Learning by creating: A case study in teaching computing to entry-level students

Vijayakumar Nanjappan
Hai-Ning Liang

Dept. of Computer Science and Software Engineering
Xi'an Jiaotong-Liverpool University

Introduction

- Learning programming concepts can be daunting
- Akin to learning a new language
- Involves multiple facets
  - Computational thinking
  - Syntactic thinking
  - Developing tools
- Chinese students have limited exposure to programming prior to entering university
- Reasons
  - Programming only introduced in the 1980s in China
  - Excluded from Gaokao (national standardized university test)

Introduction

- Chinese students face many challenges in learning programming
- Chinese educational system primarily focuses on
  - Exam writing
  - Memorization
  - Repetition
- No practical, hands-on, and experiential learning
- Underperform in programming related subjects
- Choose subjects where programming is kept in a very small role

Introduction

- Introduced a new course in CSE003 to change this trend
- Smoothly transition the students into the realm of computer programming
- Encourage them to learn by constructing non-tangible artifacts
- Scratch Programming
  - Visual programming language
  - Developed by Lifelong Kindergarten Research Group at MIT media lab
- Normally a computer program contains text based instructions
- Written using a programming language (C, C++, Java and etc.)

Text-based Computer Program

- To print simple words on the computer screen

Text-based Instructions

```
#include<stdio.h>

int main()

   {
      printf("Hello World\n");

      return 0;

   }
```

Instructions Set

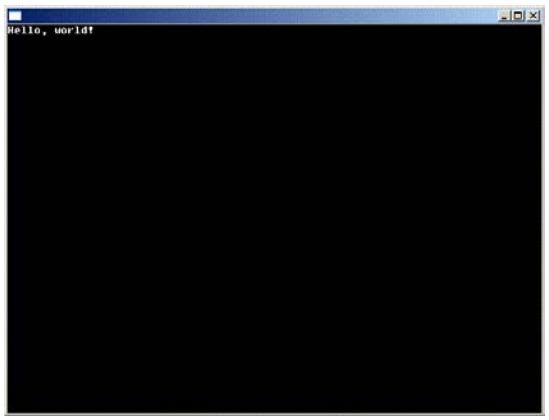


Figure 1: C program to print "Hello World" on the screen

Why Scratch?

- Avoids of the use of text based environments
- Skips learners to acquire syntax of programming language
- Graphical blocks represents different constructs
  (statements, loops, conditions, and variables)
- Allows students to concentrate on the principles of programming
- Observe the behaviour of a piece of code in their creations
- Supports novices in the acquisition of programming and computational thinking skills
- Allows learners to create Interactive stories and games

