Snap! Broadcasting

UBMS Game and Robo Programming with Snap! and Python

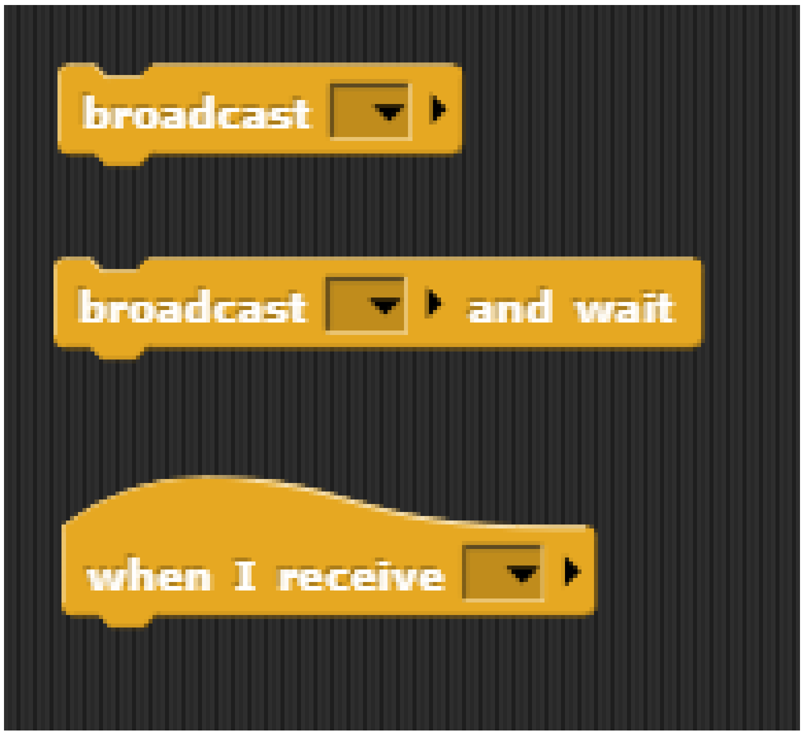
June 27, 2023

Synchronize sprites with broadcasting

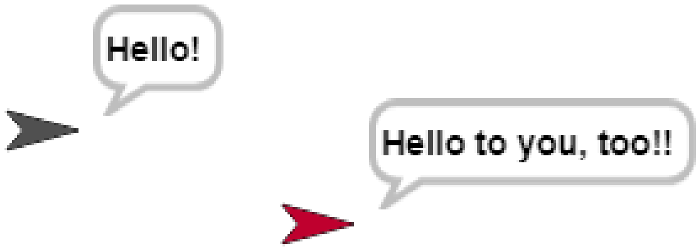
- Two sprites are not automatically synchronized
- We can synchronize manually or rely on *broadcasting*
- Broadcasting = sending message to a listener
- This is also the basis of OOP [Object Oriented Programming]

Figure 1: Ernest Briggs and bucherbird listening to gramophone

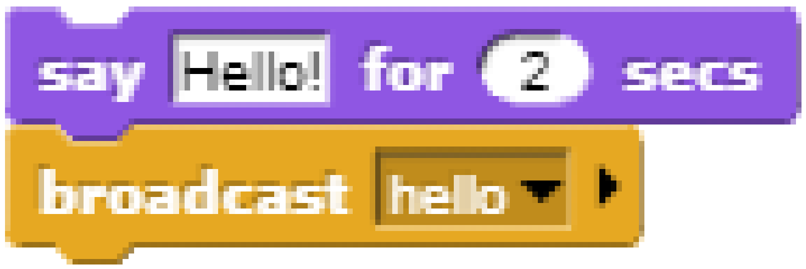
- Snap! broadcasting commands:



Example: greeting

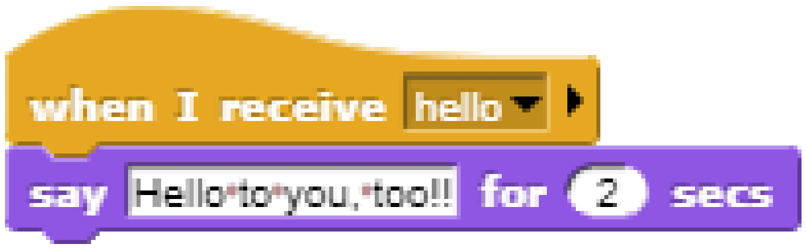


- `Sprite_1`:



