Computational Thinking: from pre-school to higher education

Teaching

Learning

Projects

Resources

Anabela Gomes
anabela@isec.pt

Department of Informatics and Systems Engineering – Polytechnic Institute of Coimbra
&
Center of Informatics and Systems – University of Coimbra

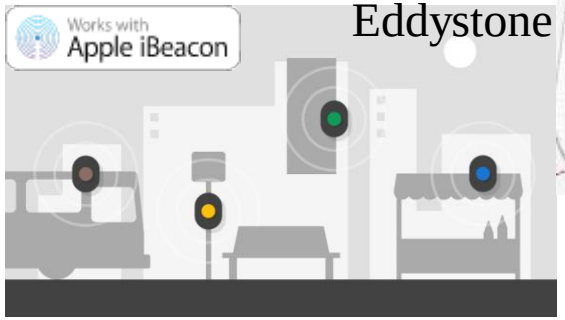

Norwegian Sea
Northwestern Passages
Kalaallit Nunaat Greenland
Ísland Iceland
Sverige Sweden
Suomi Finland
Norge Norway
Россия Russia
Canada
Hudson Bay
Danmark Denmark
United Kingdom
Ireland Éire
Беларусь Belarus
Polska Poland
Deutschland Germany
Україна Ukraine
Österreich Austria
France
România Romania
Italia Italy
Қазақстан Kazakhstan
Монгол Улс Mongolia
O'zbekiston Uzbekistan
Кыргызстан Kyrgyzstan
Ελλάς Greece
Türkiye Turkey
United States
North Atlantic Ocean
中国 China
대한민국 South Korea
East China Sea
Tunisia
Morocco
Algeria
Libya
Egypt
Iraq
Iran
Pakistan
Nepal
India
Western Sahara
Saudi Arabia
Oman
México Mexico
Gulf of Mexico
Cuba
Puerto Rico
Mali
Niger
Tchad Chad
Sudan
Yemen
Gulf of Aden
Arabian Sea
Bay of Bengal
ประเทศไทย Thailand
Việt Nam Vietnam
Pilipinas Philippines
Guatemala
Caribbean Sea
Nicaragua
Burkina Faso
Guinée Guinea
Nigeria
South Sudan
Ethiopia
Gulf of Thailand
Venezuela
Guyana
Suriname
Colombia
Ghana
Gulf of Guinea
Laccadive Sea
Malaysia
Ecuador
Gabon
République démocratique du Congo DR Congo
Kenya
Tanzania
Indonesia
Laut Banda
Brasil Brazil
Perú Peru
Angola
Zambia
Moçambique Mozambique
Bolivia
Namibia
Zimbabwe
Botswana
Madagasikara Madagascar
Indian Ocean
Paraguay
Chile
South Atlantic Ocean
South Africa
Uruguay
Argentina

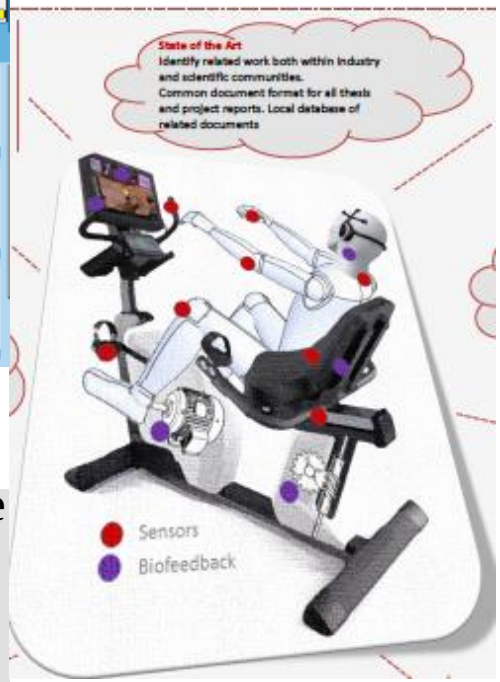
Portugal – Coimbra University

Portugal – Coimbra Polythechnic

Interests

- Computer Science Education
- Learning Styles
- Learning Taxonomies
- Assistive Technologies
  - ExoBike Project
  - BlueEyes Project
  - Symbolum Project
- Human Computer Interaction
- Brain Computer Interaction
- **Computational Thinking**





Agenda

- Computational Thinking
  - Terms
  - Concepts
  - Misconceptions
  - Importance
  - Underlying theories
  - Elements
  - Competencies & Skills
  - Assessing
  - Projects
  - Resources

CT - Terms

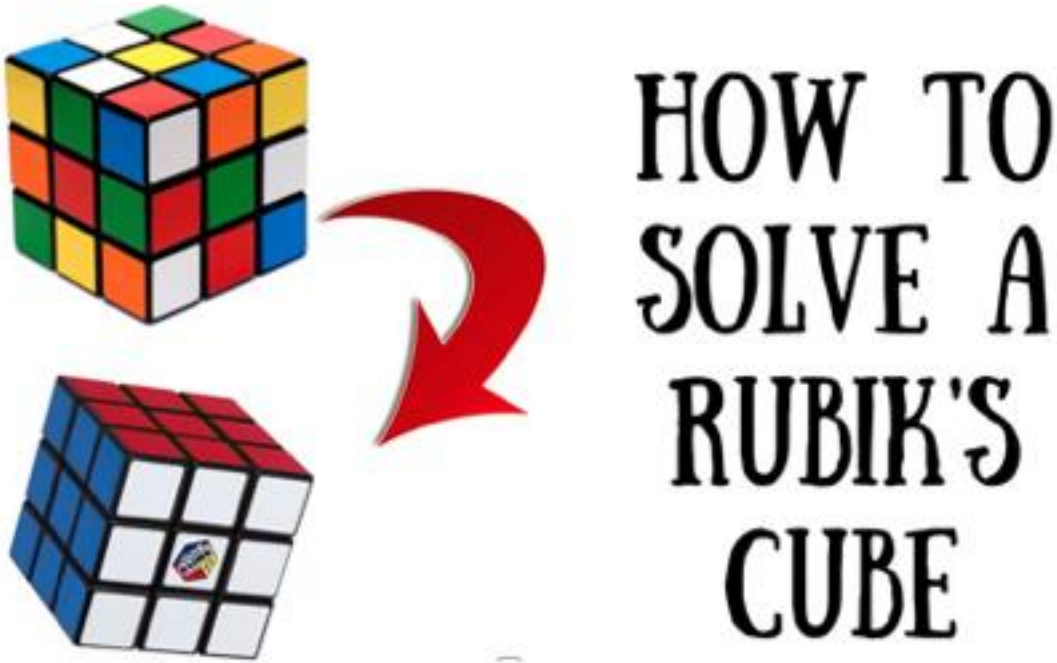
- **Thinkering terms:** Logical thinking, Algorithmic thinking, Engineering thinking, Analogical thinking, Mathematical thinking, Procedural thinking, Critical thinking
- **Problem solving terms**: Analysis, Generalization, Evaluation,…
- **Computer Science terms**: systems design, automation, recursion and recovery through redundancy,...
- **Imitation terms:** modelling, simulation, visualisation,…

CT – Concepts

- Solving problems
- Algorithms & Abstraction
- Decomposition
- Thinking Conceptually
- Repetition & scale
- Dealing with errors

