Computational Thinking and STEM Education

Prof. KONG Siu Cheung
The Education University of Hong Kong
Content

- Computational Thinking (CT)

- Three ways of developing CT
  - CT & Mobile Computing
  - CT & Programmable Robotics
  - CT & Modeling/ Simulation

- Role of CT in STEM Education

- Curriculum Design for CT Development in STEM Education
CT Framework

Computational Thinking Knowledge
Fundamental programming knowledge

Computational Thinking Practices
Process and outcomes of practices for logical thinking and problem solving skills development

Computational Thinking Perspectives
Evolving understandings of themselves, their relationships to others, and the technological world around them

Knowledge in programming language context
1. Sequence
2. Events
3. Repetition
4. Conditional
5. Parallelism
6. Naming (Variables)
7. Operators
8. Manipulation of data and elementary data structure

Outcome of programming experiences with fundamental programming knowledge
1. Programming fundamentals, such as sequencing, branching and looping, and interacting among objects
2. Algorithms in programming, such as sorting and searching

Processes of practices
1. Recognize computational problems, find workable means
2. Understand the order of precedence in solving the problem
3. Decomposing tasks into subtasks
4. Gather & arrange relevant information for problem solving
5. Use the programming environment/language to code accurately and with clear documentation
6. Recognize logical sequence, branching & looping relationships
7. Put the program for testing & debugging

Outcomes of practices with skills for logical thinking and problem solving development
1. Reusing and remixing (building on other’s work)
2. Being incremental and iterative
   (a) Iterative, recursive, and parallel thinking
3. Abstracting and modularizing
   (a) Abstraction and pattern generalization
   (b) Structured problem decomposition
4. Testing and debugging
5. Algorithmic thinking
   (a) Systematic processing of information
   (b) Symbol systems and representations
   (c) Algorithmic notions of flow of control
   (d) Conditional logic

Experiences in coding
1. Expressing
2. Questioning
3. Connecting

Outcomes of experiences with following key perspectives
1. Personal interests in coding
2. Willingness to engage further in coding
3. Positive perception of coding
4. Confidence in coding
5. Computational identity
6. Digital empowerment
Constructionism as the Learning Theory Underlying CT Development

- Constructionism refers to the cognitive development and learning involved in building knowledge structures through interacting with natural and designed environment (Papert, 1991)

- Students take part in construction of knowledge by building a meaningful product (e.g., a program) for others or themselves (Kafai & Resnick, 1996; Lye & Koh, 2014)

- Constructionism emphasizes hands-on practice for conceptual development and learning by making
Examples of CT Development

**Finland**
Coding was defined in the core curricula for 2016, with more general abilities of CT added.

**England**
A new computing curriculum was implemented in 2014.

**USA**
A Computer Science curriculum for K–12 with programming content was proposed in 2003. A new K-12 CS framework was announced in 2016.

**India**
New CS curriculum was proposed in 2013. Textbooks with CT curriculum were used by 300 schools from K1-8 in 2016.

**Singapore**
CT-based curriculum for K-12 was proposed in 2014.

**Estonia**
An initiative to bring programming to K-12 began in 2012.

**China**
Researchers began to discuss the necessity of including CT in K-12 curriculum from 2013.

**South Korea**
Coding will be made a required course in 2017. High schools will adopt coding as elective from 2018.

**Japan**
Computer programming will be compulsory in 2020 for primary schools, 2021 for middle schools and 2022 for high schools.