Scratch – Visual Programming

- Graphical blocks are used instead of text-based instructions

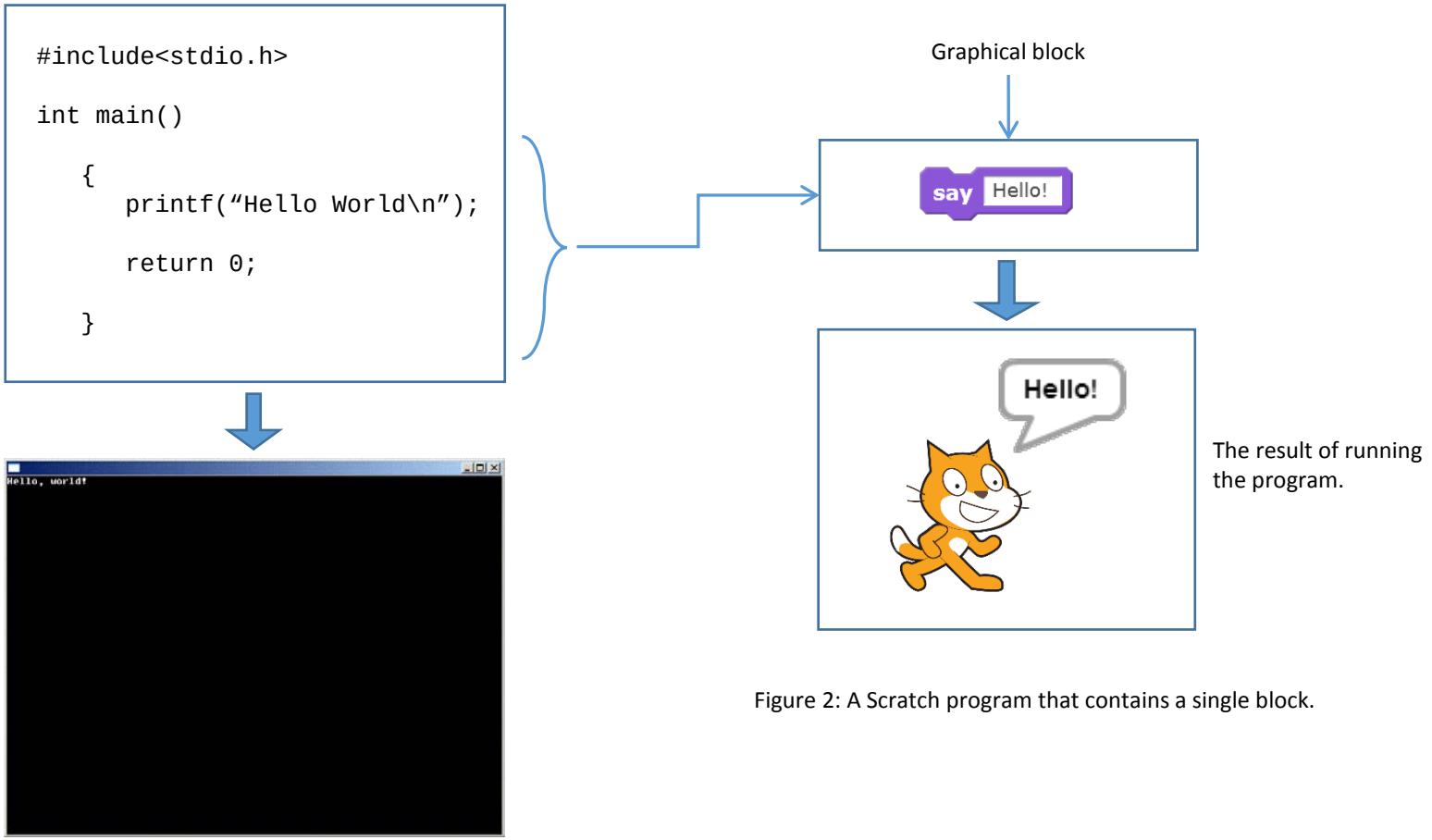


Figure 2: A Scratch program that contains a single block.

