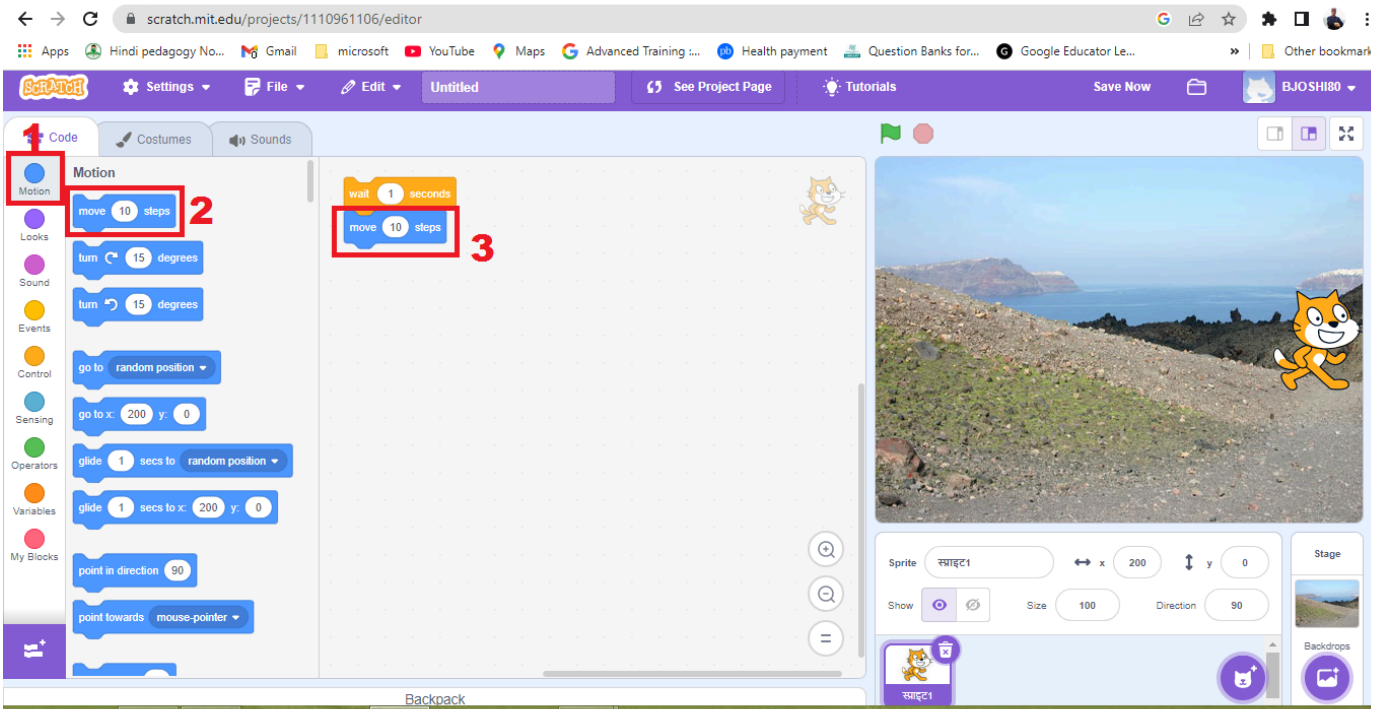
1. **Add a Control Block**:
   - Click the **Control** button on the left side of the Blocks Palette.
   - Drag the **wait 1 seconds** block into the Code Area and attach it below the existing blocks. This block pauses the program for a moment, creating a delay that makes the cat's movements visible.
   - If you want the cat to move faster, reduce the delay to **0.5 seconds**.



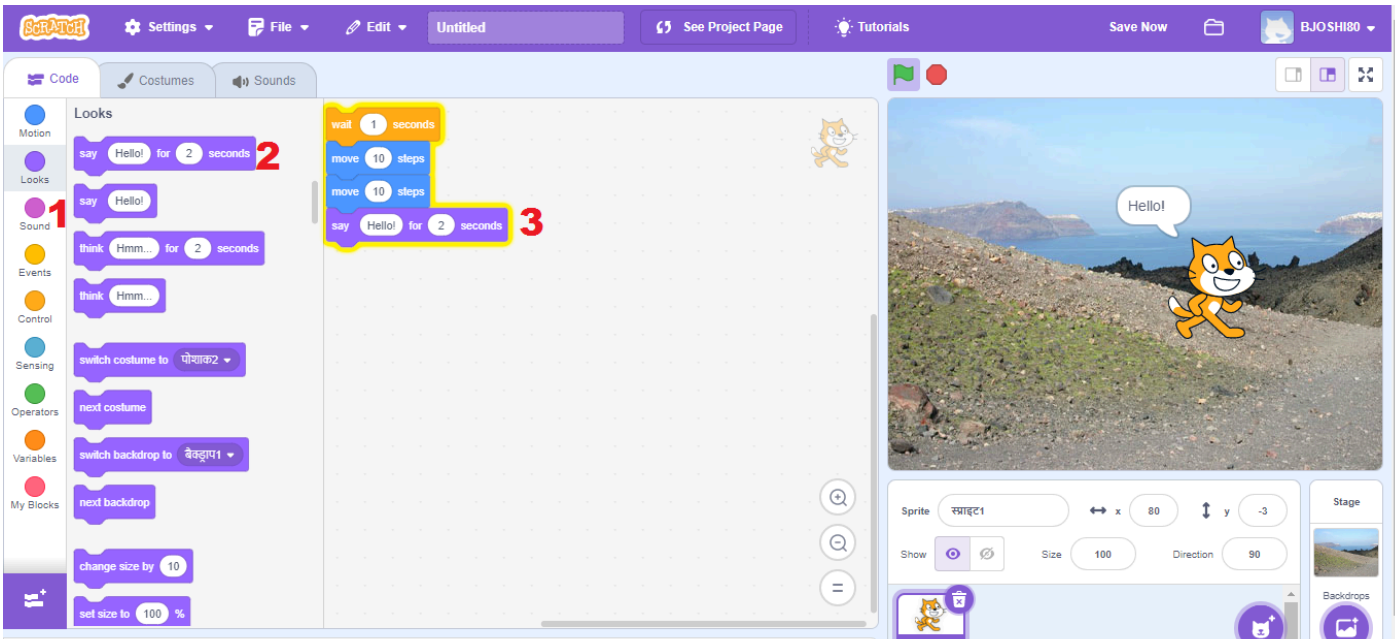
2. **Duplicate Blocks**:

- Right-click (or tap and hold on touchscreens) the **move 10 steps** block. Select **Duplicate** from the menu. This will copy the block along with any blocks connected below it, such as the **wait** block.
- Drag the duplicate to the bottom of the script and snap it into place. Repeat this step to make the cat move further.





3. **Add a Speech Bubble**:
   - Click the **Looks** button in the Blocks Palette.
   - Drag the **say Hello! for 2 seconds** block into the Code Area and attach it to the end of your script.
   - Click on the text “Hello!” to edit it, and change it to “Phew!” to make the cat express relief after its walk.

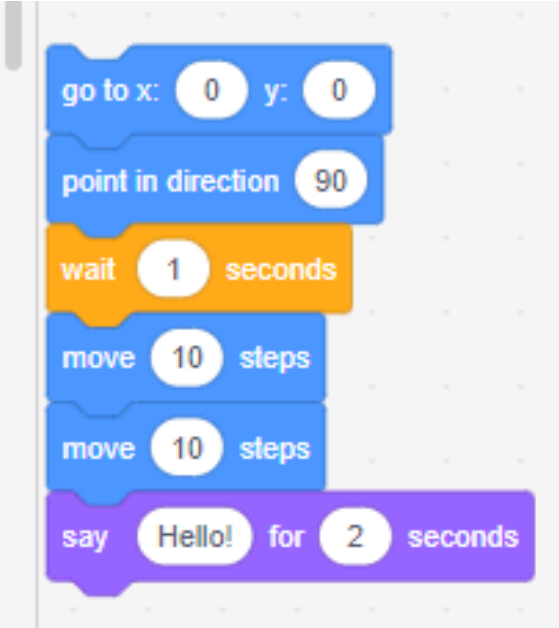


Note: Although repeating blocks works for now, later you'll learn more efficient ways to simplify scripts.

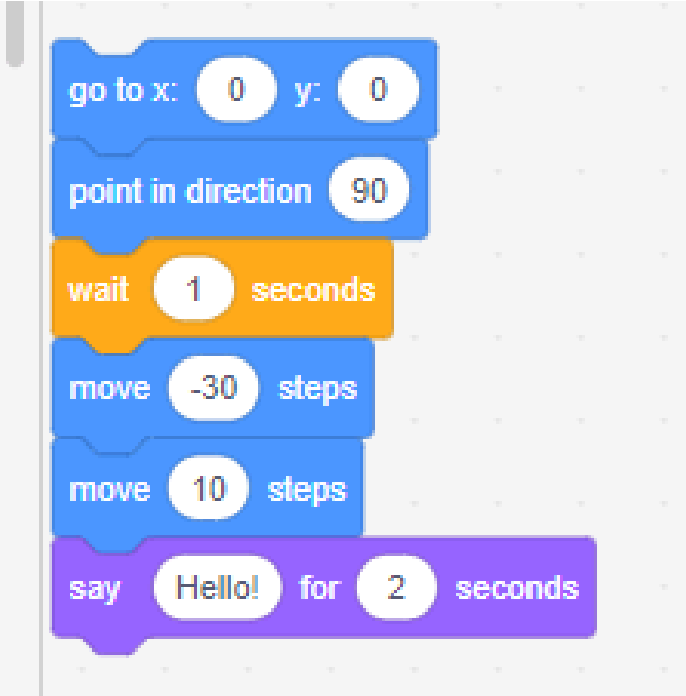
4. **Run Your Script**:
   - To execute your program, click any joined blocks in the Code Area. Scratch will carry out the instructions sequentially, starting from the top and working downward.