**Hong Kong**
A 3-year programme developing CT via coding education was held since 2016 among senior primary students in 32 schools.
CT and STEM Education

3 possible ways of developing CT

- CT Skills & mobile computing
- CT Skills & programmable robotics
- CT Skills & simulation
CT Development through Coding

- CT is originally based on Seymour Papert’s work (1980) referring to the expression of powerful ideas through computational representations (Papert, 1996)

- It is the thought processes involved in formulating a problem and expressing its solution(s) so that a computer-human or machine can implement (Wing, 2014)

- It involves designing systems, understanding human behavior and solving problems by applying the concepts fundamental to computer science (Wing, 2006)

- Coding is an effective means for developing computational thinking

- Block-based programming environments (e.g., Scratch, App Inventor) allow even young learners to learn how to code or write programs
CT & Mobile computing

- Examples of platform: App Inventor, Swift playground
- Students to develop mobile apps by means of coding

1. Input: collect data (detect events) from:
   - sensors of mobile devices (e.g. Accelerometer sensor, location sensor)
   - user input (e.g. Submitting answers, pressing certain keys)

2. Process: automated computation based on learners’ programs and data collected

3. Output: application to respond to the input
CT & Mobile Computing

- A framework of CT (Brennan & Resnick, 2012):

<table>
<thead>
<tr>
<th>CT concepts</th>
<th>CT practices</th>
<th>CT perspectives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequences</td>
<td>Being incremental &amp; iterative</td>
<td>Expressing</td>
</tr>
<tr>
<td>Events</td>
<td>Testing &amp; debugging</td>
<td>Connecting</td>
</tr>
<tr>
<td>Loops</td>
<td>Reusing &amp; remixing</td>
<td>Questioning</td>
</tr>
<tr>
<td>Parallelism</td>
<td>Abstracting &amp; modularizing</td>
<td></td>
</tr>
<tr>
<td>Conditionals</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CT & Programmable Robotics

- Examples: Lego Mindstorms, mBot, iRobot, Botball, FIRST, Dash & Dot

- Students to build a robot with sensors and interact with it with coding

1. Input: collect data from:
   - different environmental sensors (e.g. temperature, ultra-sonic, rotation sensors);
   - input modules (e.g. switches, buttons)

2. Process: automated computation based on learners’ programs and data gathered from smart mainboard (e.g. arduino) via wireless connection (e.g. bluetooth/ wifi)

3. Output: robot reacts accordingly via output modules (e.g. motor, wheels, buzzers)
A learning progression in educational robotics (Sullivan & Heffernan, 2016):

**Sequencing**
- Ability to put items into specific order
- Foundation of programming

**Causal Inference**
- Based on comparison between the expected movement and immediate feedback from the programmed object
- Set up hypothesis on why the expected movement was not observed

**Conditional Reasoning**
- Ability to abstract a rule for the behavior
- Use of environmental sensors to work with robotic devices
- Eg. sensory-reason-action-loop

**Systems Thinking**
- Understanding of interacting functions of related parts of robotics device
- Interaction between microcomputer, actuators (eg. motor, bulbs) & sensors
CT & Modeling/Simulation

- Example of platform: NetLogo

- Example: Modeling the ecosystem of food chain & waste cycle
  (Sengupta, 2011; Tan & Biswas, 2007)

  ◦ Learning task: being able to sustain a number of fish over a period of time
CT & Modeling/Simulation

1. Input: parameters and commands set by learners

2. Process: rule-based simulation of natural phenomenon based on input parameters

3. Output: generation of behaviors that support or contradict learners’ hypothesis
CT Skills & Modeling/ Simulation

Categories of CT practices in STEM (Weintrop et al., 2016):

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Collecting Data</td>
<td>Using CT models to understand a concept</td>
<td>Preparing problems for CT solutions</td>
<td>Integrating a complex system as a whole</td>
</tr>
<tr>
<td>Creating Data</td>
<td>Constructing CT models</td>
<td>Programming</td>
<td>Understanding the relationships within a system</td>
</tr>
<tr>
<td>Manipulating Data</td>
<td>Assessing CT models</td>
<td>Choosing effective CT tools</td>
<td>Thinking in levels</td>
</tr>
<tr>
<td>Analyzing Data</td>
<td>Designing CT models</td>
<td>Developing monocular CT solutions</td>
<td>Communicating information about a system</td>
</tr>
<tr>
<td>Visualizing Data</td>
<td>Using CT models to find and test solutions</td>
<td>Assessing different approaches/solutions to a problem</td>
<td>Defining systems and managing complexity</td>
</tr>
</tbody>
</table>

