Coding for STEM:
Simulating falling objects

MAV'24

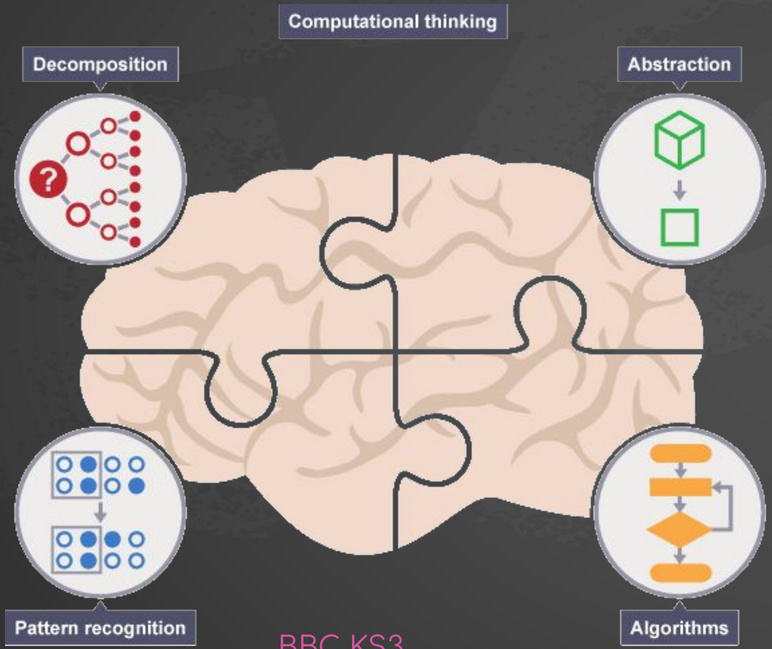
Dec 6, 2024 (Session H11 - 3.10pm)

Dr. Max Stephens - University of Melbourne
Prof. Sebastian Sardiña - RMIT University

POSTERA CRESCAM LAUDE
THE UNIVERSITY OF
MELBOURNE

RMIT
UNIVERSITY

The four cornerstones of Computational Thinking

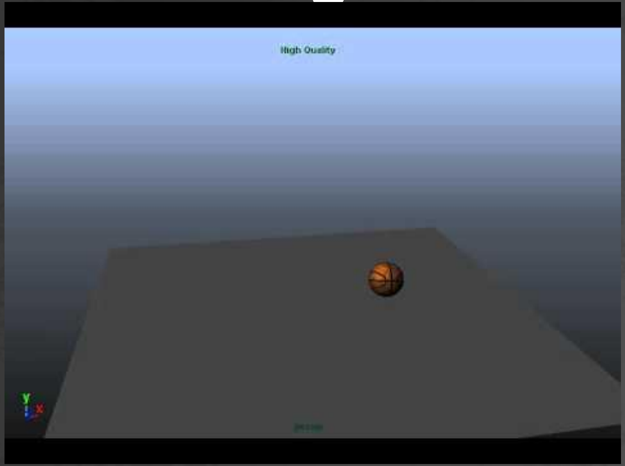


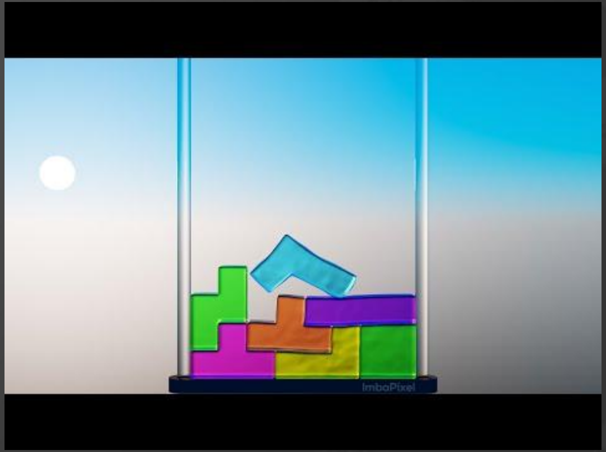
BBC KS3

- **Decomposition:** Break down complex problems; prevent from becoming overwhelmed.
- **Abstraction:** Strip away unnecessary details to see core features.
- **Pattern recognition:** find similarities, differences, trends, repetitions.
- **Algorithms:** step-by-step process to solve a problem or task.

Also builds and support: *confidence* (lots of "aha" moments!), *tenacity*, *communication* skills, *curiosity*, *intentional attitude*, *growth mindset.*

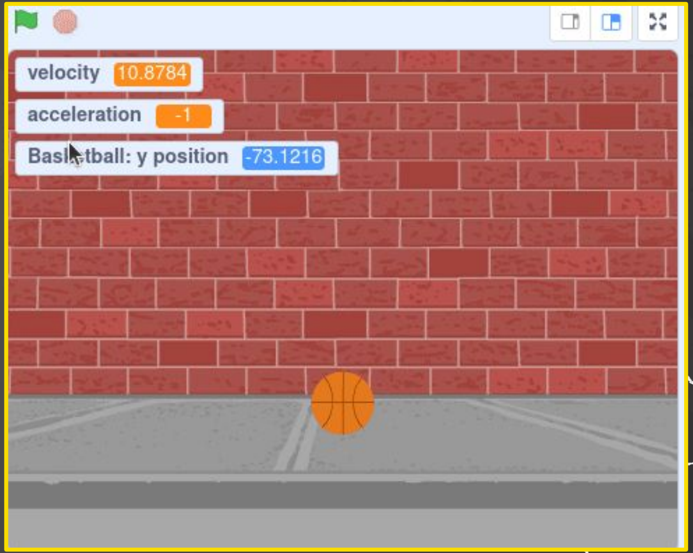
Falling objects..





Break task down into key steps

1. Set-up the stage by placing ball (and background).
2. Drop ball at constant speed.
3. Hit floor and stop.
4. Added acceleration due to gravity.
5. Bounce back when hitting the floor.
6. Loss of energy when bouncing.
7. Extensions.



velocity 10.8784

acceleration -1

Basketball: y position -73.1216

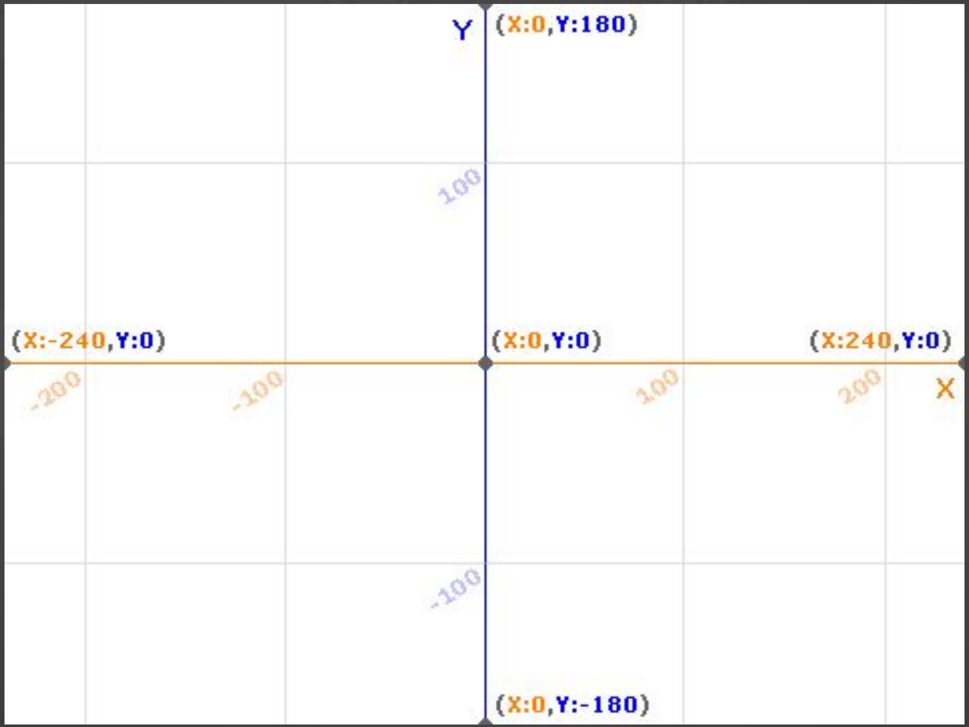
This tutorial is _not_ a sample lesson, but a list of meaningful self-contained steps that constitutes the STEM coding ***horizon*** for the teacher to guide implementation. Each step may take multiple sessions.

Specific timeline will depend on: year level, experience with coding and Scratch, familiarity with STEM and mathematical ideas (e.g., variables and equations).

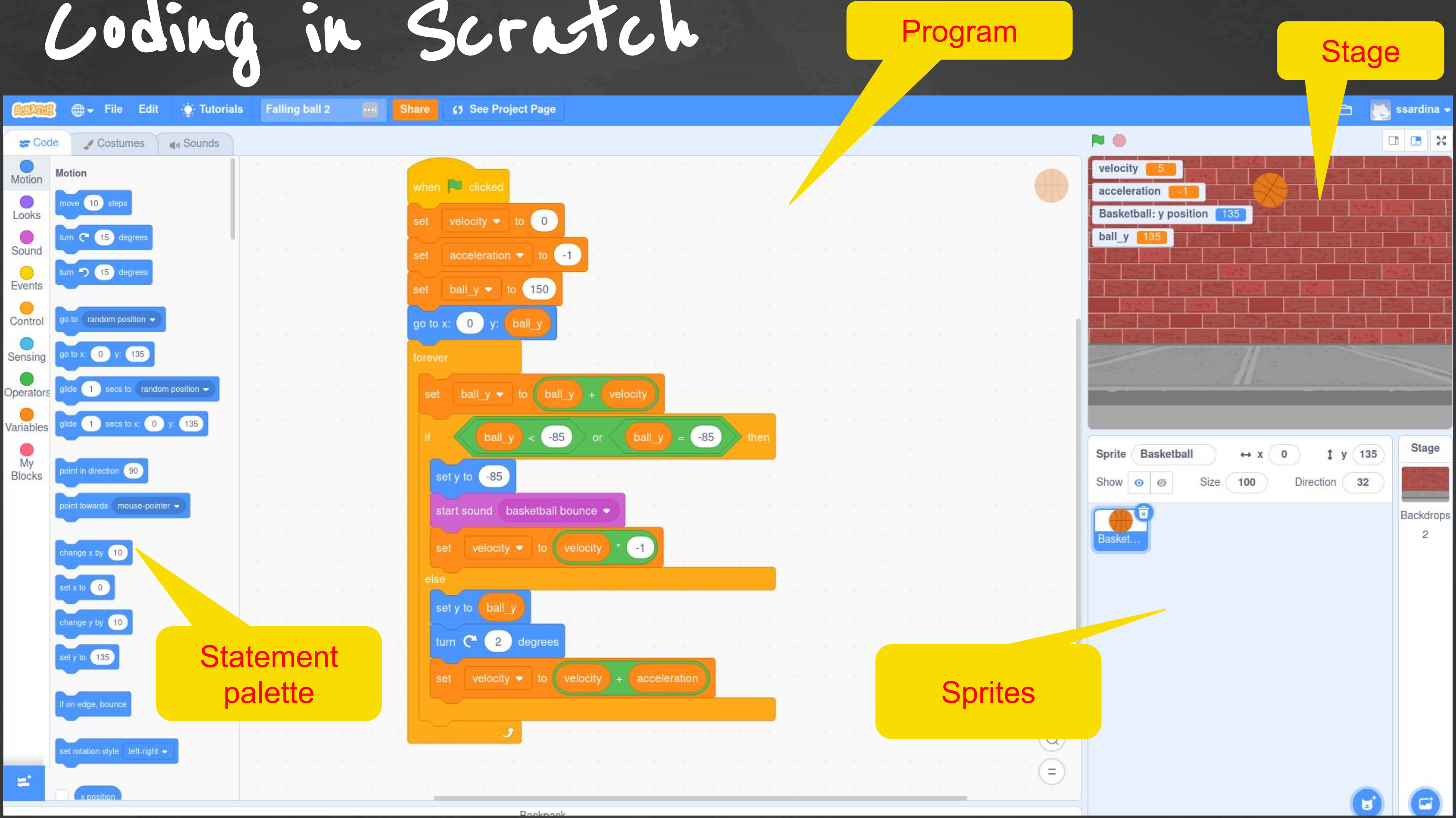
Mathematical components promoted

1. Cartesian plane & coordinates (to place objects) - VC2M6N01
2. Positive & negative numbers (to represent direction, position).
3. Basic math operations (addition, multiplication, etc) - VC2M4N06.
4. Number comparison (e.g., greater than) - VC2M5N01
5. Boolean logical expressions (e.g., or, and).
6. Use of variables for modeling - VC2M4A01.
7. Proportion/percentage (to implement loss of energy) - VC2M5N04.
8. Decimal numbers - VC2M5N01.
9. Multiplying by -1 (to implement reversing of direction).