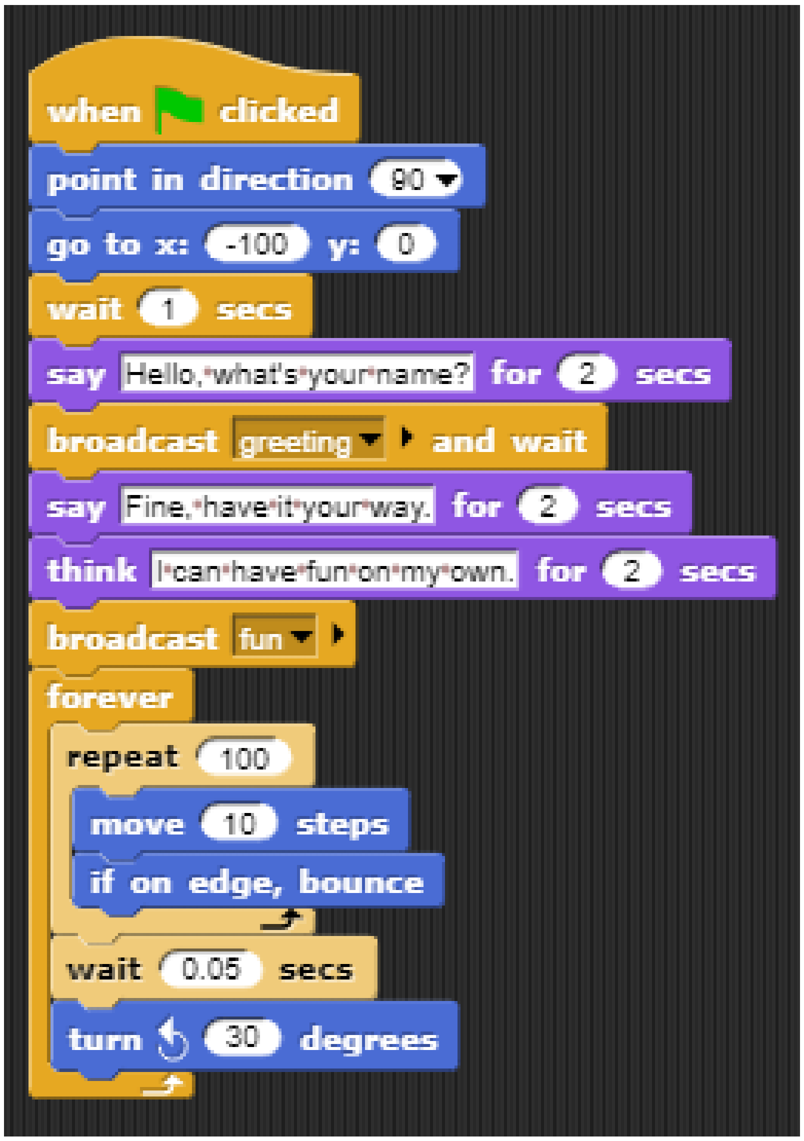
```
say [Hello!] for (2) secs
broadcast [hello ▼]
```

- `Sprite_2`:

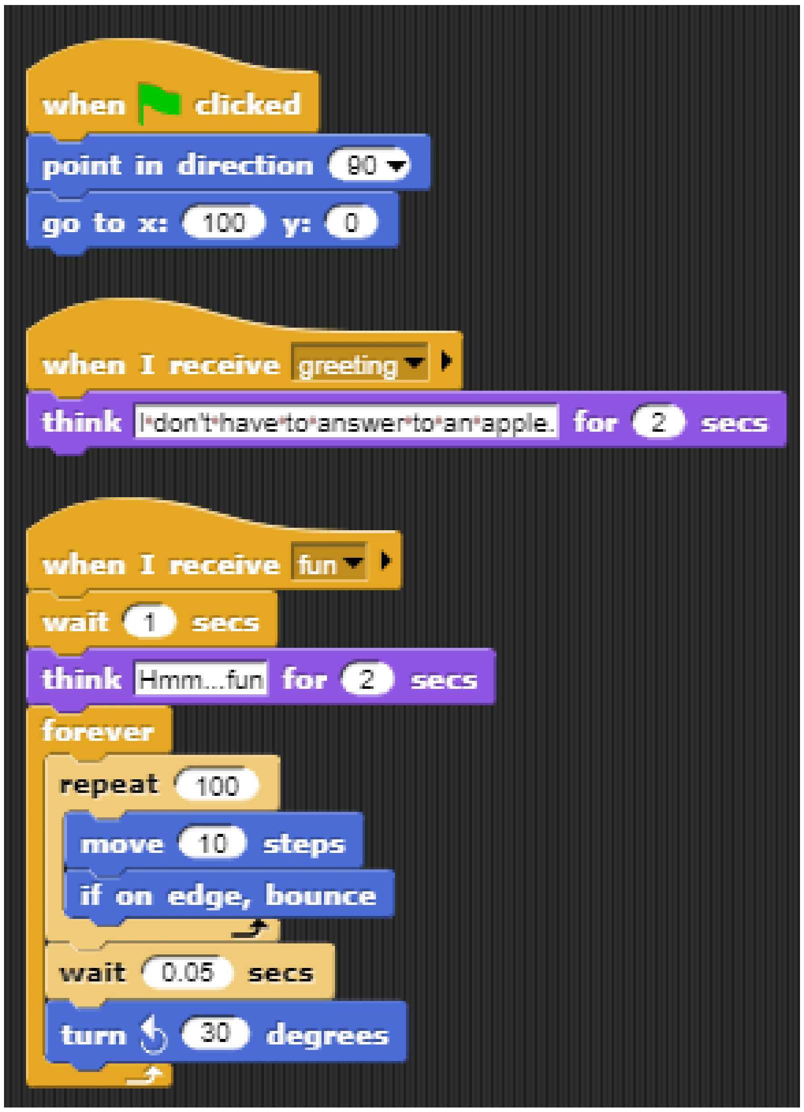


```
when I receive [hello ▼]
say [Hello to you, too!!] for (2) secs
```

- Here is a more elaborate example program:



```
when ⚑ clicked
point in direction (90 ▾)
go to x: (-100) y: (0)
wait (1) secs
say [Hello, what's your name?] for (2) secs
broadcast [greeting ▾] and wait
say [Fine, have it your way.] for (2) secs
think [I can have fun on my own.] for (2) secs
broadcast [fun ▾]
forever
    repeat (100)
        move (10) steps
        if on edge, bounce
    wait (0.05) secs
    turn ↺ (30) degrees
```



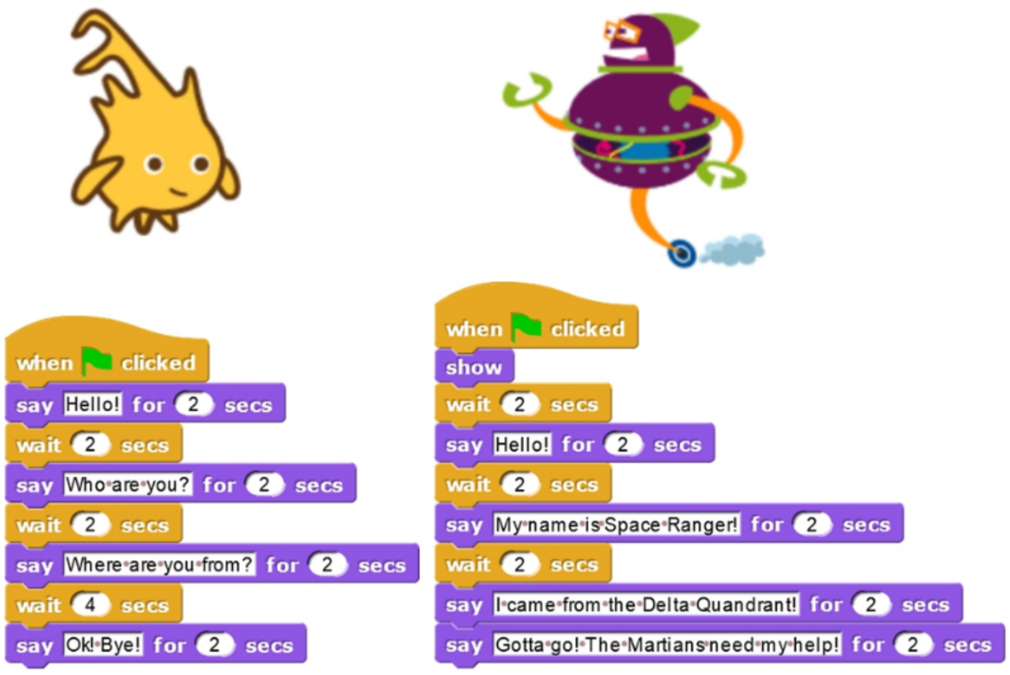
```
when [flag] clicked
point in direction (90)
go to x: (100) y: (0)

when I receive [greeting]
think [I don't have to answer to an apple.] for (2) secs

when I receive [fun]
wait (1) secs
think [Hmm...fun] for (2) secs
forever
    repeat (100)
        move (10) steps
        if on edge, bounce
    wait (0.05) secs
    turn ↺ (30) degrees
```