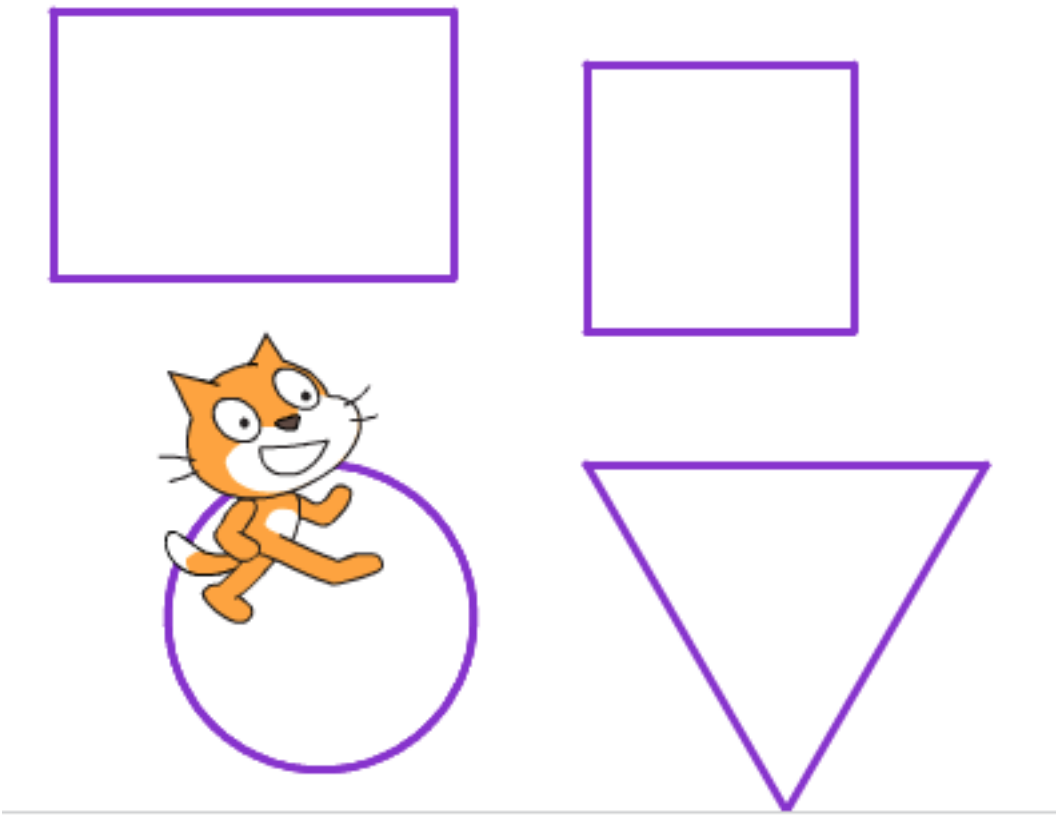
CT – Concepts

Solving Problems



CT – Concepts

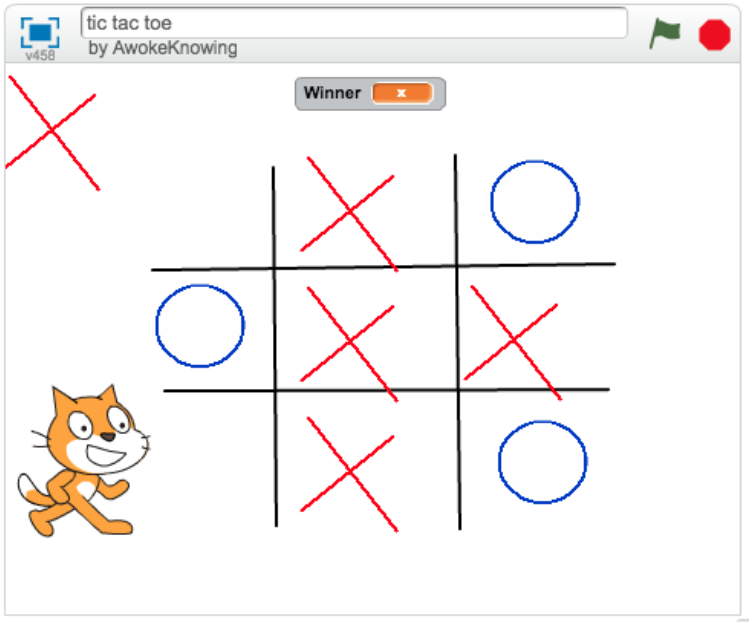
Algorithms & Abstraction



CT – Concepts

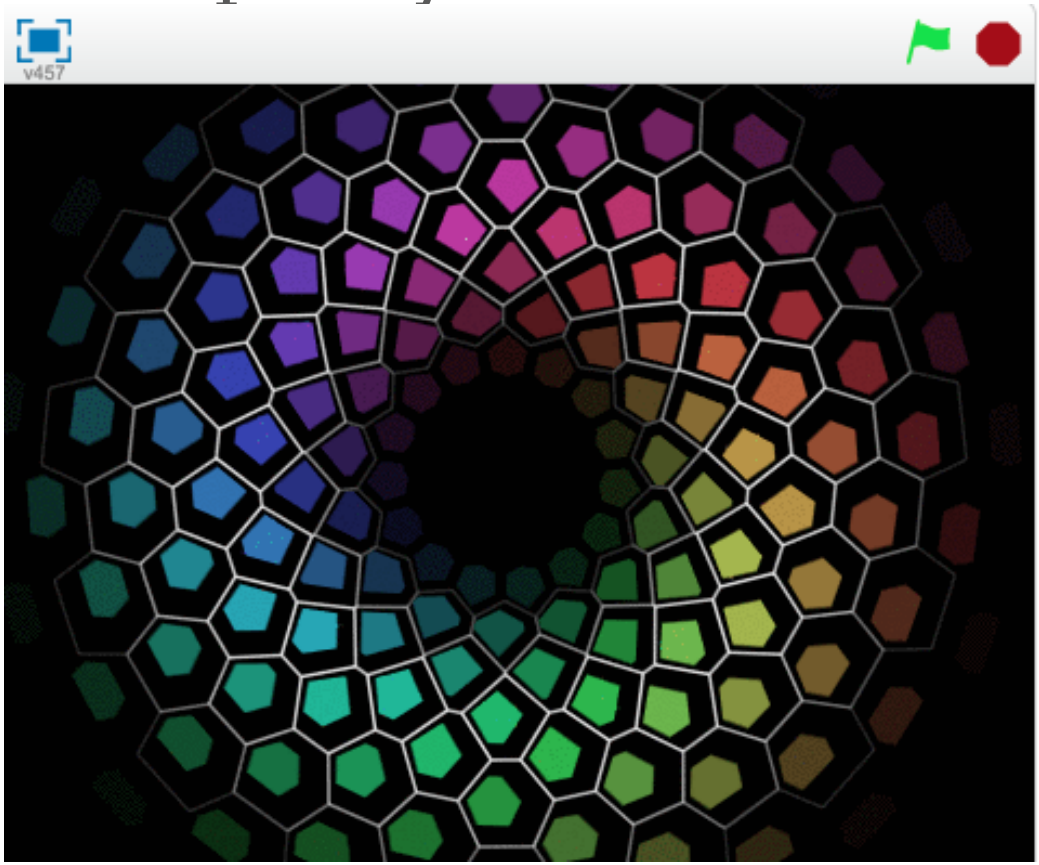
Decomposition

1. Display the board
2. Make a move
    - 2 a. Ask/Wait for a move
    - 2 b. Update the board
3. Decide winner
    - 3 a. Check 3 in a row
    - 3 b. Check 3 in a column
    - 3 c. Check 3 diagonally



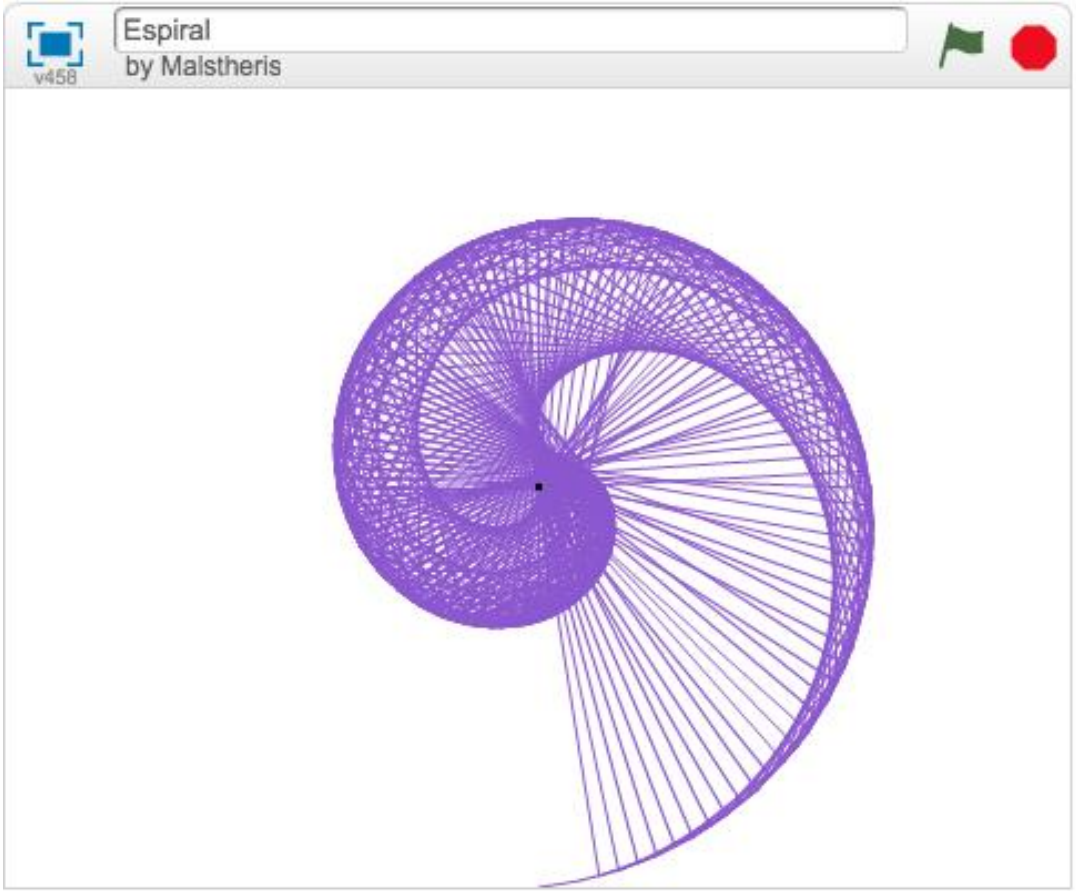
CT – Concepts

Thinking conceptually



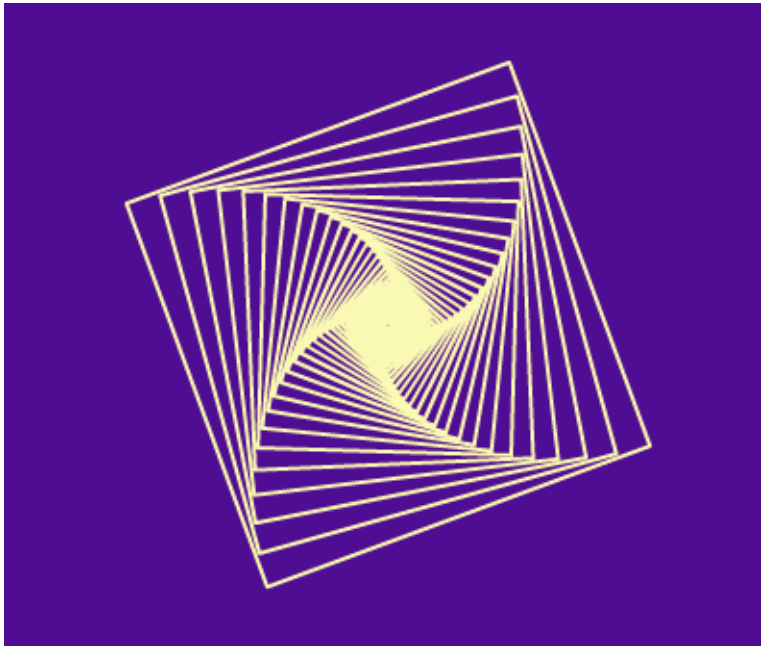
CT – Concepts

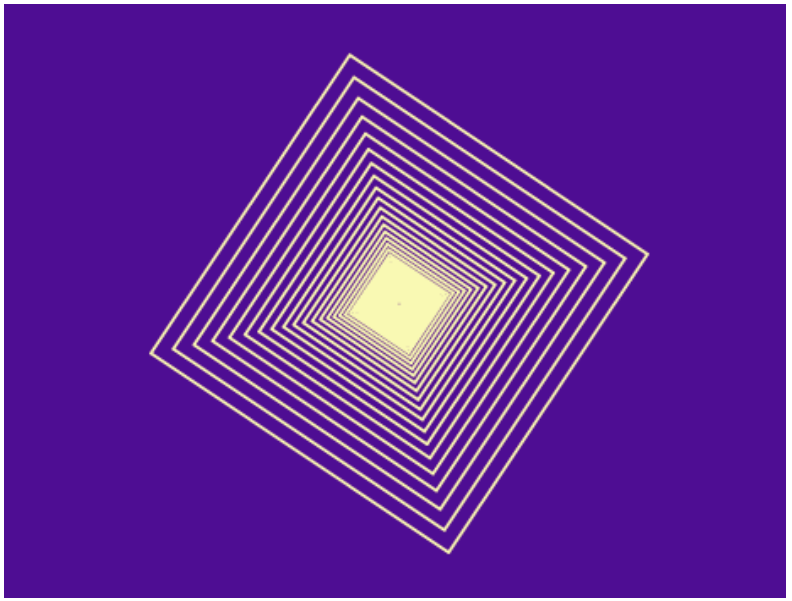
Repetition & Scale



CT - Concepts

Dealing with errors





CT – Definitions

- Wing (2006) - CT entails "solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science"
- Cuny, Snyder, and Wing (2010) - CT is a thinking process where "… solutions are represented in a form that can be effectively carried out by an information-processing agent"
- Berland and Wilensky (2015) - CT is "the ability to think with the computer-as-tool"
- …

CT – Conceptions

- It is a thought process
- It goes beyond programming
- It is the process involved in formulating a problem and expressing its algorithmic solution
- It helps us to understand the world and to understand ourselves
- It is not easy to teach

CT – Misconceptions

- It is not Computer literacy
- It's not just more technical details for using software
- It's not thinking like a computer
- It's not programming (necessarily)
- It doesn't always require a computer
- It's not one more thing to add to your curriculum

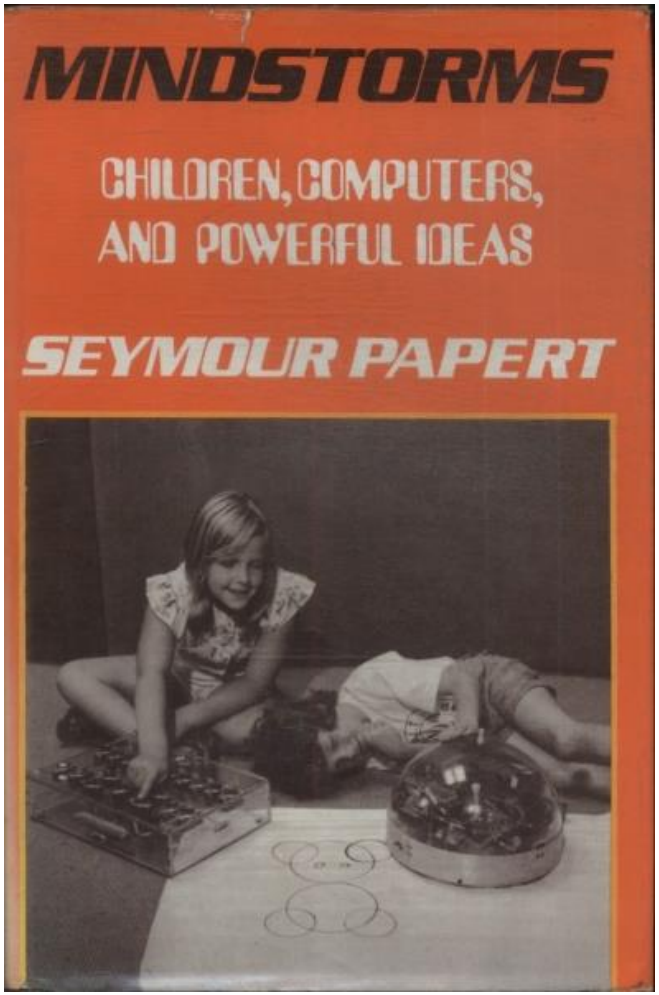
CT – Importance

- It emphasizes creating knowledge rather than using information
- It moves students beyond technology literacy
- It creates problem solvers instead of software technicians
- It presents endless possibilities for creatively solving problems
- It enhances the problem-solving techniques you already teach
- Without it you might be using inadequate models, doing inadequate empirical research, developing misguided strategies and failling to prepare learners for the rest of the 21st century.

CT - Theories

- Constructivist and experiential learning (Piaget)
  - Constructivist theories of psychology take a view of learning as a reconstruction rather than as a transmission of knowledge.

