snap

COR100 Snap! Programming

README

- A COR100 course on snap! programming
- Includes transfer of Scratch3 ML applications to Snap!
- To open Snap!, go to snap.berkeley.edu and register

Sources

- Snap! intro and projects by Joshi / Video sample solutions
- openSAP course media computation and data science
- Berkeley home page
- ml-for-kids (notes)
- Al Sweigart on Udemy (Scratch 2)

Snap! vs. Scratch

- Web-based online application (HTML5), no desktop app
- Comparison between Snap! and Scratch (About Snap!):
  - Snap! allows the user to define **data structures** like trees, heaps, hash tables etc. because you can define a "list of lists".
  - In Snap!, you can also create **control structures** like functions that allow you to automate things.
  - Data and control structures are the backbone of any form of **imperative** computer programming.

Scratch interface

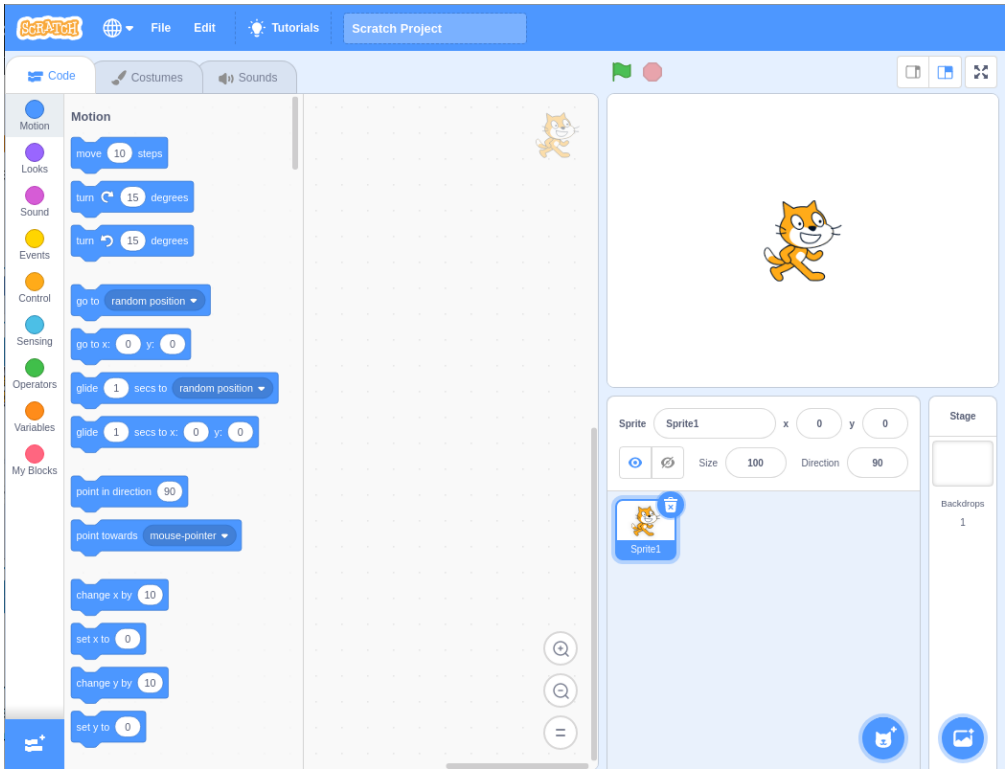


Figure 1: Scratch 3 interface

Snap! interface

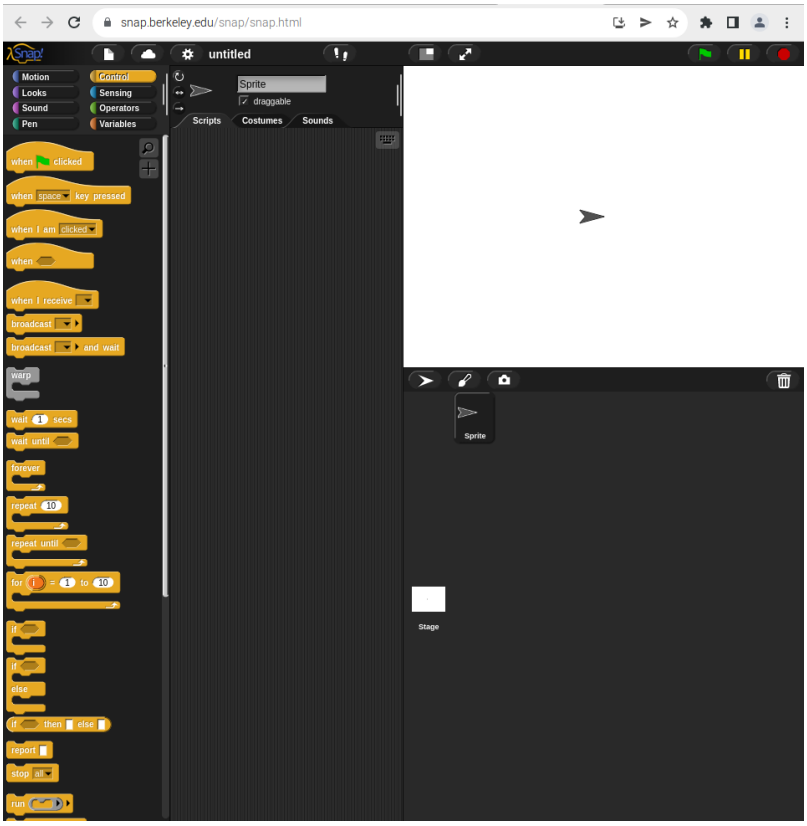


Figure 2: Snap! 7 interface

Introduction

Programming

- This could be your first ever programming course. Is it?
    - What were your other courses about?
    - What did you take away from them?
    - What's your view towards programming?
- Why should you bother to learn how to program?
    - Understand the relationships of humans and machines[1]
    - Develop critical thinking skills[2]
    - Create games and animations[3]
- The diagram shows different relevant levels of programming and computing including hardware (bottom half) and software (top half). In this course, we're working on *"Applications"* that use the computer to solve problems. Languages other than *Snap!* on this level include *C++*, *Java*, and *Python* (all of these are OOP languages[4]).

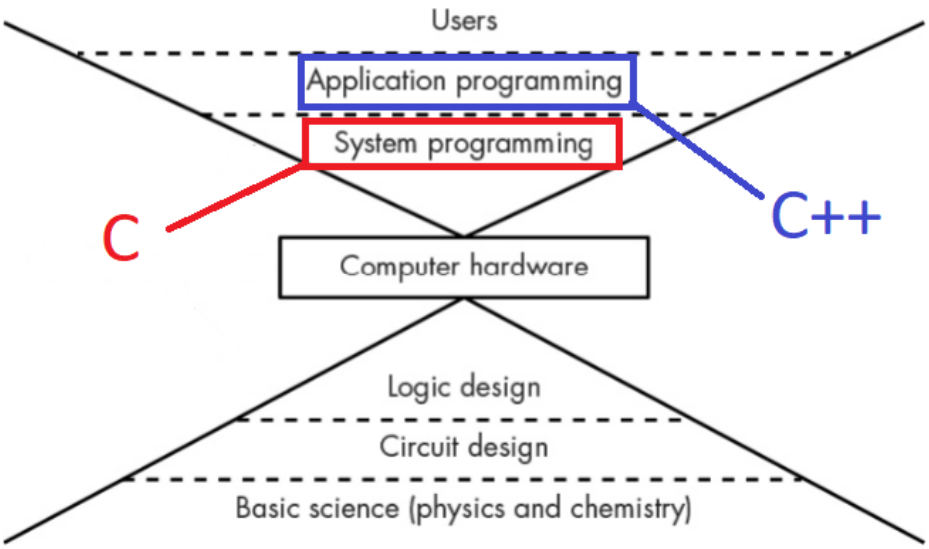


Figure 3: levels of computing (Source: Steinhart, 2019).