Scratch – Visual Programming

- The Scratch environment

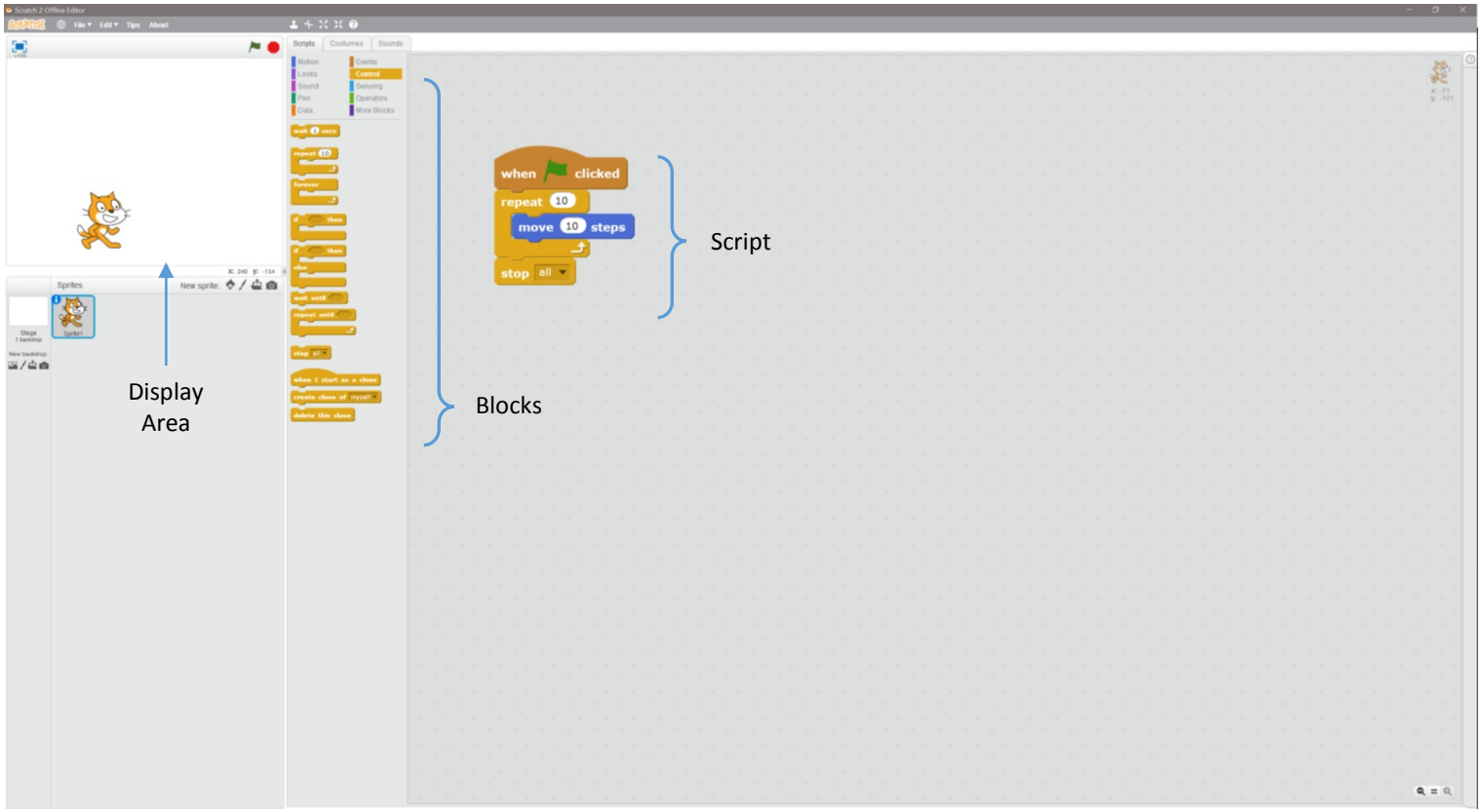


Figure 3. The Scratch environment (version 2 and above). Scratch is a block-based programming environment. Blocks (color-coded objects in the middle component) are dragged onto the right component to create scripts. In the script above, for example, when there is a mouse click action on the sprite object (orange cat) will move 10 steps.

Context of the case study

- CSE003 – Fundamentals of Computer Programming
  - Foundational course – 2.5 module
  - Runs 7 weeks at second half of the first semester
  - Open to all but compulsory to Engineering and Computer Science students
- Fall of 2015 – 450 students
  - STEM and Non-STEM students
- Designed to prepare students to cope with CS1 programming course
  - Computer science students learn Java
  - Engineering students learn C and C++

Teaching programming to entry-level students

- Scratch component
  - Takes place in first 2 weeks
  - 2 hours lectures
  - 2 hours lab

Week	Day	Activity	Learning Outcomes
1	Tuesday	Lecture 1	Introduction to Scratch
	Thursday	Lab 1 – Tutorial 1	Introduction to Scratch (Practical exercises)
	Friday	Assignment Out	
2	Tuesday	Lecture 2	Discussion about the lab exercises
	Thursday	Lab 2 or Tutorial 2	Practice exercises (or work on assignment)
3	Tuesday	Submission	Scratch program and a short reflection report needed to be submitted

Table 1. Overview of first 2 weeks of learning activities.

Assignment

- Interactive program – animated storytelling/storyboard of their choice
  - Student as the main character
  - Length of the play (between 90 to 120 seconds)
- Sprites (objects used in their programme)
  - Students own photo – minimum three costumes (different poses)
- Backdrops and sounds
  - Minimum of 5 different background images
  - Different music to begin and end the story

Compulsory Requirements	User-created	Scratch Library	Other Sources
Sprites	3	Unlimited	Unlimited
Costumes	3	Unlimited	Unlimited
Backdrops	Minimum 5		
Sounds	Minimum 5		

Table 2. Assignment's compulsory requirements when using the sprites, costumes, backdrops and sounds.