Scratch Coordinate system



Coding in Scratch



Program

Stage

Statement palette

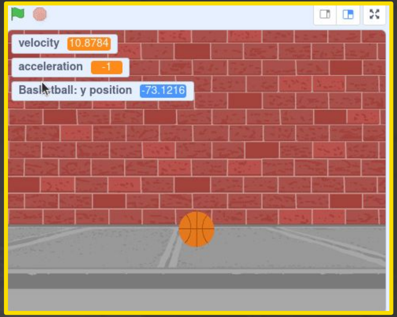
Sprites

```
when ⚑ clicked
set velocity to 0
set acceleration to -1
set ball_y to 150
go to x: 0 y: ball_y
forever
    set ball_y to ball_y + velocity
    if ball_y < -85 or ball_y = -85 then
        set y to -85
        start sound basketball bounce
        set velocity to velocity * -1
    else
        set y to ball_y
        turn ↻ 2 degrees
        set velocity to velocity + acceleration
```

velocity 5
acceleration -1
Basketball: y position 135
ball_y 135

Sprite Basketball ↔ x 0 ↕ y 135
Show Size 100 Direction 32

Two ways to use the project



velocity 10.8784

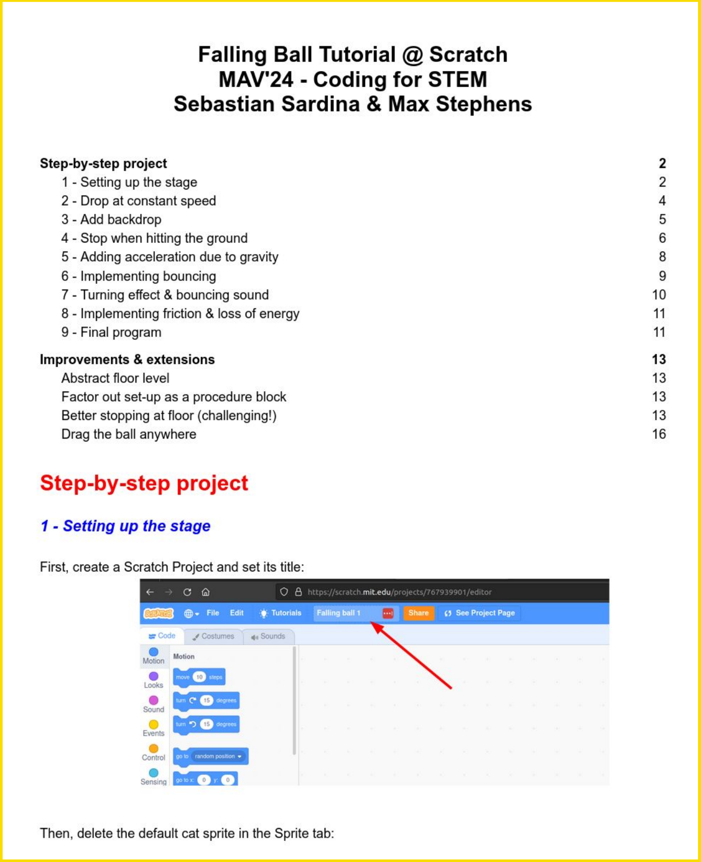
acceleration -1

Basketball: y position -73.1216

1. **Create the code incrementally.**
   - More demanding.
   - Several sessions depending on existing knowledge and skills.
   - Potentially higher-level of achievement.
   - Step-by-step tutorial provided here: https://bit.ly/3VcMNGx
2. **Remix/modify existing code.**
   - Full code provided by the teacher.
   - Students first understand the code.
   - Then, student modify code to achieve various objectives.
     - Different ball location.
     - Faster fall.
     - Less/more energy loss at bouncing.
     - Further abstraction (introduce new variables).
     - Fix existing bugs (maybe introduced by teacher in original code)
   - After re-mixing, students may create their code from zero.

Resources from today's session

- Tutorial: https://bit.ly/3VcMNGx
- Final program:
  - https://scratch.mit.edu/projects/770606600/
- Scratch: https://scratch.mit.edu/
- Similar project: https://bit.ly/3GPrKn2



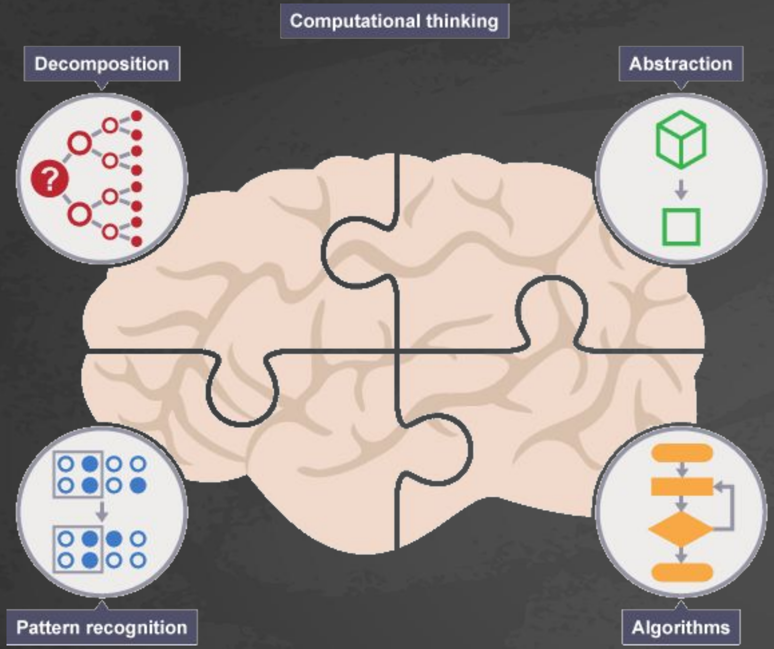
Falling Ball Tutorial @ Scratch
MAV'24 - Coding for STEM
Sebastian Sardina & Max Stephens

Step-by-step project

1 - Setting up the stage

First, create a Scratch Project and set its title:

Then, delete the default cat sprite in the Sprite tab:



Thank you! Q & A

Slides: *https://bit.ly/mav22-fball*

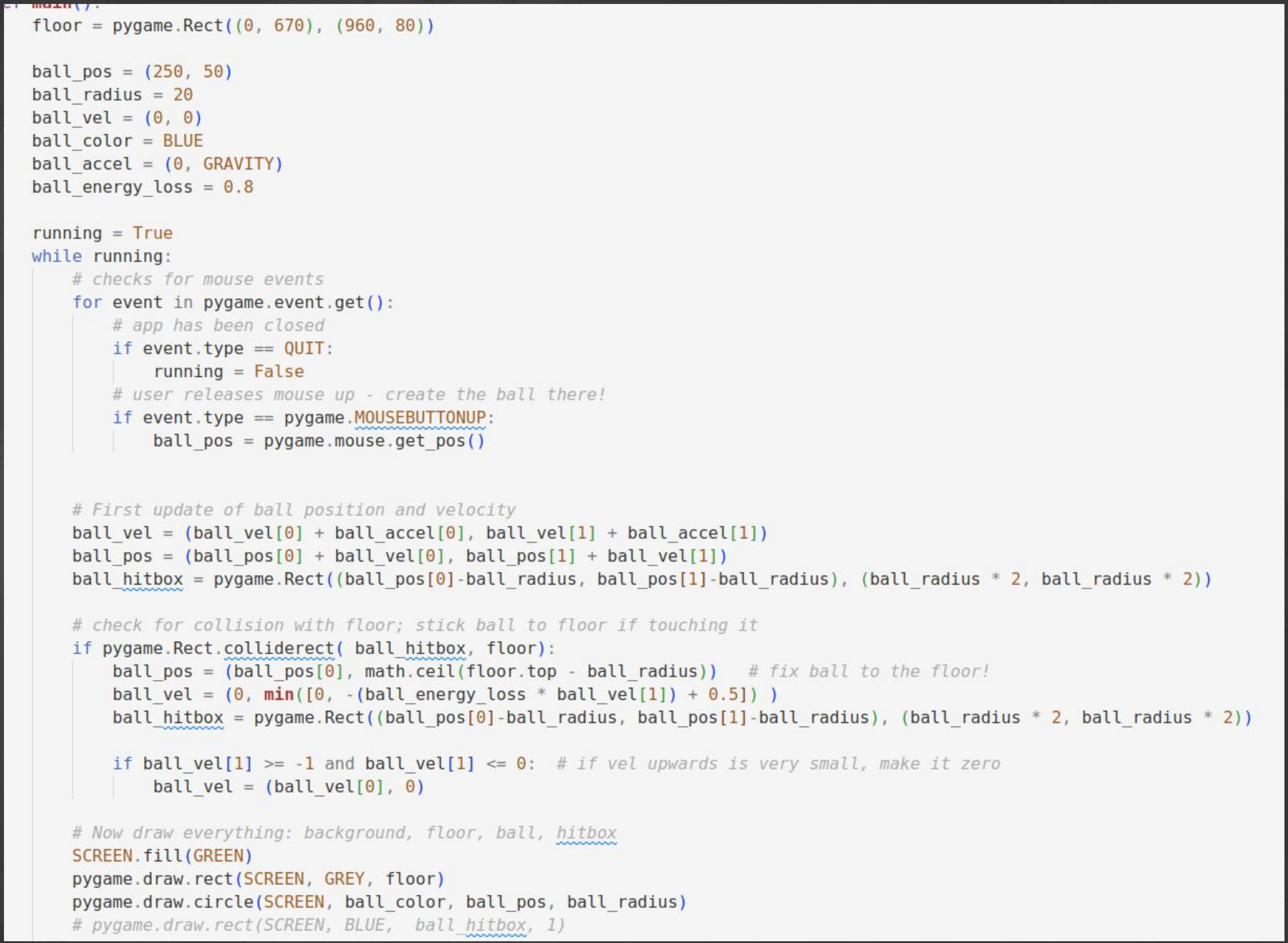
Questions? Comments? Suggestions? Share experiences?