- The top level "Users" refers to most people who only use computers (mostly when operating their smart phones, or driving their cars). As with phones or cars, most of the power is under the hood.
- I recommend maintaining a (digital) notebook for this course. That's exactly what I did when working through the textbook, using the GNU Emacs editor with Org-mode.

Why Snap!

- It's full-fledged programming language
- It's easy to build animation and games in it
- It's instantly, freely available online
- It trains pseudocode and modular design
- It's good to go for data science applications (= my thing)
- I've always wanted to get into it

Unit 1: Looping, Broadcasting, Animation

Concepts

- [ ] Snap user interface (UI)
- [ ] Paint editor
- [ ] Sequence of commands
- [ ] Motion commands
- [ ] Simple looping (repeat, forever)
- [ ] Absolute motion
- [ ] Relative motion
- [ ] Smooth motion using repeat
- [ ] Nested looping
- [ ] XY geometry
- [ ] Costume-based animation

First look at Snap!

What is a User Interface?

- A *user interface* (UI) is the dashboard or platform that allows a user to interact with an application. It's the first thing that you, as a user, see.
- UI/UX is an important, relatively new, interdisciplinary field that includes art and design, usability analysis, etc. UX focuses on the user's path to solving a problem (like shopping online), while UI focuses on the look of the surface of an interactive product (like a web site for online shopping). More: freecodecamp.org video course.

Snap! user interface

- Connect to *snap.berkeley.edu* and register using your name and Lyon student email address.
- For offline use - on any computer that you can administer, i.e. where you can download and install programs as you please - download the source files from GitHub, unpack the files, and open `snap.html` in a browser.

- This is how the interface looks like:

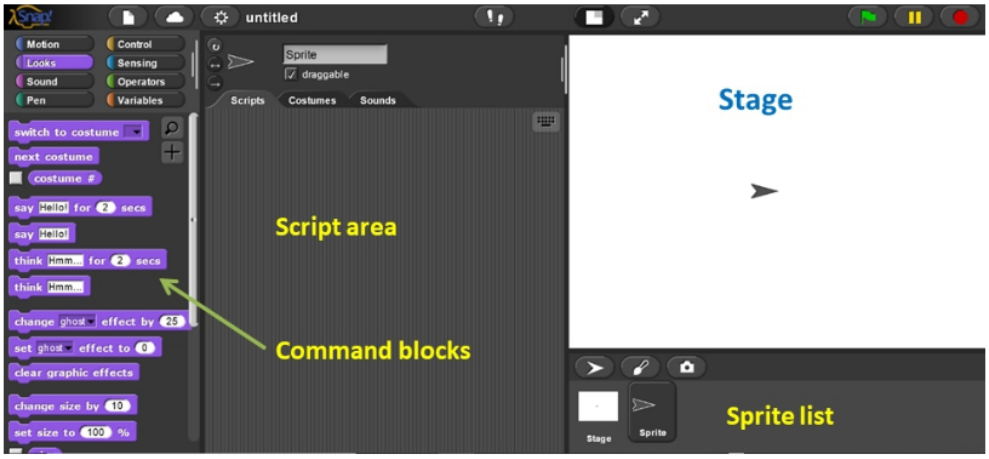


Figure 4: Snap! user interface (Source: Joshi, 2018)

- The interface is reminiscent of a movie maker's studio: *commands* are assembled in the *script* area, and the resulting action plays out on a *stage* with a cast of characters called *sprites*. Every sprite has a script associated with it.

- Compare with Windows Movie Maker - commands on the left, script in the lower half of the screen, sprites/characters in the middle, and stage on the right hand side.

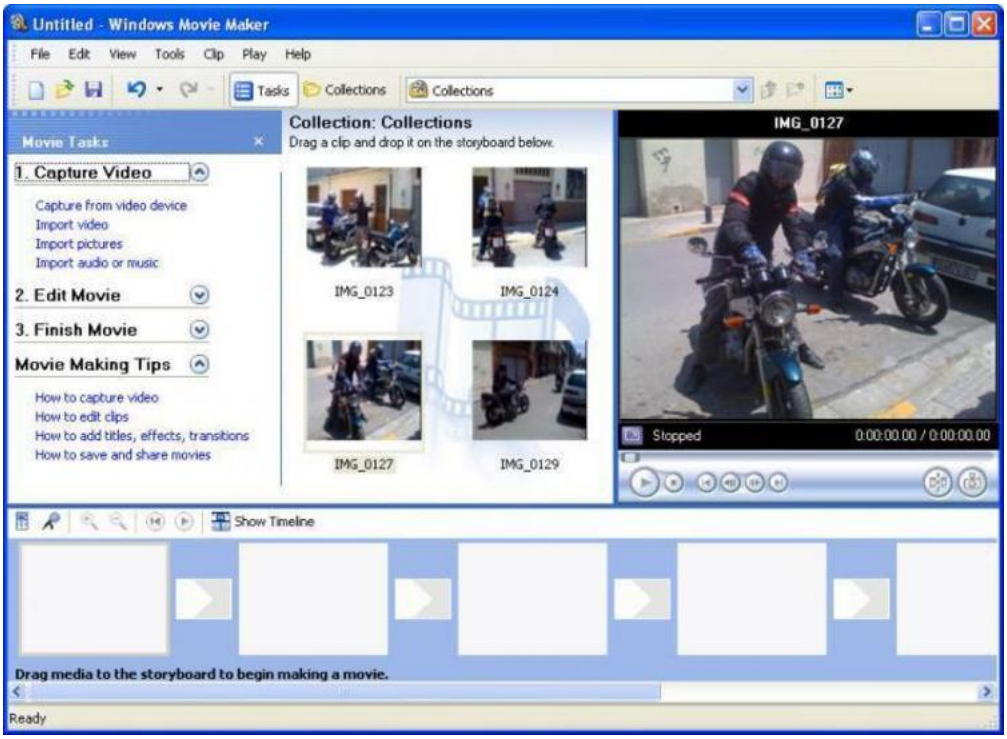


Figure 5: Windows Movie Maker

- As a programmer, you are writing the script for each sprite, including movements, sounds, and costumes, but you are also the producer, casting director, and editor.

Building a script

- To build a script, you drag and drop the blocks from the commands area into the script area and connect them like a jigsaw puzzle.
- Scripts are programs, sets of instructions for the computer. They need to be absolutely flawless: you need to be 100% diligent and careful when programming. Otherwise, the computer will refuse to cooperate.

Saving a Snap! project

- A Snap! *project* is a collection of scripts for sprites.