Assignment

- User-interface controls
  - To begin and stop the program
  - Pause/resume program at any point

Device	Key and Icons	Function
Keyboard	Spacebar	Pause and Resume
	S	Play
	X	Stop
	R	Re-play
Mouse	GreenFlag	Play
	StopButton	Stop

Table 3. Descriptions of the required user controls in the Scratch programs.

- Script
  - The code composed of programming blocks
  - Must contain at least one block from each the nine-block palette
  - 11 types of blocks in Scratch (only 10 blocks are visible)

Figure 4. The different types of programming blocks in the palettes in the Scratch programming environment.

Data collection instruments

- Surveys (Questionnaires)
  - First Survey
    - Demographic information
    - Prior programming experience
    - Interested program of study in the future
  - Second Survey
    - Students' experience using Scratch
    - Effectiveness and usefulness in terms learning programming concepts
    - Time spent learning Scratch
  - Students were informed:
    - Surveys were not compulsory
    - Answers would not affect their grades
- Submission
  - Individual Scratch projects
  - Short reflection report
- University records
  - Student's final programme choice

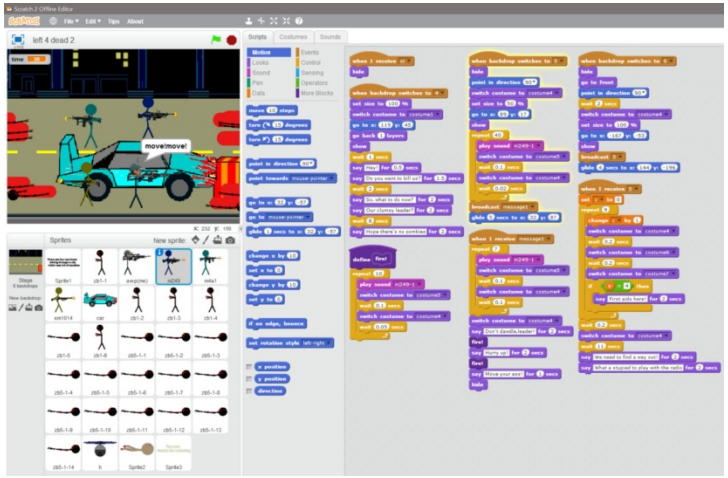


Figure 5. A sample Scratch program completed by a student. The final output is in the top left component. The bottom component shows all the sprites and backdrops. The right window shows the scripts of the highlighted sprite.

Results

- Enrolled students
  - 450 students
  - 356 students used in the data collection
  - Excluded: incomplete surveys – non-submissions

Gender	Frequency (N)	(%)
Male	265	74.4
Female	91	25.6

Table 4. Frequency distribution of the male and female students.

Programme	Frequency (N)	%
STEM	320	89.1
Non-STEM	36	10.1

Table 5. Frequency distribution of STEM and Non-STEM students.

Programme	Male		Female	
	N	%	N	%
STEM	246	92.8	74	81.3
Non-STEM	19	7.2	17	18.7

Table 6. Frequency distribution of the male and female students in STEM and Non-STEM programs.

Results

- Prior programming experience of the students

Programming knowledge	Frequency (N)	(%)
No	285	80.1
Yes	71	19.9

Table 7. Frequency distribution of previous programming knowledge of CSE003 students.