Please contact us at:

- *Sebastian Sardina: sebastian.sardina@rmit.edu.au*
- *Max Stephens: m.stephens@unimelb.edu.au*

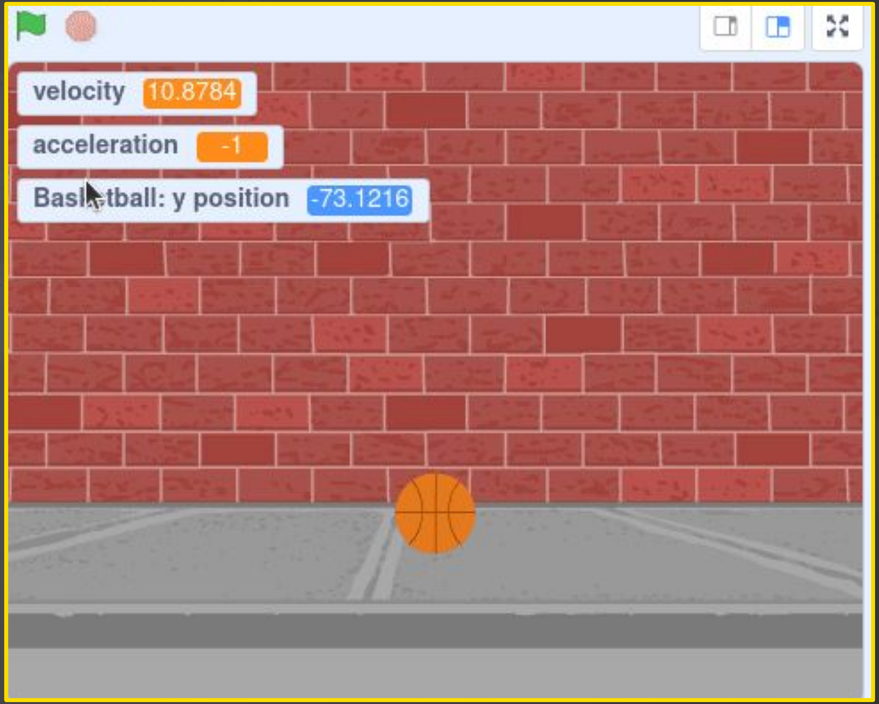
CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.

Python



```
    floor = pygame.Rect((0, 670), (960, 80))

    ball_pos = (250, 50)
    ball_radius = 20
    ball_vel = (0, 0)
    ball_color = BLUE
    ball_accel = (0, GRAVITY)
    ball_energy_loss = 0.8

    running = True
    while running:
        # checks for mouse events
        for event in pygame.event.get():
            # app has been closed
            if event.type == QUIT:
                running = False
            # user releases mouse up - create the ball there!
            if event.type == pygame.MOUSEBUTTONUP:
                ball_pos = pygame.mouse.get_pos()


        # First update of ball position and velocity
        ball_vel = (ball_vel[0] + ball_accel[0], ball_vel[1] + ball_accel[1])
        ball_pos = (ball_pos[0] + ball_vel[0], ball_pos[1] + ball_vel[1])
        ball_hitbox = pygame.Rect((ball_pos[0]-ball_radius, ball_pos[1]-ball_radius), (ball_radius * 2, ball_radius * 2))

        # check for collision with floor; stick ball to floor if touching it
        if pygame.Rect.colliderect( ball_hitbox, floor):
            ball_pos = (ball_pos[0], math.ceil(floor.top - ball_radius))   # fix ball to the floor!
            ball_vel = (0, min([0, -(ball_energy_loss * ball_vel[1]) + 0.5]) )
            ball_hitbox = pygame.Rect((ball_pos[0]-ball_radius, ball_pos[1]-ball_radius), (ball_radius * 2, ball_radius * 2))

            if ball_vel[1] >= -1 and ball_vel[1] <= 0:  # if vel upwards is very small, make it zero
                ball_vel = (ball_vel[0], 0)

        # Now draw everything: background, floor, ball, hitbox
        SCREEN.fill(GREEN)
        pygame.draw.rect(SCREEN, GREY, floor)
        pygame.draw.circle(SCREEN, ball_color, ball_pos, ball_radius)
        # pygame.draw.rect(SCREEN, BLUE,  ball_hitbox, 1)
```

Step by step tutorial



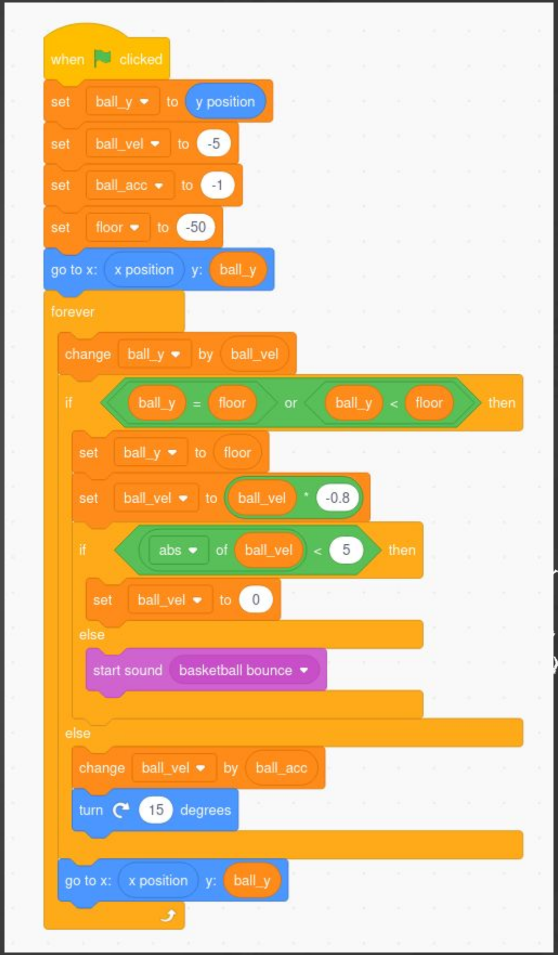
velocity 10.8784

acceleration -1

Basketball: y position -73.1216

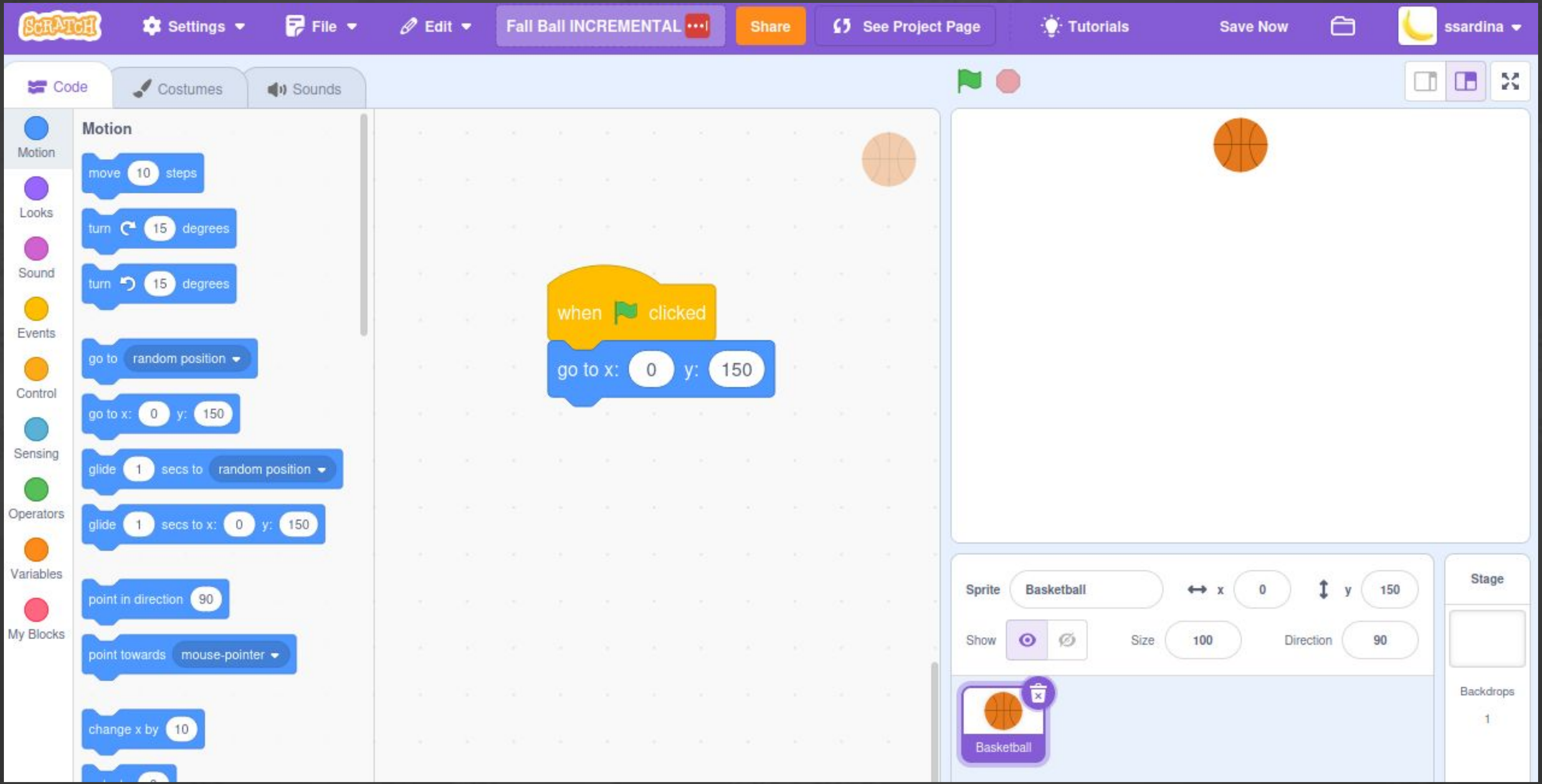
Steps

1. Set a ball in the upper-middle of the stage.
2. Naive (finte) fall.
3. Abstract ball's y-coordinate (variable bball_y).
4. Constant free fall (forever).
5. Add a backdrop.
6. Collision with floor (if ball_y = 80) - first try!
7. Collision with floor (if ball_y = 80) - works!
8. Abstract velocity (variable ball_acc).
9. Add acceleration (change ball_vel via ball_acc).
10. Perfect bouncing (multiplication by -1).
11. Add bouncing sound and rolling.
12. Model loss of energy at bounce (multiply by < -1).
13. Stabilize at floor.
14. Allow any initial location.
15. Abstract acceleration & floor.



```
when 🏴 clicked
set ball_y ▾ to y position
set ball_vel ▾ to -5
set ball_acc ▾ to -1
set floor ▾ to -50
go to x: x position y: ball_y
forever
    change ball_y ▾ by ball_vel
    if <<ball_y = floor> or <ball_y < floor>> then
        set ball_y ▾ to floor
        set ball_vel ▾ to (ball_vel * -0.8)
        if <(abs ▾ of ball_vel) < 5> then
            set ball_vel ▾ to 0
        else
            start sound basketball bounce ▾
    else
        change ball_vel ▾ by ball_acc
        turn ↻ 15 degrees
    go to x: x position y: ball_y
```

