Snap! Introduction

COR100 Snap! Programming

INTRODUCTION

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.
- How To Make A Video Game in Scratch (MIT course)

Scratch interface

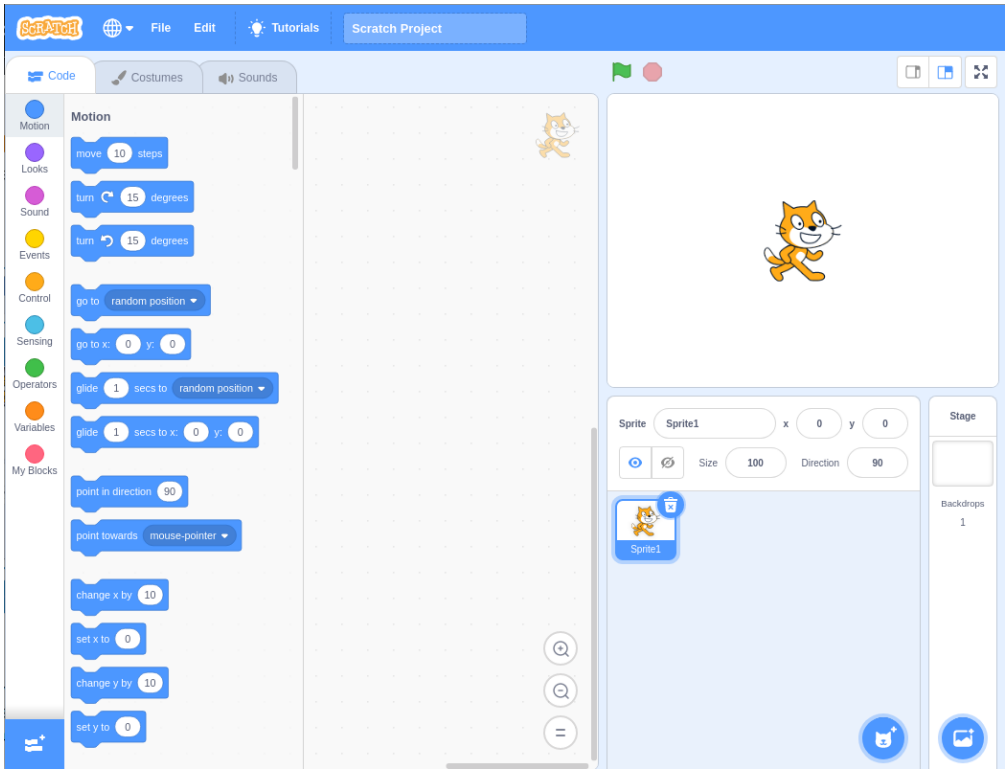

Figure 1: Scratch 3 interface

Snap! interface

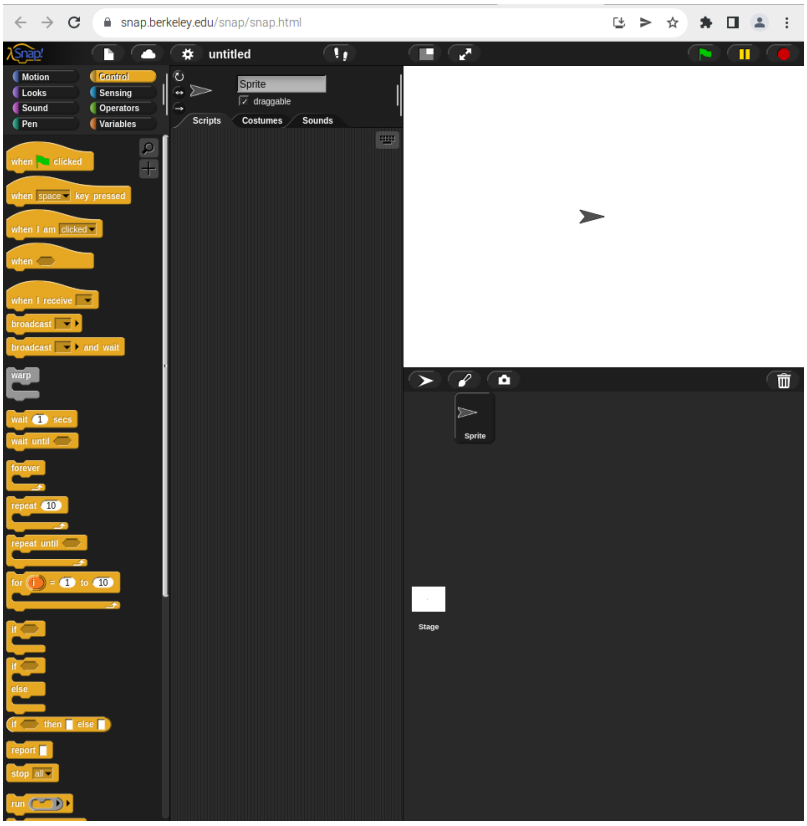

Figure 2: Snap! 7 interface

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]).