Practice broadcasting

- Turn the conversation between Alonzo and the Robot into a broadcast/receive script.
- Open this project to get the original script: bit.ly/greenFlag
- Duplicate the whole script for each sprite before changing it



```
when [flag] clicked
say [Hello!] for (2) secs
wait (2) secs
say [Who are you?] for (2) secs
wait (2) secs
say [Where are you from?] for (2) secs
wait (4) secs
say [Ok! Bye!] for (2) secs
```

```
when [flag] clicked
show
wait (2) secs
say [Hello!] for (2) secs
wait (2) secs
say [My name is Space Ranger!] for (2) secs
wait (2) secs
say [I came from the Delta Quandrant!] for (2) secs
say [Gotta go! The Martians need my help!] for (2) secs
```

Figure 2: A conversation between two characters

- Once completed and tested, SAVE the project to your cloud repo for later reference
- Practice solution

Program 4: Circus

- Write a "circus" program in which the ringmaster calls animals (like an elephant) or actors (like a clown) one by one and each performs some act. Use at least one animal and one actor, and a suitable background.
- The animal/actor should be hidden at the start and appear upon being called.
- The actor/animal should hide when the performance is over.
- Upload your solution to Canvas
- Sample solution: Circus by Bryceton Church (Fall'22)
- Tip: when programming under time pressure, do NOT begin by making yourself feel good (looking for pics, applause sounds, etc.) but first lay down the logic and the code and then customize if you have time to spare (which in class or in a job interview, you don't).

Pair programming: project animation

1. What are the elements of a story? Look at this short (1:20 min) example: "A Warning" - Plotagon animation (2017), URL: youtube.

   Must have elements:

   - Plot (storyline with start and end event)
   - Characters (participants, more or less developed)
   - Message (meaning, could be a list of options)

   Some optional elements:

   - Climax, denouement (a way of ending/untie the knot)
   - Mood (humor, tragedy, farce)
   - 3-stage plot (Aristotelian poetics)

2. Team up with one other partner

3. Design a short story animation. Something like:

   - a nursery rhyme
   - a folk story

- your own story
- a clip from your favorite story or film or game

4. The story should be narrated as sub-titles or as part of the animation (you can create subtitles with the `write [..]  size [..]` code block in the `Pen` category.

5. Requirements:

   - Green flag should start the animation
   - At least 2 sprites
   - Draw at least one background (using the paint editor)
   - Use as many commands learnt so far as possible
   - Use sounds and costumes
   - Sprite names must be as per the story (not sprite1, sprite2 etc)
   - As each scene of the story appears anywhere on the screen, the animation of that scene should be visible
   - The story should be slow and easy to understand
   - The animation should take at least 10 seconds but not more than a minute or two.
   - Put a short description (or the story itself) in the project notes

   Here is a story example "High Ground" by Isaac Rice and Jakobe Alcorn (Fall 2022)

6. Upload your solution to Canvas when you're done.

7. This exercise may go over two sessions - if you can't finish on day one, we'll take as much as 1/2 of day 2 to complete.

8. At the end, we'll have a show of each short story!

9. Some of the best games are based on the best stories (check: Fallout 4, The Witcher, Ghost of Tsushima, Red Dead Redemption etc.)

10. You need to organize cooperation yourself: you can either work in one Snap! editor together, or you can split the task up (one of you works on the props/images/sounds, the other one on the code, or two people work on different parts of the code).