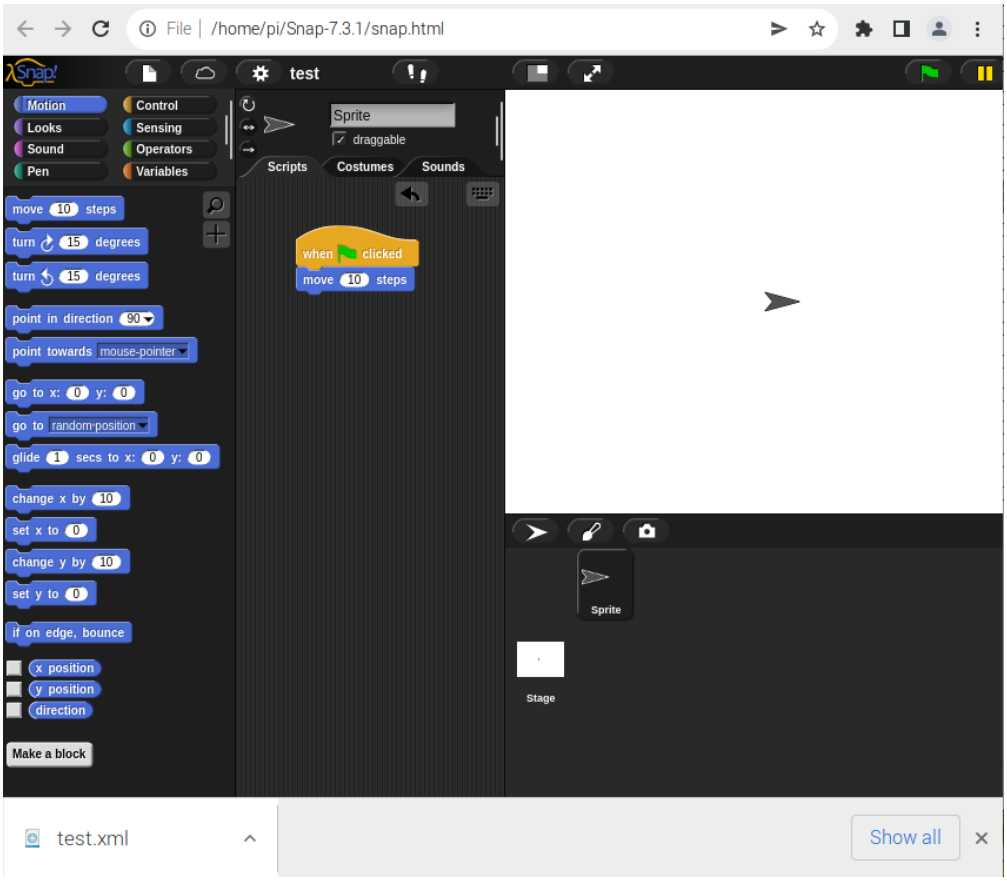


Figure 6: Snap! project example

- You can save your projects in your cloud account (if you are using the cloud version of Snap!), or you can save it locally as an XML file[5].

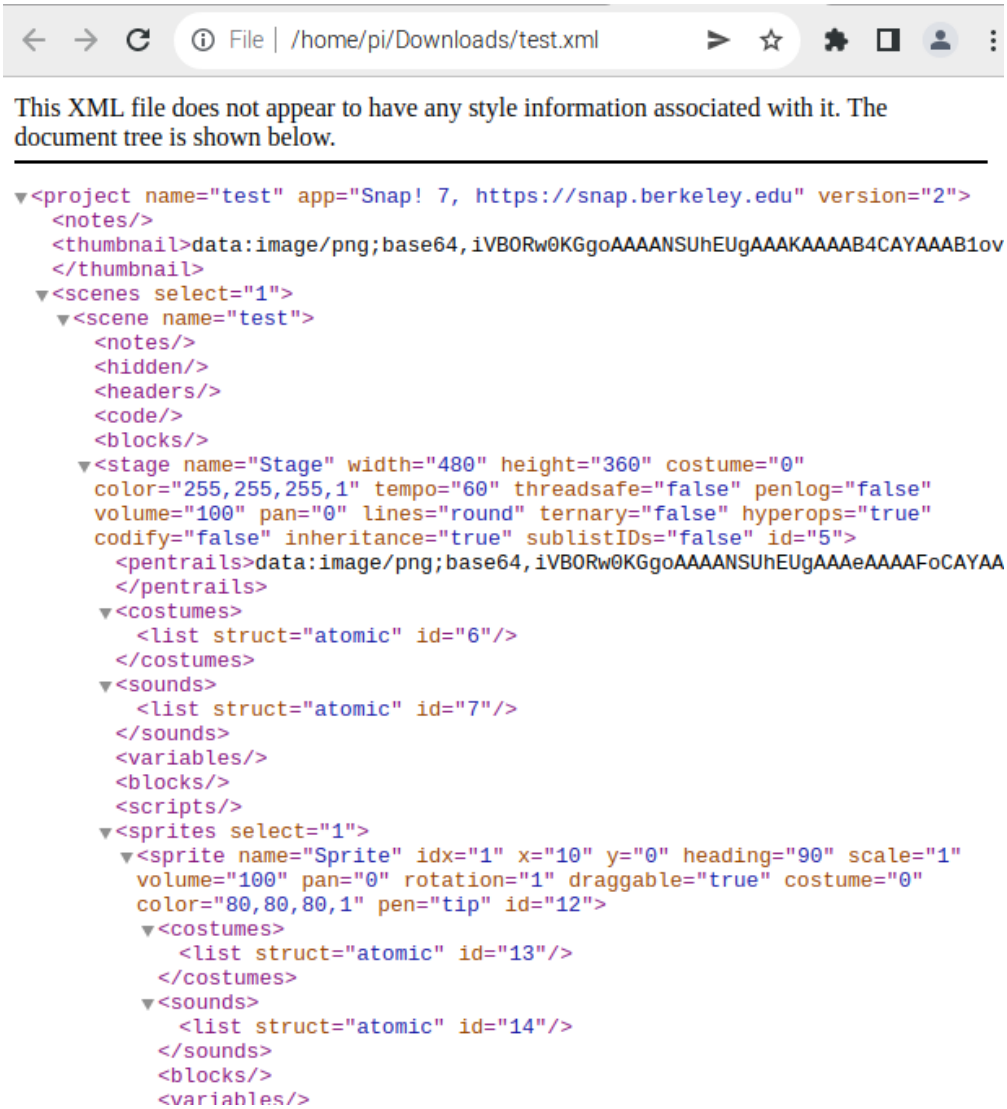


Figure 7: Snap! project example

Sprites and costumes

- When you add a new sprite, it always comes up as a "Turtle", a triangular shape.
- Every new Turtle sprite appears at a random place on the screen, facing a random direction, and has a random color.

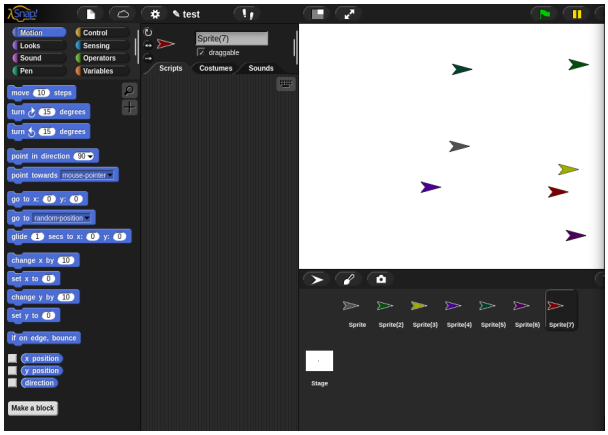


Figure 8: Snap! sprites.

- You can also use your camera to create a sprite.

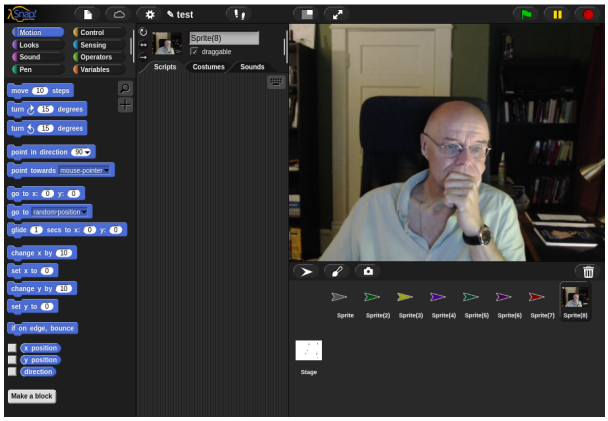


Figure 9: Snap! sprite, created with camera

- To change the appearance of the standard Turtle sprite, load a new costume. There are readymade costumes provided.