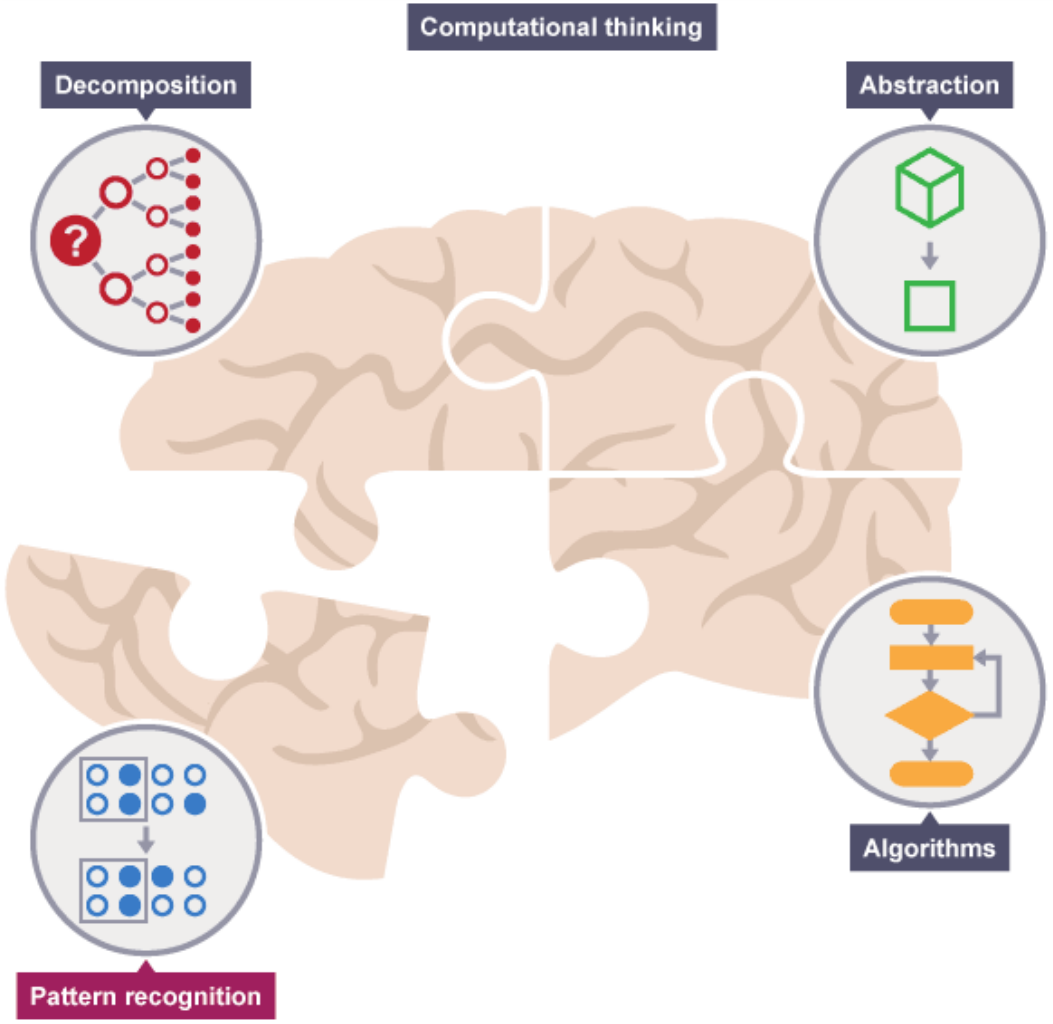
CT - Theories


MINDSTORMS
CHILDREN, COMPUTERS,
AND POWERFUL IDEAS
SEYMOUR PAPERT

CT - Theories

- **Constructionist learning** (Seymour Papert)
  - Learners construct mental models to understand the world around them
  - **Active learning**, Student-centered, focusing the responsibility of learning on learners
  - **Project-based learning,** students learn through participation in projects where they make connections between different ideas and areas of knowledge facilitated by the teacher through coaching rather than using lectures or step-by-step guidance
  - **Problem-based learning**, students learn about a subject through the experience of solving an open-ended problem found in trigger material
  - **Discovery learning** where students use information they already know to acquire more knowledge
  - Providing learning environments enabling **Discovery learning**
  - **Learning-by-making**, Learning by doing (John Dewey)

CT - Elements



- ImageSource: http://www.bbc.co.uk/education/guides/zp92mp3/revision

CT - Elements

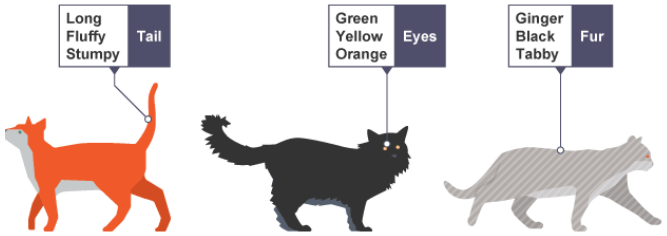
- Decomposition
  - Breaking down data, processes, or problems into smaller, manageable parts
- Patterns
  - Observing patterns, trends and regularities
- Abstraction
  - Identifying the general principles that generate these patterns
- Algorithm
  - Developing the step by step instructions for solving this and similar problems

CT – Across the curriculum

- **Decomposition: Break a problem into parts or steps**
  - **Literature**: Break down the analysis of a poem into analysis of meter, rhyme, imagery, structure, tone, dictation, meaning
  - **Maths**: apply order of operations in a complex expression
  - **Science**: Do a species classification (“genus”, a “family”, an “order”, a “class” a “branch” and a “kingdom”)

CT – Across the curriculum

- **Patterns: Recognize and find patterns or trends**
  - **Literature:** The relationship between characters, actions and ideas within a book connects with ideas outside the story.
  - **Maths:** A specific procedure is repeatedly used with n-1 discs of Hanoi Towers
  - **Science:**



CT – Across the curriculum

- **Abstraction: Generalize patterns and trends into rules, principles, or insights**
  - **Literature:** Write a story with branches
  - **Mathematics**: Figure out the rules for factoring 2nd-order polynomials.
  - **Science:** Build a model of a physical entity (the double helix model of DNA)

CT – Across the curriculum

- **Algorithm: Develop instructions to solve a problem or steps for a task.**
  - **Literature:** Steps needed to sort the names in a list
  - **Maths**: Do long division, factoring; do carries in addition or subtraction
  - **Science**: Steps needed to do an experimental procedure

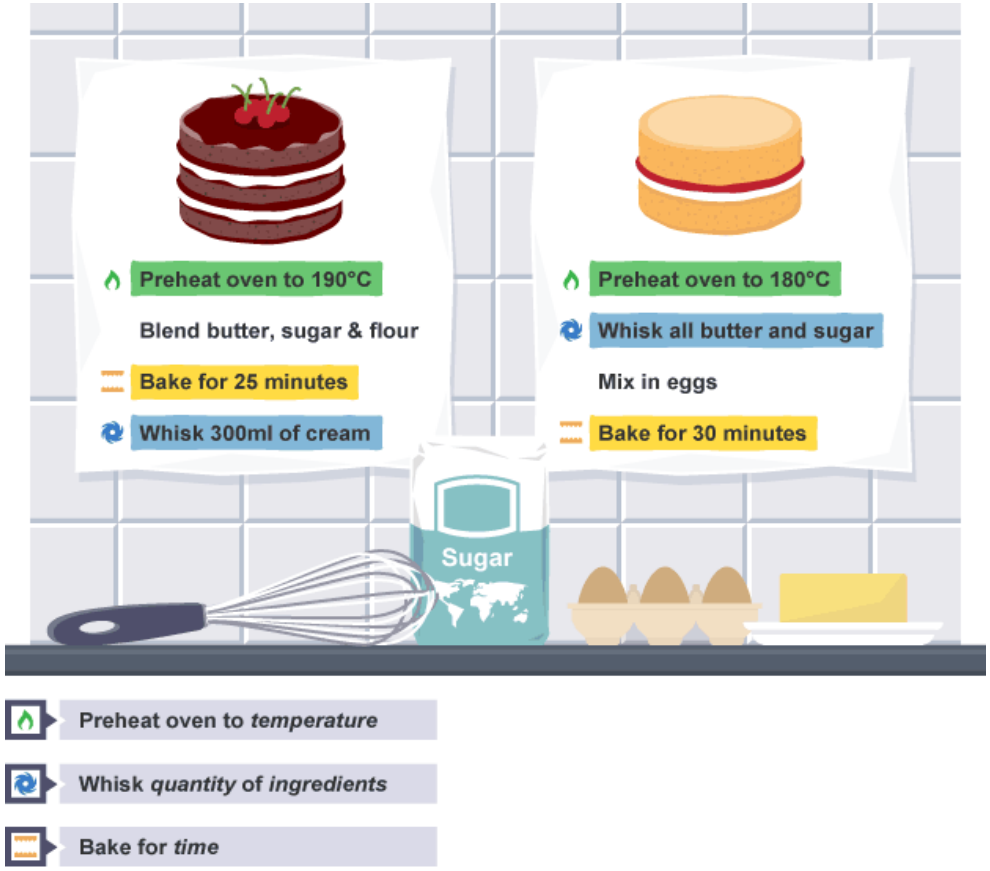
CT – Example: Decomposition

- The task of **baking a cake** would highlight the need for us to know the solutions to a series of smaller problems:

Decomposition
What kind of cake we want to bake?
What ingredients we need and how much of each?
How many people we want to bake the cake for?
How long we need to bake the cake for?
When we need to add each ingredient?
What equipment we need?

- Once we know how to bake one particular type of cake, we can see that baking another type of cake is not that different - because **patterns** exist.

CT – Example: Patterns



- Image Source: http://www.bbc.co.uk/education/guides/zxxbgk7/revision/3

CT – Example: Abstraction

Specific Details	General Patterns
A particular Banana cake has a layer of chocolate cream in the middle.	A cake needs ingredients.
A particular Banana cake uses 600 g of sliced bananas.	A cake needs specific ingredient quantities.
A particular Banana cake takes 1h: 48 m to cook.	A cake needs a specific time to bake.

CT – Example: Algorithm

- When you write an algorithm you need to include precise, step-by-step instructions.

Step 1	Mix Margerine and sugar
Step 2	Add the eggs to the mixture
Step 3	Beat the mixture
Step 4	Add the flour to the mixture …

CT – Example: Algorithm

- "Program your teacher" to make a Jam Sandwich (**Sandwich Bot)** Junior Computer Science
  - Failing in precision, uniqueness, unambiguity

CT – Concepts & Competencies

Concepts	Competencies
Decomposition	Breaking complex artefacts, processes or systems into their component parts.
Pattern recognition	Identifying the patterns and commonality between artefacts, processes or systems.
Abstraction	Dealing with complexity by reducing unnecessary detail
Algorithm	Identifying the processes and sequence of events
Evaluation	Systematically make substantiated value judgements

CT – Competences & Skills

1970 - Most Valued Skills	2009 - Most Valued Skills
Writing	Teamwork
Computational Skills	Problem Solving
Reading Skills	Interpersonal Skills
Oral Communication	Oral Communication
Listening Skills	Listening Skills
Personal Career Development	Personal Career Development
Creative Thinking	Creative Thinking
Leadership	Leadership
Goal Setting/Motivation	Goal Setting/Motivation
Teamwork	Writing
Organizational Effectiveness	Organizational Effectiveness
Problem Solving	Computational Skills
Interpersonal Skills	Reading Skills