1 - Set a ball in the upper-middle

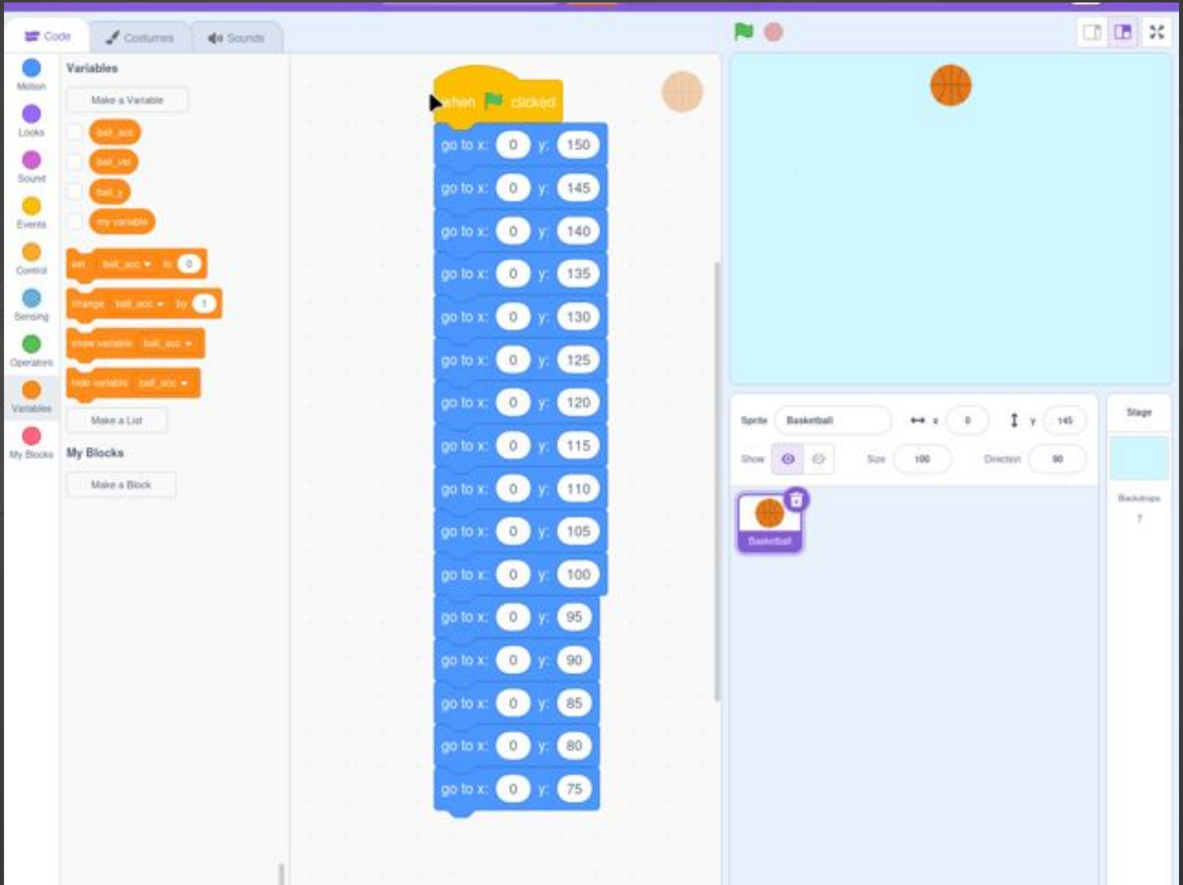


```
when 🏁 clicked
go to x: 0 y: 150
```

Sprite: Basketball x: 0 y: 150

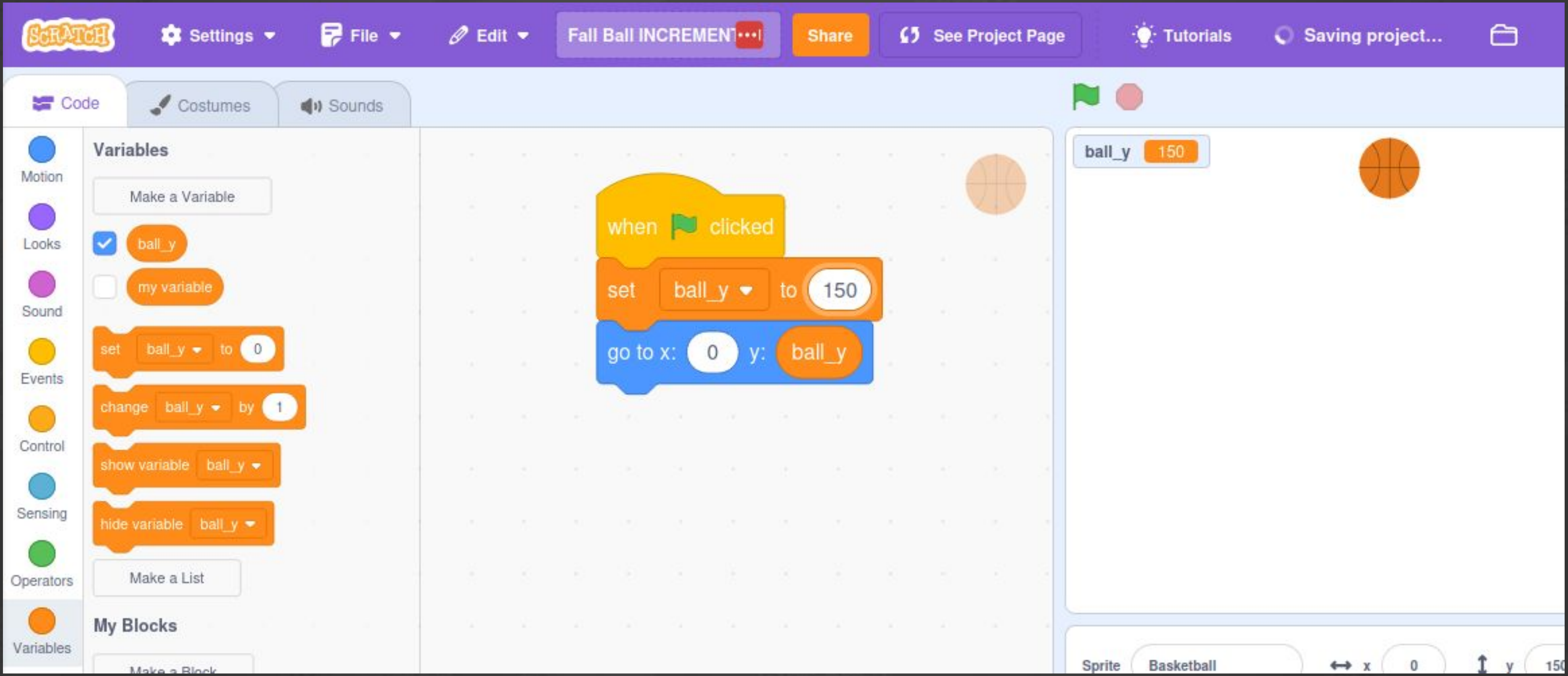
Show Size: 100 Direction: 90

2 - Naive (finite) fall



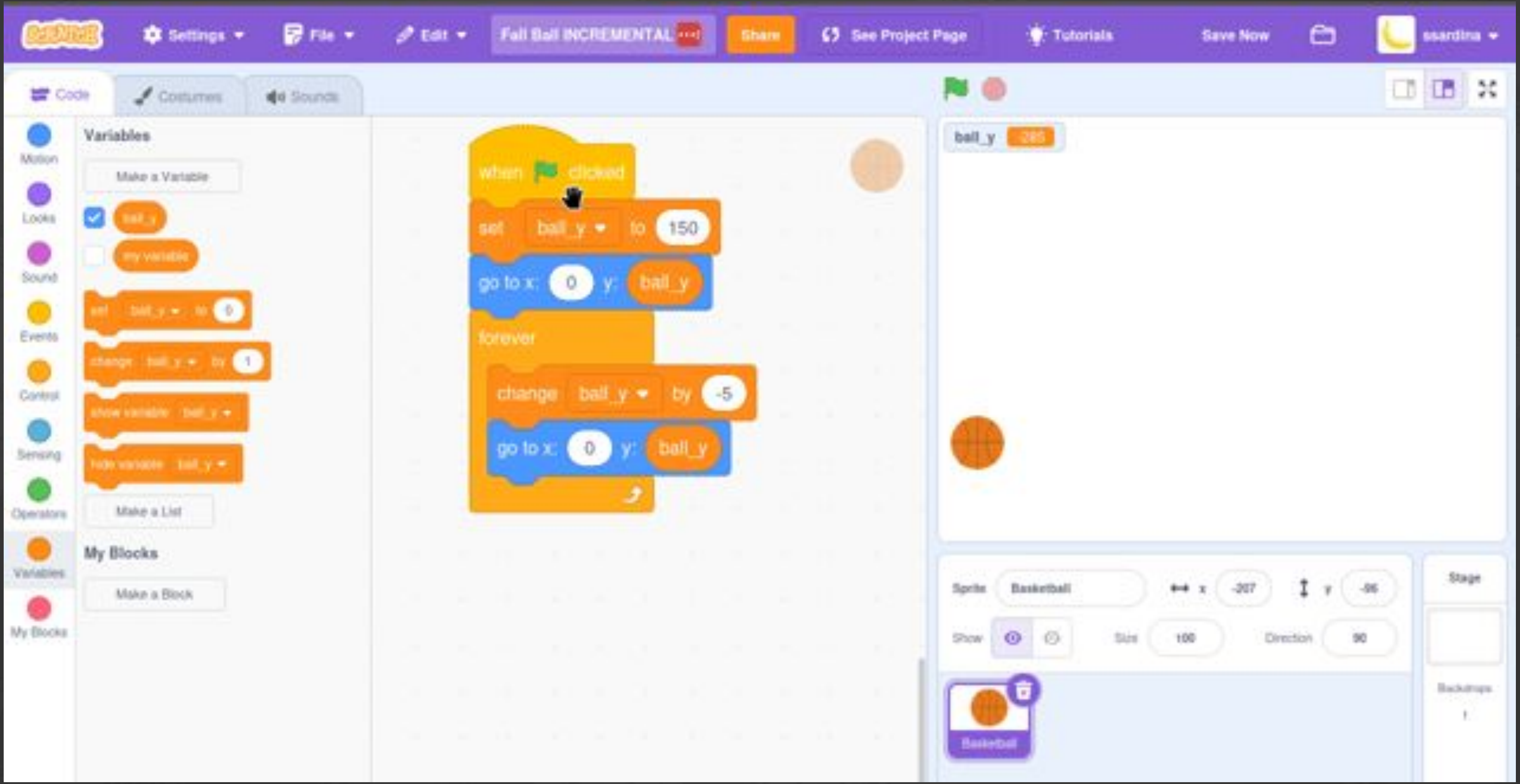
```
when 🏁 clicked
go to x: 0 y: 150
go to x: 0 y: 145
go to x: 0 y: 140
go to x: 0 y: 135
go to x: 0 y: 130
go to x: 0 y: 125
go to x: 0 y: 120
go to x: 0 y: 115
go to x: 0 y: 110
go to x: 0 y: 105
go to x: 0 y: 100
go to x: 0 y: 95
go to x: 0 y: 90
go to x: 0 y: 85
go to x: 0 y: 80
go to x: 0 y: 75
```

3 - Abstract ball's y-coordinate



```
when 🏁 clicked
set ball_y to 150
go to x: 0 y: ball_y
```

4 - Constant free fall



Variables

- ☑ ball_y
- □ my variable
- set ball_y to 0
- change ball_y by 1
- show variable ball_y
- hide variable ball_y

My Blocks

```
when 🏁 clicked
set ball_y ▾ to 150
go to x: 0 y: ball_y
forever
    change ball_y ▾ by -5
    go to x: 0 y: ball_y
```