- Programming
- Choosing effective CT tools
- Developing monocular CT solutions
- Assessing different approaches/solutions to a problem
- Creating CT abstractions
- Troubleshooting and debugging
# CT Skills & Modeling/ Simulation

Data Practices (Weintrop et al., 2016):

<table>
<thead>
<tr>
<th>Data Practices</th>
<th>Descriptions</th>
<th>Egs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collecting Data</td>
<td>• Propose systemic data collection protocols</td>
<td>• Decide on what and how the data should be collected</td>
</tr>
<tr>
<td></td>
<td>• Articulate how the protocols can be automated with CT tools</td>
<td>• Propose to collect rainfall on a daily/ weekly/ annual basis</td>
</tr>
<tr>
<td>Creating Data</td>
<td>• Define CT procedures</td>
<td>• Set/ add conditions for the simulation, eg. if no. of wolf&gt;0</td>
</tr>
<tr>
<td></td>
<td>• Run simulations</td>
<td>and no. of sheep =0, wolf will die</td>
</tr>
<tr>
<td>Manipulating Data</td>
<td>• Sorting, filtering, cleaning, normalizing, joining separate datasets</td>
<td>• Sort data from ascending order</td>
</tr>
<tr>
<td></td>
<td>• Manipulate datasets with CT tools</td>
<td>• Find out the highest/ lowest rainfall and present separately for</td>
</tr>
<tr>
<td></td>
<td>• Reshaping dataset to be a more useful configuration for further investigation</td>
<td>easier referencing</td>
</tr>
<tr>
<td>Analyzing Data</td>
<td>• Look for patterns and anomalies</td>
<td>• Generalize rules from data</td>
</tr>
<tr>
<td></td>
<td>• Define rules to categorize data</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Identifying trends and correlations</td>
<td></td>
</tr>
<tr>
<td>Visualizing Data</td>
<td>• Use conventional visualizations or interactive displays to present data</td>
<td>• Graphs and charts</td>
</tr>
</tbody>
</table>
## Comparing the 3 ways of developing CT

<table>
<thead>
<tr>
<th>Platform/example</th>
<th>Mobile computing / app or game design</th>
<th>Programmable/ educational robotics</th>
<th>Modeling/ simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT framework</td>
<td>App Inventor, Alice, iGame, Scratch, Logo, Game Maker</td>
<td>Lego Mindstorms, mBot, iRobot, Botball, FIRST</td>
<td>NetLogo</td>
</tr>
<tr>
<td>Renowned Framework</td>
<td>CT framework (Brennan &amp; Resnick, 2012)</td>
<td>Computational learning progression in robotics (Sullivan &amp; Heffernan, 2016)</td>
<td>CTSiM framework (Sengupta et al., 2016a)</td>
</tr>
</tbody>
</table>
| Role of learner  | As a designer: - Build one’s own app  
As an inventor: - Create something new which can benefit the society | As a maker/ designer: - Build and program a robot  
As an inventor: - Create something new which can benefit the society | As a user/ designer: - Manipulate parameters on the platform |
| Role of teacher  | As a facilitator: - Equip learners with skills in mobile computing  
- Provide sample app as scaffold of learning  
- Provide technical support | As a facilitator: - Equip learners with skills in robotics  
- Create challenges/ setting conditions for learners  
- Provide technical support | As a facilitator: - Model the use of platform  
- Guide students through the process of scientific inquiry |
Role of CT in STEM Education

- CT skills help to deepen the understanding of STEM content areas for students (Guzdial, 1994; National Research Council, 2011; Repenning et al., 2010; Sengupta et al., 2013; Wilensky & Reisman, 2006)

- CT skills development help students to reflect productively on STEM activities which in turn help to cultivate problem solving and logical thinking ability of students:
  - E.g., when controlling a robot to run a maze, students engage in a problem-solving process, which requires skills such as decomposing a problem into sub-problems, and testing and debugging.