5. **Reset the Starting Position**:
   - Each time you run the script, the cat continues from where it last stopped, potentially ending up in awkward positions (like walking on its head). To fix this, add the following blocks to the top of your script:
     - Click the **Motion** button in the Blocks Palette.

- Drag the **point in direction 90** block into the Code Area to make the cat face the right way.
- Drag the **go to x:0 y:0** block into the Code Area and set both coordinates to **0**. This ensures the cat starts from the center of the Stage each time the script runs.



6. **Experiment with Negative Numbers**:
   - Try entering a negative value in the **move 10 steps** block. This will make the cat move backward. For fun, see if you can make the cat walk backward up the hill!



Your First Program

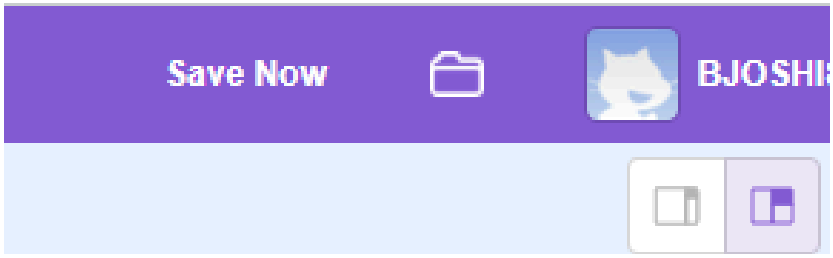
With a few blocks and some experimentation, you've successfully created your first Scratch program! Keep exploring to see what else you can make the cat do.

8. Saving Your Project

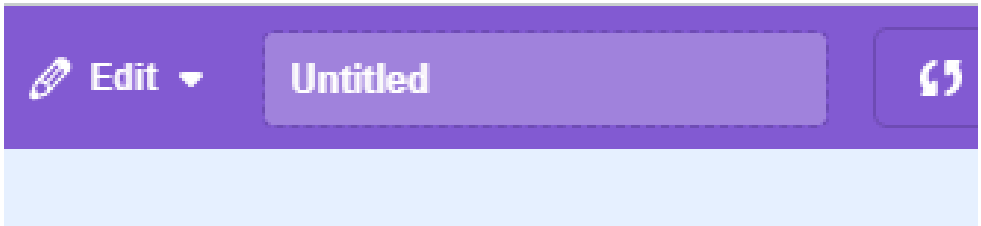
A Scratch project contains all the sprites, scripts, and backdrops that you've used. It's important to save your work so you can revisit and modify it later. Here's how to manage your projects effectively:

Saving in Scratch 3 (Online Version)

- **Automatic Saving**:
  In the online version of Scratch 3, your work is saved automatically. If changes haven't been saved, a **"Save Now"** link will appear in the blue bar at the top of the screen, next to the folder icon where you can access your projects.



- **Renaming Your Project**:
  By default, projects are saved with the name **Untitled** followed by a number. You can give your project a meaningful name by editing the text box in the blue bar above the Code Area, to the right of the tabs.

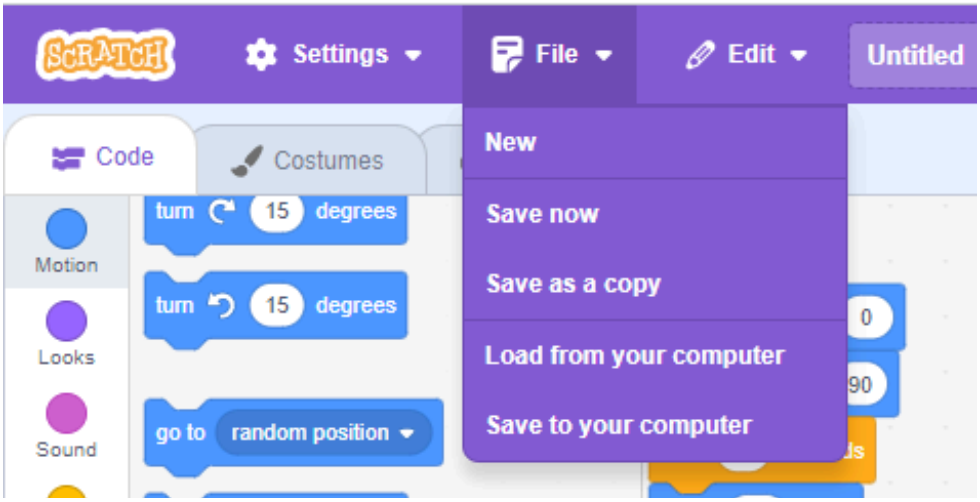


Additional Saving Options in the File Menu

To manually save or manage your project, use the options in the **File** menu, located above the Blocks Palette. These include:

1. **Save as a Copy**:

- This creates a duplicate of your project under a new name, leaving the original untouched.
- Use this option if you want to experiment with changes without affecting the working version of your project.

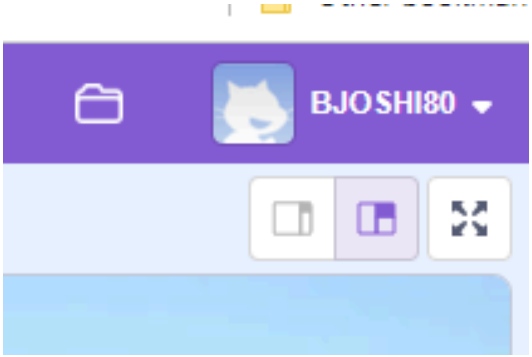


2. **Save to Your Computer**:
   - This option allows you to download your project as a file to your computer.
   - It's especially useful if you're working offline or want to ensure your progress is safe in case of an internet issue.
3. **Load from Your Computer**:
   - If you've previously downloaded a Scratch project file or created one using the offline version of Scratch, you can upload it back to the Scratch website to continue editing or share it online.

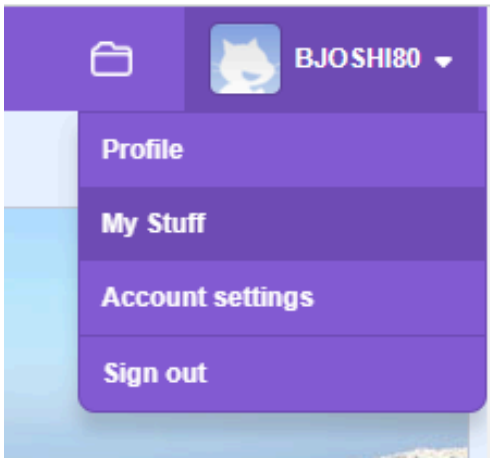
9. Opening Projects

Accessing Your Projects on the Scratch Website

1. **Finding Your Projects**:
   - Click the **folder icon** next to your username in the top-right corner of the screen.

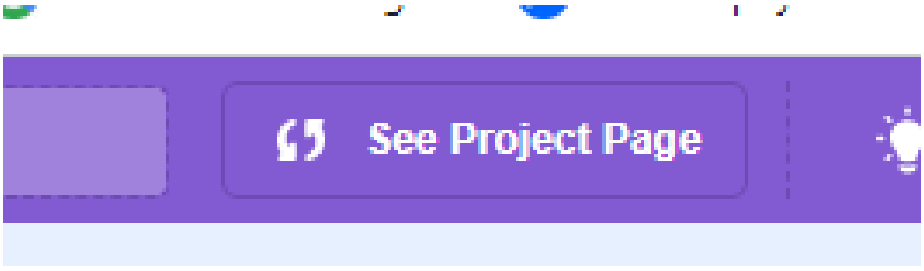


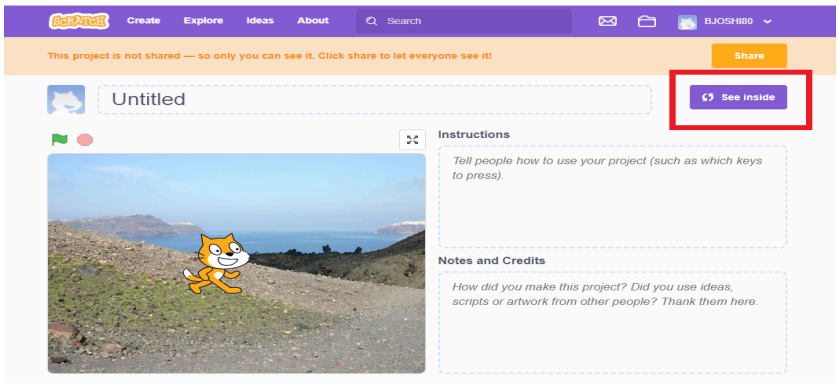
- Alternatively, click your username and select **My Stuff** from the dropdown menu.



- If "Sign in" appears instead of your username, click it to log in first.

2. **My Stuff Section**:
   - This area lists all your projects, with the most recently edited ones at the top.
   - Use the scrollbar on the right side of your browser to browse through your projects. If needed, click **Load More** at the bottom of the list.
3. **Opening Projects**:
   - **Edit Mode**: Click the **See inside** button on a project to open it in the editor.
   - **Run Mode**: To simply play a project, click on its **screenshot**.