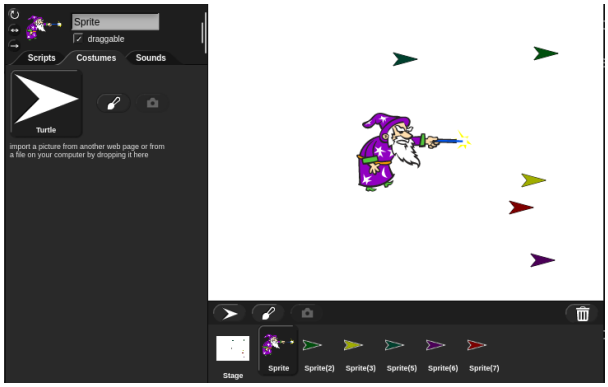


Figure 10: Snap! costume from the media library

- This is where the costumes library resides on my computer at home: `/home/pi/Snap-7.3.1/Costumes`[6].

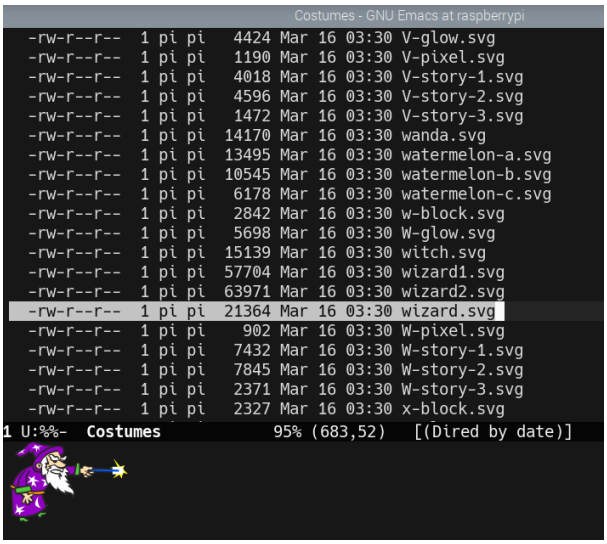


Figure 11: Snap! top menu

- You can also create or modify an existing costume using the paint editor.

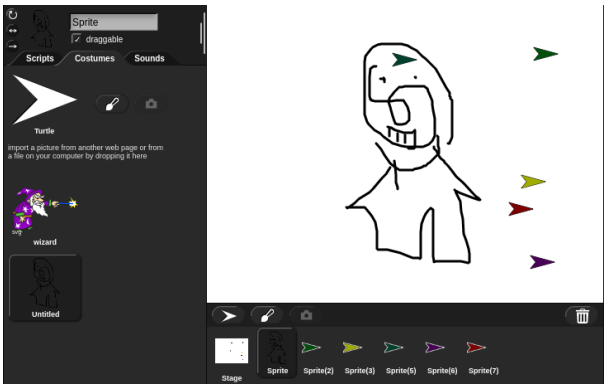


Figure 12: Self-drawn Snap! costume

- To import an image or go to the Costumes library, open the top (or "file") menu next to the Snap! logo, marked by a document symbol.

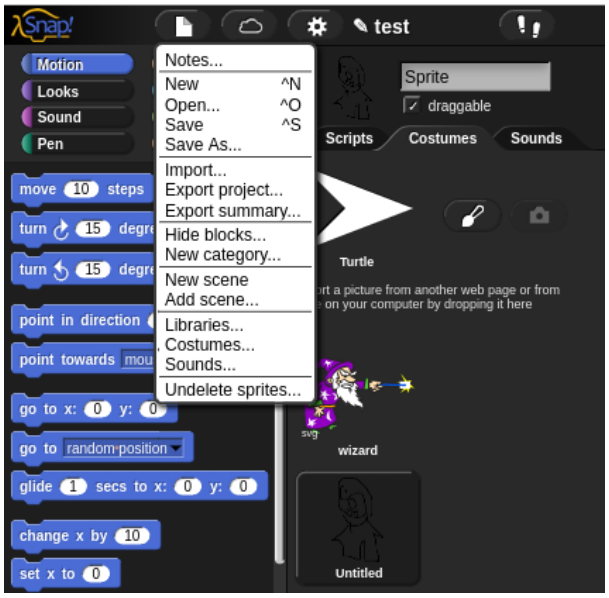


Figure 13: Snap! top menu

Stage or background

- Similar to the costume library, Snap! comes with backgrounds that you can load for your stage.

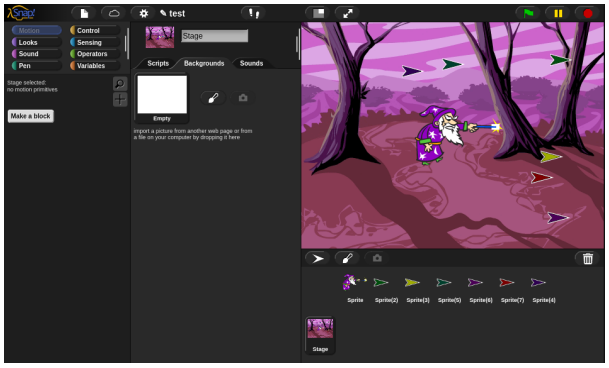


Figure 14: Snap! standard background `woods.gif`

- You can also modify or import backgrounds from your computer.

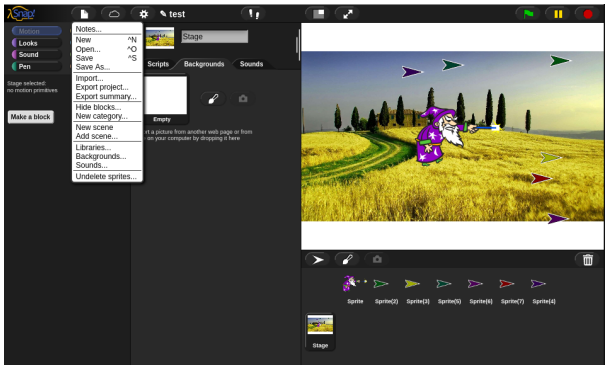


Figure 15: Snap! standard background `woods.gif`

Command blocks and scripts

- Scripts control the action of sprites (characters)
- Scripts are created by dragging command blocks into the script area and snapping them together

- You can run any command block (aka *programming statement*) by clicking on it. This Gif shows that for `"turn 90 degrees"`.

Figure 16: Snap! motion command to turn sprite clockwise by 90 degrees

Figure 17: GIF screenshot

- When a script is running, the command blocks used are glowing. Clicking on a running script again will stop it.

Figure 18: Snap! motion command that runs forever

Practice 1 - first script

1. Register an account with `snap.berkeley.edu`. Use your Lyon College email address and FirstnameLastname as Username, e.g. MarcusBirkenkrahe.

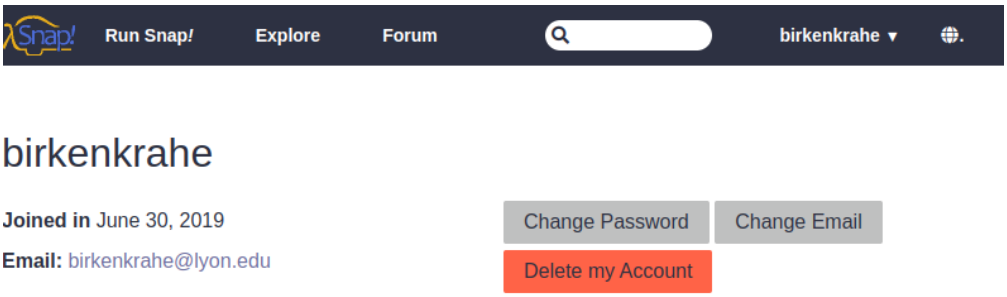


Figure 19: snap.berkeley.edu profile page

2. Create a new named project:
   - Open the main menu at the top
   - Click on `New` (a new project page opens)
   - Click on `Save As ...` and enter the name `FirstProject`
   - Save the project on your computer.
   - Open the file location to see where `FirstProject.xml` was saved