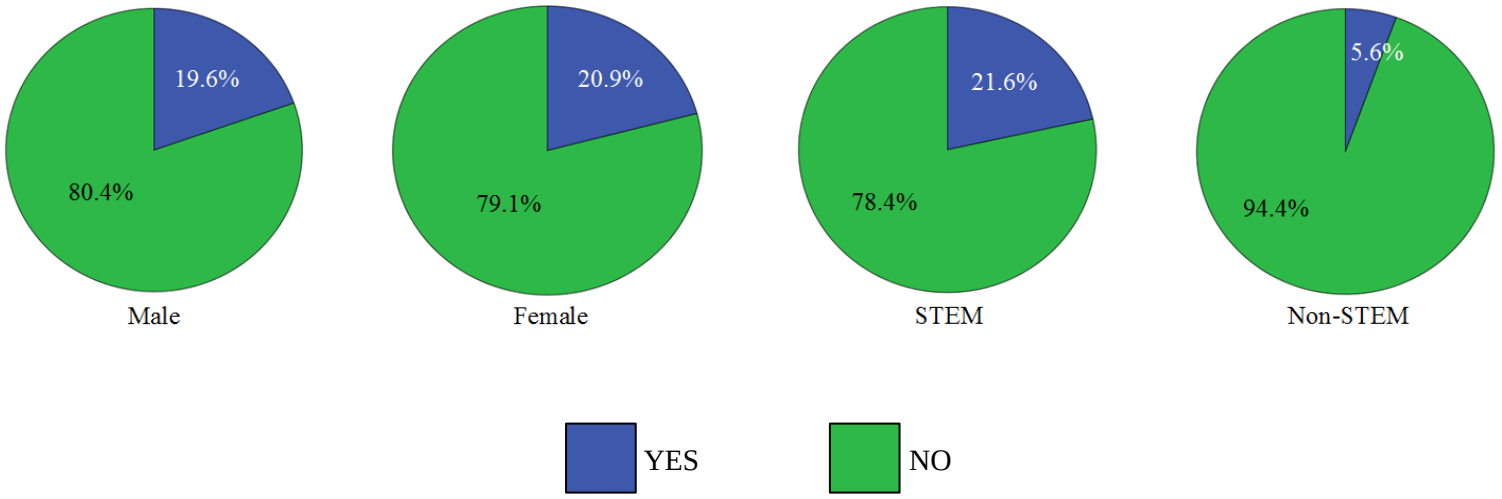


Figure 6. Students response to the question on "Have you learned programming before?"

Results

- Types of Scratch creations

Themes	Total	
	N	%
Personal Interests	111	31
Daily Experiences	97	27
Hollywood Inspired	72	20
Game-based	57	16
Socio-political concerns	19	0.05

Table 8. Types of creations based themes (percentage rounded to the nearest number, except for the bottom-right one).

Themes	STEM		Non-STEM	
	N	%	N	%
Personal Interests	94	**29**	17	**47**
Daily Experiences	88	28	9	25
Hollywood Inspired	68	**21**	4	**11**
Game-based	52	16	5	14
Socio-political concerns	18	6	1	3

Table 8. Types of creations based on STEM/Non-STEM students

Themes	Male		Female	
	N	%	N	%
Personal Interests	74	**28**	37	**41**
Daily Experiences	69	26	28	31
Hollywood Inspired	62	**23**	10	**11**
Game-based	45	17	12	13
Socio-political concerns	15	6	4	4

Table 9. Types of creations based on gender

Sample Projects

- Sample Scratch programs developed by the students:

a) social-political concern

b) personal interests

Figure 7. Sample Scratch programs developed by students (a-b)

Sample Projects

- Sample Scratch programs developed by the students:

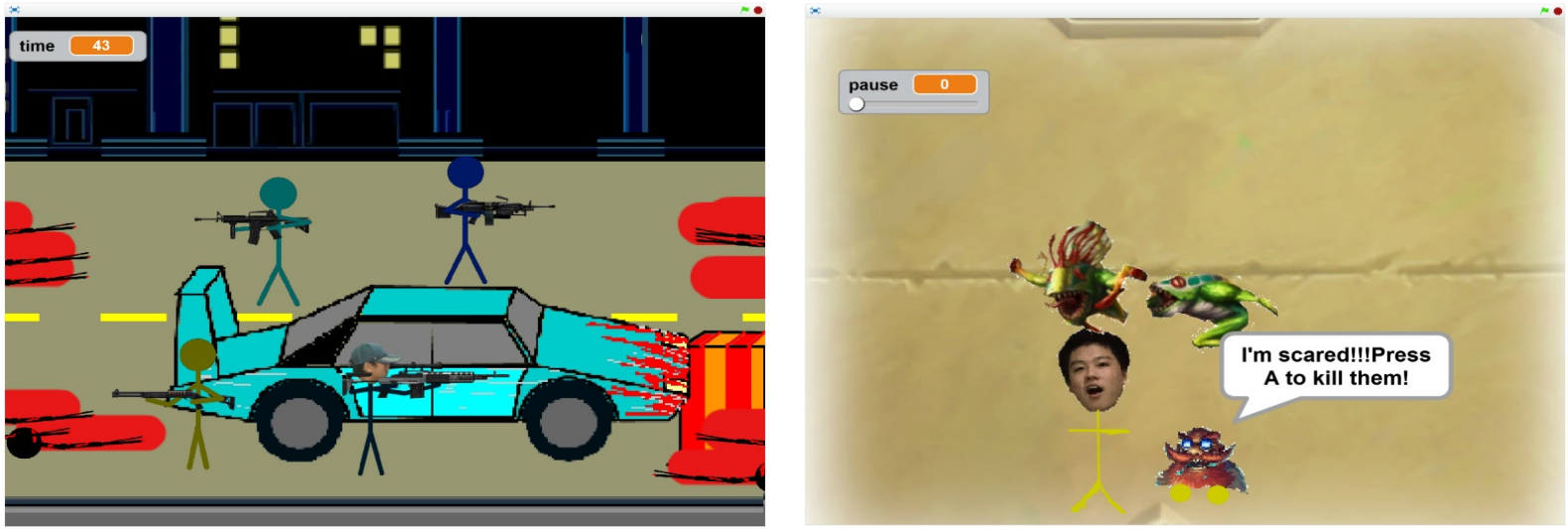
c) Hollywood-inspired

d) Daily life

Figure 7. Sample Scratch programs developed by students (c-d)

Sample Projects

- Sample Scratch programs developed by the students:



e) Game-based projects

Figure 7. Sample Scratch programs developed by students (e)

Factors influencing students marks

- Female students achieved higher grades
- STEM students also received better grades on average
- Game-based projects were best
- Socio-political concerns scored lowest