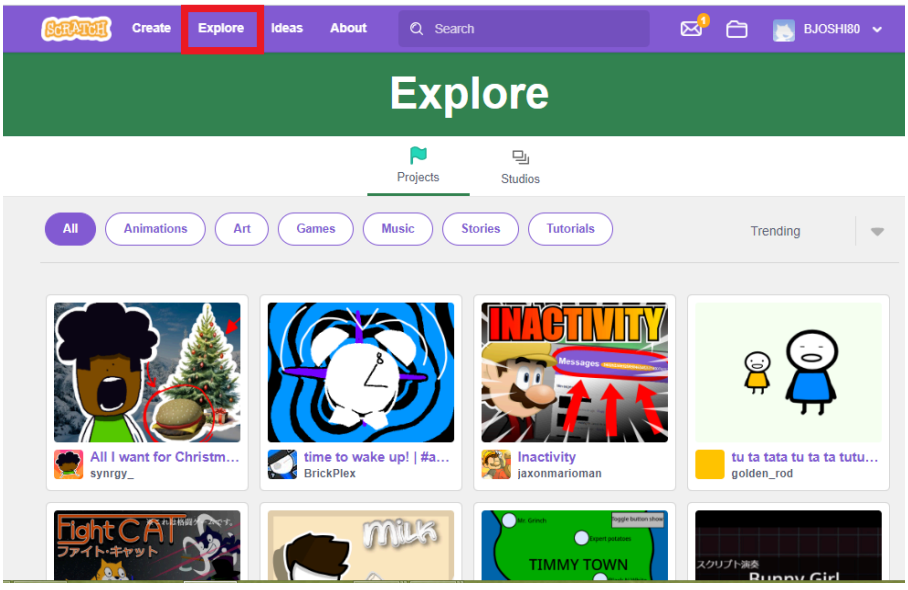
4. **Managing Projects**:
    - To delete unwanted projects, click the **Delete** link on the right side of the project in the My Stuff section.
    - If you accidentally delete a project, navigate to the **Trash** folder on the left and click **Put back** to restore it.
    - Note: You can't delete a shared project unless you unshare it first.

Opening Projects in the Offline Version of Scratch

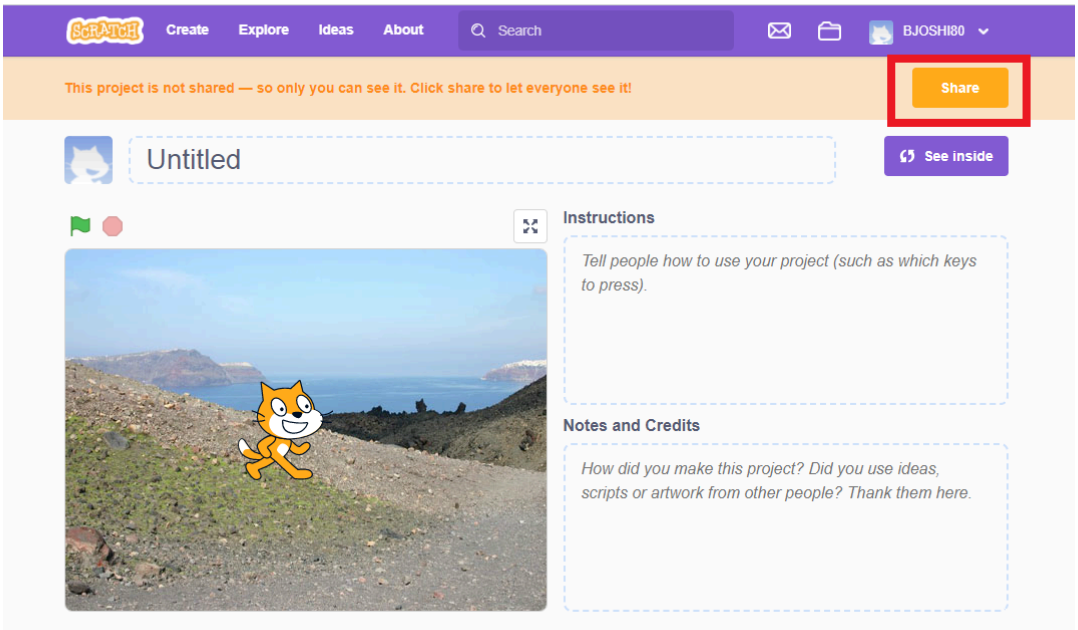
- Open the File menu at the top of the screen and select **Open** to access saved projects on your computer.

Opening Shared Projects on the Scratch Website

1. **Browsing Shared Projects**:
    - Visit Scratch's website and click **Explore** at the top of the page.
    - Use the category buttons or the dropdown menu to find projects by **trending**, **popular**, or **recent** filters.



2. **Interacting with Shared Projects**:
    - Click a project to go to its page.
    - **Run the Project**: Use the green flag button either within the project display or just above it to play the project.
    - **Learn from the Code**: Click the **See inside** button in the top-right corner to open the project in the editor and explore how it works.
3. **Sharing Your Own Projects**:
    - Click the **Share** button on a project to make it available for others to view, use, and remix.



By following these steps, educators can introduce Scratch programming effectively, enabling students from Classes 3 to 10 to learn coding concepts while exploring their creativity.

10. Sample project idea

Here are some beginner-friendly Scratch project ideas with sample scripts that students can use to learn Scratch programming and build their own creative projects. Each program introduces essential Scratch concepts such as movement, loops, animations, and user interaction.

1. Moving a Sprite (Basic Movement)

Objective: Make the cat move and bounce off the edges of the stage.

Code Steps:

- Blocks:
    1. `when green flag clicked`
    2. `forever`
        - `move 10 steps`
        - `if on edge, bounce`
- Script:
    1. Drag a "when green flag clicked" block from the Events category.
    2. Attach a "forever" loop from the Control category.
    3. Inside the loop, add a "move 10 steps" block and an "if on edge, bounce" block from the Motion category.

What Happens: The sprite moves in a straight line, bounces when it hits the edge, and keeps moving indefinitely.

2. Talking Sprite (Speech Animation)

Objective: Make a sprite say a greeting and react.

Code Steps:

- Blocks:
    1. `when green flag clicked`
    2. `say Hello! for 2 seconds`
    3. `say How are you? for 2 seconds`
- Script:
    1. Drag a "when green flag clicked" block.
    2. Attach a "say Hello! for 2 seconds" block from the Looks category.
    3. Attach another "say How are you? for 2 seconds" block.

What Happens: The sprite speaks two messages one after the other, creating a short conversation.

3. Sprite Following the Mouse (Interaction)

Objective: Make a sprite follow the mouse pointer wherever it goes.

Code Steps:

- Blocks:
    1. `when green flag clicked`
    2. `forever`
        - `point towards [mouse pointer]`
        - `move 5 steps`
- Script:
    1. Drag a "when green flag clicked" block.
    2. Add a "forever" loop.
    3. Inside the loop, add "point towards [mouse pointer]" and "move 5 steps" blocks from the Motion category.

What Happens: The sprite moves towards the mouse pointer, creating an interactive experience.

4. Simple Animation (Dancing Sprite)

Objective: Make the sprite appear as if it is dancing using costumes.

Code Steps:

- Blocks:
    1. `when green flag clicked`
    2. `forever`
        - `next costume`
        - `wait 0.5 seconds`
- Script:
    1. Drag a "when green flag clicked" block.
    2. Add a "forever" loop.
    3. Inside the loop, add a "next costume" block from the Looks category.
    4. Add a "wait 0.5 seconds" block from the Control category.

What Happens: The sprite switches between its costumes, creating a dancing animation.

5. Creating a Simple Game (Click the Balloon)

Objective: Make a balloon sprite disappear when clicked.

Code Steps:

- Blocks:
    1. `when green flag clicked`
    2. `go to [random position]`
    3. `show`
    4. `forever`
        - `go to [random position]`
        - `wait 1 seconds`
    5. `when this sprite clicked`
        - `hide`
        - `play sound [pop]`
- Script:
    1. Drag a "when green flag clicked" block.
    2. Add "go to [random position]" and "show" blocks from the Motion and Looks categories.

3. Add a "forever" loop containing "go to [random position]" and "wait 1 seconds".
4. Add another script: "when this sprite clicked", followed by "hide" and "play sound [pop]".

What Happens: A balloon moves randomly across the screen. When clicked, it disappears and plays a popping sound.

6. Color-Changing Sprite

Objective: Change the sprite's color every time it moves.
Code Steps:

- **Blocks:**
  - `when green flag clicked`
  - `forever`
    - `move 10 steps`
    - `if on edge, bounce`
    - `change color effect by 25`

Script:
scratch

```
when green flag clicked

forever

  move 10 steps

  if on edge, bounce

  change color effect by 25
```

What Happens:
The sprite moves across the screen, bounces on the edges, and changes color continuously.

7. A Countdown Timer

Objective: Display a countdown from 10 to 0.
Code Steps:

- **Blocks:**
  - `when green flag clicked`
  - `set [variable: timer] to 10`

    - repeat 10
        - say [timer] for 1 second
        - change [timer] by -1
    - say "Time's Up!" for 2 seconds

Script:
scratch

```
when green flag clicked

set [timer] to 10

repeat 10

  say (timer) for 1 second

  change [timer] by -1

say "Time's Up!" for 2 seconds
```

What Happens:
The sprite displays a countdown timer, saying each number for one second, and ends with a message.

8. Build a Virtual Piano

Objective: Create a piano that plays different notes when clicked.
Code Steps:

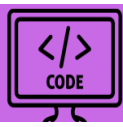
- Blocks:
    - when this sprite clicked
        - play note [60] for 0.5 beats
          (Repeat this for different sprites or keys with different notes.)

What Happens:
Clicking on different sprites (keys) plays various musical notes.

9. Flying Bat Animation

Objective: Make a bat sprite flap its wings as it moves.
Code Steps:

- **Blocks:**
    - when green flag clicked



    - forever
        - move 10 steps
        - next costume
        - if on edge, bounce

Script:
scratch

```
when green flag clicked

forever

  move 10 steps

  next costume

  if on edge, bounce
```

What Happens:
The bat sprite moves across the screen, bouncing off edges while flapping its wings.

