Snap! Looping and costumes

UBMS Game and Robo Programming with Snap! and Python

June 27, 2023

Green flag

- The project GreenFlag contains two sprites. Each sprite has its own script. There is no way to run both these scripts simultaneously.

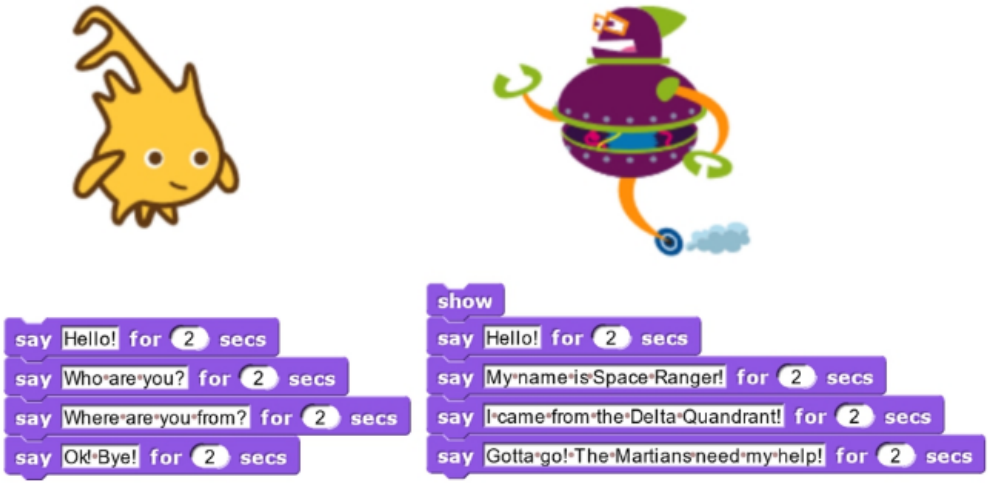


Figure 1: Two scripts for two sprites on one stage

- When attaching the *green flag* block to each of the scripts, they will start at the same time.
- But this is a conversation: to build in pauses, alter the program and use the `wait N secs` command:
- See project GreenFlag in the cloud (video/GDrive)
- Having a conversation, even between digital beings, relies on pausing to listen to the other - who'd have known!?
- Object orientation is about classes and methods (things the classes can do, but also about exchanging messages between objects (the ability to send and receive a message is a method, too).
- In Snap!, this will be mimicked by the `broadcast` and `receive` commands

Process model of a conversation

- There are two ways to implement this process in BPMN:
  1. the process has two lanes, one for each participant

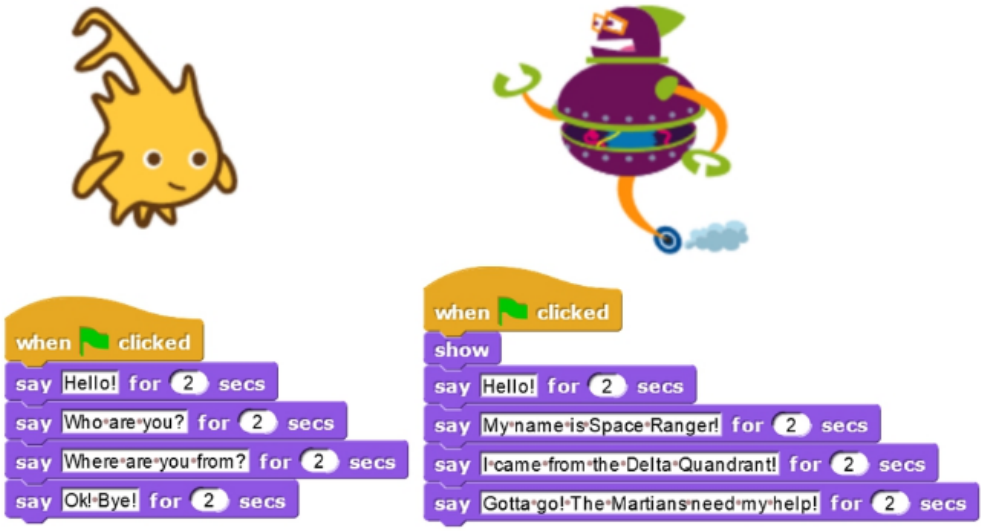


Figure 2: two scripts for two sprites on one stage with green flag

2. the process has two pools, one for each participant
3. ground rules: inside a pool, only sequence flow, between pools only message flow.