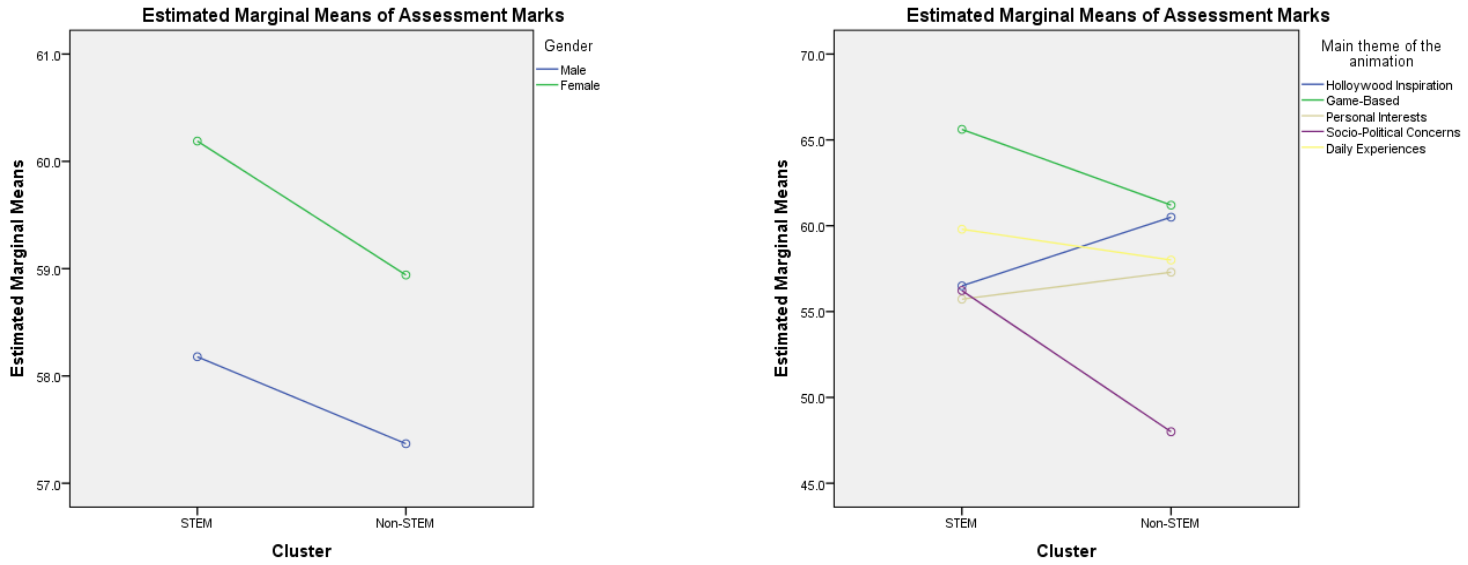


Figure 8. (Left) Mean differences between the marks of STEM/Non-STEM students; (Right) Mean differences between STEM/Non-STEM students classified based on types of Scratch programs.

Factors influencing students marks

- Male students scored well in game-based projects
- Female students scored better in personal interests types of creations
- Not much different in other types of creations

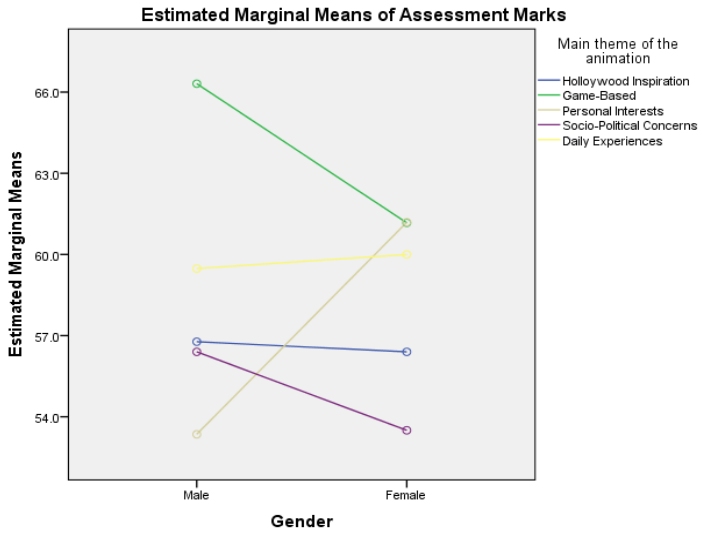


Figure 9. Mean difference between the marks of female and male students based on the type of Scratch creations.

Factors influencing students marks

- STEM vs. Non-STEM students
  - STEM students with prior programming experience performed better
- Male vs. Female students
  - Female students with prior programming experience performed better