10. Sprite Follows a Path

Objective: Make a sprite follow a specific path drawn on the stage.
Code Steps:

- **Blocks:**
    - when green flag clicked
    - forever
        - move 5 steps
        - if touching [color]
            - turn right 15 degrees

Script:
scratch

```
when green flag clicked

forever

  move 5 steps

  if <touching [color]> then
```

```
turn right 15 degrees
```

What Happens:
The sprite moves along a colored path drawn on the backdrop.

11. Random Sprite Movement

Objective: Make a sprite move randomly across the screen.
Code Steps:

- **Blocks:**
  - `when green flag clicked`
  - `forever`
    - `point in direction [pick random 1 to 360]`
    - `move 10 steps`

Script:
scratch

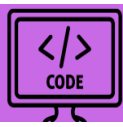
```
when green flag clicked

forever

  point in direction (pick random 1 to 360)

  move 10 steps
```

What Happens:
The sprite moves in random directions on the stage.

12. Creating a Maze Game

Objective: Move the sprite through a maze to reach the goal.
Code Steps:

- **Blocks:**
  - `when green flag clicked`
    - `go to x: (-200) y: (0)`
  - `forever`
    - `if <key [up arrow] pressed?>`
      - `change y by 5`
    - `if <key [down arrow] pressed?>`
      - `change y by -5`



        - if <key [right arrow] pressed?>
            - change x by 5
        - if <key [left arrow] pressed?>
            - change x by -5
        - if touching [color]
            - say "Oops!"
            - go to start

What Happens:
Players control the sprite through a maze while avoiding obstacles.

13. Sprite Cloning

Objective: Create multiple clones of a sprite.
Code Steps:

- **Blocks:**
    - when green flag clicked
    - forever
        - create clone of [myself]
        - wait 1 second
    - when I start as a clone
        - go to [random position]
        - delete this clone

Script:
scratch

```
when green flag clicked

forever

  create clone of [myself]

  wait 1 second

when I start as a clone

  go to [random position]

  delete this clone
```

What Happens:
The sprite creates clones that appear randomly on the stage.

14. Animated Story

Objective: Create a story using multiple sprites and backdrops.
Code Steps:

- **Blocks for Sprite 1:**
  - `when green flag clicked`
    - `say "Once upon a time..." for 2 seconds`
    - `switch backdrop to [next backdrop]`
- **Blocks for Sprite 2:**
  - `when green flag clicked`
    - `wait 2 seconds`
    - `say "Let's go on an adventure!" for 2 seconds`

What Happens:
Sprites interact with dialogue and changing backdrops to tell a story.

15. Space Shooter Game

Objective: Create a basic shooting game.
Code Steps:

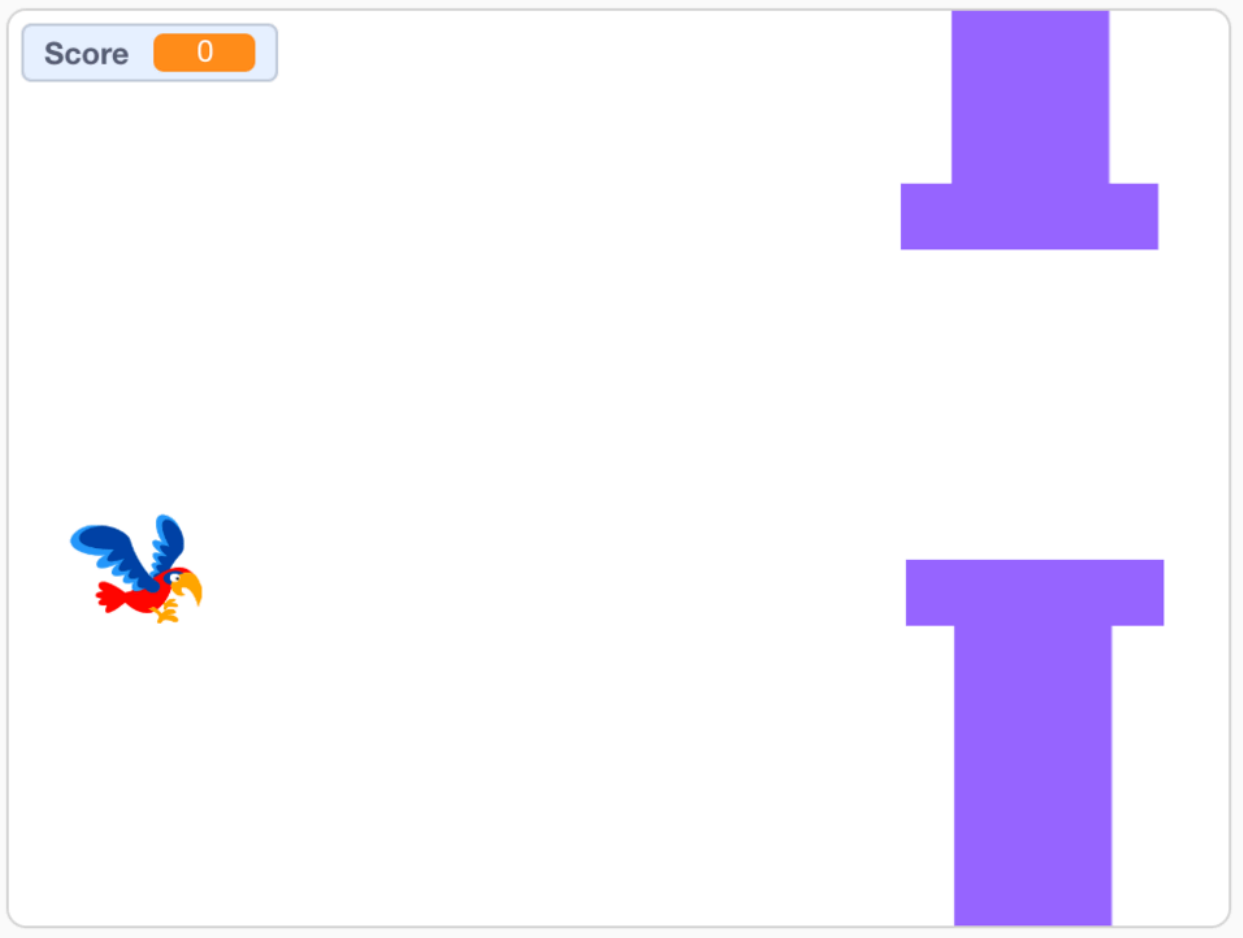
- **Spaceship:**
  - `when green flag clicked`
    - `forever`
      - `if <key [left arrow] pressed?>`
        - `change x by -10`
      - `if <key [right arrow] pressed?>`
        - `change x by 10`
      - `if <key [space] pressed?>`
        - `create clone of [bullet]`
- **Bullet:**
  - `when I start as a clone`
    - `repeat until <touching edge>`
      - `move 10 steps`
    - `delete this clone`

What Happens:
Players control a spaceship that shoots bullets to hit enemies.

These programs not only serve as a guide for Scratch programming but also encourage students to use their creativity to expand on these ideas!

11.Some more interesting running projects

1.How to Make a Flappy Bird in Scratch



Here's how you can make a simple Flappy Bird-style game in Scratch, step by step:

Step 1: Choose a Character

First, select a sprite to act as the "bird" that will fly through the obstacles. It's best to use a sprite with two costumes so you can create a flying effect. For this example, we'll use the **Parrot** sprite.

To ensure the sprite fits well between the pillars, reduce its size. In the lower-right corner of the editor, set the **Size** property to 30.

Step 2: Add Code for Flying Motion

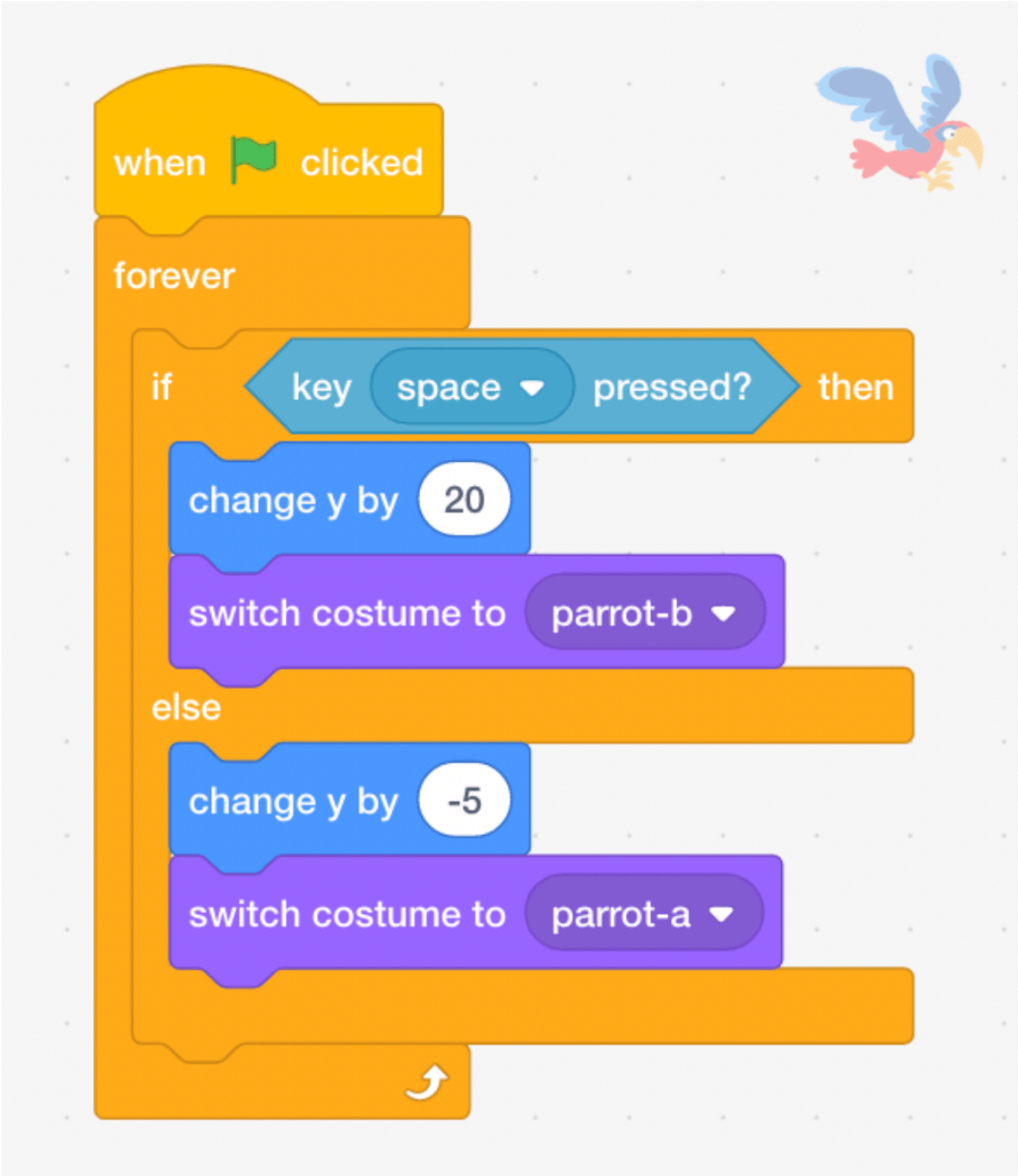
Now, let's program the "flappy" movement for the parrot.