- Your job (bonus points) - I'll show you how to get started:

1. draw the process model for this conversation in bpmn.io
2. draw the 2-lane version
3. save the model as a `.bpmn` file to your PC
4. upload the file here to Canvas

- Beginning diagrams for 2-lane and for 2-pool solutions:

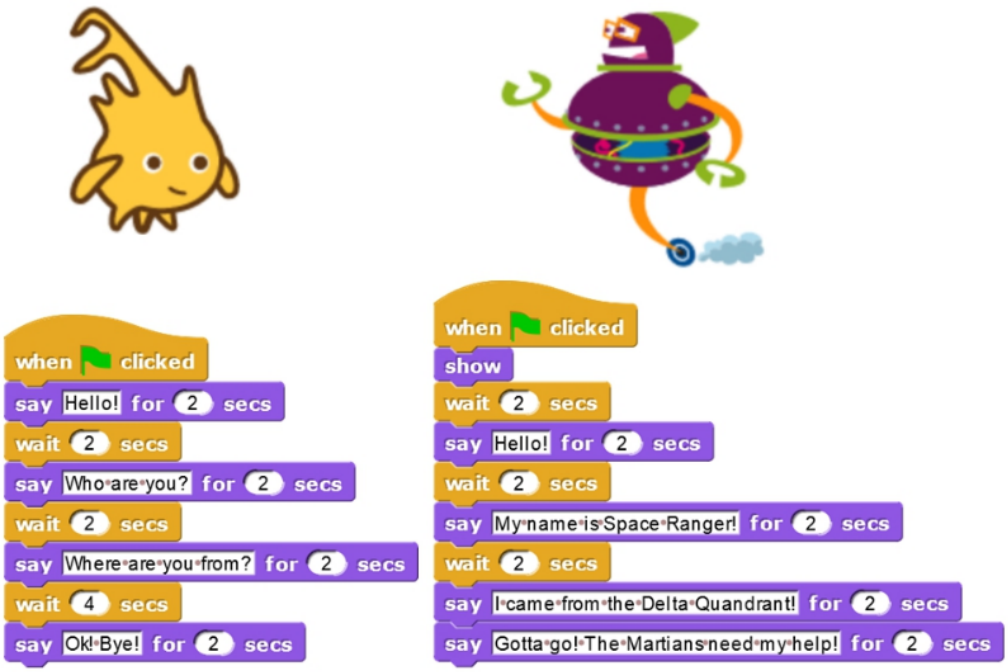
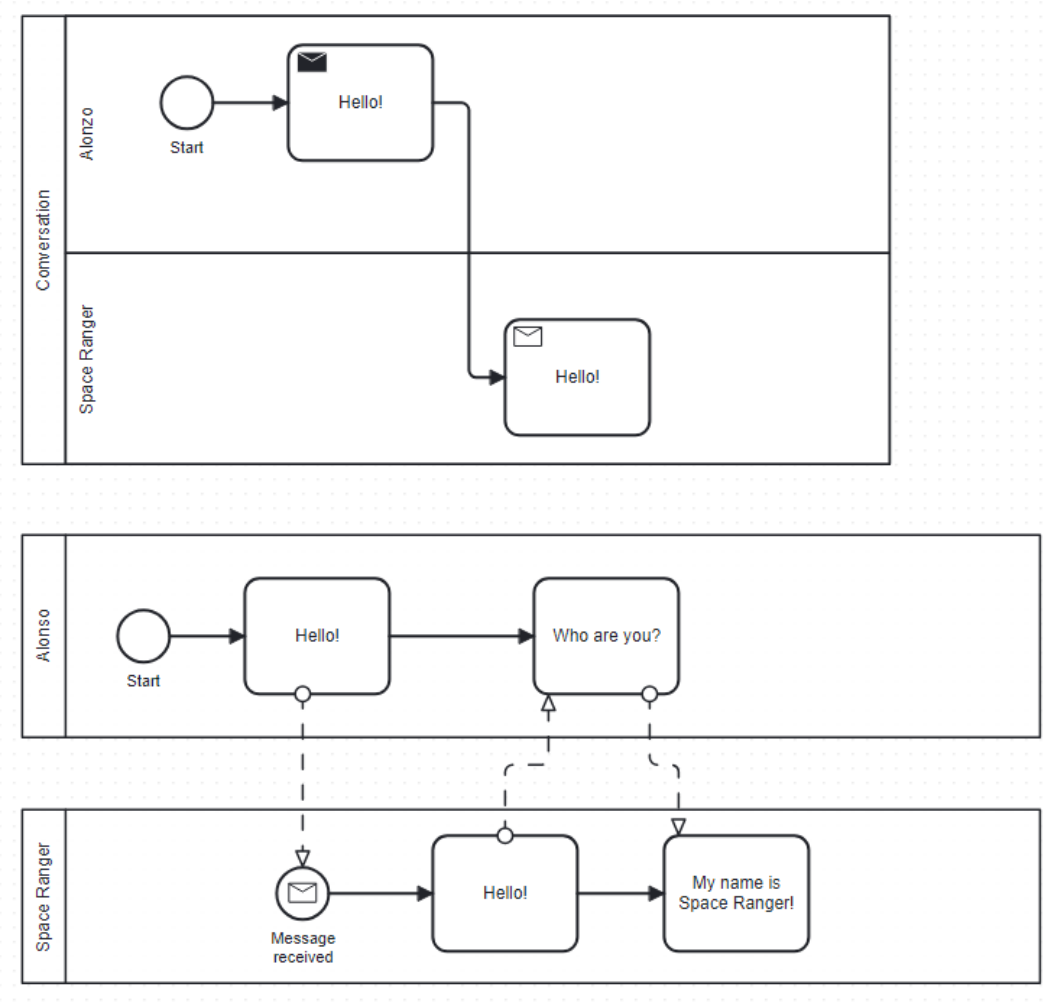