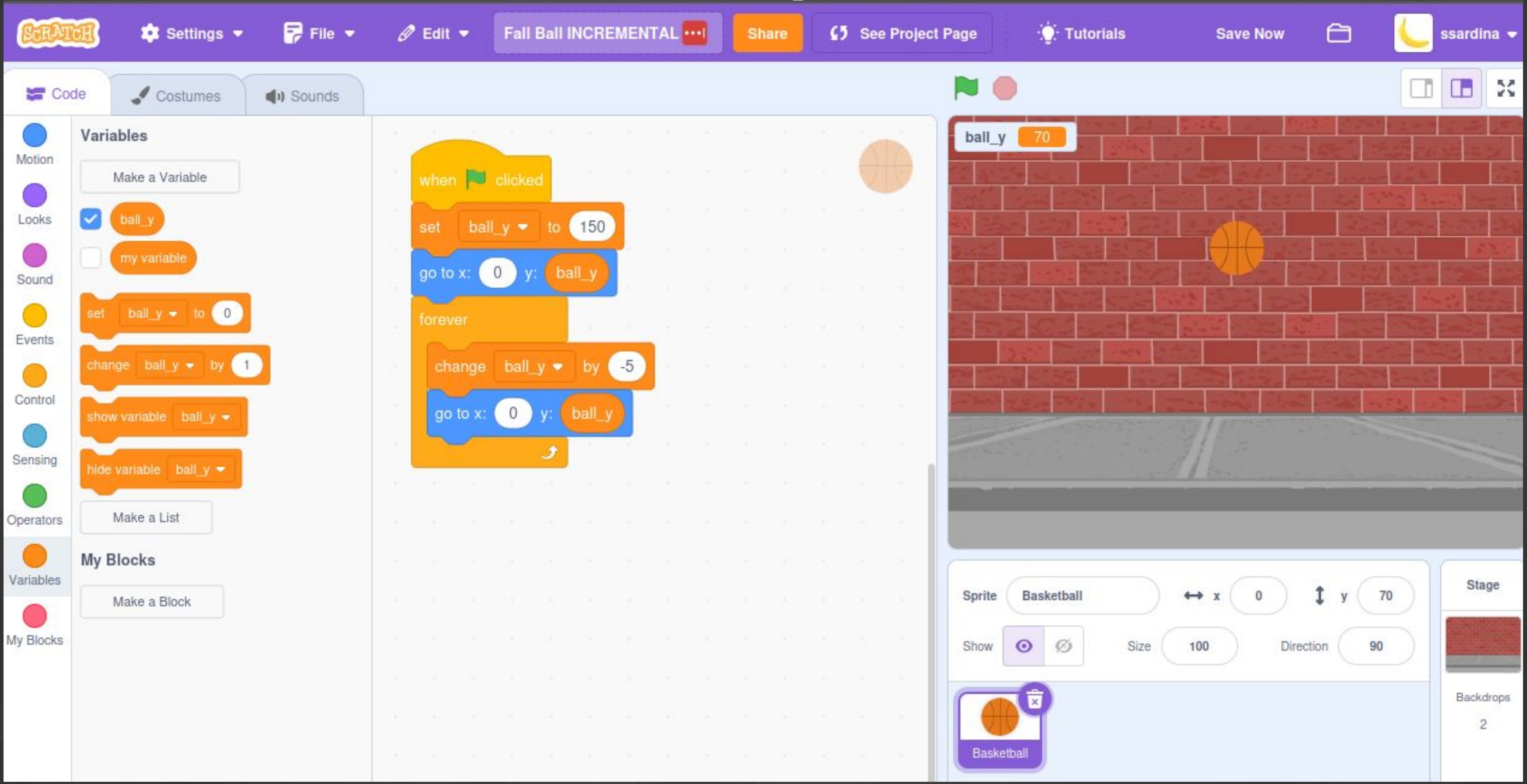
ball_y -285

Sprite Basketball ↔ x -207 ↕ y -96

Show Size 100 Direction 90

Basketball

5 - Add a backdrop



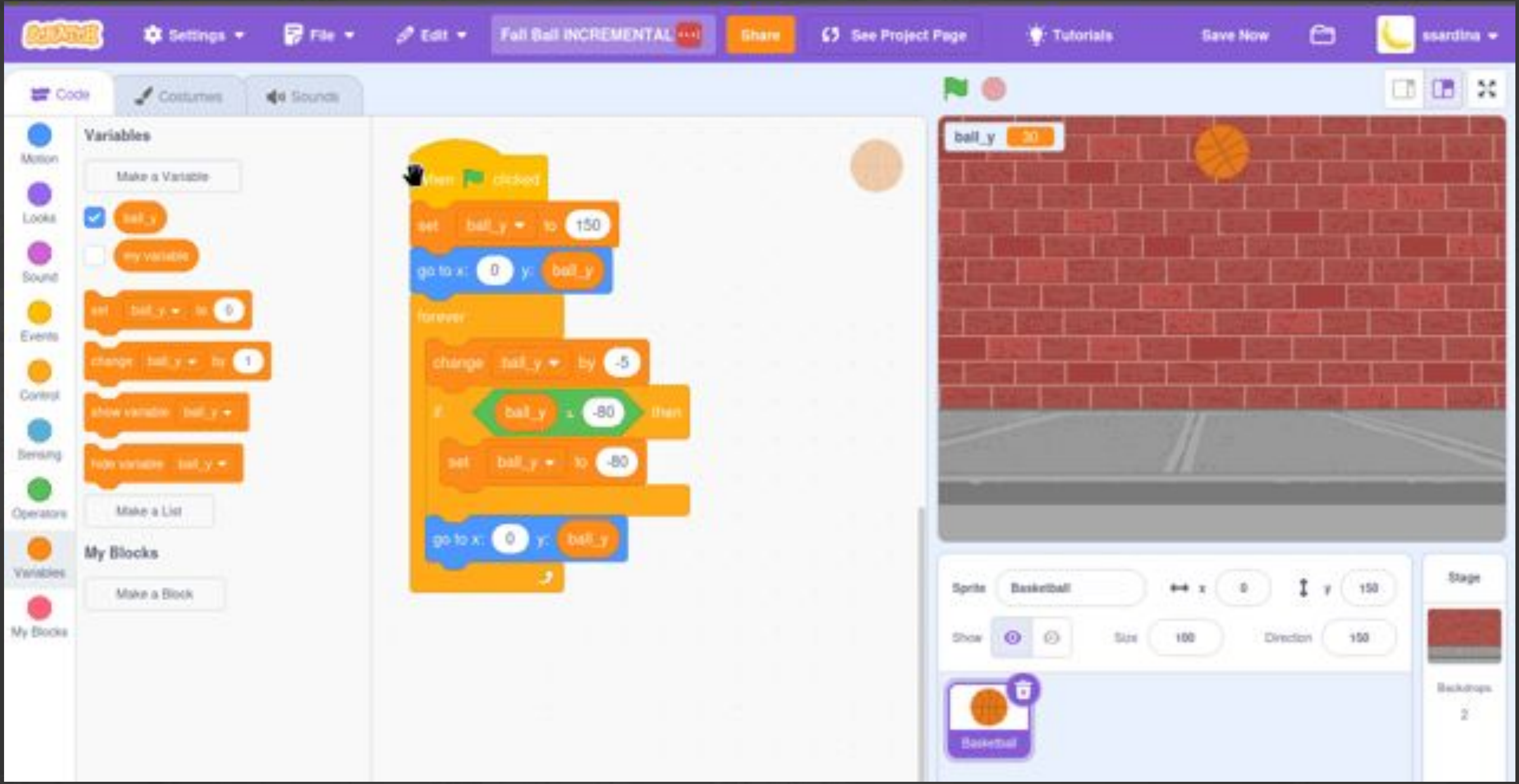
```
when ⚑ clicked
set ball_y to 150
go to x: 0 y: ball_y
forever
    change ball_y by -5
    go to x: 0 y: ball_y
```

ball_y 70

Sprite: Basketball   x: 0   y: 70

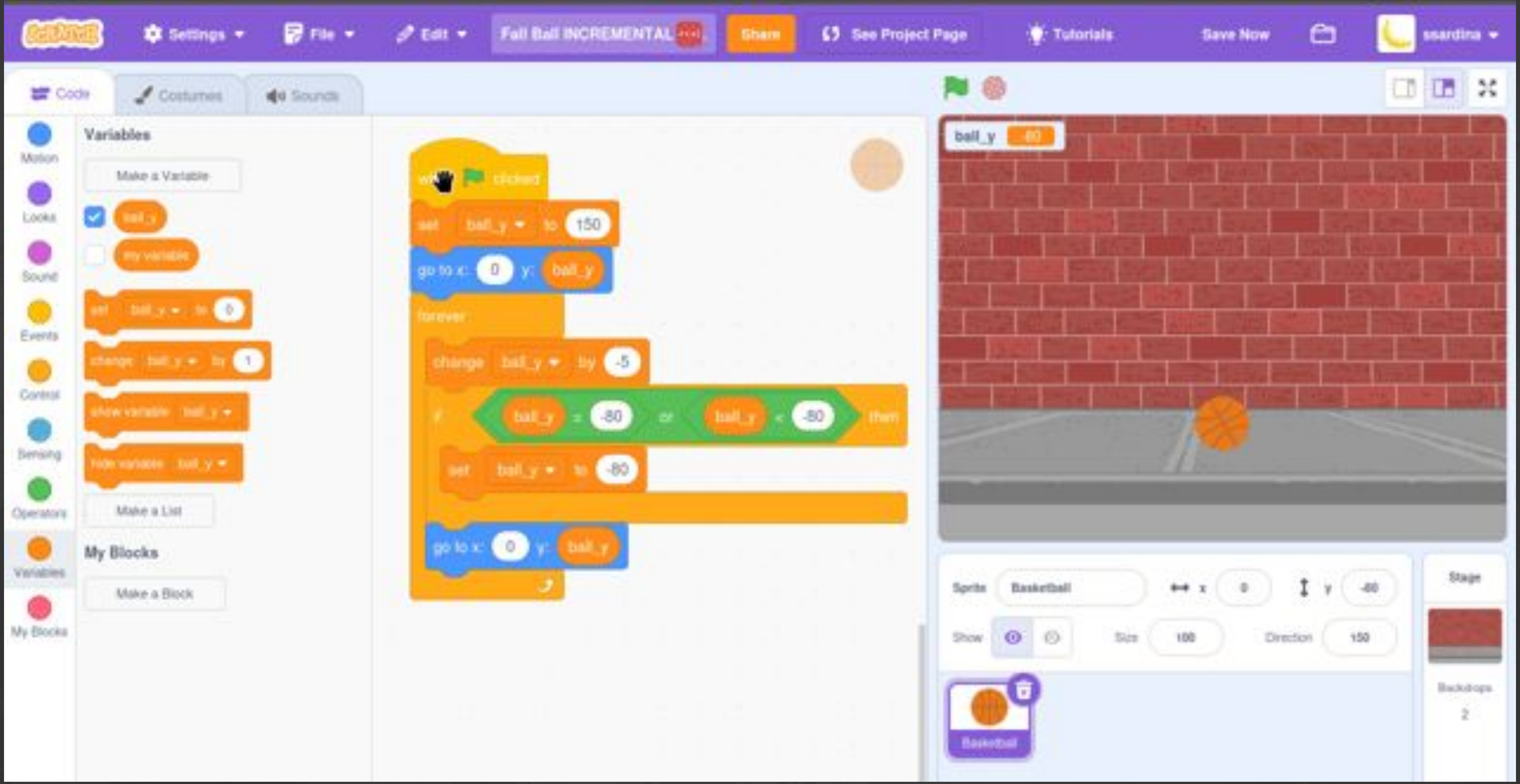
Show   Size: 100   Direction: 90

6 - Collision with floor - first try!



```
when 🏁 clicked
set ball_y to 150
go to x: 0 y: ball_y
forever
    change ball_y by -5
    if ball_y < -80 then
        set ball_y to -80
    go to x: 0 y: ball_y
```

7 - Collision with floor - works!