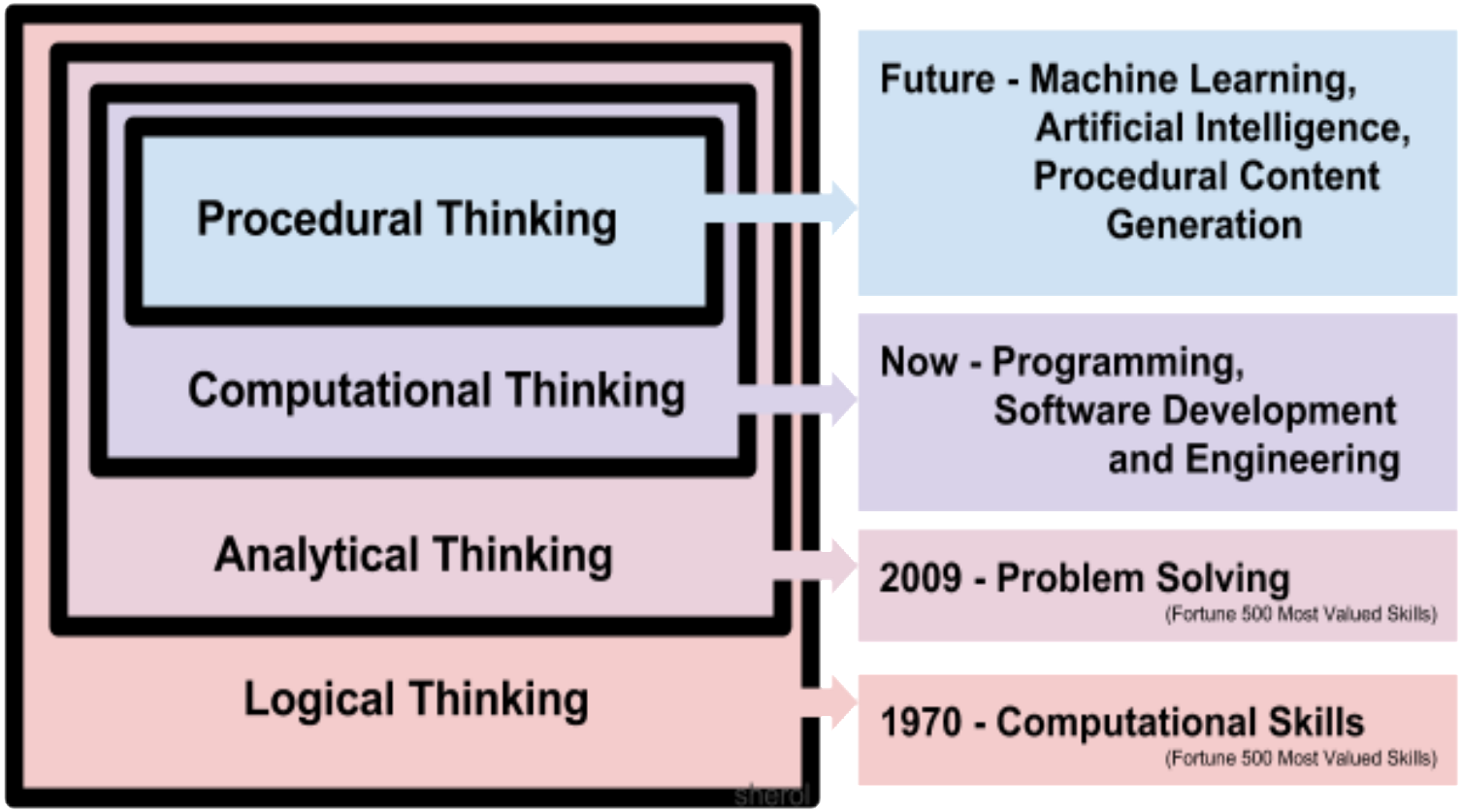
- Source: Fortune 500 most valued skills, cited in Linda Darling-Hammond et al, Criteria for High-Quality Assessment (2013)

CT – Competences & Skills

2015 - Most valued skills	2020 - Most valued skills
Complex Problem Solving	Complex Problem Solving
Coordinating with Others	Critical thinking
People Management	Creativity
Critical thinking	People Management
Negotiation	Coordinating with Others
Quality Control	Emotional Intelligence
Service Orientation	Judgement and Decision making
Judgment and Decision Making	Service Orientation
Active Listening	Negotiation
Creativity	Cognitive Flexibility

Source: Future of Jobs Report; World Economic Forum

CT – Competences & Skills



CT - Challenges

- How can we incorporate computational thinking into an already crowded curriculum? **Strategies**?
- How do we **train** a generation of **teachers** who have no background in this sort of thing?
- How will we **assess** this – and where does it 'fit' within our competency framework?"

CT – Teaching/Learning Strategies

- **Project-based learning methods**
  - Real-world challenges and problems
  - Interdisciplinary problems
    - Extended period of time for students to investigate/plan, try,…
- **Engineering Design Process**
  - Planning, Trying, Modifying, Communicating
  - Giving time for each phase
- **Inquiry and discovery-based approaches**
  - Posing questions, problems or scenarios—rather than presenting established facts or information
    - Why? How? What?

CT – Teaching/Learning Strategies

- **Active learning**
  - Students actively involved
  - Responsibility of learning on students
  - Engage students in learning (KSA - Knowledge, Skills and Attitudes)
    - **Why? How? What?**
  - Encourage students to introduce their own questions to the lessons
- **Discovery learning**
  - The domain should not directly be offered to learners but learners have to deduct the domain from experiences or examples
  - Teachers provide adequate guidance, clarify doubts and supervise students
  - Gradually remove the scaffolds to encourage greater independence and resilience in students.

CT – Train teachers

- Applying engaging/motivational approaches
  - Implement Role-Playing activities, Cooperative and Collaborative tasks
  - Teach according to students' cognitive levels (**Bloom's Taxonomy**)
  - Teach according to students' **learning styles** preferences (ex: Felder-Silverman)
- Attending workshops and training
- Sharing pedagogical experiences with other teachers of both what works and what doesn't work.

CT – Train teachers

- The International Society for Technology in Education (ISTE), Computer Science Teachers Association (CSTA) and the UK Computing at School working group (CAS) have collaborated with representatives from education and industry to develop computational thinking resources for educators.
  - ISTE Computational Thinking Page
  - CSTA Computational Thinking Page
  - CAS Computational Thinking Page
  - Google's Exploring Computational Thinking (ECT) page

CT – Train Teachers

- Mitchel Resnick

CT – Train Teachers

- Mitchel Resnick (4 P's)
  - **Projects**. People learn best when they are actively working on meaningful projects – generating new ideas, designing prototypes, refining iteratively.
  - **Peers**. Learning flourishes as a social activity, with people sharing ideas, collaborating on projects, and building on one another's work.
  - **Passion**. When people work on projects they care about, they work longer and harder, persist in the face of challenges, and learn more in the process.
  - **Play**. Learning involves playful experimentation – trying new things, tinkering with materials, testing boundaries, taking risks, iterating again and again.

CT – How to assess?

- Videos explaining how to test CT
  http://scratched.gse.harvard.edu/ct/assessing.html
  - "Experimenting & Iterating", "Testing & Debugging", "Reusing & Remixing"
- Computational Thinking Assessment Grid
  - Brenann, K., Balsch, C & Chung, M. (2014) Creative Computing. An Introductory Computing Curriculum Using Scratch.
    - "Experimenting & Iterating", "Testing & Debugging", "Reusing & Remixing" and "Abstracting & Modularizing "

CT – How to assess?

- **Experimenting & Iterating**
  - Build a project step by step
    - Describe how you built your project, step by step
  - Try things as you go
    - What other things have you been experiencing during project design?
  - Make revisions based on what happens
    - What revisions did you make and why did you do them?
  - Try different ways to do things, or try new things
    - Describe the different approaches you experienced in your project, or when you tried to do something new.

CT – How to assess?

- **Testing & Debugging**
  - Observe what happens when you run your project
  - Describe what is different from what you want
    - Describe what happened to your project differently than you intended.
  - Read through the "scripts" to investigate
    - Describe how you read the code to find the cause of the problem
  - Make changes and test to see what happens
    - Describe how you made the changes and tested to see the results.
  - Consider other ways to solve the problem
    - Describe how you would consider other ways to solve the problem

CT – How to assess?

- **Reusing and remixing**
  - Find ideas and inspirations by trying other projects and reading the scripts
    - Describe if you found inspiration in other projects and reading the available code.
  - Select a part of another project and adapt it for your project
    - Describes how you selected a part of another project and adapted it to your project
  - Modify an existing project to improve or enhance it
  - Give credit to people whose work you build on or are inspired by
    - Describe as you mention / cite people whose work inspired your own

CT – How to assess?

- Bebras (http://www.bebras.org/)
  - An international initiative aiming to promote Computer Science and Computational Thinking among school students of all ages.
    - Participants are supervised by teachers who may integrate the *Bebras* challenge in their teaching activities.
      - The challenge is performed at schools using computers or mobile devices.
  - Each country can join the initiative within certain rules

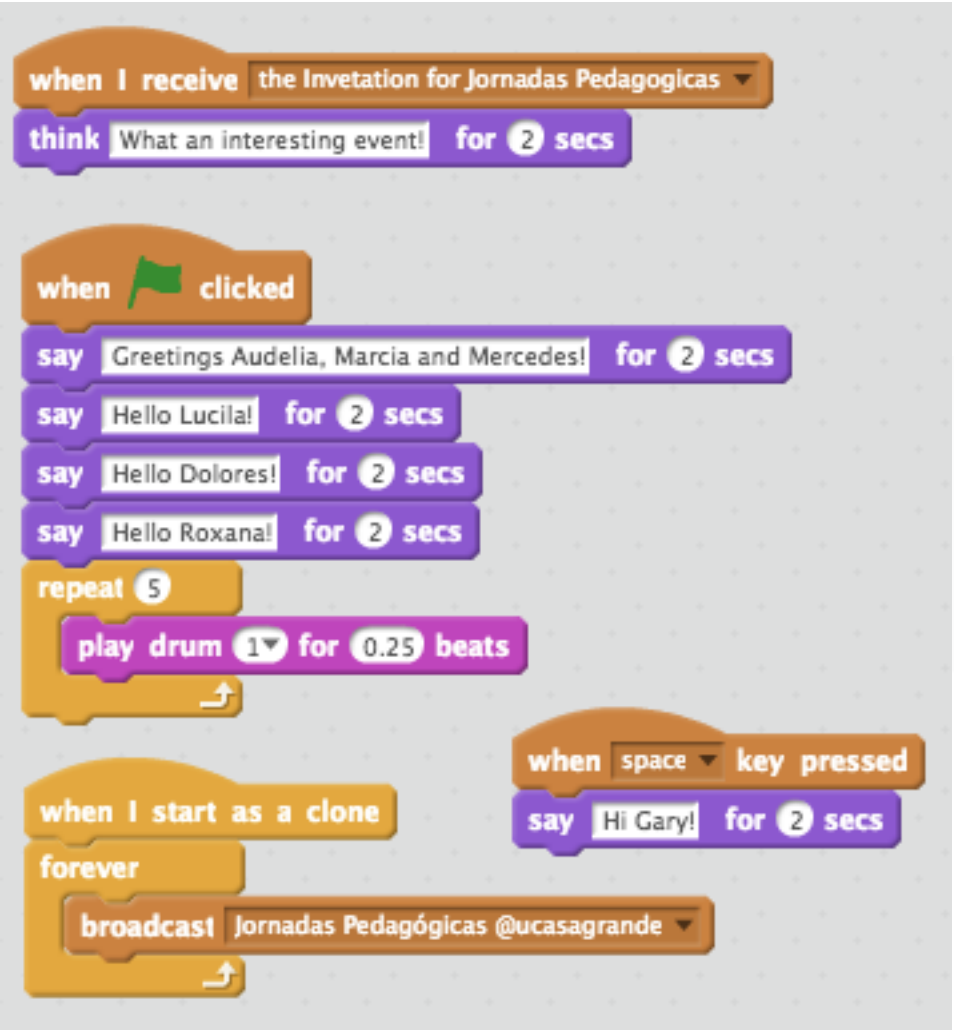
CT – How to assess?

- Other instruments…
  - CT Pattern Analysis (**CTPA**) in K-12 schools (Ioannidou et al., 2011).
  - **PECT** (Progression of Early Computational Thinking) is a Scratch-based assessment designed for elementary school students (Seiter & Foreman, 2013).
  - DrScratch
  - Computational Thinking Test (**CTT**) (Román-González, Pérez-González, & Jiménez-Fernández, 2016).
  - Computational Thinking Skills Scale (**CTSS**) (Korkmaz, Çakir, and Özden, 2017).