Figure 3: a conversation between two characters



- Bonus points will be applied at the end of the course only.

Simple Looping

- See project Looping online
- If you want to code along, open a new Snap! project now

Looping commands

- Looping (or *iteration*) is the repetition of a sequence of commands
- The commands `repeat` and `forever` in Snap! allow simple looping

☐ What does this command do? The sprite will move around in a square. Each move takes it 100 steps before it turns 90 degrees.

☐ What does this command do? The sprite will spin around itself forever - one full spin will take $360/5 = 72$ iterations.

Figure 4: Simple loop with 'repeat'

Figure 5: Simple loop with 'forever'

Jumping up and down

- To make a sprite jump up and down repeatedly:
  1. get the basic command sequence (what the sprite should do)
  2. repeat the sequence
- This command moves the sprite along the y-axis (vertically):

Figure 6: Gliding in 1/2 second to (0,100)

- This command brings the sprite back to the origin:
- Finally, we use `repeat` to iterate four times:
- See the complete code here. The XY-geometry of the background will be covered in an upcoming lesson. When animating, add the X-Y-grid as a background as I've done it here.

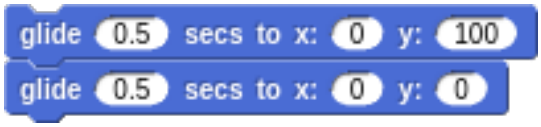


Figure 7: Gliding in 1/2 second between (0,100) and (0,0)

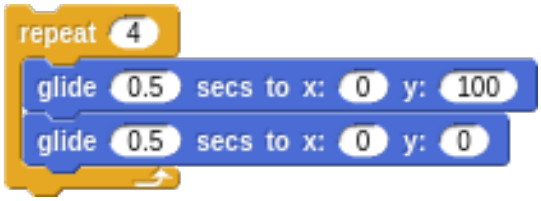


Figure 8: Gliding in 1/2 second between (0,100) and (0,0) 4 times

Smooth motion

- Place your sprite somewhere near the left edge, add and click the following script:

Figure 9: Seemingly instantaneous jump to the right

- Looping reveals that these commands are not instantaneous: the sprite moves the same number of 300 = 10 * 30 steps.
- Reducing the number of steps per iteration and increasing the number of iterations still moves 300 = 30 * 10 steps, but the movement is now much smoother.

Practice - looping

1. Define a new project called "Looping".
2. Create three `sprites` and pick different `costumes` for them so that you can distinguish them. Name them according to the action.
3. Make the `spinning` sprite spin around itself really slowly `forever`. Tip: slow motion means many small degree changes.

Figure 10: Few loops, large steps make for choppy motion

Figure 11: Many loops, small steps make for smooth motion

4. Make the `jumping` sprite jump up and down 10 times: the sprite should move down slowly (over 2 seconds) and bounce up fast (over 0.5 seconds).

5. Make the `smooth` sprite move smoothly 200 steps to the right. Tip: to bring a sprite back that has left the stage, move it with a negative number of steps - e.g. `-200`.

6. When you're all finished, save your project to the Snap! cloud and share the URL in the Google `Snap!Chat` **in the thread that I'll start** in class. Thanks!

Practice solution - looping

Rotation style

- If you want your sprite to walk horizontally (East-West) forever, let it bounce upon hitting an edge.

