Computational Thinking

Joe McCade and Adam Kennedy

Why Computational Thinking?

In 2015, national educational policies in the United States specifically included references to CS as part of a well-rounded science, technology, engineering, and mathematics (STEM) education, and the term "computational thinking" was added to the Next Generation Science Standards as a core scientific practice that could be applied across many science content areas (Weintrop et al. 2015)

Why Computational Thinking?

- After a two-year process, Carnegie Mellon Robotics Academy and the University of Pittsburgh's Learning Research and Development Center have had a research study published by the prestigious Association for Computing Machinery, entitled *Developing Computational Thinking through a Virtual Robotics Programming Curriculum*.
- ITEEA has endorsed the engineering levels of Robomatter's Robotics Curriculum Continuum through its Engineering byDesign™ (EbD) standard, The Engineering Endorsement Matrix.

Is Computational Thinking learning about computers?

- No – it is an approach to solving problems
- It combines mathematics, logic and algorithms, and teaches you a new way to think about the world. (Crow. 2014)

Who proposed Computational Thinking?

- CT was originally developed by a computer scientist – Jeannette Wing
- CT can be defined as “an approach to solving problems in a way that can be solved by a computer. .” (Barr and Stephenson 2011)
- but the computer can be a human

What can we compare CT to?

- *Scientists* seek to show that theories fit the data;
- *mathematicians* seek to show logical proof of abstract connections;
- *engineers* seek to demonstrate that designs work
- (AAAS – Science for All Americans)

Computer Scientist are equally evangelical about their approach

Computational thinking teaches you how to tackle large problems by breaking them down into a sequence of smaller, more manageable problems. It allows you to tackle complex problems in efficient ways that operate at huge scale. It involves creating models of the real world with a suitable level of abstraction, and focus on the most pertinent aspects. It helps you go from specific solutions to general ones.

What have we done at MU?

- A few years ago we split our EPT class dividing Transportation from Energy and Power. We linked Automation with Transportation to create a new course which replaced Power Conversion and Control.
- For the Automation portion of the course we adopted the RobotMatter Curriculum using both the Virtual world and VEX robots to match the popular platform for robotics competitions.

This semester Adam’s project is to covert the programming challenges into engineering/design challenges that include CT

We adopted a four part model for CT:

1. Decomposition: Breaking down data, processes, or problems into smaller, manageable parts
2. Pattern Recognition: Observing patterns, trends, and regularities in data
3. Abstraction: Identifying the general principles that generate these patterns
4. Algorithm Design: Developing the step by step instructions for solving this and similar problem

We map those against nine design strategies Design Strategies (Chrismond & Adams, 2012)

1. Understand the Challenge
2. Build Knowledge
3. Generate Ideas
4. Represent Ideas
5. Weigh Options & Make Decisions
6. Conduct Experiments
7. Troubleshoot
8. Revise/Iterate
9. Reflect on Process

Design Strategies	Computational Thinking	Integration (six strategies)
Understand the Challenge	Decomposition	**Ask**: investigate the problem breaking it down including specifications and component parts
Build Knowledge	Pattern recognition	**Research**: find information related to the challenge including trends and commonalities
Represent Ideas	Pattern generalization and abstraction	**Plan**: brainstorm potential solutions
Weight Options and make decisions	Algorithm design	**Create**: select a solution Additional steps include: test, evaluate