CT – Projects with Scratch

- Scratch is a free educational programming language that was developed by the Lifelong Kindergarten Group at the Massachusetts Institute of Technology (MIT)
- Scratch is designed to be fun, educational and easy to learn. It has the tools for creating interactive stories, games, art, simulations, and more, using block-based programming, without programming knowledge but helping to the development of Computational Thinking.
- Users of all ages (if they know how to read and write) can program in Scratch by dragging blocks from a block palette and attaching them to other blocks like a jigsaw puzzle.
- https://scratch.mit.edu/

CT – Projects with Scratch



CT – Portuguese Scratch Projects

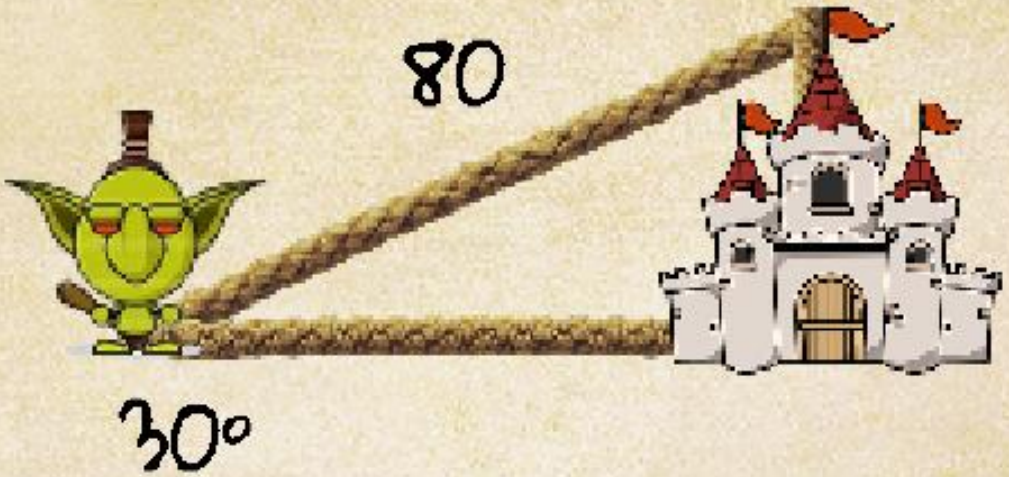
- A- Multidisciplinary Projects at **1º- 4º year** (Fátima **2015/2016**) – involving certain topics of the curricular subjects (**Sciences**, **Maths**, **Portuguese**) around a chosen theme
- B - **PhD** (Access the **Satisfaction**, **Motivation** and **Learning** of the English language and linguistic acquisitions) - English Project at 3º- 4º year (Fátima **2015/2016 onwards**)
- C - Multidisciplinary Projects at **3º-4º year** (Coimbra- **2015/2016**) – involving certain topics of all curricular topics (**Mathematics**, **Natural Sciences**, **Portuguese Language**, **School Area** and **Civics**) around the socio-cultural realities, environmental and economic of the Coimbra Council

CT – Portuguese Scratch Projects

- D – Computer Science + Maths Project in Higher Education (**2016/2017**)
  - The Scratch was introduced in the first year of the degree in **Biomedical Engineering of ISEC** in a joint action in **two curricular units**: Introduction to Information Technology (Introduction to programming) and Calculus I.
  - The idea is to use Scratch as a visualization tool intended to help students **in their early programming stages**, **lowering the levels of abstraction** through concrete representations. Hence the subproject "**MathScratch**" arouse.


Pontuação
0
55
Há um Goblin!

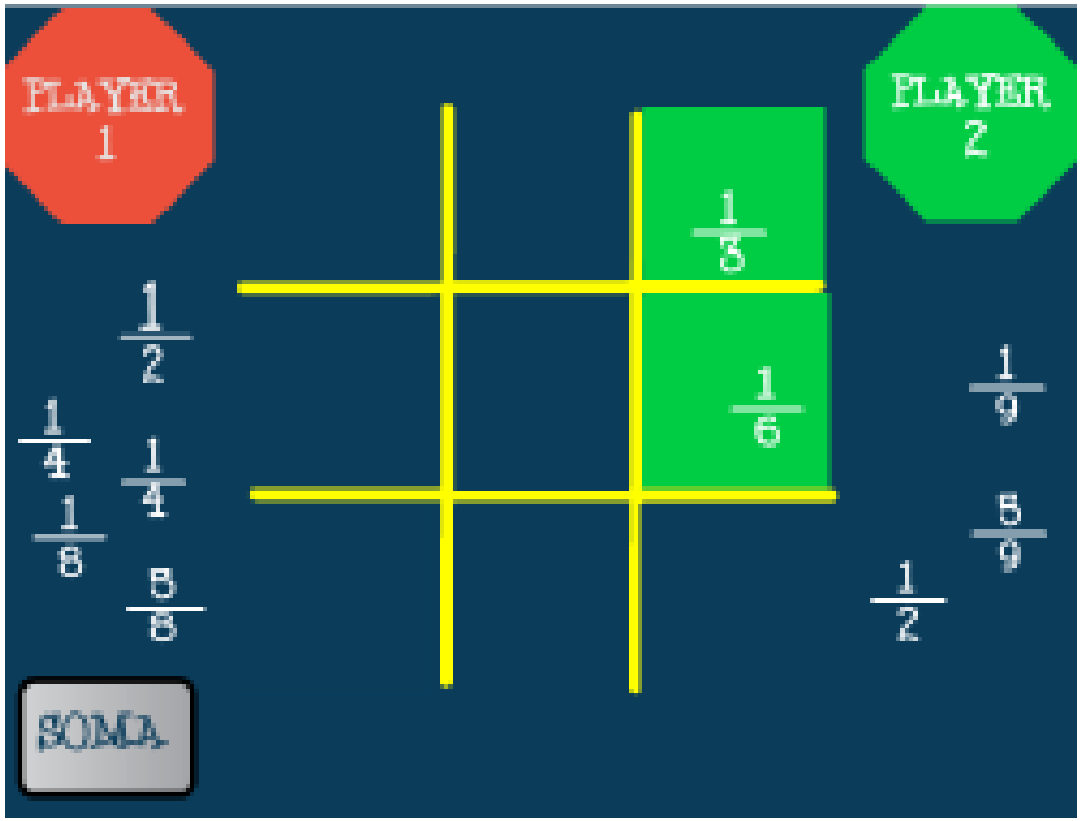
Pontuação 0



15 40 50

X A B

CT – Portuguese Scratch Projects



CT – Portuguese Scratch Projects

- E – "All in Scratch" project (**2016/2017**) to train **future professionals** in the degree in Educational Sciences of the Faculty of **Psychology** and Educational Sciences of the University of Coimbra, in the area of Educational Technology.

- F - **PhD** (To know the potential of Scratch in the professional development of the graduates in Educational Sciences) - Coimbra **2017/2018 onwards**

CT – Portuguese Scratch Projects

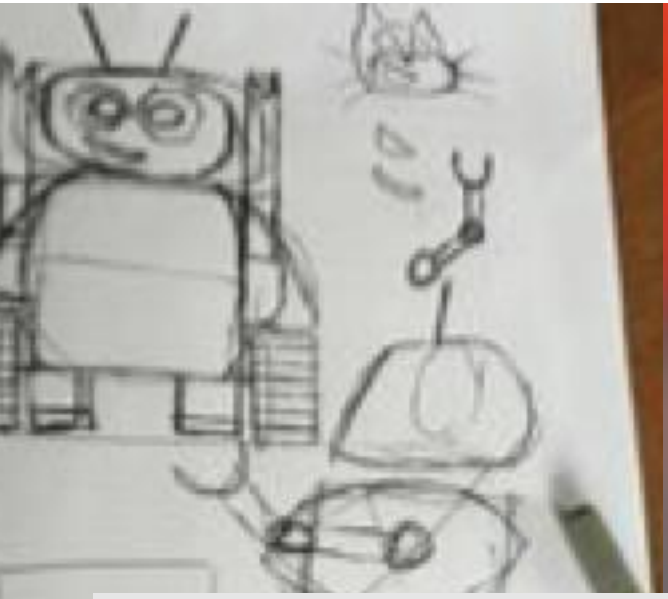
- G - Projects for Special Needs (**2016/2017 onwards**)
  - I teach a subject about Usability and Accessibility where students have to plan and build a Scratch project having a user with special needs
  - Cognitive Deficits
  - For Blind (with Makey Makey)
  - For coordination (with Leap Motion)
  - For rehabilitation (with Kinect and Arduino)
- *Some other projects…*

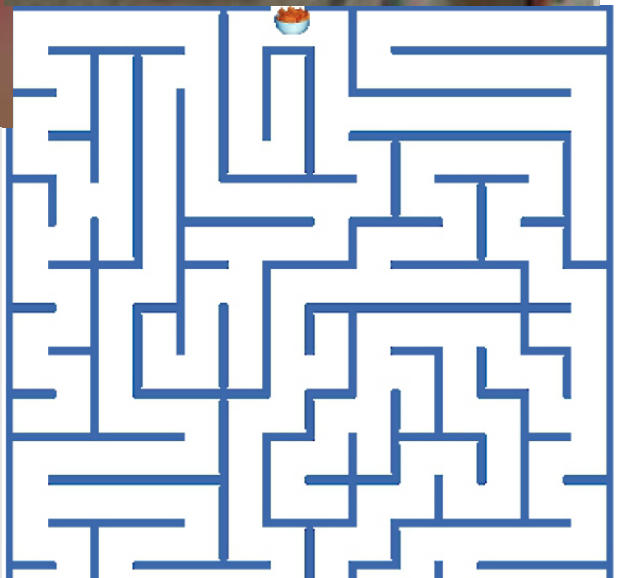
CT – Scratch for Cognitive Deficits

CT - Scratch for Blind people with Makey-Makey

CT – Scratch for coordination (leap Motion)

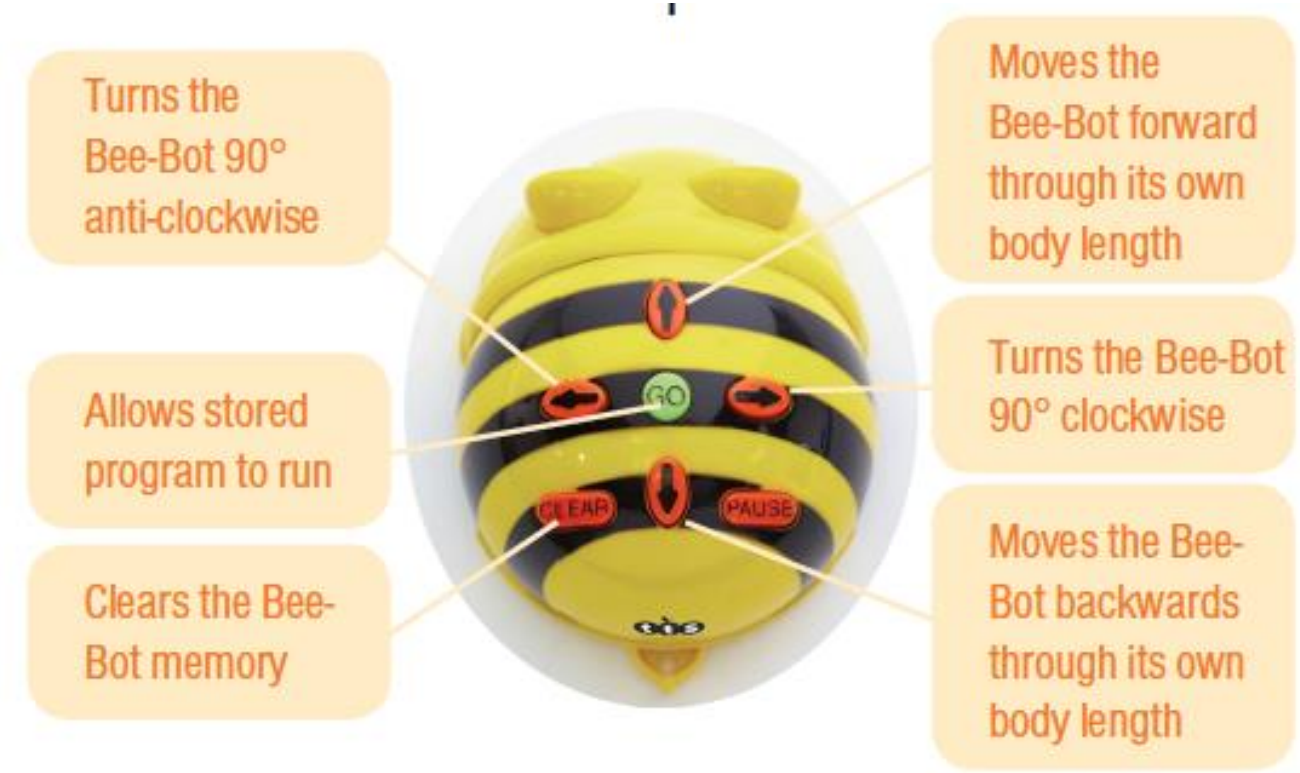
Scratch for rehabilitation with Kinect and Arduino