Variables

☑ ball_y

□ my variable

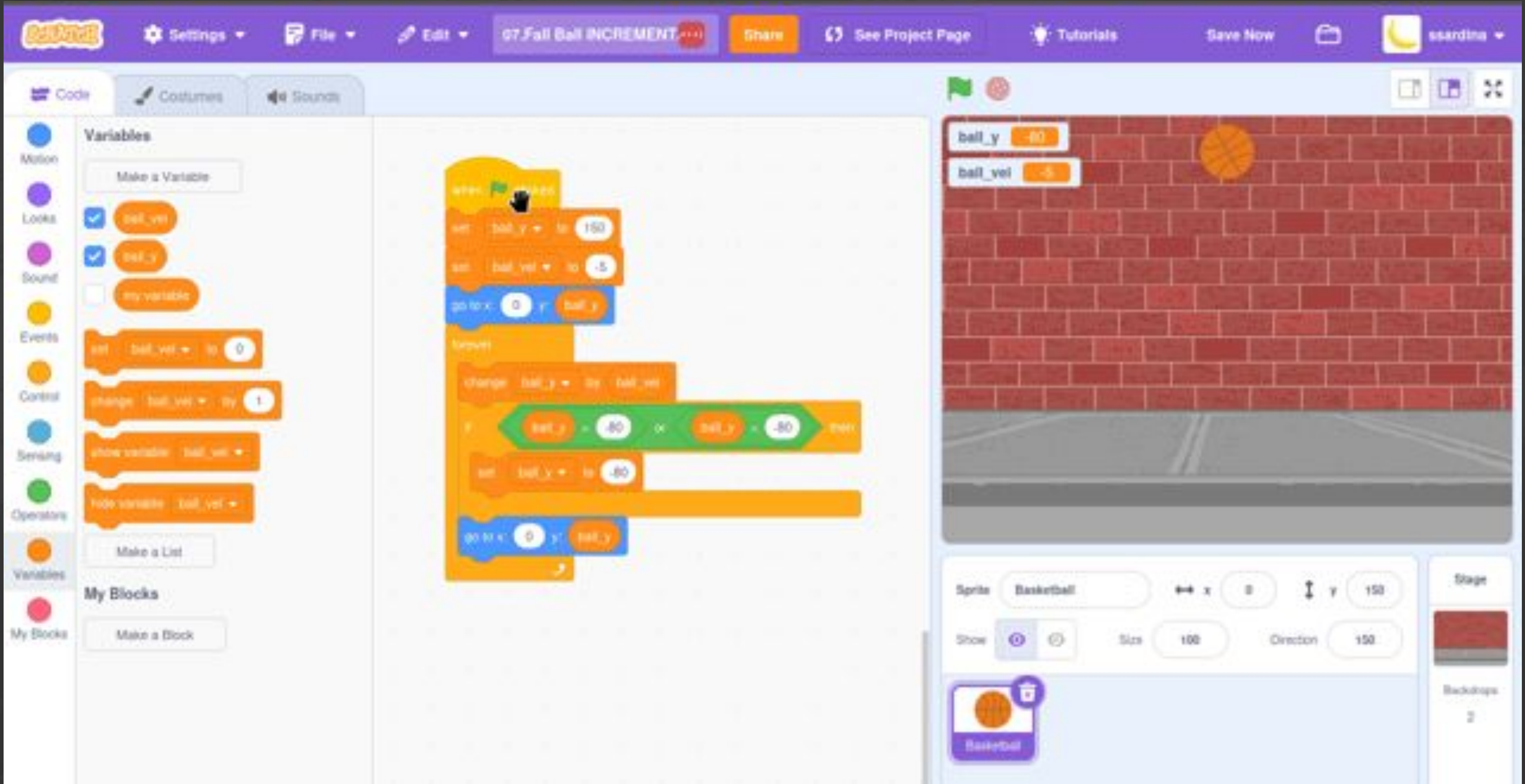
```
when 🏁 clicked
set ball_y to 150
go to x: 0 y: ball_y
forever
    change ball_y by -5
    if <<ball_y = -80> or <ball_y < -80>> then
        set ball_y to -80
    end
    go to x: 0 y: ball_y
```

ball_y -80

Sprite: Basketball x: 0 y: -80

Show Size: 100 Direction: 150

8 Abstract velocity: bell_vel



```
when 🏳 clicked
set ball_y to 150
set ball_vel to -5
go to x: 0 y: ball_y
forever
    change ball_y by ball_vel
    if <<ball_y < -80> or <ball_y = -80>> then
        set ball_y to -80
    go to x: 0 y: ball_y
```