1. Select the **Parrot** sprite by clicking on its icon in the sprite list.
2. Open the **Code** tab and start by adding these blocks:
    - Drag the **"when green flag clicked"** block from the **Events** category.
    - From the **Control** section, attach a **"forever"** block beneath it.
3. Inside the forever loop, we'll use an **"if/then/else"** block to control the parrot's motion.
    - The movement will be triggered by the space bar. So, drag the **"key [space] pressed?"** block from the **Sensing** category and place it in the condition of the **if/then** statement.
4. **What happens when the space bar is pressed?**
    - Add a **"change y by [20]"** block from **Motion** to make the bird move upward.
    - To simulate flapping, switch the sprite's costume. Use the **"switch costume to [parrot-b]"** block from the **Looks** category.
5. **What happens when the space bar is NOT pressed?**
    - Inside the **else** section, add a **"change y by [-5]"** block to make the parrot fall slightly.
    - To maintain the flapping effect, switch the costume back using **"switch costume to [parrot-a]"**.

Complete Script for Flying Motion

scratch

```
when green flag clicked
forever
  if <key [space] pressed?> then
    change y by 20
    switch costume to [parrot-b]
  else
    change y by -5
    switch costume to [parrot-a]
  end
```



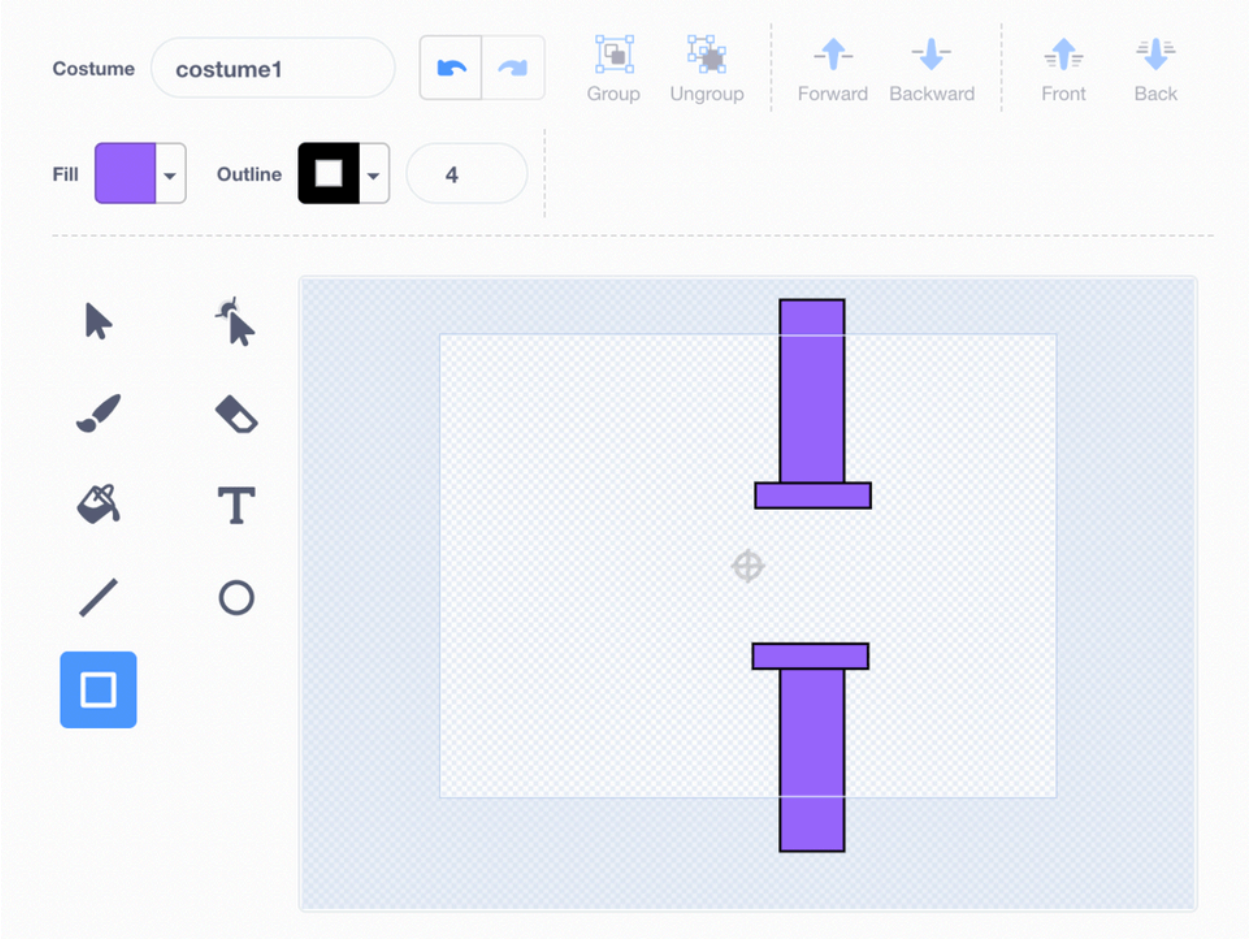
What This Does:

- When you press the space bar, the parrot will move upward and switch to the next costume, giving the appearance of flapping its wings.
- If the space bar is not pressed, the parrot will gradually fall while alternating costumes, mimicking the effect of flying downward.

This code creates the core mechanics of the game. Next, you can work on adding obstacles, scoring, and more to complete the game!

Step 3: Create the Pillars for the Bird to Fly Through

Now that we've programmed the parrot's flappy motion, let's design the pillars as obstacles the bird will fly through. Here's how to do it step by step:



Step 1: Create a Custom Sprite for the Pillars

1. **Add a New Sprite:**
   - Hover over the circle at the bottom-right corner of the screen where you add sprites.
   - Select the Paintbrush icon (third option from the top). This opens the sprite editor, where you can create custom sprites.
2. **Draw the Pillars:**
   - In the sprite editor, use the Rectangle Tool (found at the bottom of the left toolbar).
   - Create two long vertical rectangles to serve as the top and bottom pillars.
   - Add two short horizontal rectangles at the edges of the pillars to connect them and give them a completed look.

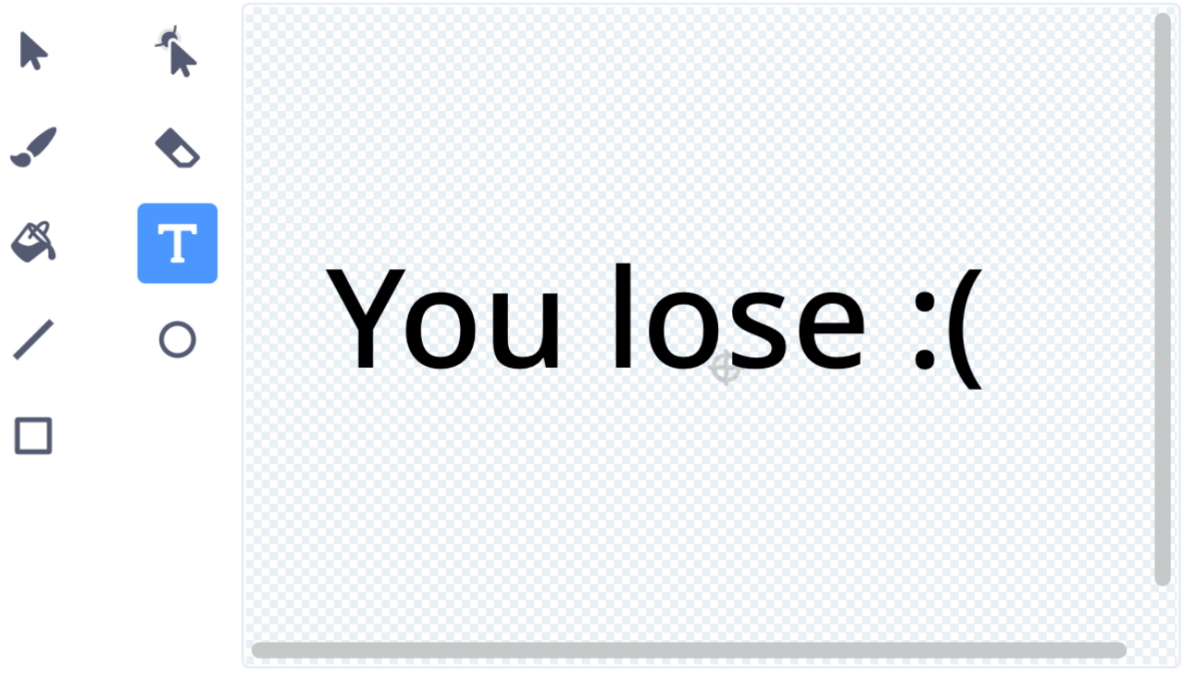
- Position the vertical rectangles in a way that leaves enough space between them for the parrot to fly through.