CT - Resources for all ages

- Bee-Bot – 3 Years
  - https://www.bee-bot.us/

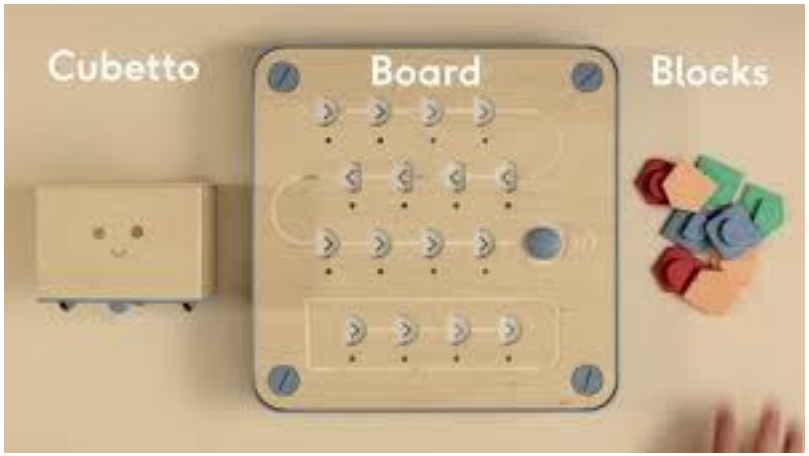


CT - Resources for all ages

- Code-a-piller: 3 Years
  - http://www.fisher-price.com/en_US/brands/think-and-learn/index.html

CT - Resources for all ages

- Cubetto (3 years)
  - https://www.primotoys.com/

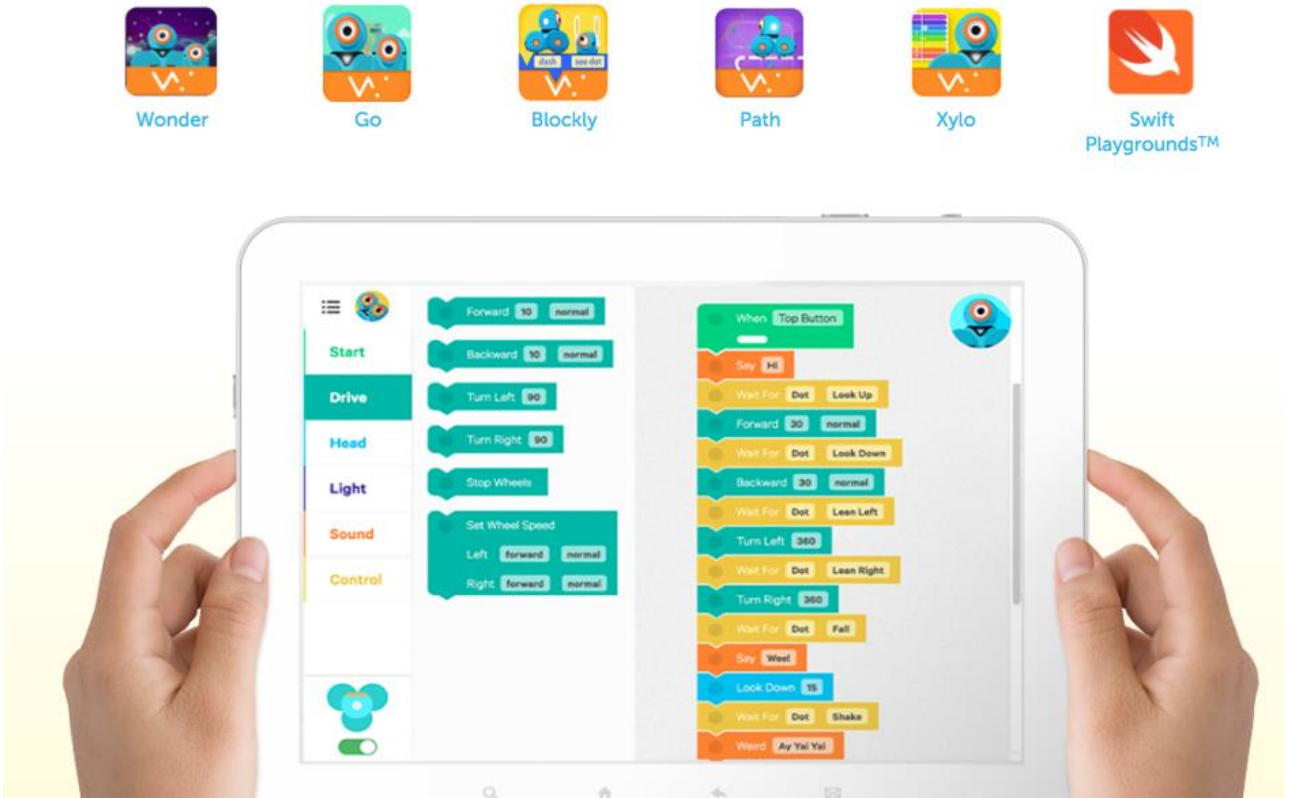


CT - Resources for all ages

- Kibo (4 years)
  - http://kinderlabrobotics.com/kibo/

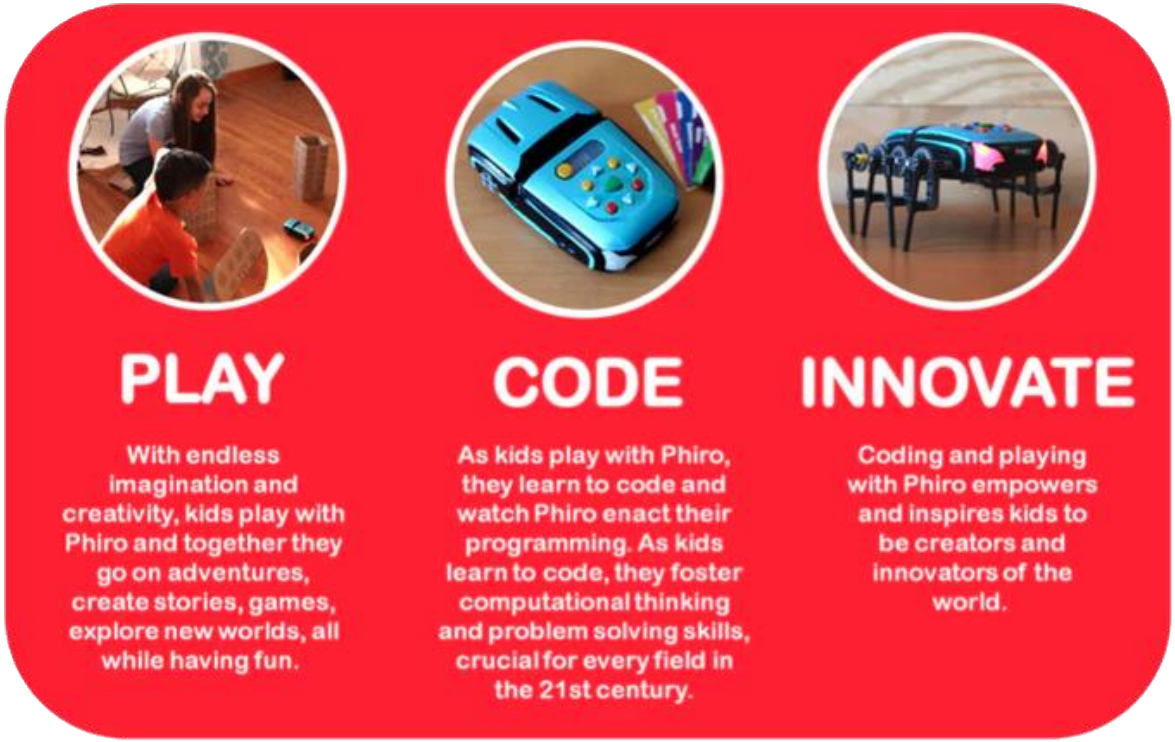
CT - Resources for all ages

- Dash-Dot: 4-12 Years
  - https://www.makewonder.com/dash



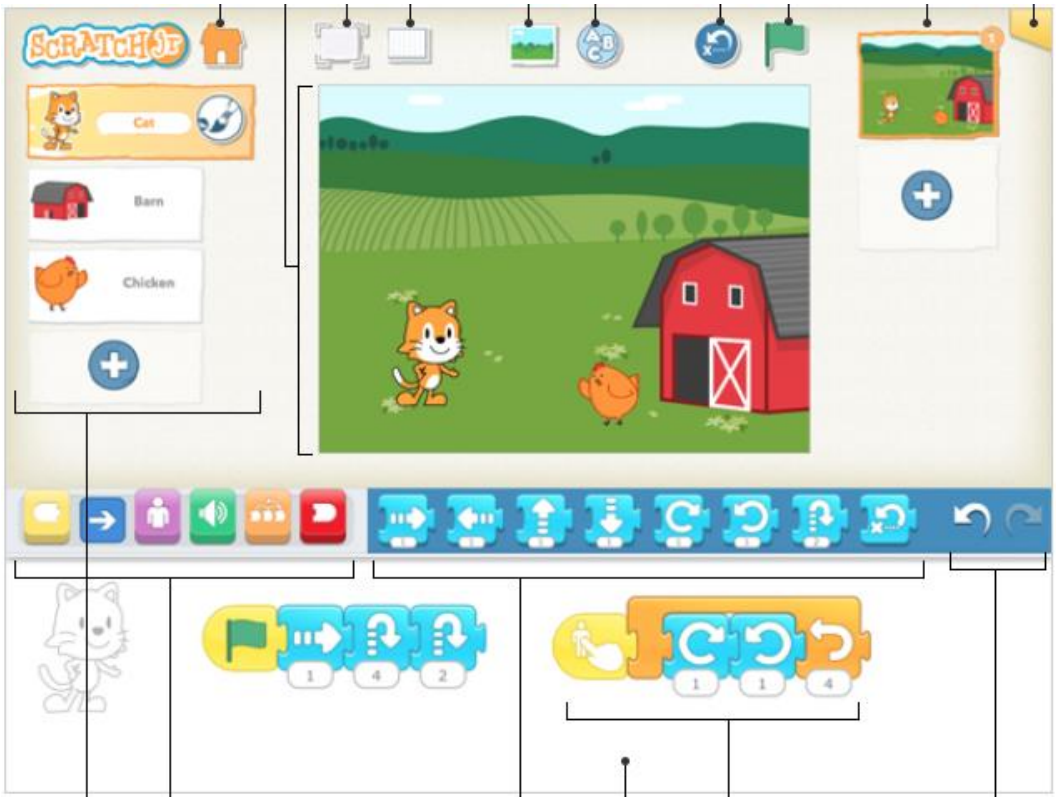
CT - Resources for all ages

- Phiro: 4-12 years
  - https://www.kickstarter.com/projects/2074714954/phiro-a-smart-robot-for-kids-learn-to-code-in-5-wa