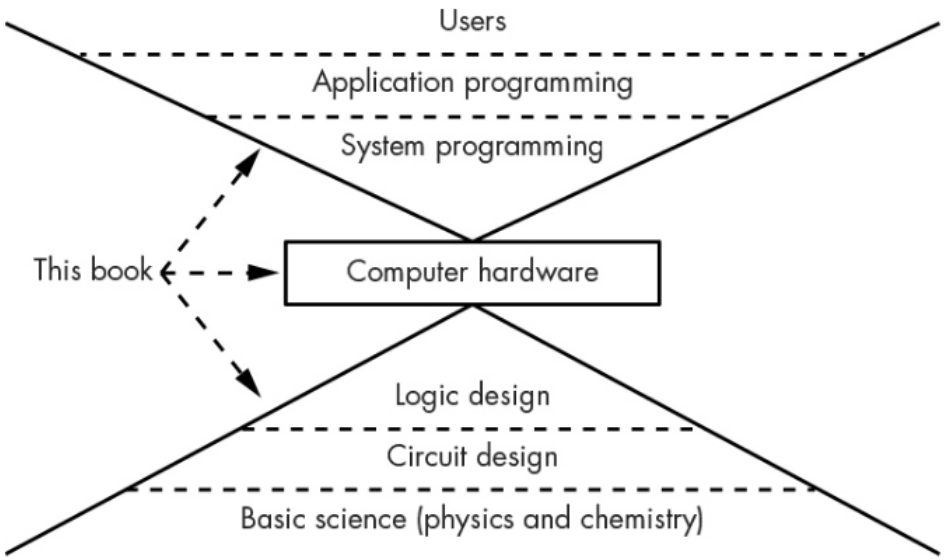

Figure 3: levels of computing (Source: Steinhart, 2019).

- 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 a 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 suited for data science applications
- I've always wanted to get into it

UNIT 1: LOOPING, BROADCASTING, ANIMATION

Concepts

- [X] 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:

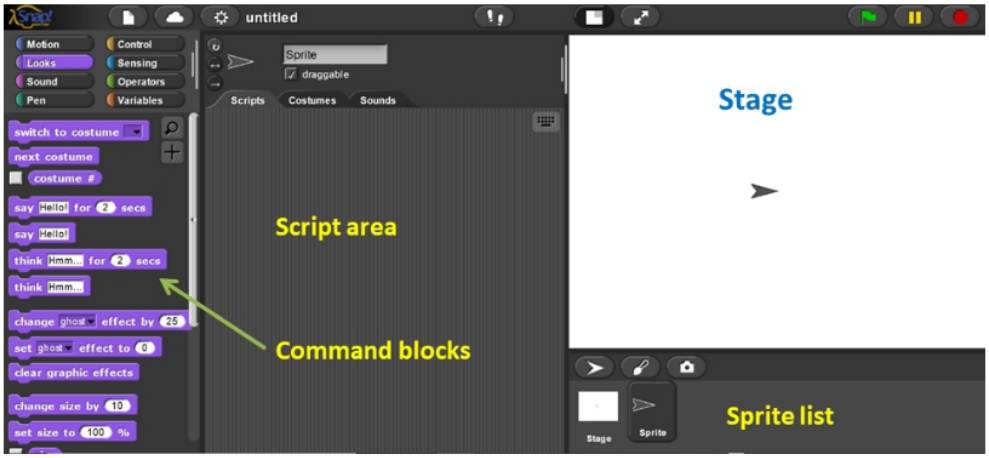

Figure 4: Snap! user interface (Source: Joshi, 2018)

- 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.

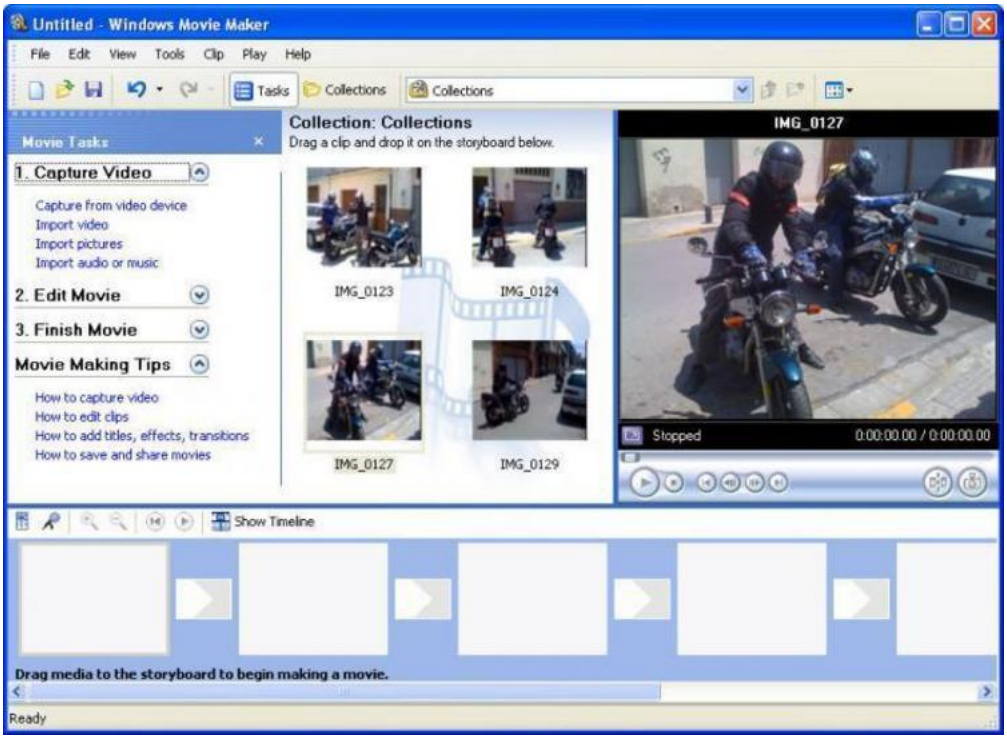

Figure 5: Windows Movie Maker

- 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.

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.

Created: 2022-10-11 Tue 10:20