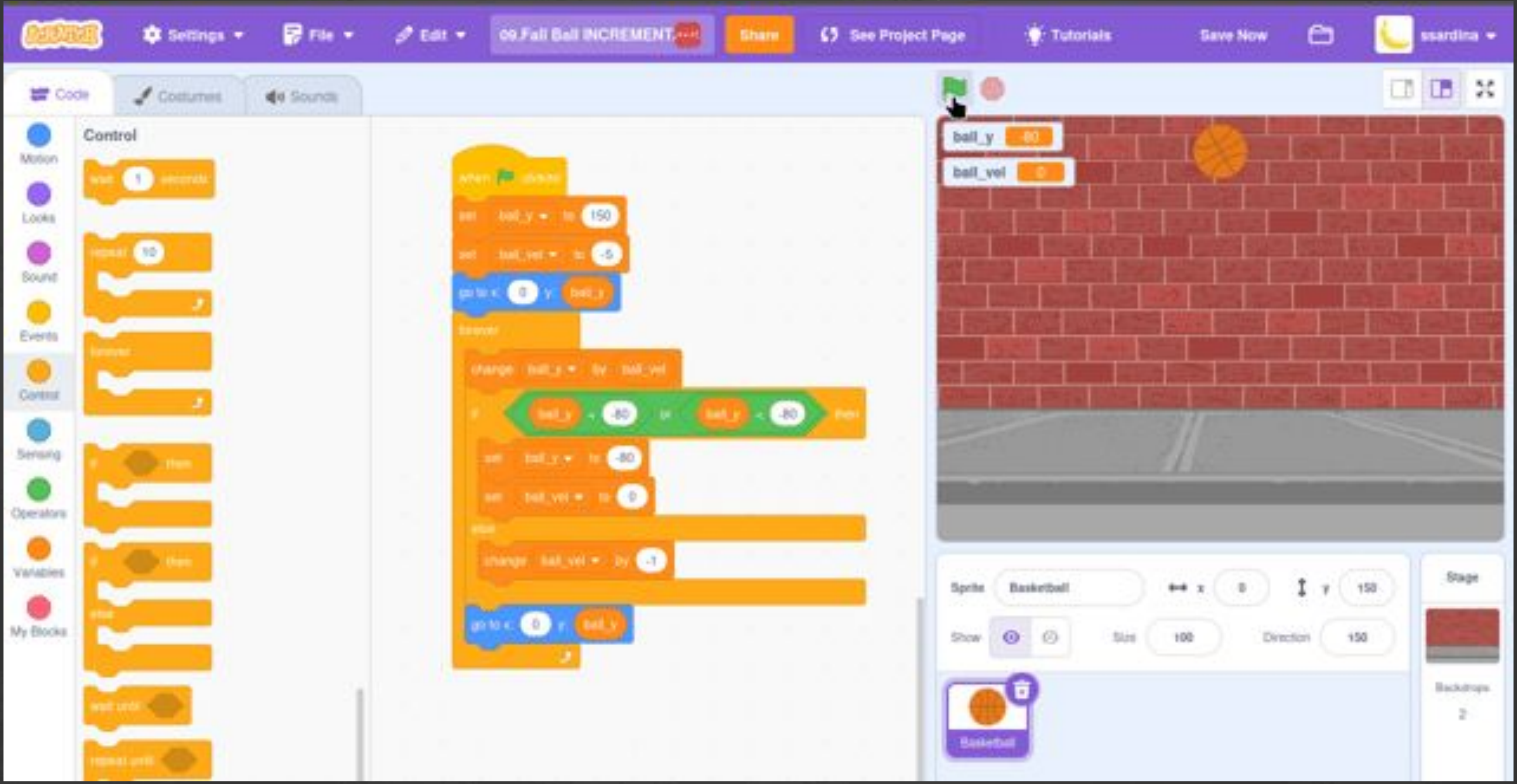
ball_y -80

ball_vel -5

Sprite Basketball x 0 y 150

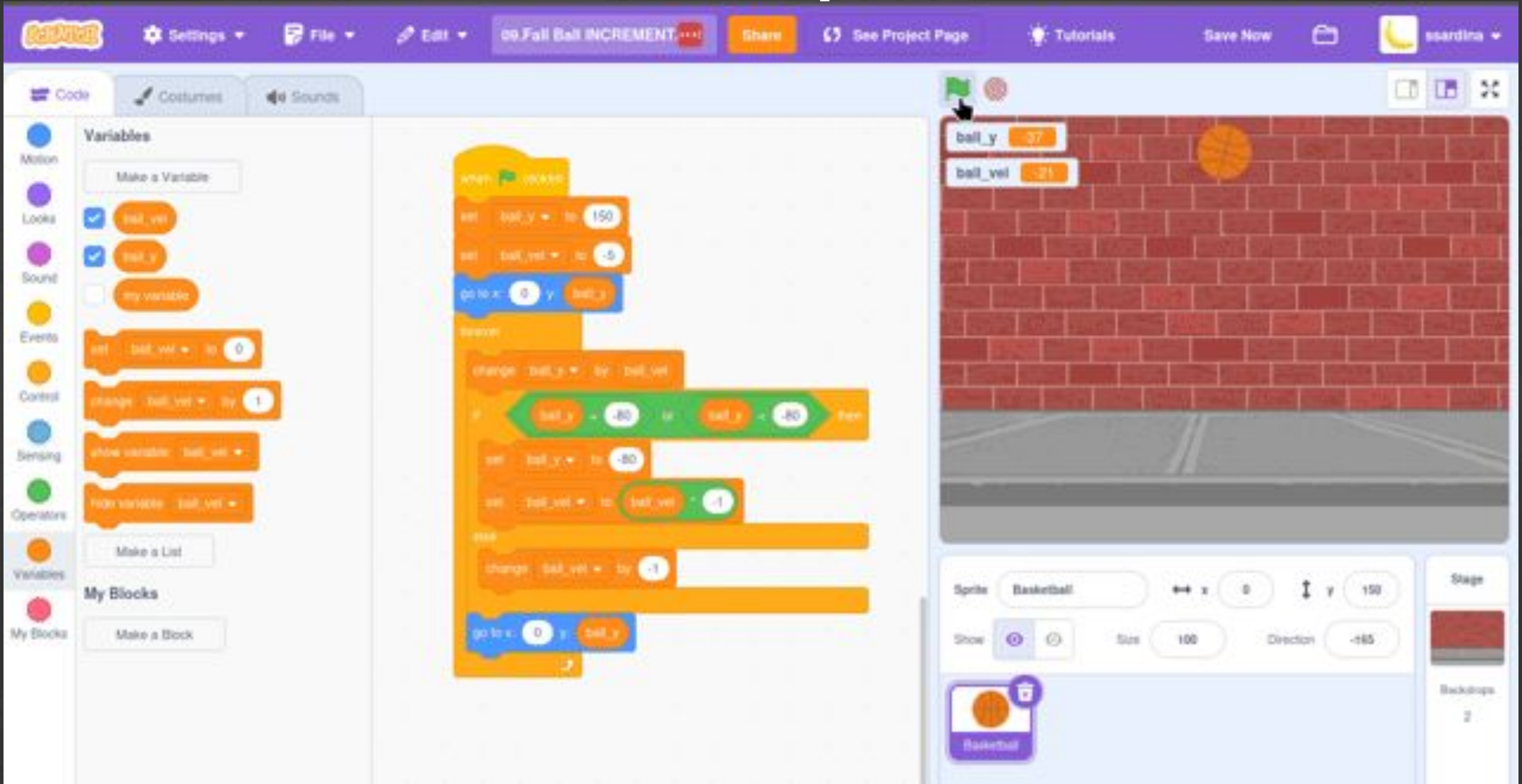
Show Size 100 Direction 150

9 - Acceleration: change velocity



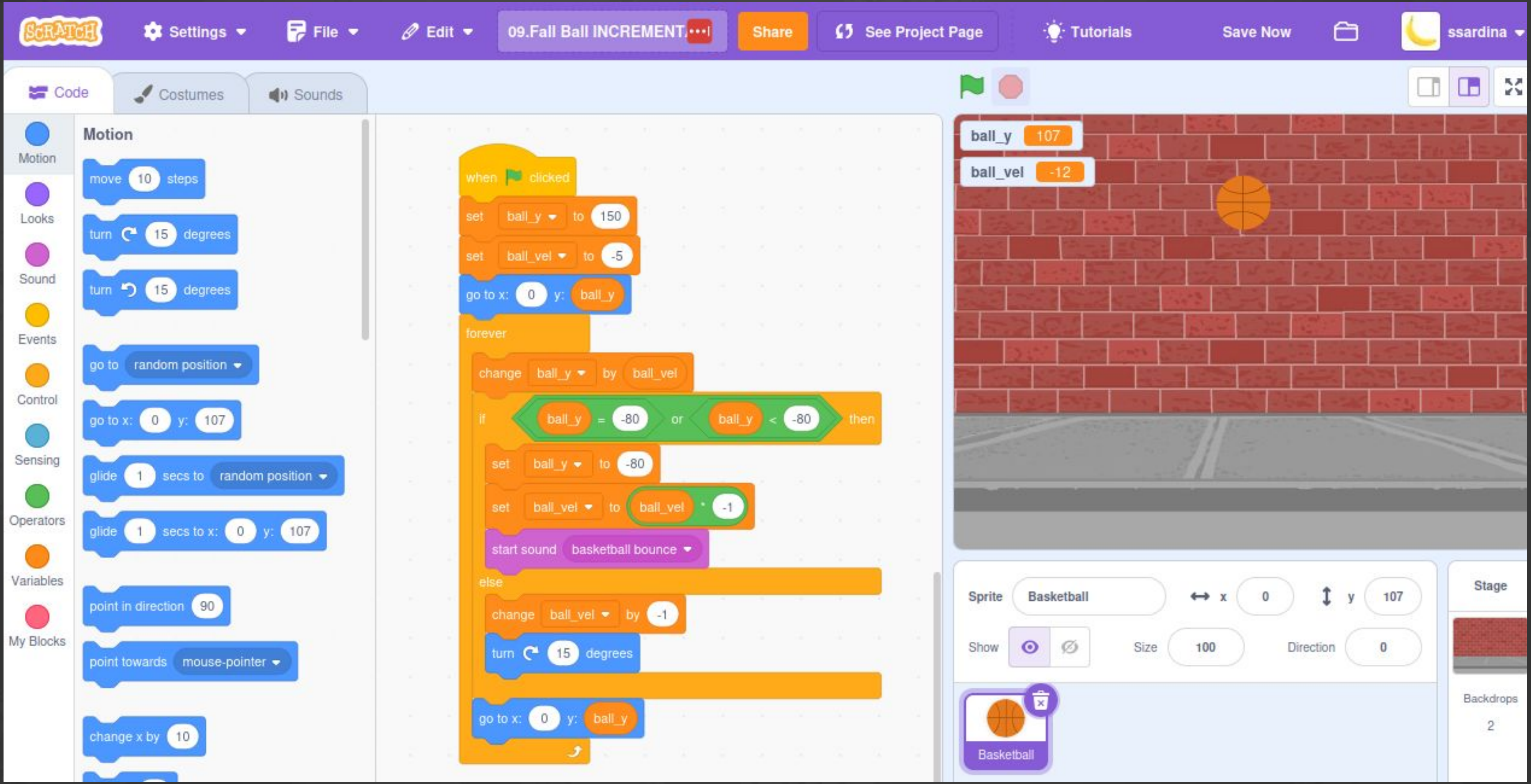
```
when [flag] clicked
set ball_y to 150
set ball_vel to -5
go to x: 0 y: ball_y
forever
    change ball_y by ball_vel
    if ball_y = -80 or ball_y < -80 then
        set ball_y to -80
        set ball_vel to 0
    else
        change ball_vel by -1
    go to x: 0 y: ball_y
```

10 - Bounce back (perfect bounce)



```
when 🏁 clicked
set ball_y ▾ to 150
set ball_vel ▾ to -5
go to x: 0 y: (ball_y)
forever
  change ball_y ▾ by (ball_vel)
  if <<(ball_y) = -80> or <(ball_y) < -80>> then
    set ball_y ▾ to -80
    set ball_vel ▾ to ((ball_vel) * -1)
  else
    change ball_vel ▾ by -1
  go to x: 0 y: (ball_y)
```

11 - Sound and roll...



```
when ⚑ clicked
set ball_y to 150
set ball_vel to -5
go to x: 0 y: ball_y
forever
    change ball_y by ball_vel
    if <<ball_y = -80> or <ball_y < -80>> then
        set ball_y to -80
        set ball_vel to (ball_vel * -1)
        start sound basketball bounce
    else
        change ball_vel by -1
        turn ⟳ 15 degrees
    go to x: 0 y: ball_y
```