- You can control the orientation of the sprite after bouncing using the rotation style buttons located left of the little sprite image that indicates the currently chosen rotation style:

    1. top = sprite can rotate when hitting the edge

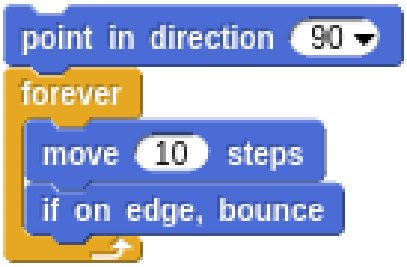


Figure 12: Sprite bounces off edge in chosen rotation style

Figure 13: Sprite bounces off edge in chosen rotation style

2. middle = sprite turns around instantly when hitting the edge
3. bottom = sprite doesn't rotate, keeps direction at edge

Practice rotation styles

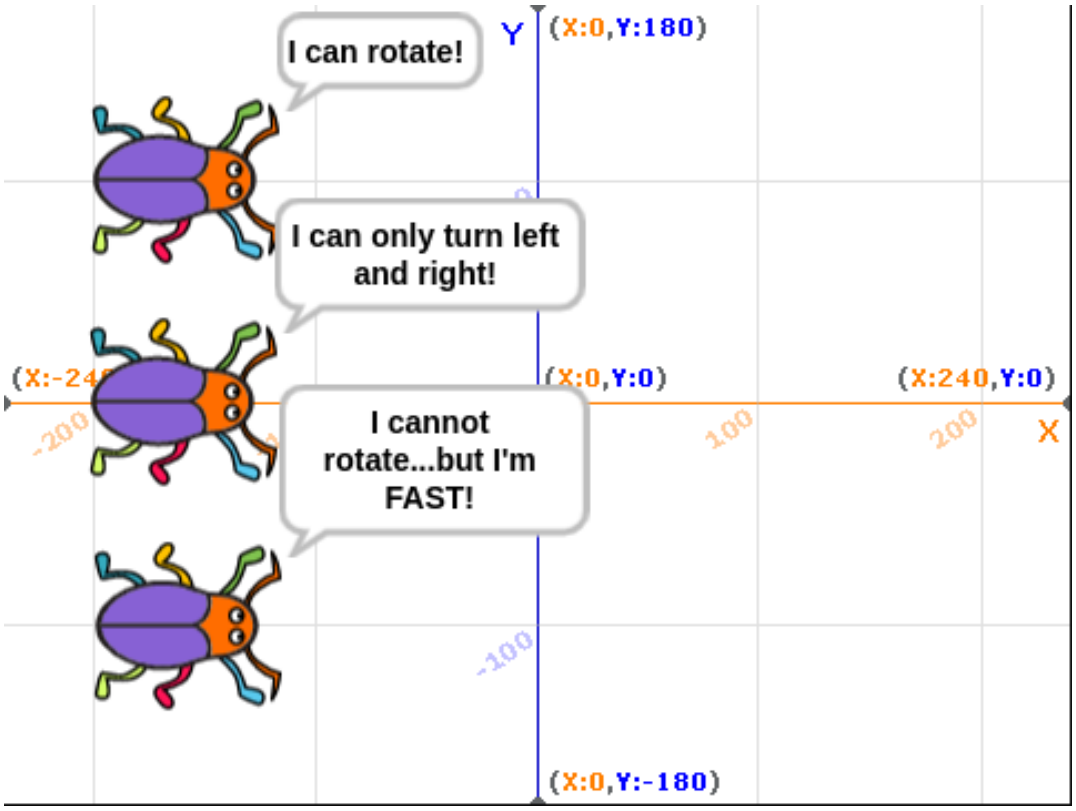


Figure 14: Illustrating three rotation styles upon hitting the edge

1. **Create** a project that implements the three rotation styles as shown in this video (GDrive).
2. **Start** by defining three sprites, and download the beetle costume. **Tip: you can download it once only and drag it onto a sprite in the sprite area below the stage.**
3. **Script:** point the beetle in the same direction (90). In a `forever` loop, make it `move 10 steps` and if on edge, bounce it.

4. Choose a **different rotation style** for each beetle.

5. You need three identical scripts (remember a script is tied to a sprite). To start all three beetles at the same time, add a Green Flag start command. **Tip: You can duplicate scripts and drag them onto sprites, too.**

6. *Optional:* once you're done, add speech bubbles and experiment with making them start at different times, move at different paces, and add the XY geometry background (as shown in my video).

7. When you're all finished, save your project to the Snap! cloud and share the URL in the Google `Snap!Chat` **in the thread that I'll start**. Thanks!

