Pedagogy, Coding and Computational Thinking

Short presentation for OCG-Cepis

Mini-conference Coding and Computational Thinking (CT) at Schools

Program:

1. Personal introduction
2. Pedagogy and teaching in general
3. Pedagogy and coding
4. Concepts of Coding and Computational Thinking (CT)
5. CT pedagogy and how to practice e.g. language
6. Questions and discussion

Personal Introduction

- Founder LearningFocus
- Cepis-SIN/CIS member
- NHL-University for Applied Science (Leeuwarden, Friesland) :(professor/lector) Changing pedagogy and ICT
- Curriculum developer for informatics at Nat.Institute for Curriculum Development
- Studied Dutch language & literature and Theoretical linguistics (Computational linguistics) at UvA

Pedagogy and instruction and learning

Looking to instruction and learning we could distinguish various approaches (sometimes overlapping):

- Instructionism (Skinner): practice and drill
- Social Constructivism (Vygotsky)
- Cognitivism (Bruner)
- (Cognitive) constructivism (Piaget, Papert)
- Constructionism (Papert. Resnick)
- Neuroscience (research validates the cognitive/ constructivistic approach)

Trailblazers
on learningpsychology and ICT

- Lev Vigotsky: Zone of Proximate Development, social environment

- Jean Piaget: Developmental psychology, Constructivism

Logo: Papert & Resnick

- Seymour Papert (student of Piaget)
- Logo: from constructivism towards constructionism: making education
- “Students control the computer not the computer controls the student”
- Pedagogy: work in projects, social interaction, exploration not pure instruction

Mitchel Resnick: Lego Mindstorms, Scratch, Computer Clubhouses

- 10 years Computerclubhuis Amsterdam: Video with Papert en Resnick
  http://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code
- http://www.youtube.com/watch?v=zWFjkwo-ck4
- Robotics: integrates physics, biology, informatics, engineering Video World Robocup Junior 2013 in Eindhoven
- https://www.youtube.com/watch?v=mR8Zyt33NRY

CT approaches?

Definitions:

- Computational thinking : computer principles, language and problem solving skills (Sharples 2015)
- Leads to learning a ‘language’ and grammar compare Niels Bohr:“Science is not to tell us about the universe, but to tell us how to talk about the universe.” compare: CT is not about how a computer works but how can ‘communicate’ with and about IT
- Computational thinking: is about problem solving, algorithms, decomposition, conceptualizing not just programming, fundamental for everyone, combi and complement of math.thinking and engineering (Wing 2006)
- 2 forms of coding. Exploratory programming (playing with Scratch) then Thinky programming (introducing tools, techniques, data-structures, algorithms. (Sloman 2014)

Questions related to computational thinking skills

- Wing (2006) Is more than just coding. What can humans do better than computers? And What can computers do better than humans?
- Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions. (Wing, 2006)
- Grasp the notion of computability (incompleteness th. Gödel)
- Sharples (2015) key-elements: decomposition, pattern recognition, abstraction, algorithm design, debugging present a solution – Refining those steps.

Problem based learning

- 1. Examine a case and clarify terms
- 2. Identify the problem
- 3. Analyse the problem
- 4. Draft an explanatory model
- 5. Establish learning goals
- 6. Work individually to collect
- additional information
- 7. Apply and discuss additional information (Sharples 2015)

The pedagogy of Scratch and the maker movement, Resnick

- Listening is forgetting
- Reading is remembering
- Making is understanding

Why focus on language within CT?

1. **History argument:** ‘50: development of higher programming language: Math/automat. theory, electronic engineering , theoretical linguistics, cognitive psychology

2. **Pedagogical argument:** discover and explain theoretical concepts for not only beta-students e.g. (formal) grammar, natural vs. artificial language, AI, recursion, algorithms, finite state and (CF)-pushdown automata, Girls like language orientated applications

3. **Sustainability** Is a sustainable element of CT (till we have quantumcomputer?)

4. **Language technology**: automatic translation, speech and speaker recognition etc.

5. **Related to CT key-issues** (Wing)

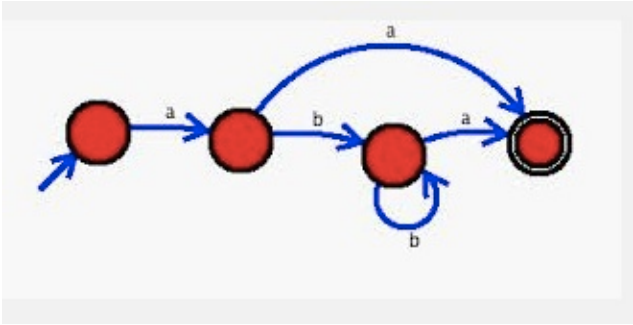
The computational capacity: a basic concept within CT/Coding

- consists of a mechanism with finite amount of terminal elements (words, number, sounds) and a set of combination rules

- that can generate an infinite amount of structured utterances (sentences, melody's) with a specific system of rules: a grammar

How?
e.g. by using a ‘toolbox’

- Using CF-sentence generators: for language (push down automata)
- Generating Haiku’s
- Rap’s
- Sentences
- Stories



- The key are the assigments, a combi of constructionism interaction, some instruction and make it meaningfull. There is not general recipe.

Conclusions

- CT, coding/programming asks for a non-instructionalist based approach
- Some instruction is always OK but not dominating
- Focus on active learning
- Let students/pupils collaborate
- Constructionist: maker education. Use 3D-printing, laser cutters.
- This demands for ‘new’ pedagogical skills of teachers
- Challenge is: develop more examples of teaching and assessment

Literature:

- Papert, Seymour. *Mindstorms. Children, Computers, and Powerful Ideas.* New York, 1980.
- Sharples, Mike*. Cognition, computers and creative writing*. Chichester, 1985
- Resnick, Mitchel. *Turtles, Termites, and Traffic Jams. Explorations in Massively Parallel Microworlds.* Cambridge (Mass.) 1997.
- Bruer, John T, *Schools for Thought*. *A science of learning in the classroom.* MIT-Press, 1997.
- Martinez S. and G. Stager. *Invent to Learn. Making, Tinkering, and Engineering in the Classroom.* Torance (CA.), 2013.
- Wing See Jeannette M. Wing, 2006, "Computational Thinking," in *Communications of the ACM* 49(3):33-35,
- Aaron Sloman, *What is computational thinking? What Forms of computational thinking will our children need when they grow up?* Invited Talk at CAS 2014 Conference Birmingham June 2014
- Sharples, Mike. Innovating Pedagogy. In: *Exploring new forms of teaching, learning and assessment, to guide educators and policy makers*. P24. About Scratch. Open University 2015.