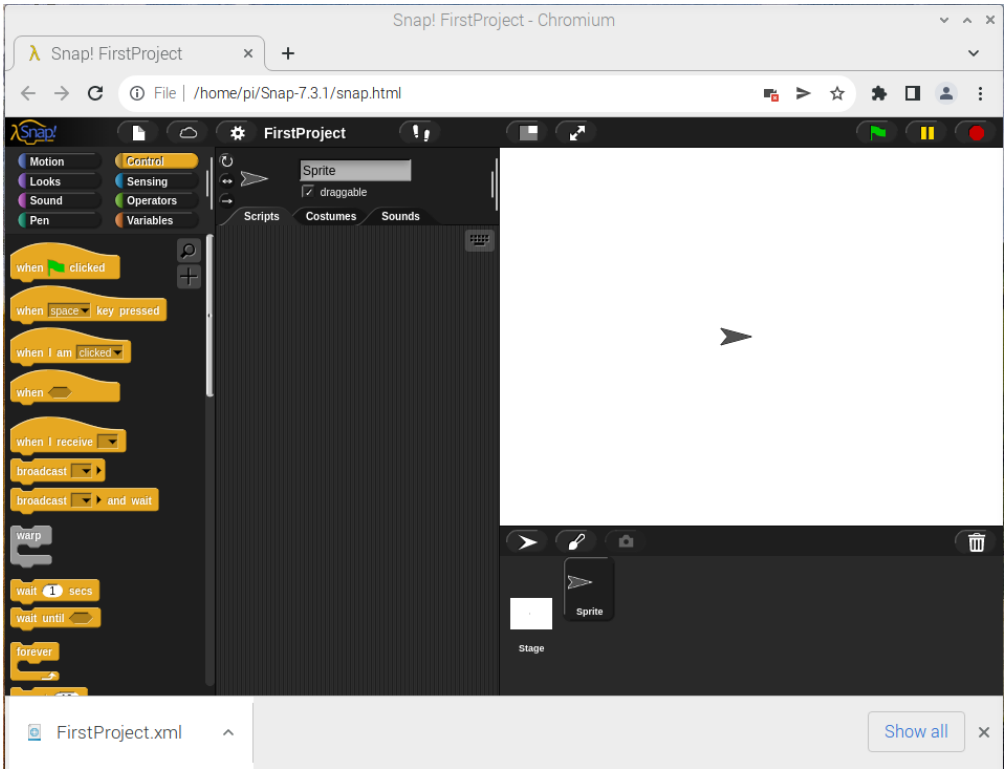


Figure 20: FirstProject in the Snap! desktop app

3. Create a new sprite and stage:

    - Add a new *Turtle* sprite
    - Open the *Costumes* menu from the main menu (at the top)
    - Click on the sprite icon and pick an animal or human *costume* for the *sprite* using the Costumes library
    - Click on the *stage* icon and pick a background for the *stage* using the Backgrounds library
    - Save your project to the cloud using `Save As ...` and then choosing the location `Cloud` instead of `Computer`
    - Go to `My Projects` on the Snap! website and find your project

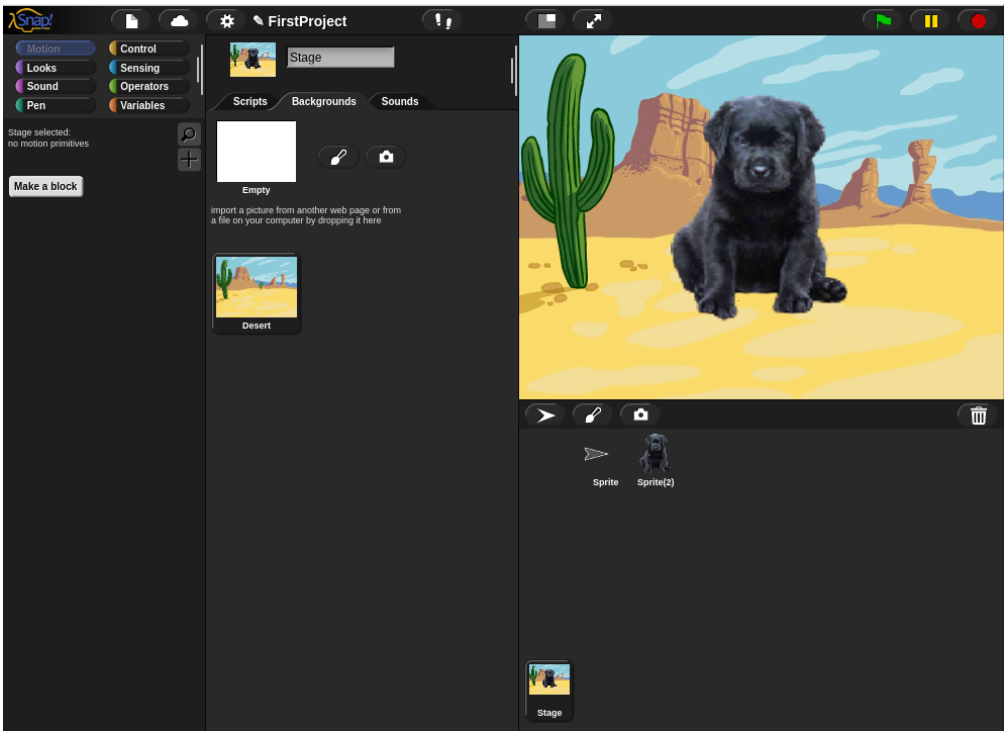


Figure 21: New sprite with new background.

4. Create a simple script with a standard Turtle:
    - Go back to the *Scripts* tab. If the `Motion` command palette is greyed out, then your chosen sprite costume cannot be moved and you need to pick another.
    - Make your sprite `point towards center` of the stage
    - Make your sprite `move 200 steps`
    - Make your sprite `go to a random position`
    - Make sure that all your statements/commands are attached to one another in the prescribed order
5. Run script:
    - Run the script a few times by clicking on any of the statements in the script
    - Go to the `Control` command palette
    - Make your sprite `wait 1 secs` between moving and going to a random position
    - Run the altered script a few times to make sure it does what it should
    - Execute the script `forever` by including it in a `forever` loop
    - Stop the program by clicking on the script, or by clicking on the red STOP symbol at the top above the stage
    - When running, the final result should look like shown in this video (with your choice of sprite and background, of course)
    - Save your project to the cloud location (with `Save As ...`)
6. Share your project and upload the location
    - Go to your projects and share the project using the `Share` button.
    - You can now publish the project, which means that it will be visible (and searchable) in the Snap! website

    - On the project page, you can `Unshare` and `Unpublish` your project.

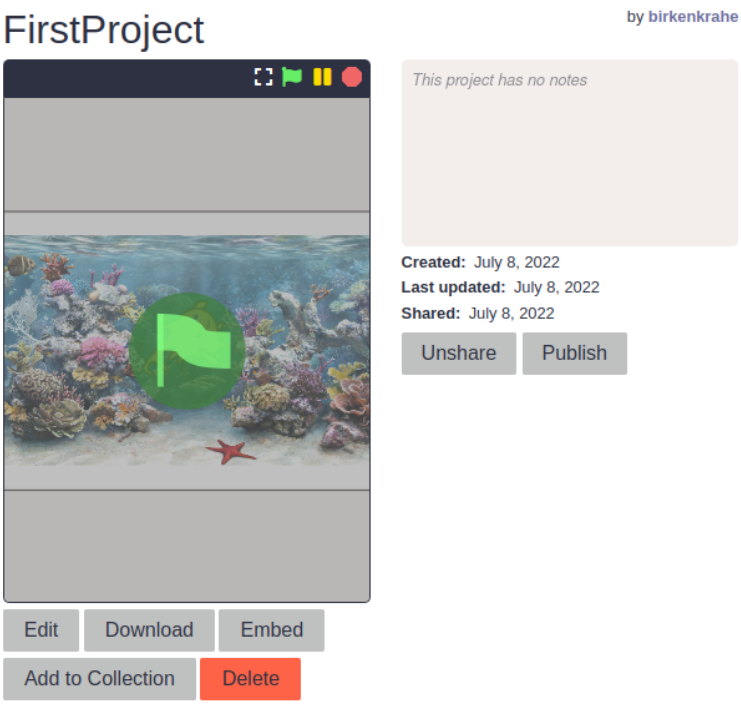


Figure 22: You can share/unshare, and publish/unpublish projects

- On the `My Projects` page, you also see if a project is shared and/or published.