3. **Set the Pillar Color:**
   - Use the Fill Tool to color the pillars. Green is a good choice to mimic grass or plants.
4. **Adjust the Pillar Size:**
   - Ensure the sprite is proportional to the game. Use the Size field in the sprite settings to scale the pillars if needed.

With this, your Flappy Bird game will now have dynamic, scrolling pillars to fly through! The next step is to add a scoring system and a game-over condition when the bird collides with a pillar.

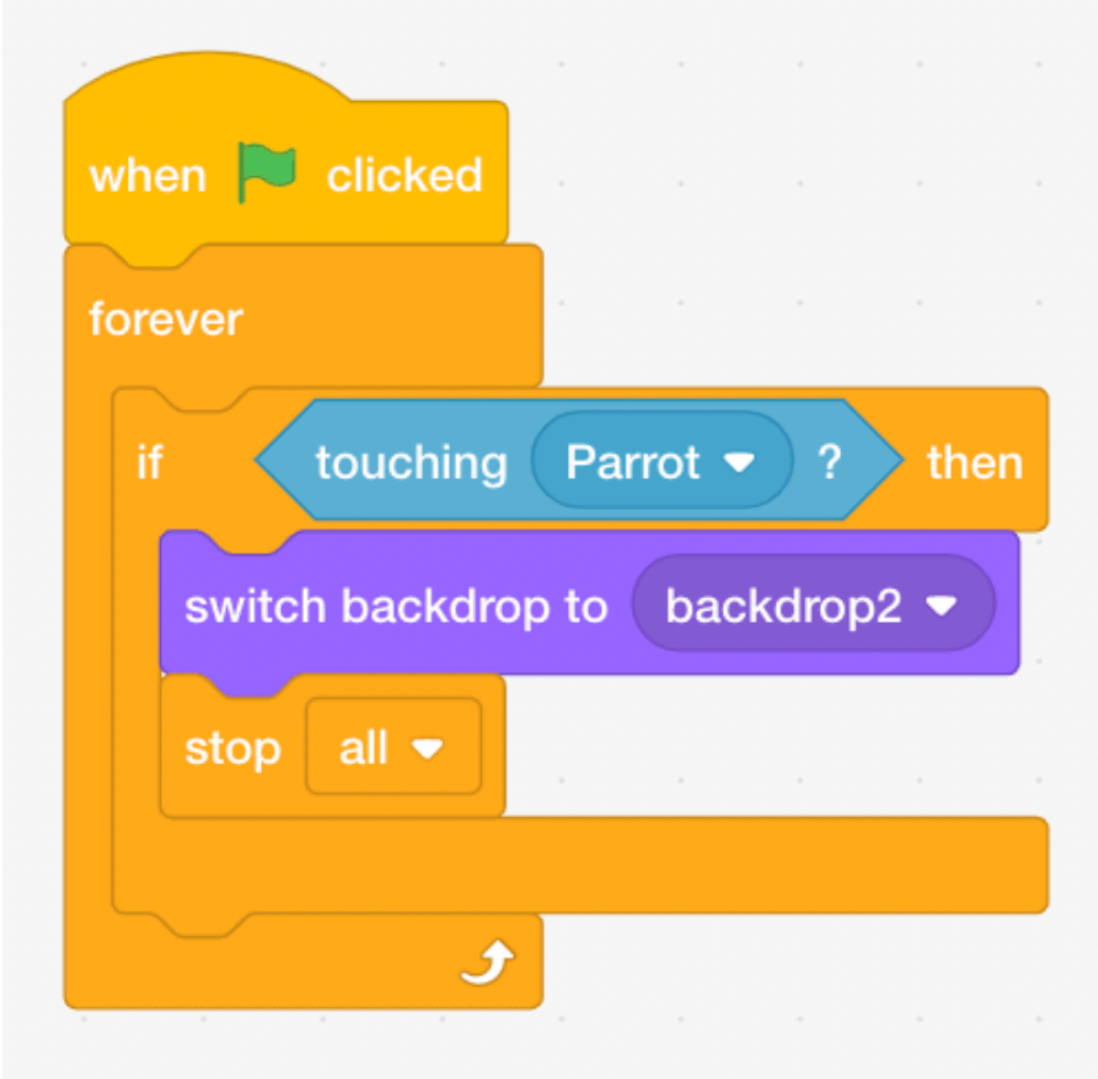
Step 4: Create a Game Over Background

1. Hover over the circle in the bottom-right corner to add a new background.
2. Select the **Paintbrush** option (third from the top).
3. In the backdrop editor, use the **Text Tool** (second option in the left toolbar).
4. Type a message, such as *"Game Over"*, to indicate the end of the game.
5. Customize the text by adjusting the size, color, and font to make it visually appealing.



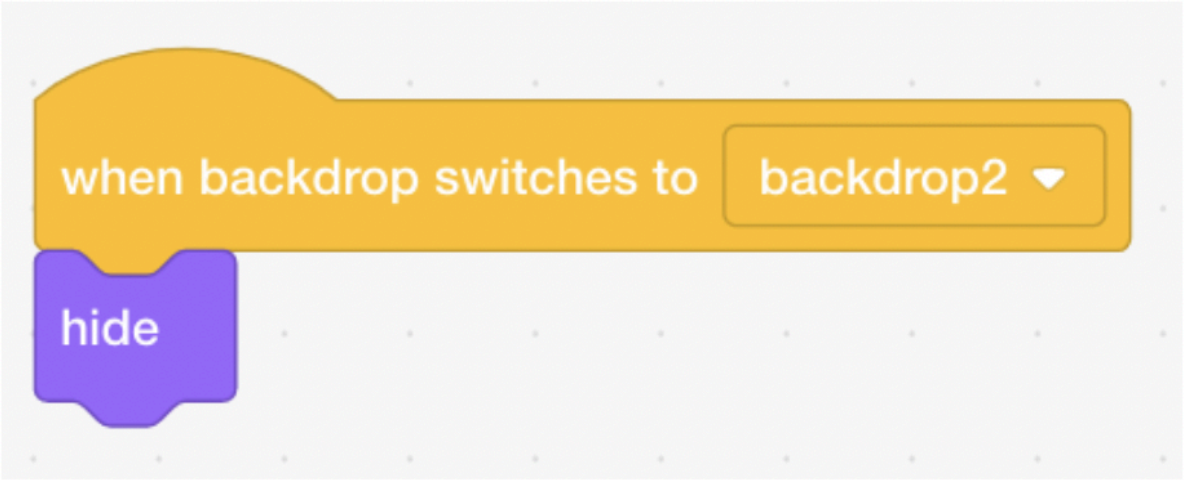
Step 5: End the Game When the Bird Hits a Pillar

