Learn Computational Thinking using Scratch: Level 1 and 2

Introductory courses for children and young adults to learn computational thinking

Summary:

This document describes the learning objectives, required resources, books, and infrastructure, and the course content provided for conducting these courses.

Audience:

This document is meant for self-learners who wish to learn by themselves, or for teachers who want to teach these courses.

Note: The material described in this document covers both the Level 1 and 2 courses. The actual curriculum and flow of each course is described in separate documents.

Resources:

Books:

- Learn CS Concepts with Scratch (Combined textbook for both levels)
- Practice CS Concepts with Scratch (Additional review questions and practice programs)
- Pen Art in Scratch Programming (Application of programming to Pen art and Turtle geometry)
- Advanced Scratch Programming (10 advanced Scratch projects)

Articles:

- Guidelines for teachers – http://www.abhayjoshi.net/articles/TeachingCS.pdf
- Benefits of learning to program: http://scratched.gse.harvard.edu/resources/handout-learning-through-programming

Infrastructure:

Scratch 2.0 should be used in the offline mode. It runs on Windows, Linux, and Mac, so OS does not matter. If sound h/w (headset) is available, students can exploit the sound features of Scratch, but, this is not a must. The instructor must have a projector to explain concepts and demonstrate sample programs.

Download Scratch 2.0 from: https://scratch.mit.edu/download/scratch2

Style of Delivery:

The courses should be run as a series of projects and practice programs. Concepts should be introduced only as required by the project. Typically 70% of the total time is spent in hands-on work where students write programs. The instructor's work consists of:

- Introducing concepts through examples

- Showing and explaining demo programs
- Helping students debug their programs/design/thinking
- Providing all technical help to students
- Evaluating students' work

Activity at every session:

In every session, the instructor should:

- Allow students to ask questions if they have any
- (Occasionally) Allow students to demo their programs to the class if they wish to
- Ask review questions about concepts learnt so far
- Discuss common mistakes, misunderstanding etc.
- (Occasionally) Debug some code
- (Occasionally) Design an algorithm for a simple problem
- Show algorithm/code of practice program of previous session
- Introduce new concepts: The instruction should be given in a variety of ways: power-point, code demo, class discussion, etc.
- Let students go to programming

Note: As mentioned earlier, the last step should get 70% of the time every session!

Concepts learnt in these courses:

CS Concepts:

- Sequence
- Simple looping (repeat, forever)
- Nested looping
- Synchronization using a timeline
- Synchronization using broadcasting
- Events
- Reset script
- Concurrency
- User interaction using keyboard
- Conditions: YES/NO questions
- Conditionals (IF)
- Conditionals (IF-Else)
- Stopping scripts
- User interaction using mouse pointer
- Click-buttons using broadcasting
- Conditionals (Wait until)
- Variables – numbers
- Variables as sliders
- Keyboard events using polling

- User input using buttons
- Relational operators (=, <, >)
- Variables as remote control
- Built-in variables - properties
- User interaction (ASK)
- String variables
- String operations (join, letter, length of)
- Variables – as counters
- Random operator
- Algorithms
- Conditional looping (repeat until)
- Basics of object oriented programming (OOP)
- Custom procedures (Scratch blocks)
- Custom procedures with inputs
- Abstraction
- Recursion
- List data structure
- Boolean logic operators (AND, OR, NOT)
- Nested conditional statements
- Mapping random numbers to a set of things
- Mouse events using polling
- User input validation
- Variables scope - local/global

Scratch and other concepts:

- Scratch User Interface: layout, create/save projects
- Sprites
- Stage and backdrops
- Paint editor
- Motion commands
- Absolute motion
- Relative motion
- Smooth motion using repeat
- If on edge bounce, rotation styles
- XY geometry
- Costume-based animation
- Multiple backdrops
- 3-D effect using repeat
- Basic sound commands
- Graphic effects
- Sensing touch
- Sound (playing files)
- Sound (creating music)

- Turtle round trip
- Motion - direction and bouncing
- STAMP - creating images
- Creating instances using clones
- Motion – piggybacking another object
- Pen commands
- Basic Turtle geometry
- Drawing a circle
- Scratch Turbo mode
- Mapping random numbers to a set of things

Practice programs:

The teacher is expected to pick programming problems provided in the textbook, and in addition, take some of the problems listed in "Practice CS Concepts with Scratch".

The following Excel file lists all available practice programs collectively in these two books:
http://abhayjoshi.net/spark/scratch/book0s/concepts.xlsx

This file specifies the following for each practice program:

1. Which CS and Scratch concepts the program uses.
2. The "difficulty" level of the program.

This information should be useful in picking suitable problems based on students' prevailing understanding of concepts.

It is sometimes helpful to demonstrate the expected behavior of the programs. The following files list videos that demonstrate this for all the practice programs:
http://abhayjoshi.net/scratch/book0/videos.htm
http://abhayjoshi.net/spark/scratch/book0s/videos.htm

Finally, the following files contain sample Scratch solutions of the practice problems:
http://abhayjoshi.net/scratch/book0/solutions.zip
http://abhayjoshi.net/spark/scratch/book0s/solutions.zip

Programming projects:

In addition to the practice programs, students should do relatively complex projects that combine a set of concepts. The textbook covers the following projects:

- Short story animation
- Interactive animation: fly a helicopter
- Game of Pacman (also known as Maze)

- Game of Bricks
- Game of Falling objects (such as coins or aliens)
- Game of Flappy bird
- Traffic Light using Pen Art

In addition, there are 10 relatively complex projects listed in the book "Advanced Scratch Programming", parts of which can be offered to students as projects.

Finally, students should be allowed to design and create their own Scratch capstone project at the end of each course. This could possibly be a group effort.

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 22 May 2019

Courses created and published by: SPARK Institute and Publications