ball_y 107

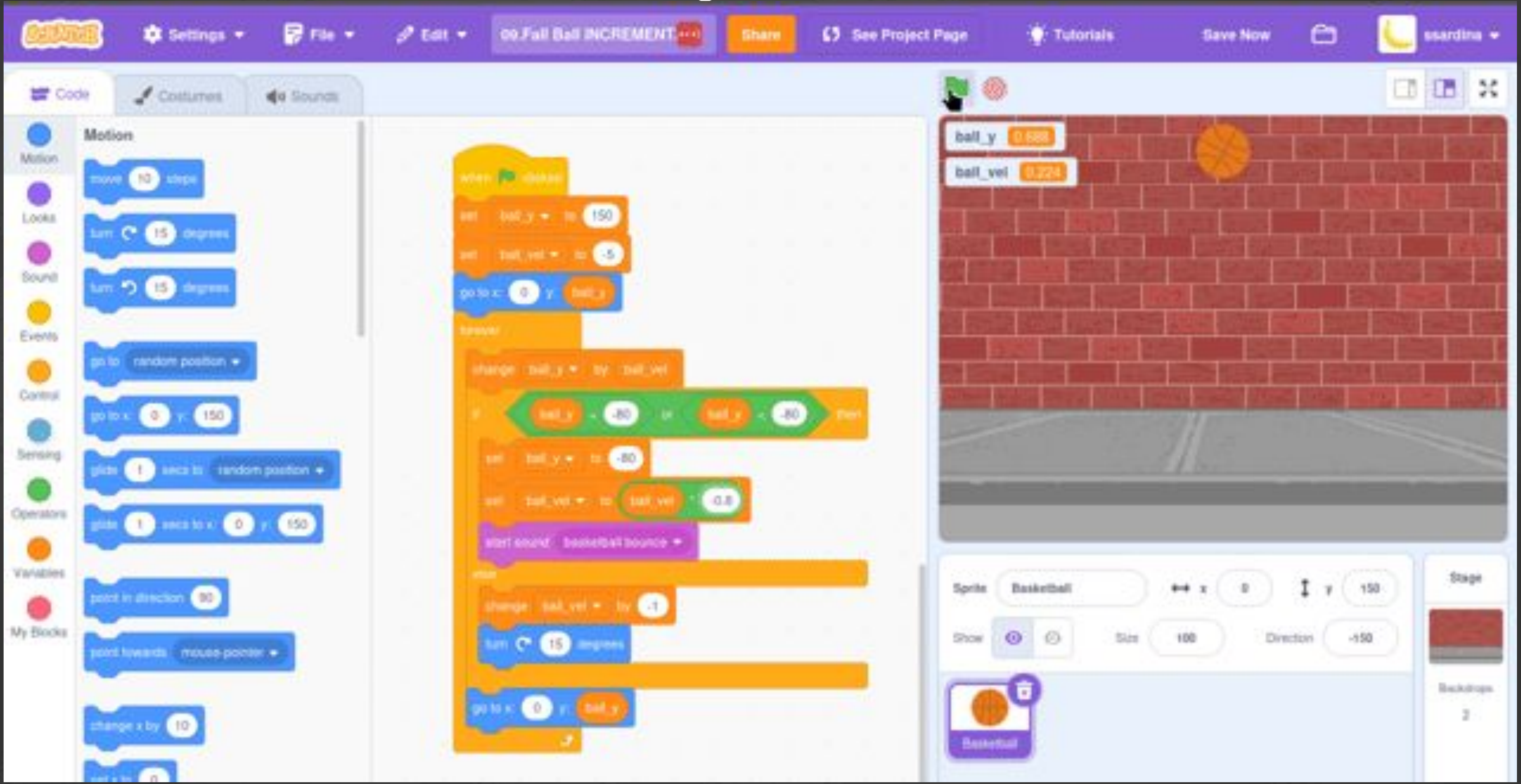
ball_vel -12

Sprite Basketball ↔ x 0 ↕ y 107

Show Size 100 Direction 0

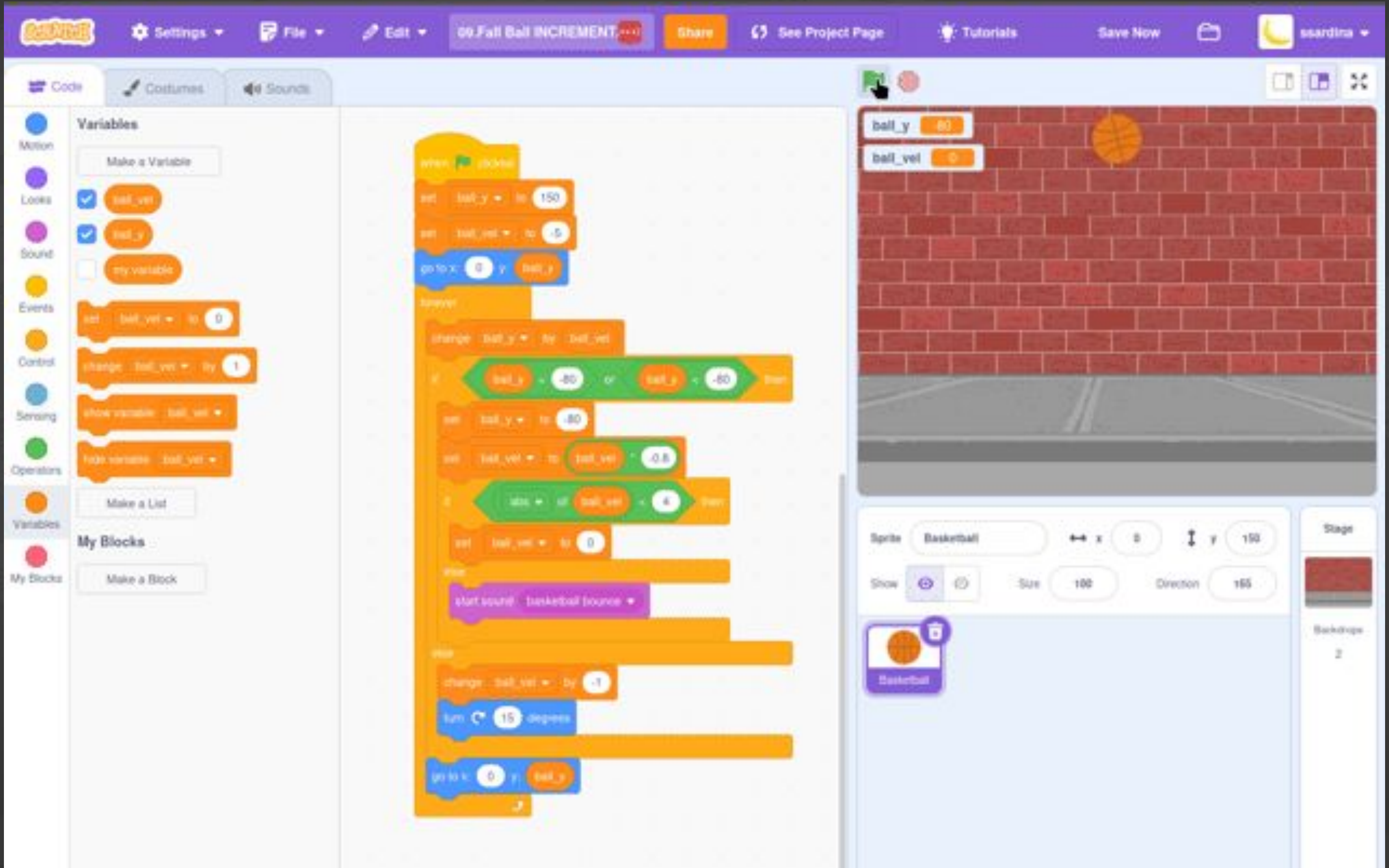
Backdrops 2

12 - Loss of energy at bounce



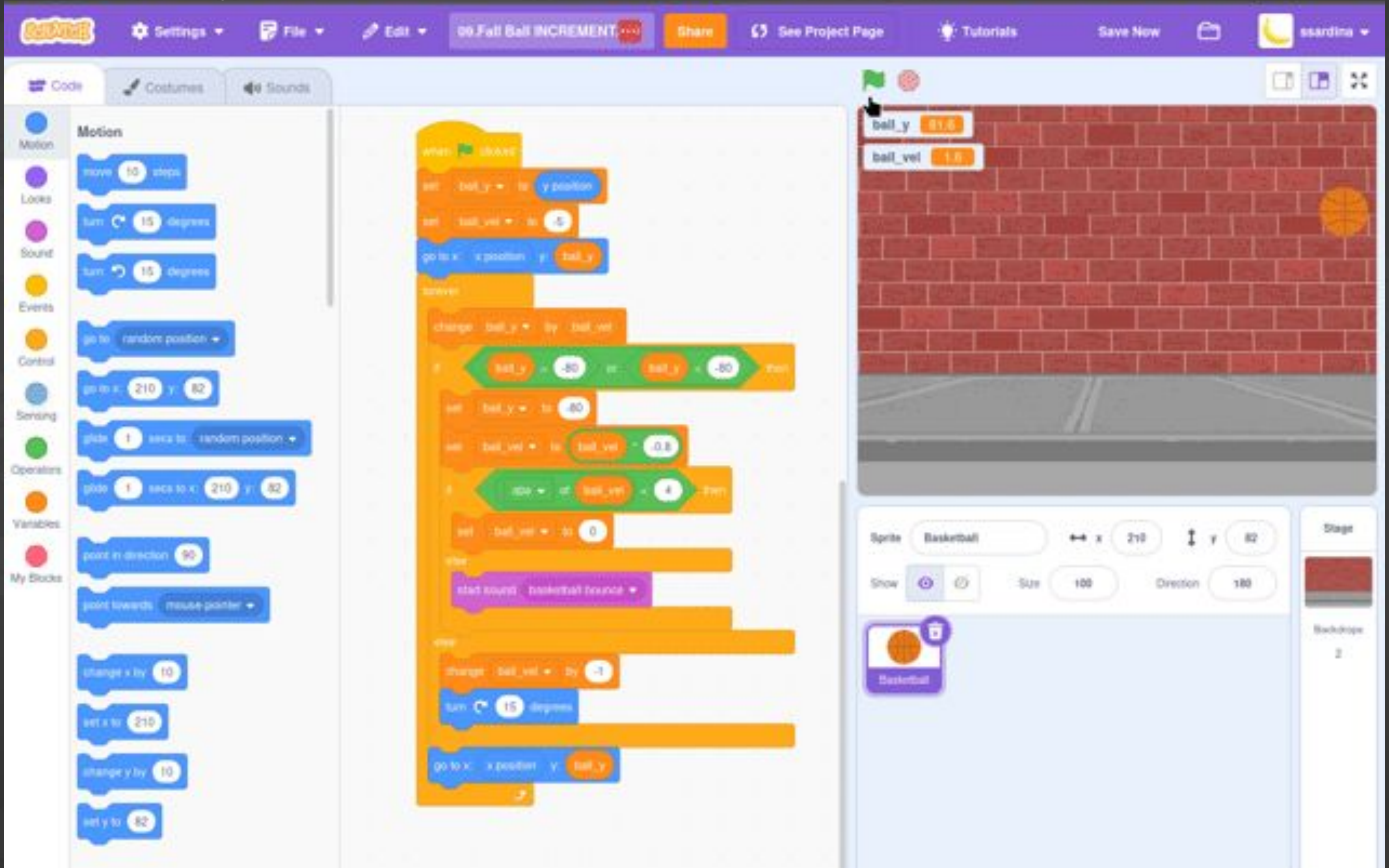
```
when 🏴 clicked
set ball_y ▾ to 150
set ball_vel ▾ to -5
go to x: 0 y: ball_y
forever
    change ball_y ▾ by ball_vel
    if ball_y = -80 or ball_y < -80 then
        set ball_y ▾ to -80
        set ball_vel ▾ to ball_vel * -0.8
        start sound basketball bounce ▾
    else
        change ball_vel ▾ by -1
        turn ↻ 15 degrees
    go to x: 0 y: ball_y
```

13 - Stabalise at floor



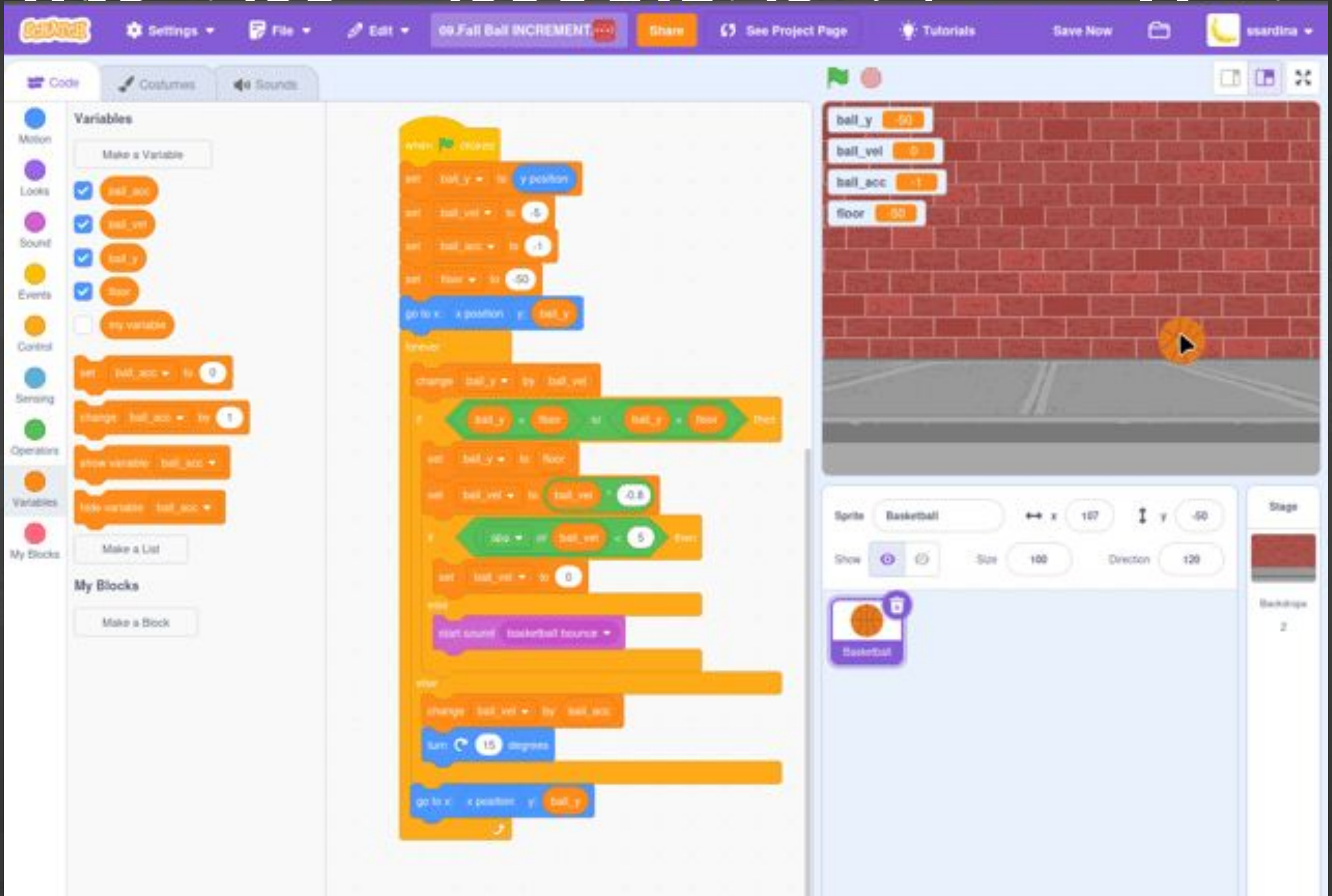
```
when [flag] clicked
set [ball_y] to (150)
set [ball_vel] to (-5)
go to x: (0) y: (ball_y)
forever
    change [ball_y] by (ball_vel)
    if <<(ball_y) = (-80)> or <(ball_y) < (-80)>> then
        set [ball_y] to (-80)
        set [ball_vel] to ((ball_vel) * (-0.8))
        if <([abs] of (ball_vel)) < (4)> then
            set [ball_vel] to (0)
        else
            start sound [basketball bounce]
        end
    else
        change [ball_vel] by (-1)
        turn ↻ (15) degrees
    end
    go to x: (0) y: (ball_y)
end
```

14 - Any initial location



```
when 🏁 clicked
set ball_y ▾ to (y position)
set ball_vel ▾ to (-5)
go to x: (x position) y: (ball_y)
forever
    change ball_y ▾ by (ball_vel)
    if <<(ball_y) = (-80)> or <(ball_y) < (-80)>> then
        set ball_y ▾ to (-80)
        set ball_vel ▾ to ((ball_vel) * (-0.8))
        if <([abs ▾] of (ball_vel)) < (4)> then
            set ball_vel ▾ to (0)
        else
            start sound (basketball bounce ▾)
    else
        change ball_vel ▾ by (-1)
        turn ↻ (15) degrees
    go to x: (x position) y: (ball_y)
```

14 - Abstract acceleration & floor



```
when 🏴 clicked
set ball_y to (y position)
set ball_vel to -5
set ball_acc to -1
set floor to -50
go to x: (x position) y: (ball_y)
forever
    change ball_y by (ball_vel)
    if <<(ball_y) = (floor)> or <(ball_y) < (floor)>> then
        set ball_y to (floor)
        set ball_vel to ((ball_vel) * -0.8)
        if <([abs] of (ball_vel)) < 5> then
            set ball_vel to 0
        else
            start sound basketball bounce
    else
        change ball_vel by (ball_acc)
        turn ↻ 15 degrees
    go to x: (x position) y: (ball_y)
```