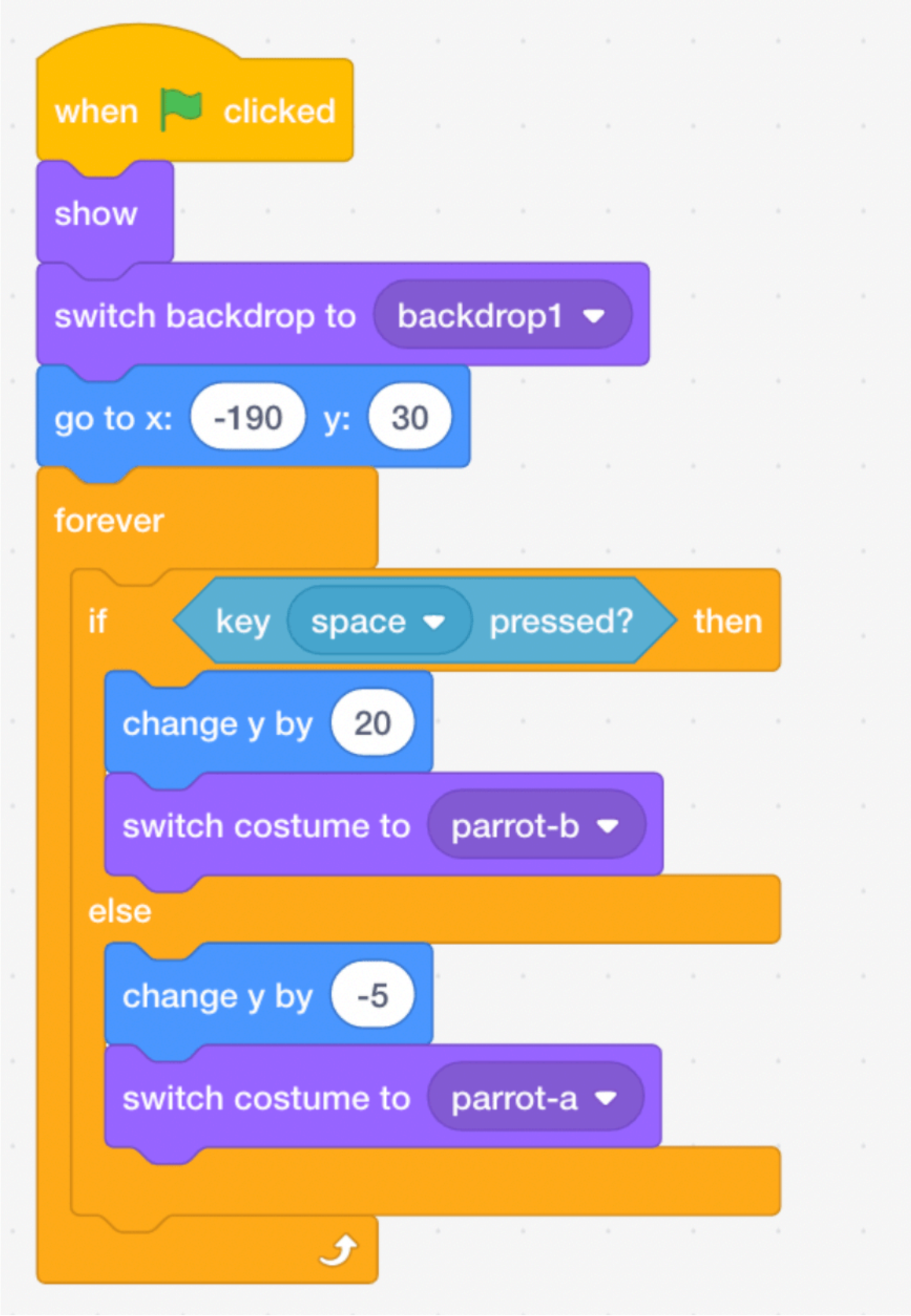
1. Click on the pillars sprite to start adding code.
2. Drag the "when green flag clicked" block from the Events section.
3. Attach a "forever" block from the Control section underneath it.
4. Inside the "forever" block, add an "if/then" block from the Control section.
5. Set the condition of the "if" block to the Sensing block "touching Parrot" (change the dropdown to match your bird sprite).
6. Inside the "if" block, add the following actions:
    - Use the "switch backdrop to [backdrop2]" block from the Looks section to show the game-over background.
    - Add the "stop all" block from the Control section to end the game.



Step 6: Adjust the Parrot for Starting and Ending the Game

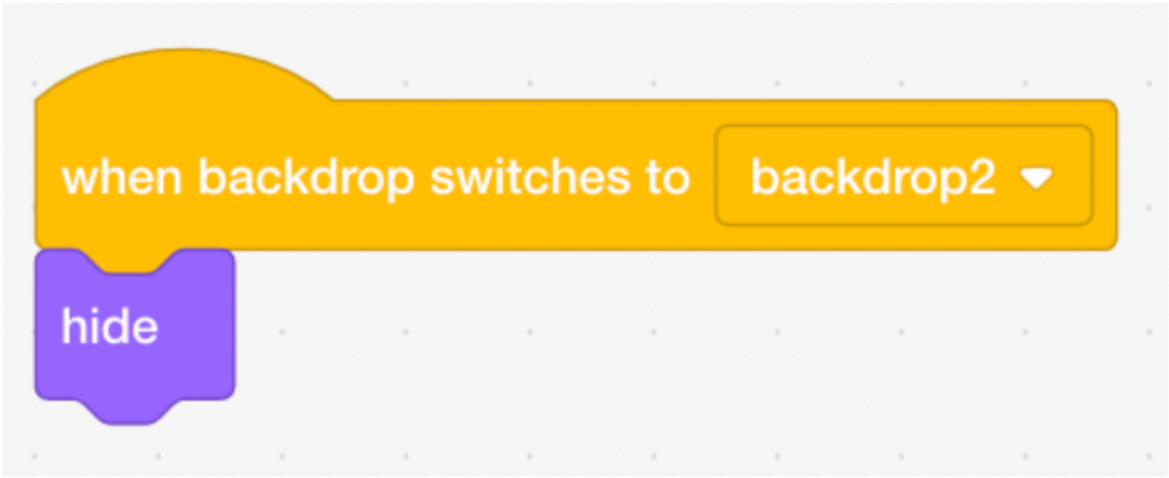
1. **Ending the Game**:
    - Go to the parrot sprite and add the **"when backdrop switches to [backdrop2]"** block from the **Events** section.
    - Attach the **"hide"** block from the **Looks** section to ensure the parrot disappears when the game ends.
2. **Starting the Game**:
    - Add a **"show"** block from the **Looks** section between the **"when green flag clicked"** and **"forever"** blocks. This ensures the parrot appears at the start of every game.
    - Attach the **"switch backdrop to [backdrop1]"** block from the **Looks** section to reset the game's backdrop to the starting background.
3. **Set Starting Position**:
    - After the **"switch backdrop to [backdrop1]"** block, add a **"go to x: [ ] y: [ ]"** block from the **Motion** section.
    - Set the coordinates for the starting position of the parrot. For example:
        - **x = -190**
        - **y = 30**
          This places the parrot on the left side of the screen and near the middle vertically.

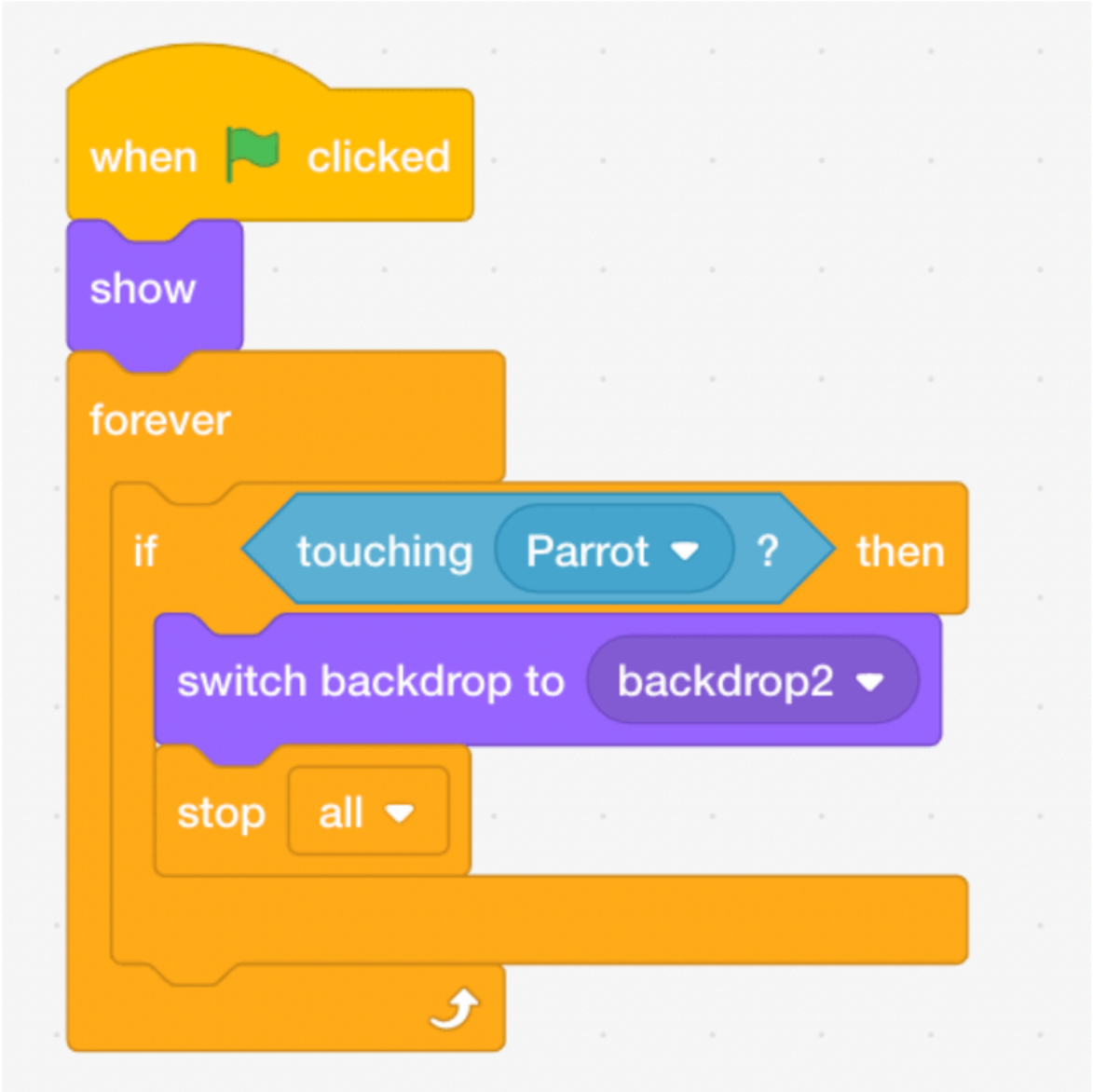

when backdrop switches to
backdrop2
hide



To add additional features to the pillars, start by ensuring they disappear when the game ends. Select the pillar sprite and add the **"when backdrop switches to [backdrop2]"** block from the **Events** section. Attach the **"hide"** block from the **Looks** section to this, so the pillars vanish when the

game-over backdrop is displayed. Additionally, to ensure the pillars reappear when the game restarts, place the **"show"** block from the **Looks** section between the previously added **"when green flag clicked"** and **"forever"** blocks. This setup ensures that the pillars are visible at the start of each game but disappear appropriately when the game ends.