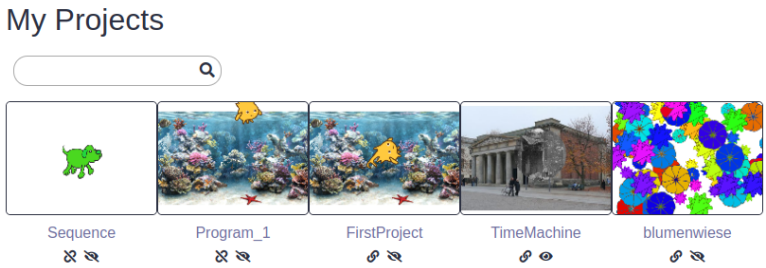


Figure 23: My "My Projects" page

  - You can add projects to collections.

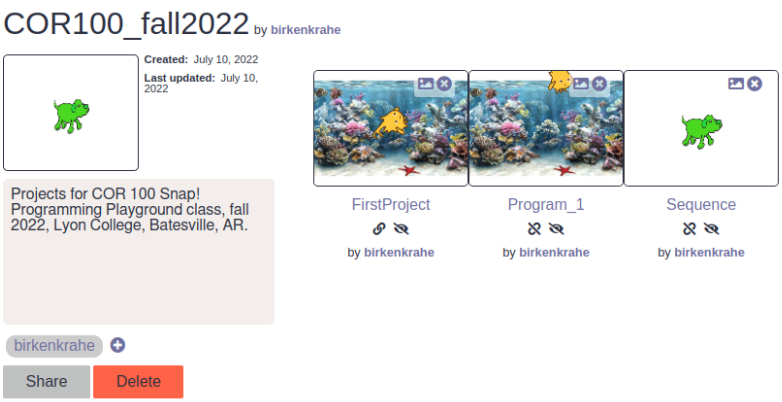


Figure 24: My collection of projects for this course

- Published projects and collections are displayed on your public page.

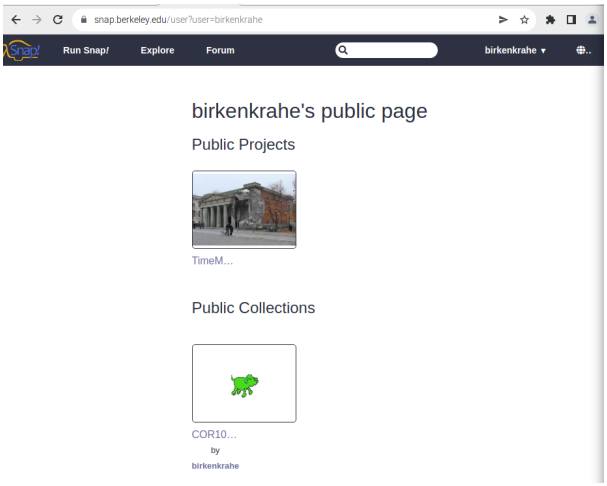
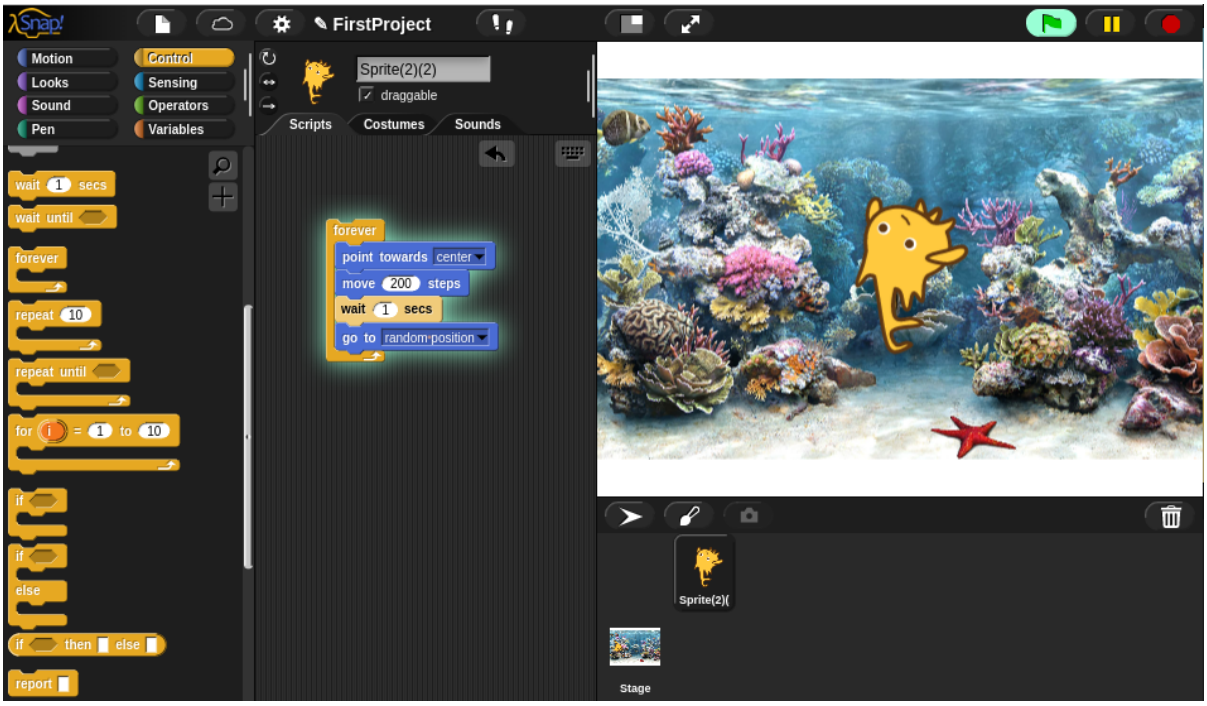


Figure 25: My collection of projects for this course

Practice 1 - sample solution

- Screenshot:



- YouTube video
- GDrive video
- Project URL

Quiz 1 - first look at Snap!

1. Match Snap! term and meaning

Snap! Command blocks	Computing instructions or statements
Snap! Script	Blocks snapped together for action
Snap! Stage	Space where the action is displayed
Snap! Sprite	Characters or actors on stage

2. Snap! allows you to use your own image/picture for new sprites

   TRUE

3. What is "Snap!"

   More than one answer is correct.

   - [X] A computer programming language
   - [X] A software in which you can create graphics and animation
   - [ ] A type of operating system like Windows or Linux
   - [ ] A collection of readymade animations and video games

4. See the script below:

The script will run when

- [X] You click on the script
- [ ] You press the ENTER key on your keyboard
- [ ] You click on the green flag symbol
- [ ] You click on the sprite

5. How can you stop a running script?
   - [X] You click on the red STOP symbol
   - [X] You click on the script
   - [ ] You click on the yellow PAUSE symbol
   - [ ] You click anywhere in the stage

Program 1 - Alonzo in an aquarium

1. Write a Snap! script for a sprite dressed in an "Alonzo" (vector) costume, with an aquarium background.
2. You can download a suitable background image from GitHub.
3. The sprite should `wait 1 second`, then `say Hello! for 1 second`, then `point towards a random position`, and finally `go to a random position` on the stage.
4. The script should run `forever`.
5. Make it possible to run the script by clicking the green flag symbol.
6. Add a description in the project notes (main menu, 1st option) that includes your name as the author, and the date.
7. Save the script to the cloud.
8. Save the script to your computer.
9. Submit the `.xml` file.