- Programming and computational modeling are effective vehicles for learning STEM concepts, e.g., friction, environmental conservation (Guzdial, 1995; Sherin, 2001; Hambrusch et al., 2009)
Curriculum Design for CT Development in STEM Education

Based on a Conceptual Framework modified from Sullivan & Heffernan (2016):

- General Studies
- Mathematics
- Programming

Knowledge in STEM

Application → Consolidation

Sequencing → Causal Inference → Conditional Reasoning → Systems Thinking

Robotics learning progression

through

Problem solving activities in robotics environment
Curriculum Design for STEM Education

- 2 cycles of robotics learning progression:

**Cycle 1:**
- Unit 5
- Unit 1
- Unit 2
- Cycle 1: Robotic car
- Unit 3
- Unit 4

**Cycle 2:**
- Unit 6
- Unit 10
- Unit 7
- Cycle 2: Electronic modules
- Unit 8
- Unit 9
## Curriculum Design for STEM Education

Modified from Brennan & Resnick (2012); Sullivan & Heffernan (2016):

<table>
<thead>
<tr>
<th>CT knowledge</th>
<th>Sequences, Repetition, Naming (Variable)</th>
<th>Conditional, Operators</th>
<th>Manipulation of data and elementary data structure, Events</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cycle 1: unit 1</td>
<td>Cycle 1: unit 2-3</td>
<td>Cycle 1: unit 4-5</td>
</tr>
<tr>
<td></td>
<td>Cycle 2: unit 6</td>
<td>Cycle 2: unit 7</td>
<td>Cycle 2: unit 8-10</td>
</tr>
<tr>
<td>CT practices</td>
<td>Testing &amp; debugging, Being incremental &amp;</td>
<td>Algorithmic thinking</td>
<td>Abstracting &amp; modularizing</td>
</tr>
<tr>
<td></td>
<td>iterative, Reusing and remixing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cycle 1: unit 1</td>
<td>Cycle 1: unit 2</td>
<td>Cycle 1: 3-5</td>
</tr>
<tr>
<td></td>
<td>Cycle 2: unit 6</td>
<td>Cycle 2: unit 7-8</td>
<td>Cycle 2: 9-10</td>
</tr>
<tr>
<td>Robotics Learning</td>
<td>Sequencing</td>
<td>Causal inference</td>
<td>Conditional reasoning</td>
</tr>
<tr>
<td>Progression</td>
<td>Cycle 1: unit 1</td>
<td>Cycle 1: unit 1-2</td>
<td>Systems thinking</td>
</tr>
<tr>
<td></td>
<td>Cycle 2: unit 6</td>
<td>Cycle 2: unit 6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cycle 1: unit 3</td>
<td>Cycle 1: unit 4-5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cycle 2: unit 7</td>
<td>Cycle 2: unit 8-10</td>
<td></td>
</tr>
</tbody>
</table>

*Progression of learning*
Sample Task 1: My Autopilot Car

Auto-piloting the robot moving along a square

Requirement: develop a program to make the robot walk through a square.
Sample Task 1: My Autopilot Car

Possible solutions:

```
Program
repeat 4
run forward at speed 95
wait 3.1 secs
run forward at speed 0
wait 1 secs
run forward at speed 85
wait 0.7 secs
run forward at speed 0
```

or

```
Program
run forward at speed 95
wait 3.1 secs
run forward at speed 0
wait 1 secs
run forward at speed 85
wait 0.7 secs
run forward at speed 0
run forward at speed 95
wait 3.1 secs
run forward at speed 0
wait 1 secs
run forward at speed 85
wait 0.7 secs
run forward at speed 0
run forward at speed 95
wait 3.1 secs
run forward at speed 0
wait 1 secs
run forward at speed 85
wait 0.7 secs
run forward at speed 0
```
### Sample Task 1: My Autopilot Car