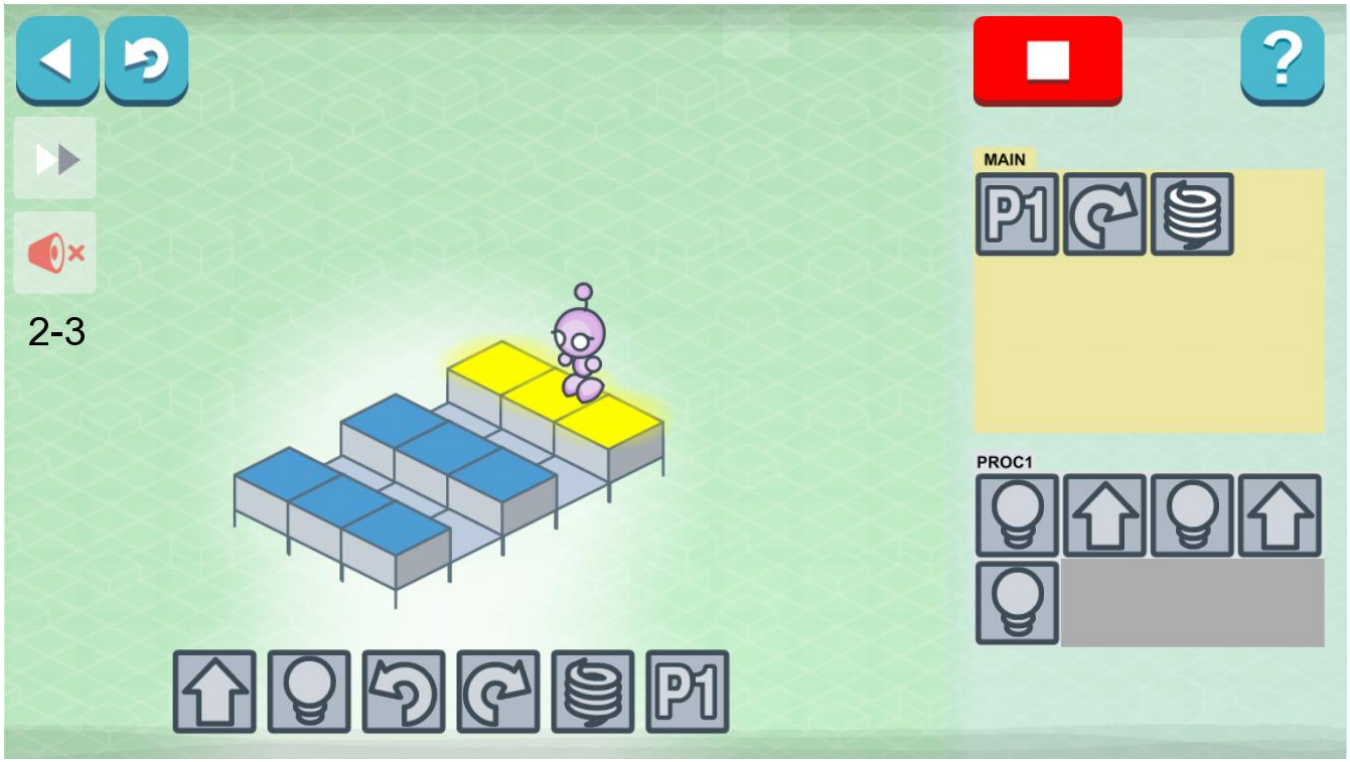
CT - Resources for all ages

- ScratchJr (5-7 years)
  - https://www.scratchjr.org/



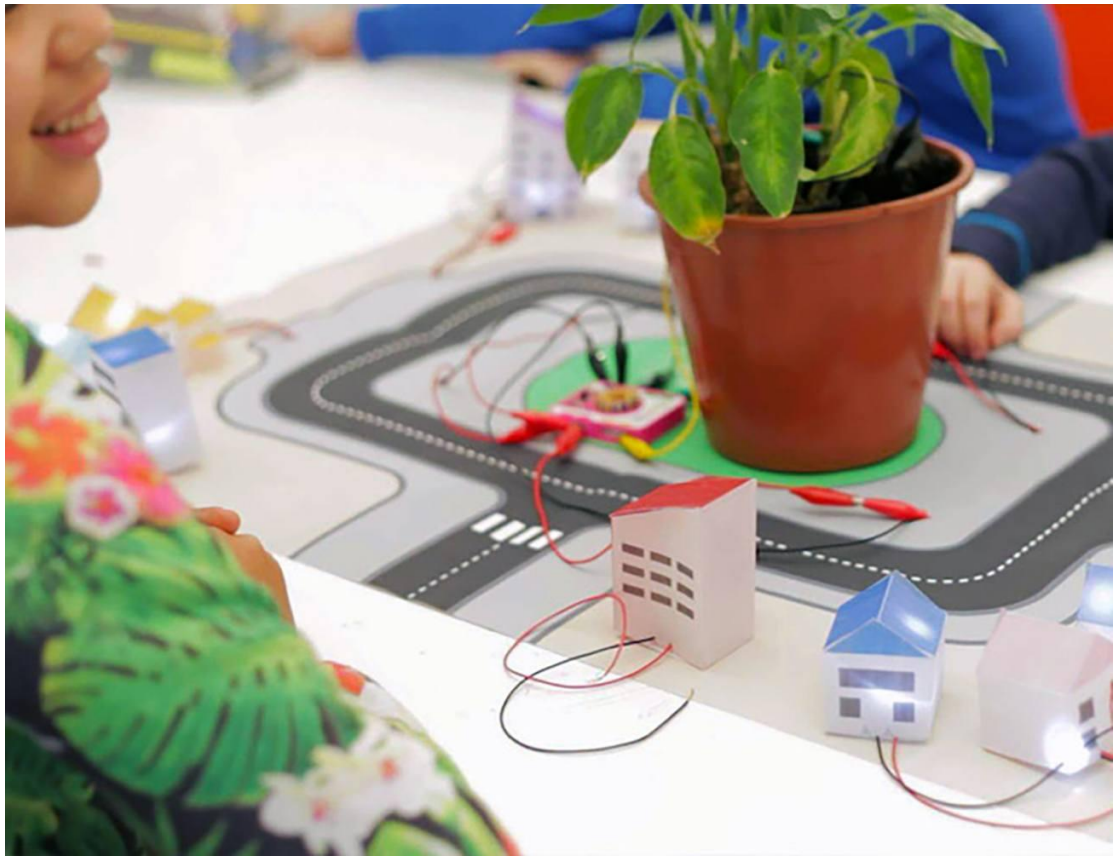
CT - Resources for all ages

- Lightbot
  - http://lightbot.com/



CT - Resources for all ages

- QMOD - https://www.kickstarter.com/projects/qmod/qmod-energy-toys-for-future-innovators?ref=NewsMay0417



CT - Resources for all ages

- Microsoft's Kodu
  - https://www.kodugamelab.com/

CT - Resources for all ages

- MineCraft
  - https://minecraft.net/en-us/

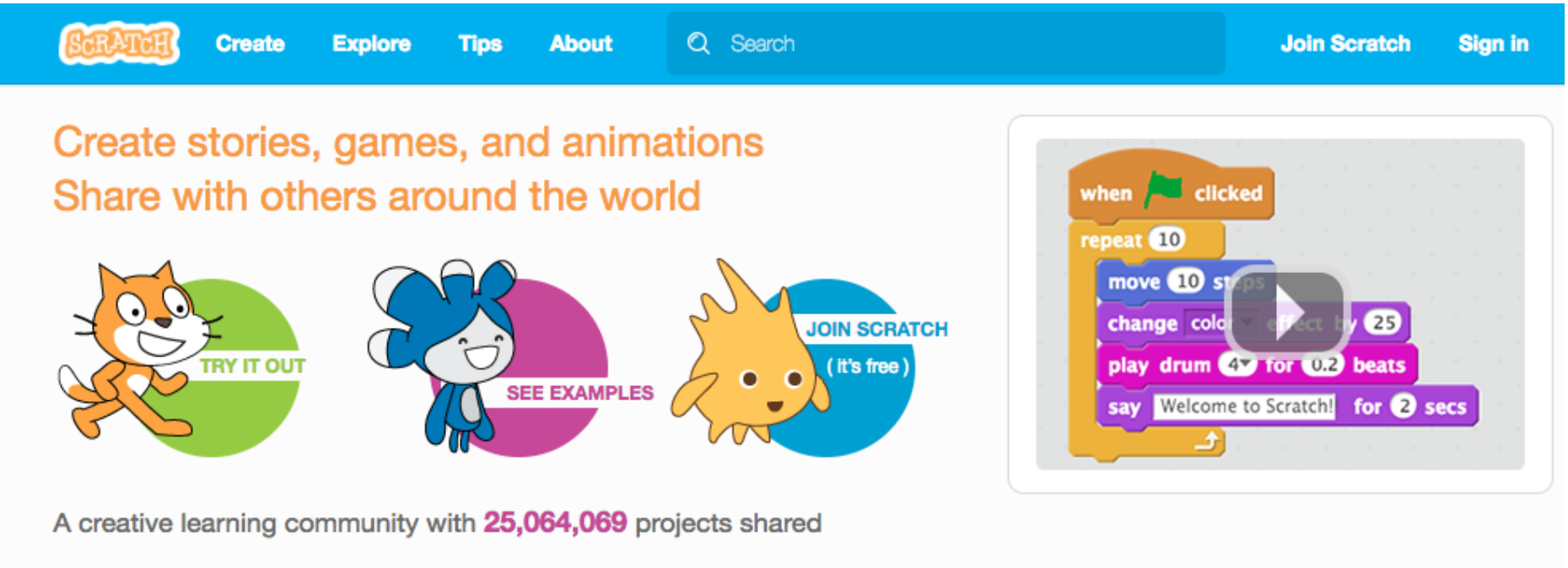
FIND LESSON EXAMPLES ACROSS SUBJECTS AND AGES