- Program 1 Solution
  - Screenshot:

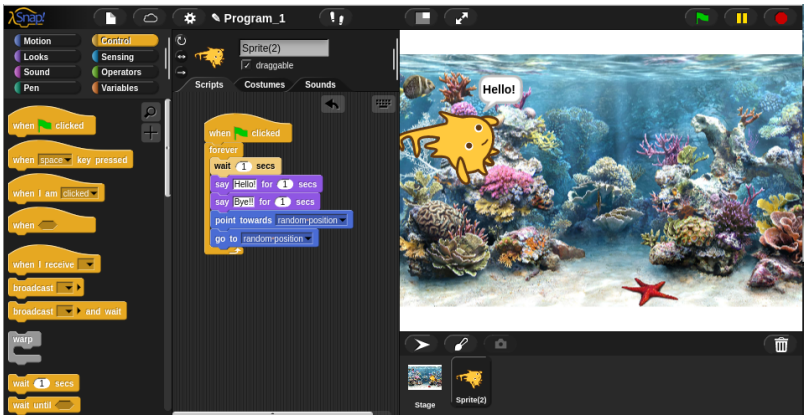


Figure 26: program_1

- Cloud with project notes description
- Video

Sequences

- The word *sequence* comes from the Latin word *sequi*, "to follow".
- Computers require exact sequences of steps or statements to work - like a to do list:
  1. Rise and wake puppy.
  2. Feed puppy.
  3. Walk puppy.
- Deviations from the prescribed sequence lead to errors.
- The worst errors are those that remain undiscovered: the script will run but it won't do what we expect it to do.
- A script like 27 below represents a series of steps.

Practice - Sequences

1. Define a new project "Sequence" and build this script in your Snap! dashboard. You can pick a non-standard sprite if you like.

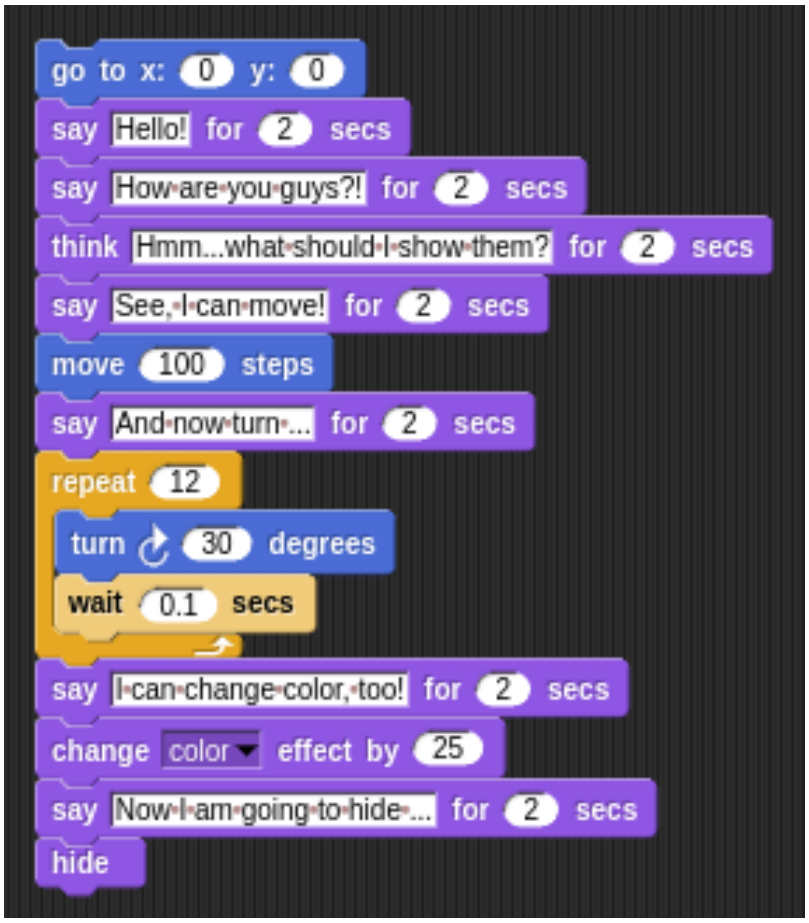


Figure 27: sample Snap! script (Source: Joshi)

2. When you try to run the script a second time, nothing happens. What could you do to re-run it properly?

Add the `Looks::show` command at the top.

3. This is how the result should look like (video).

Figure 28: screenshot - sample Snap! script (Source: Joshi)

4. Make the change and re-run the script.
5. Save the Sequence project to the cloud and find it in **My Projects**.

Sounds

- Every sprite has a `Sounds` tab
- You can *import* an existing sound from the Snap! library (`Sounds` option in the main menu), or you can *upload* your own sound (mp3).

Figure 29: dog sprite with library barking sound

- The `play sound ... until done` command moves do the next statement only if the sound file has been played.

Figure 30: Play sound … until done

- The `play sound...` command starts playing the sound file and moves immediately on to the next statement.

Figure 31: Play sound …

- You can stop all sounds with the `stop all sounds` command.

Figure 32: Play sound …

- `[ ]`

  What does this script sound like?

Figure 33: Script with sounds

- `[ ]`

  What does this script sound like?

Figure 34: Script with sounds

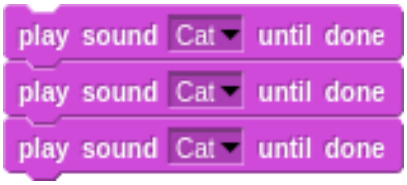
Practice 2 - Sounds

1. Define a new project "SoundCheck".
2. Pick a costume and a suitable sound.
3. Write a script that produces 5 consecutive sounds **without loop**.
4. `Duplicate` the script and add a `repeat` loop with only one sound command
5. Save the project to your `My Projects` list

Quiz 2 - sequence and sounds

1. Playing sounds 1

   How many times will you hear "Cat" sound when you run the script below?

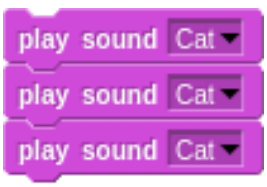


   - `[X]` 3
   - `[ ]` 2
   - `[ ]` 1
   - `[ ]` 0

   Feedback: the next command is executed as soon as the whole sound file has been played.

2. Playing sounds 2

   How many times will you hear "Cat" sound when you run the script below?



- [ ] 3
- [ ] 2
- [x] 1
- [ ] 0

Feedback: the next command is immediately executed and the sounds bleed into one another.

3. Motion and sound

   What will this program do when you click the script?

   - [x] It will move, wait for 1 second, and then meow
   - [ ] It will move and meow at the same time
   - [ ] It will not do anything
   - [ ] It will meow, wait for 1 second and then move

4. Hiding in plain sight

   You cannot `hide` a sprite on the stage.

   FALSE

5. Composing

   You can record a composition (a sequence of musical notes) and play it in Snap!

   TRUE

   - Example: harmonica melody

Figure 35: harmonica melody script

Program 2 - Soundbites

1. Create a new project named "`Soundbites`", with a script that does the following when clicked:
2. The sprite will greet you and tell its name.

3. It will walk `100` steps and then grow in size.
4. It will jump to `x = -100` and `y = -100`. It will then shrink in size.
5. It will say "`Hey, I am down here!`"
6. It will bid farewell with a sound and `hide`.
7. Add some `notes` to explain what the script does.
8. Save the `Soundbites` project to your projects in the cloud.
9. Save the script to your computer.
10. Submit the `.xml` file.