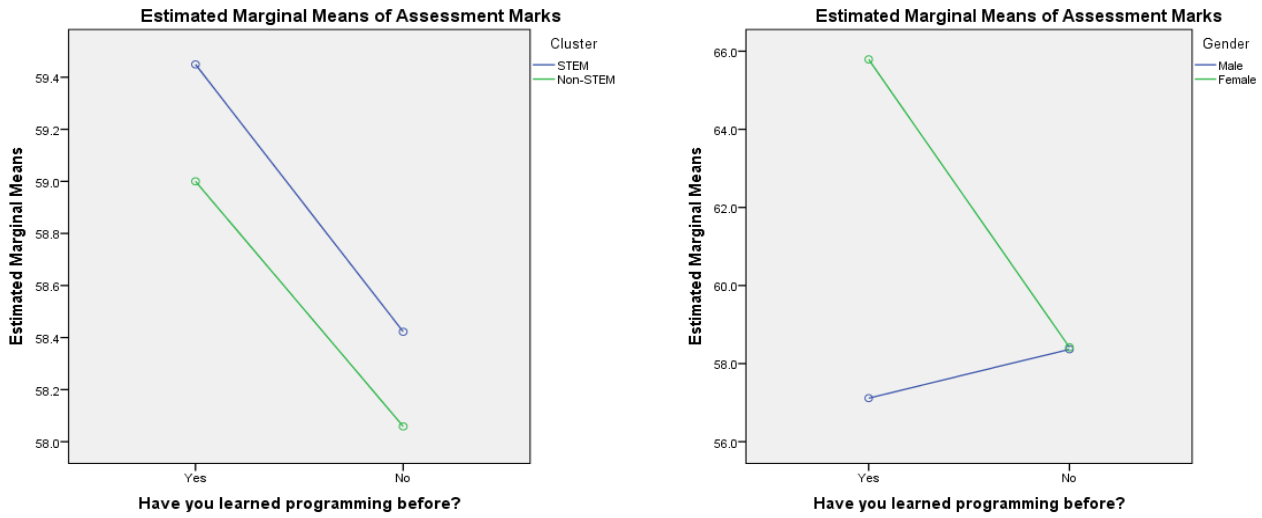


Figure 10. Mean difference in marks based on prior programming experience; (LEFT) for STEM/Non-STEM students; and (RIGHT) Male/Female students.

Usefulness of Scratch

- Students' perception of the usefulness of Scratch on helping them to learn

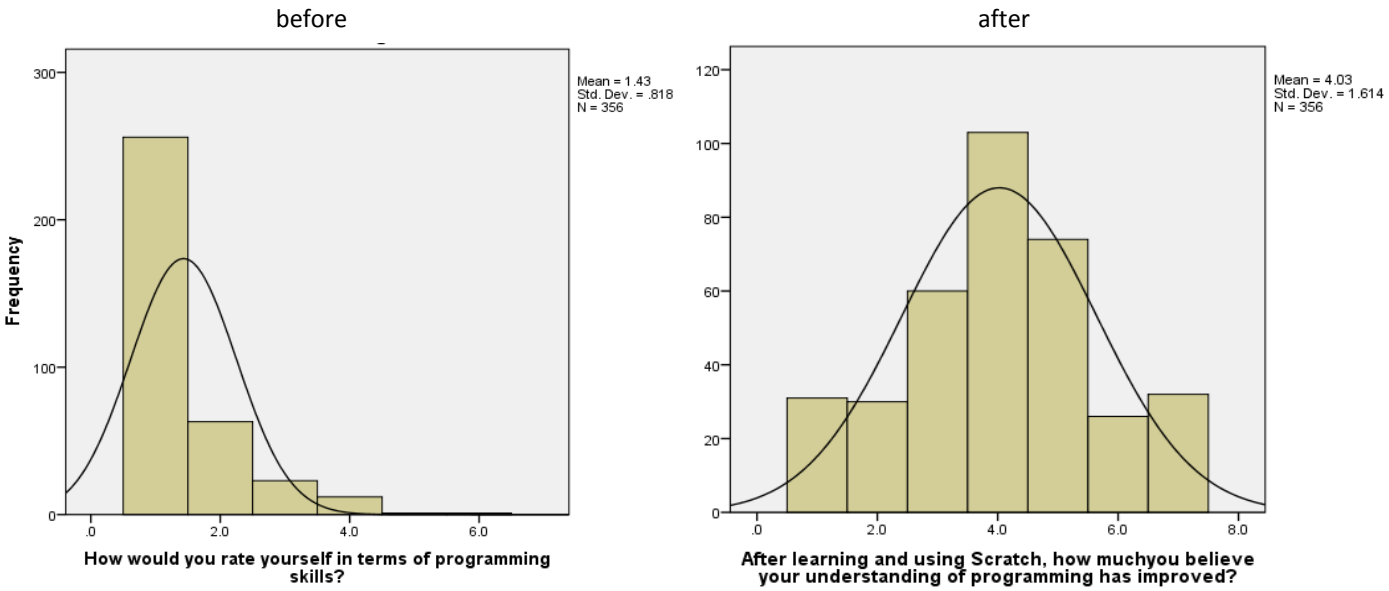


Figure 11. Frequency distribution of the students' response before and after using Scratch to complete their assignments in terms of their knowledge of programming

Findings

- Students regardless of gender and program of study benefited from Scratch
- They found it interesting and useful to build things to support their learning of programming
- Female students, Non-STEM students and students without prior background in programming can do equally well (and sometimes even better)
- Most students prefer themes related to their personal interests and daily experiences