Practice solution - rotation style

Animation using costumes

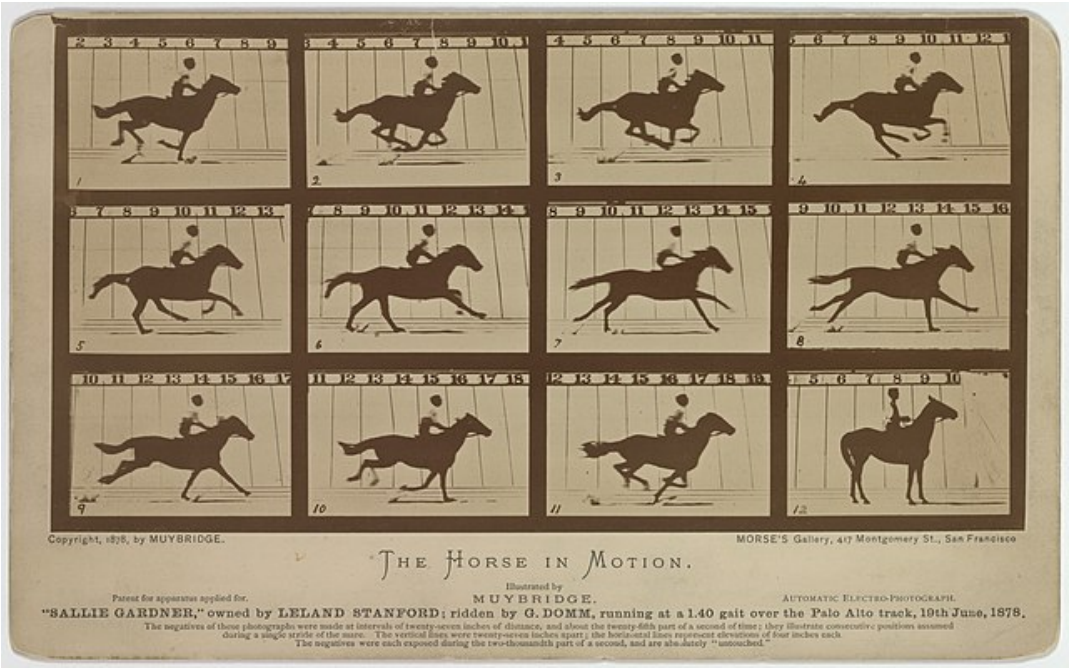


Figure 15: Horse in motion by Eadweard Muybridge (1878)

- Animation (images brought to life) is an illusion of action or motion, a trick played on our eyes (and ears, with sounds).
- To create this illusion visually, you can use a series of sprite costumes - going through the sequence is like Muybridge's horse:

Practice animation using costumes

1. Define a new project and call it `animation`.
2. Make a sprite and name it "walk".
3. Open the *Costumes* menu of the sprite. Download 4 costumes of *"avery walking a/b/c/d"* from the library for the same sprite.
4. Show avery dragging her feet: write a script that makes her move to the right 10 steps at a time **whenever the space bar is pressed** (that's a control command). Do this a few times, then drag *avery* back to her starting position.
5. Add the command "next costume" at the end of the script and run it again: avery now seems to walk to the right side of the stage. In fact, you move through four different static costumes.
6. Make avery walk *frantically* off stage: enclose the last script in a "forever" loop and add a green flag starting command at the top.

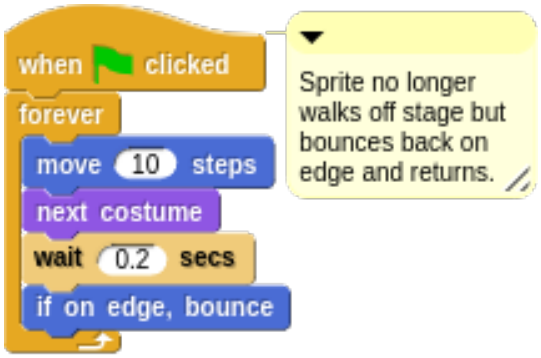
Figure 16: Four "Avery walking" costumes from the Snap! library

7. To bring the sprite back to the stage, right click in the stage area and choose "Show all". You'll have to drag the sprite to the starting position.

8. To stop the frantic motion, add a "wait 0.2 secs" command at the end of the script. Avery now walks normally.

9. Finally, use your knowledge of rotation styles to stop Avery from walking off stage and give her a suitable background to walk in.

   Add "if on edge, bounce" after the "wait" command inside the "forever" loop, and change the rotation style to "only face left/right":



10. When you're all finished, save your project to the Snap! cloud and share the URL in the Google `Snap!Chat` **in the thread that I'll start**. Thanks!

Practice solution: Project animation