3-5 years | 6-7 years | 8-10 years | 11-13 years | 14+ years

Math | Science | Lang Arts | History | Visual Art

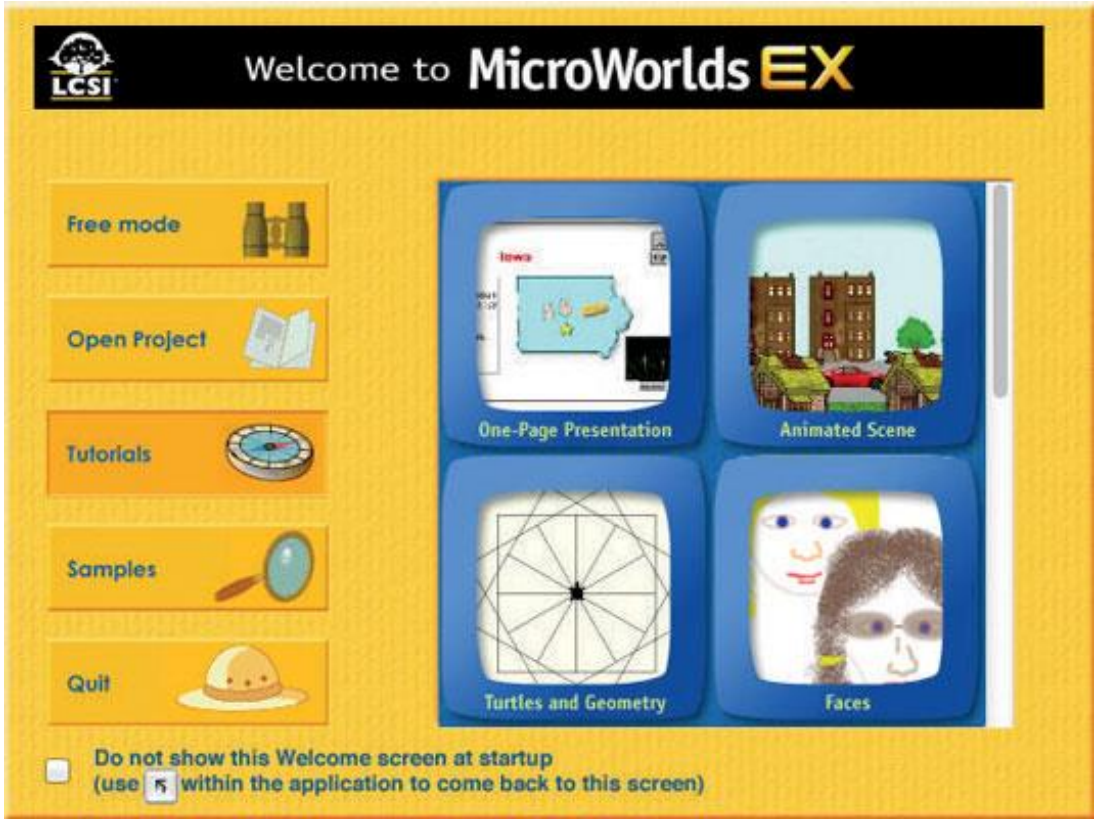
CT - Resources for all ages

- Scratch
  - https://scratch.mit.edu/



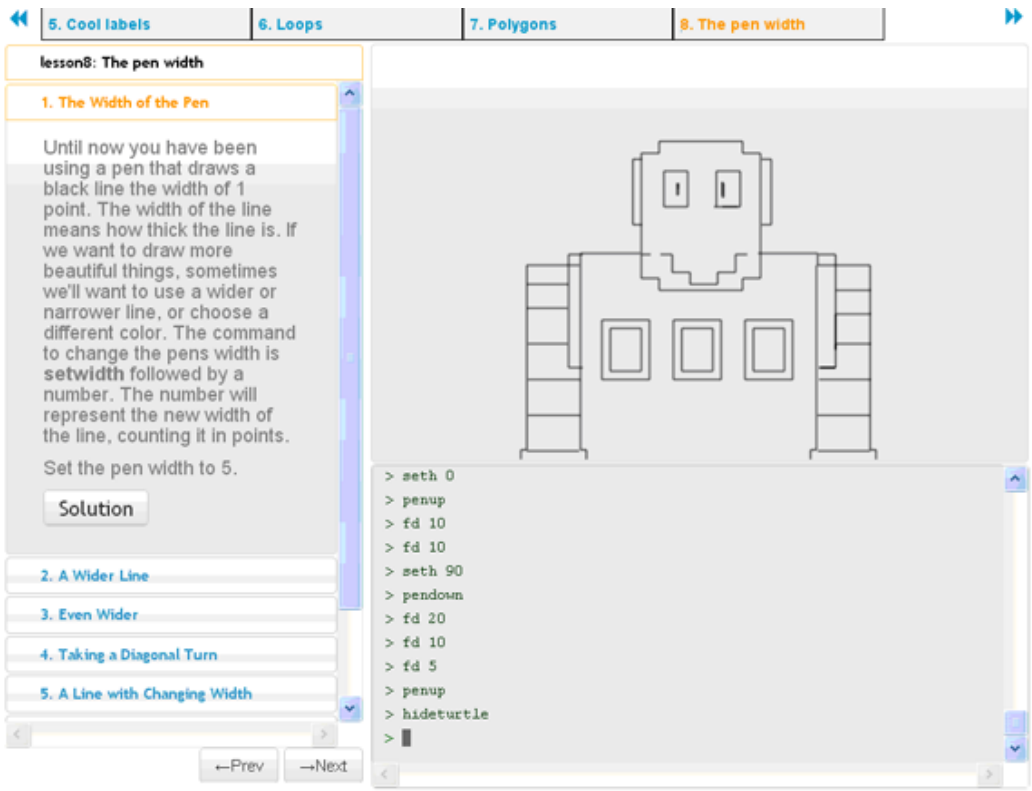
CT - Resources for all ages

- MicroWorlds EX (LOGO)
  - http://www.microworlds.com/



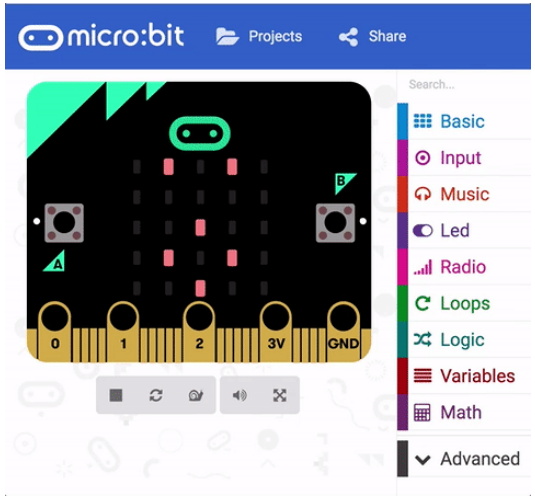
CT - Resources for all ages

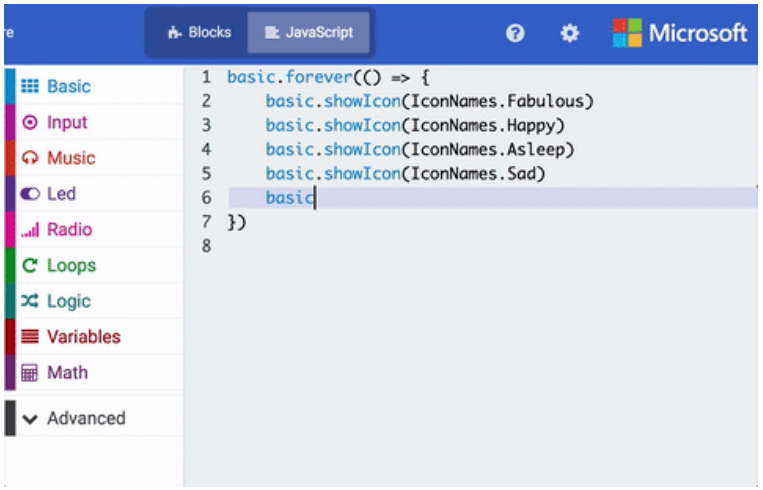
- TurtleAcademy (LOGO)
  - https://turtleacademy.com/

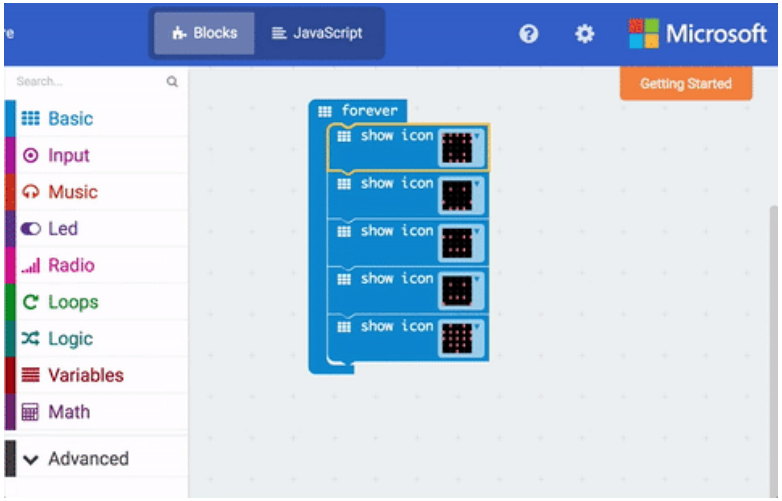


CT - Resources for all ages

- TouchDevelop -> MakeCode
  - https://www.touchdevelop.com/

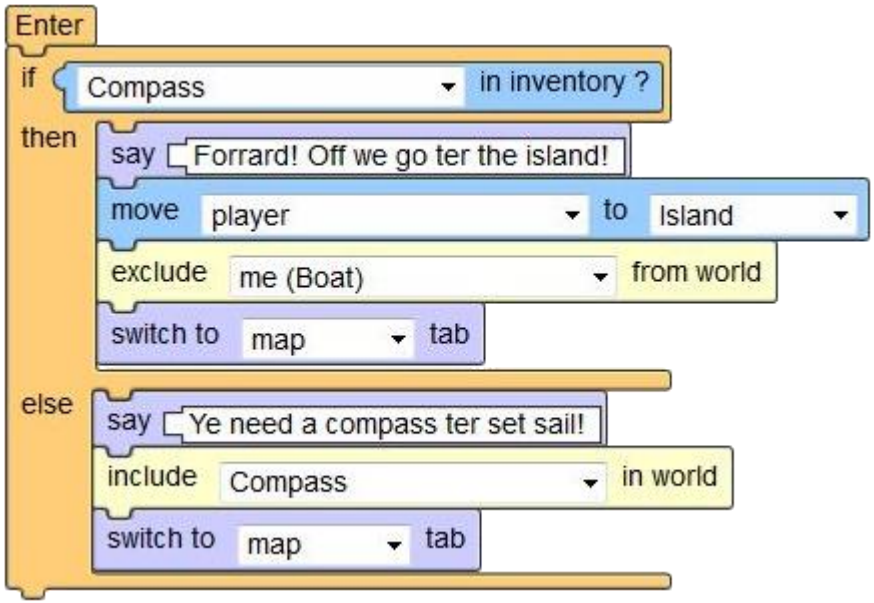




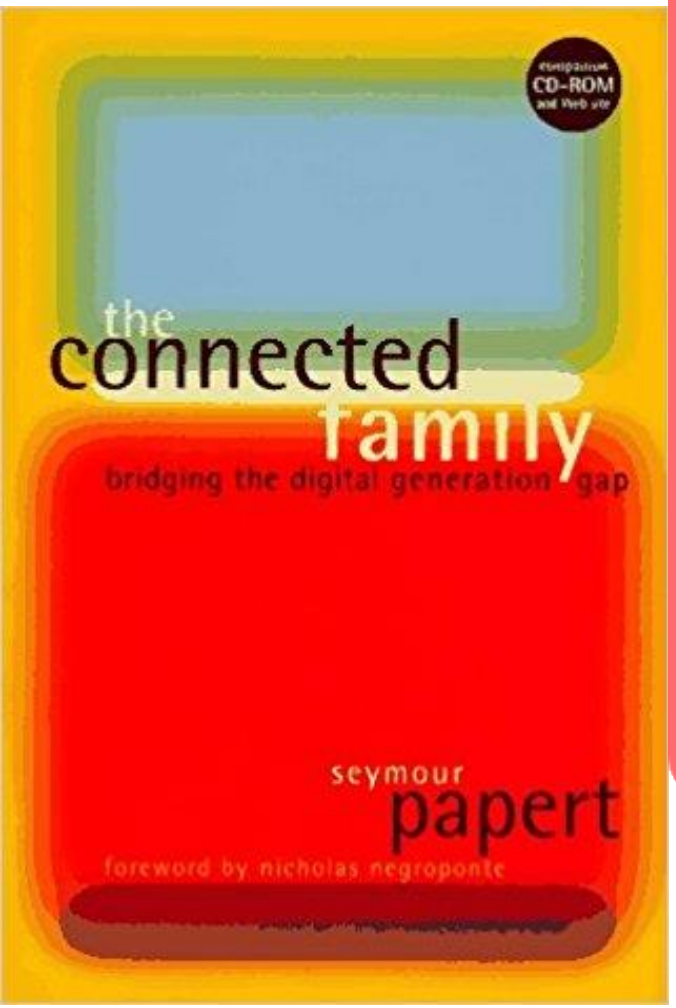


CT - Resources for all ages

- TaleBlaze
  - http://taleblazer.org/



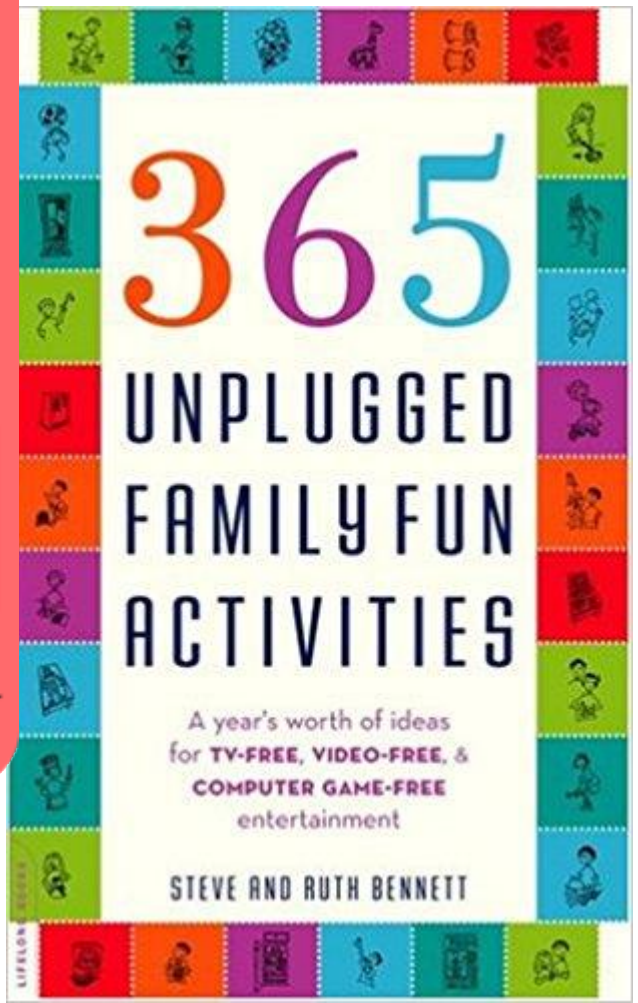
CT - Resources



Teaching Kids of All Ages How to Code

With and without a computer!

www.shareitscience.com



CT - Resources

- Beginner Developer Learning Center
  Bit & Bytes and Kids Corner, Microsoft
- LifeLong Kindergarten Mitch Resnick, MIT
- Great Principles of Computing Peter Denning, Naval Postgraduate School
- CodeAcademy
- Google for Education
- …

CT - Conclusions

- CT is recognized as vital to our students and our world's
- CT should be part of the curriculum in our schools and lives
- It's fundamental to prepare teachers to incorporate CT in their teaching practices
- CT is not easy to teach and assess
- What can we do to foster CT?
  - Talk about it
  - Implement it
  - Share ideas/experiences
  - Share resources









Portugal

Computational Thinking: from pre-school to higher education

Teaching

Learning

Projects

Resources

Anabela Gomes
anabela@isec.pt

Department of Informatics and Systems Engineering – Polytechnic Institute of Coimbra
&
Center of Informatics and Systems – University of Coimbra