**Knowledge and skills involved:**

<table>
<thead>
<tr>
<th>Course Design based on CT Progression in Robotics Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CT Knowledge</strong></td>
</tr>
<tr>
<td>Sequences, Repetition, Naming (Variable)</td>
</tr>
<tr>
<td><strong>CT Practices</strong></td>
</tr>
<tr>
<td>Testing &amp; debugging, Being incremental &amp; iterative, Reusing &amp; remixing</td>
</tr>
<tr>
<td><strong>Robotics Learning Progression</strong></td>
</tr>
<tr>
<td>Sequencing</td>
</tr>
<tr>
<td><strong>STEM areas</strong></td>
</tr>
<tr>
<td>S: Manipulate &amp; operate electronic toys domain as in general studies</td>
</tr>
<tr>
<td>T: programming using scratch</td>
</tr>
<tr>
<td>E: iterative modification of the program design</td>
</tr>
<tr>
<td>M: measurement of speed and distance; use table to record statistics</td>
</tr>
</tbody>
</table>
Sample Task 2: Car Race

Requirement: develop a program to make the car run on the track. It will also slow down automatically when there is another car in front of it.
Sample Task 2: Car Race

- If the returned value is 1 (black color is detected on the left hand side)
  - it will turn right
Sample Task 2: Car Race

- If the returned value is 2 (black color is detected on the right hand side)
  - it will turn left
Sample Task 2: Car Race

- If the returned value is 3 (only white color is detected)
  - it will keep moving forward
- If there is another car in front of it
  - it will slow down
  - it will stop when too close
Sample Task 2: Car Race

- If returned value is 0 (only black color is detected)
- It will stop
Sample Task 3: Searching for Light

Requirement: develop a program to make the robot avoid obstacles in a dark environment. It will circle around when the light is detected and will stop when the light intensity is high enough.
Sample Task 3: Searching for light

- the light intensity value is lower than 100
- the code inside SensorControlClock will be executed
- avoid obstacles which are 10 cm ahead
Sample Task 3: Searching for light

- Light intensity is higher than 100
  - code inside LightClock will be executed
  - robot will stop when the light intensity value is higher than 200
- Light intensity between 100 to 200
  - Robot circling around.
- Light intensity is lower than 100
  - LightClock will be disabled
  - SensorControlClock will be enabled
  - the robot will focus on avoiding obstacles again
To install a light bulb...
Conclusion: What is the Core of CT?

- Wing’s definition of CT (2006): It involves “the thought processes involved in formulating problems and their solutions”.

1. Problem formulation
   - The ability to discover problems in the real life or raise questions in the digital world

2. Problem solving
   - The ability to systematically process information and design the algorithm to solve problems
Conclusion: How Can We Develop CT?

1. Develop **CT skills** in order to facilitate problem formulation and problem solving.

2. Accumulate **experiences** in programming mobile apps/STEM/simulation on testing scientific hypothesis.

3. Consolidate these precious experiences as a **thinking skill** without delving deep into the technical knowledge.

4. Integrate the CT skills into one’s **expertise**.
Conclusion

- STEM education emphasizes the integration of Science, Technology, Engineering, and Mathematics into a cohesive learning paradigm based on real-world applications;

- CT plays a significant role in STEM education;

- Students can build digital physical devices with sensors, and use coding to control them to react with input data from sensors;

- In completing tasks such as controlling digital physical devices to run a maze, students engage in a problem-solving process through coding, which requires skills such as decomposing a problem into sub-problems, and testing and debugging;

- Both CT and STEM education stress learning by making. Students are required to build and create artifacts in these contexts, which is in line with the constructionist view of learning and teaching.
Computational Thinking and STEM Education

Thank You!

Prof. KONG Siu Cheung
The Education University of Hong Kong