- Program 2 - Solution

  Workflow (always test script after adding to it):

  1. Open the main menu, select ~Costumes and pick a character
  2. Change to the `Scripts` tab
  3. Add `Looks::say "I'm Dino!`
  4. Add `Motion::move...steps` and enter `100`
  5. Drag the sprite back to a position on the left side of the stage (for this to work, the sprite must be `draggable`).
  6. Add `Looks::set size to 200%`
  7. Add `Motion::go to x:-100 y:-100`
  8. Enter `Control::wait 2 secs` after growing the size (otherwise you won't see the growth)
  9. Add `Looks::say "Hey, I'm down here!"`
  10. Open main menu, select sounds and import a sound ("Chord")
  11. Add `play sound "Chord" until done`
  12. Add `hide`

  To run the program again from the original position,

  - add `show` at the top, then
  - add `say "" for 2 secs`
  - add `clear graphic effects` at the very end

Figure 36: Program 2 - soundbites - script

- Video (GDrive)

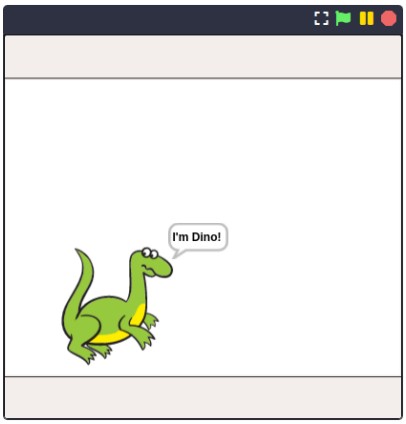


Figure 37: Program 2 - soundbites - screenshot

Bonus program: composition

Write a program that when clicked plays a composition (a sequence of musical notes) - uploaded or self-created. Save the project to the cloud and submit the `.xml` file.

- Example: harmonica melody

Green flag

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

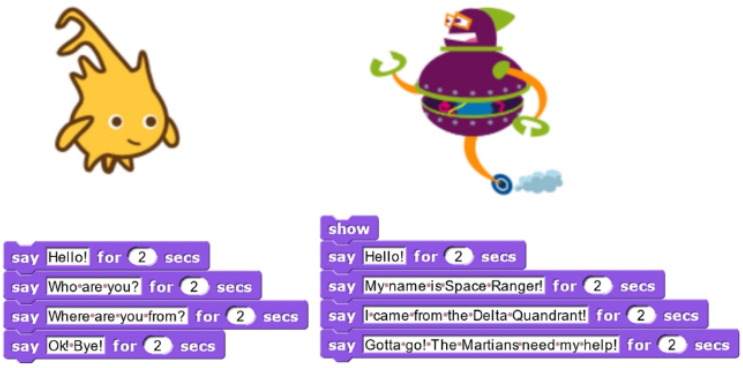


Figure 38: two scripts for two sprites on one stage

- When attaching the *green flag* block to each of the scripts, they will start at the same time.

Figure 39: two scripts for two sprites on one stage with green flag

- But this is a conversation: to build in pauses, use the `wait N secs` command:

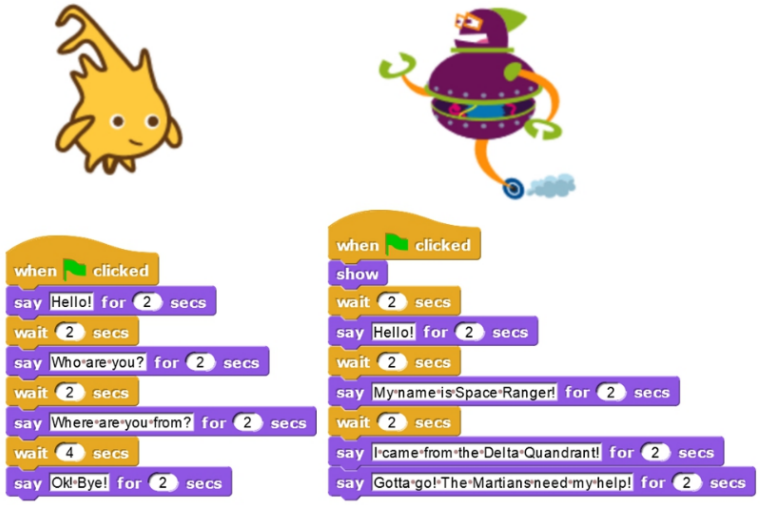


Figure 40: a conversation between two characters

- See project GreenFlag in the cloud (video/GDrive)

Looping

- See project Looping online

Looping commands: `repeat` and `forever`

- 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?

Figure 41: simple loop with repeat

  The sprite will move around in a square. Each move takes it 100 steps before it turns 90 degrees.

- [ ]

What does this command do?

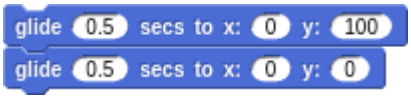
Figure 42: simple loop with forever

The sprite will spin around itself forever - one full spin will take 360/5 = 72 iterations.

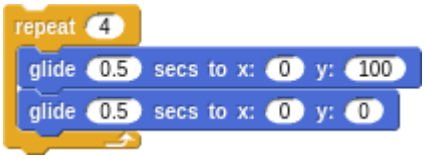
Jumping up and down

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

- This command brings it back to the origin:



- Finally, we use `repeat` to iterate four times:



- See the code here. The XY-geometry of the background will be covered in an upcoming lesson.

Practice jumping up and down

1. Define a new project called "Looping"

Smooth motion

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

  #caption: 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.

  #caption: loop over a move to the right 

- 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.

  #caption: loop over a move to the right 

Practice looping

Animation using costumes

TODO Practice animation using costumes

TODO Quiz 3: green flag, looping, and costumes

TODO Program 3:

References

- Joshi (2018). Learn CS Concepts with Snap! URL: abhayjoshi.net.

TERM	MEANING
Data structure	Order data to make the computer's job easier
Control structure	Order program flow to make jobs possible
Imperative programming	Telling the computer what to do
Application	Program telling the computer to do a job
Interface	Program between a user and the machine
Snap! Command blocks	Computing instructions or statements
Snap! Script	Assembled commands leading to action

TERM	MEANING
Snap! Stage	Where the action is shown
Snap! Sprite	Characters or actors on stage
Snap! Costume	Character image (default: "Turtle")
Snap! Sound	Library or recorded sound for a sprite
Green Flag	Function call (starts scripts together)
Loop	Iteration - repetition of command sequence

Footnotes:

[1] This issue leads into deep questions of philosophy, science, and even theology. People in all of these fields are split with regard to fundamental questions like "will machines ever be truly intelligent?", or "are humans not just very complex machines?" As a starter, check out the Stanford encyclopedia on AI (2018).

[2] This sounds kind of abstract but it is not. Just consider the that you can not not think (except perhaps when you sleep), and that thinking can have very different qualities, compare e.g. the statements *"I'm thinking of you"*, or *"I think therefore I am"*, or *"I think that programming is an important skill for any job."*

[3] We will do some of this in this course, but we're only scratching the surface. Fortunately, Snap! makes it fairly easy to develop fun games and animations.

[4] OOP = Object Oriented Programming is a programming paradigm that looks at the world as a collection of objects exchanging messages. This eases code reuse and allows you to define object class hierarchies.

[5] XML, or eXtensible Markup Language is a layout language that looks a lot like HTML, but instead of web page display its focus is on wrapping layout information in text-based, tagged files.

[6] This is a file address: the computer needs to keep track of all its files. To do this, it uses a hierarchy, like a tree turned upside down, with the *root* at the top. This particular address, `/home/pi/Snap-7.3.1/Costumes` means that the costumes files are located in a directory `/Snap-7.3.1` (which contains all files for the Snap! version 7.3.1), which is contained in a directory `/pi` (that's my username on this computer), which is contained in the directory `/home` right below the root directory `/`.

Created: 2022-07-14 Thu 20:41