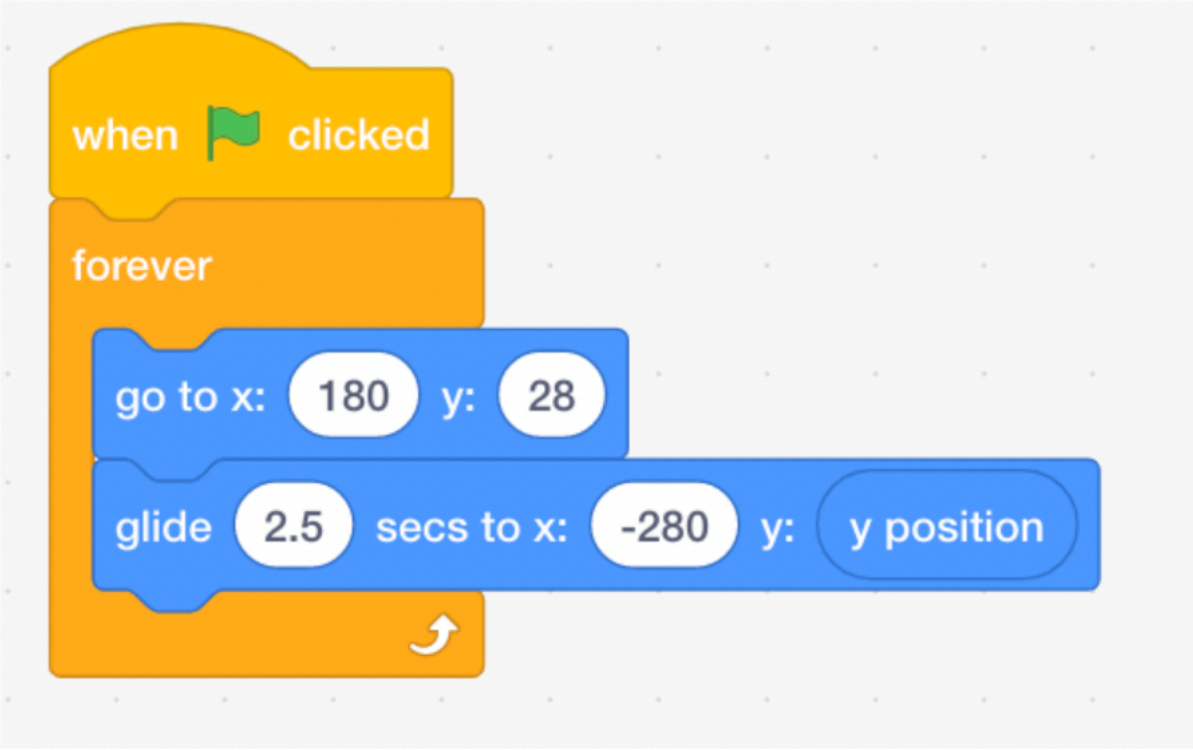

when backdrop switches to backdrop2
hide


when clicked
show
forever
if touching Parrot ? then
switch backdrop to backdrop2
stop all

step7. Get the pillars to move across the screen

To make the pillars move across the screen, we begin by getting them to move from the right to the left side. Start with the **"when green flag clicked"** block from the **Events** section and attach a **"forever"** block from the **Control** section. This ensures the pillars will keep moving as long as the game is running. Inside the **"forever"** block, first place a **"go to x: [ ] y: [ ]"** block from the **Motion** section. Set the x-coordinate to **180** (right side of the screen) and the y-coordinate to **28** to set the starting position of the pillars.

Next, add the **"glide [ ] secs to x: [ ] y: [ ]"** block from the **Motion** section to create the movement. Set the time to **2.5 seconds** (or adjust this for different speeds). The x-coordinate should be set to **-280** (left side of the screen) so the pillars move off-screen to the left, while the y-coordinate should remain unchanged. To ensure the y value remains constant, drag the **"y position"** block from **Motion** into the y space. This setup makes the pillars glide smoothly from right to left, looping back to their starting position once they reach the left side of the screen.



Step 8: Create a Score Variable

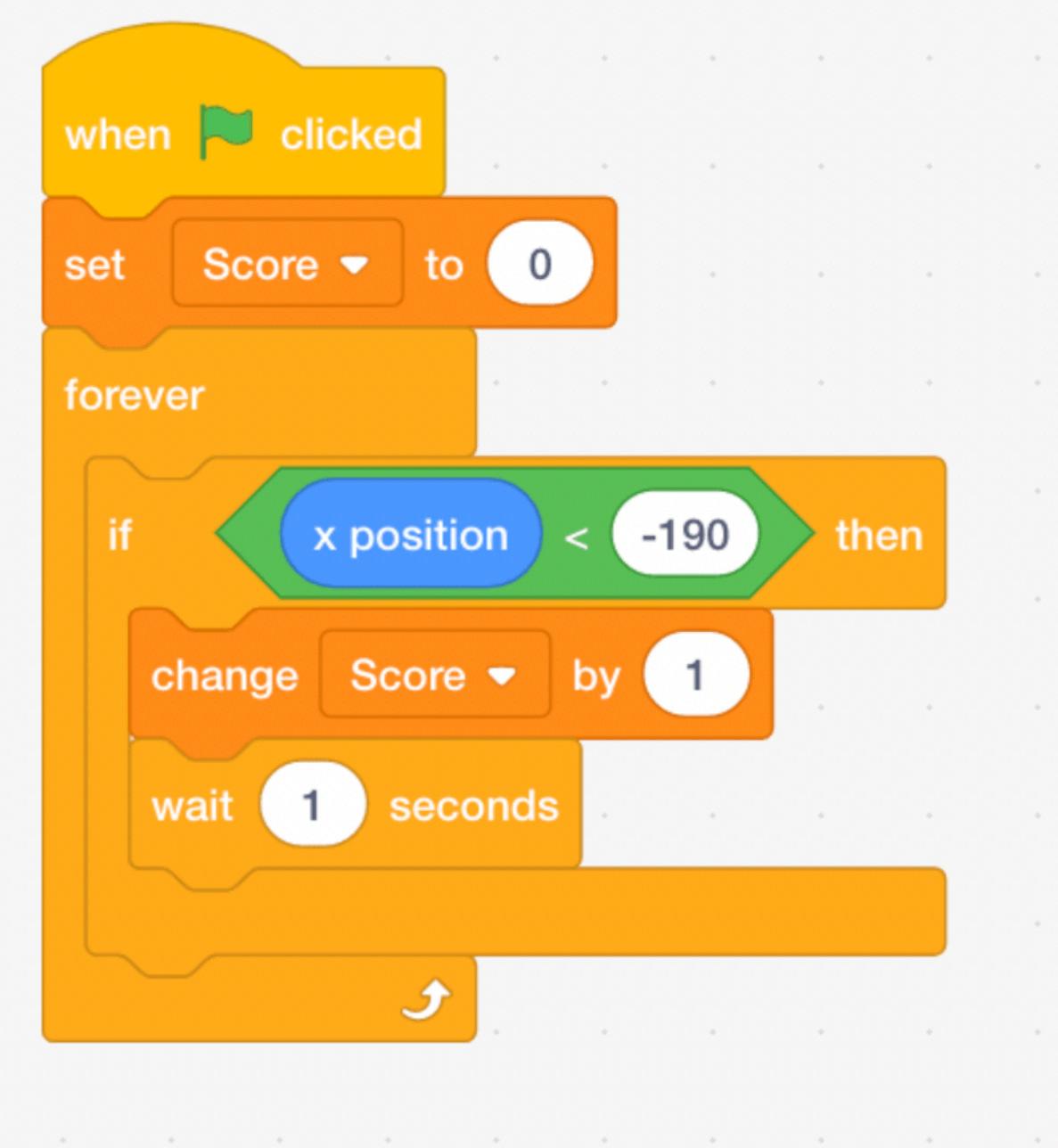
To track the score in your Flappy Bird game, follow these steps:

1. **Create the Score Variable**:

- Go to the **Variables** tab and click **“Make a Variable”**.
- Name the variable **"Score"** and click **OK**. This will create a variable that will keep track of the score throughout the game.

2. **Set the Score to 0 at the Start**:
   - Add a **“when green flag clicked”** block from the **Events** category. This will ensure the score is reset every time the game starts.
   - Inside the green flag block, add a **“set Score to 0”** block from the **Variables** category. This will reset the score to 0 at the start of each game.
3. **Check for Scoring Condition**:
   - Add a **"forever"** block from the **Control** category to continuously check for the scoring condition.
   - Inside the **"forever"** block, add an **“if/then”** block from **Control**.
   - To detect when the parrot flies through the pillars, use the **"x position"** block from **Motion** to get the parrot’s position. Place the **"<"** operator from the **Operators** category to compare the **x position** of the parrot to **-190**.
   - This condition will check when the pillars have moved past the parrot’s location.
4. **Update the Score**:
   - Inside the **if** part of the statement, add a **“change Score by 1”** block from the **Variables** category. This will increase the score by 1 every time the parrot successfully flies through the pillars.
   - After increasing the score, add a **“wait 1 second”** block from **Control**. This will ensure the score updates only once every time the parrot passes through the pillars, avoiding multiple score increments in a short time.

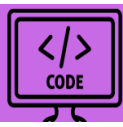
By following these steps, you’ll be able to track the score each time the parrot flies through the pillars successfully.



Final Thoughts

And that's it! You've now created your own version of Flappy Bird in Scratch. By following these steps, you've learned how to make a character move, create obstacles, add a score counter, and handle the game's start and end conditions. This project is a great introduction to game development in Scratch and gives you the foundation to build even more complex games in the future.

We hope you enjoyed building your Flappy Bird game and had fun experimenting with different elements and features. Now, it's your turn to try adding your own unique features or changing the game's mechanics. Whether



it's changing the parrot sprite, adding more levels, or tweaking the pillar speed, the possibilities are endless!

Happy coding!