Computing Computational Thinking using Computational Thinking Patterns

Authors: Kyu Han Koh et. al.

Presented by : Ali Anwar

Virginia Tech

ABOUT ME

B.Sc. Electrical Engineering, University of Engineering and Technology Lahore, Pakistan

M.Sc. Computer Engineering, University of Engineering and Technology Lahore, Pakistan

PhD. Computer Science, Virginia Tech.

- I am from Lahore, Pakistan
- Technical Lead Software Engineer, Mentor Graphics/CodeSourcey
- Tools for embedded software development
- Open-source software development mainly GDB

Lahore Fort

OVERVIEW

Recognizing Computational Thinking Patterns

Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning

Computing Creativity: Divergence in Computational Thinking

End-user game design to learn Computational Thinking

- End-user game designing is a motivator to learn computer science but do students learn computational thinking while designing a game?
- Programming games to creating science simulation using CT
- Computational Thinking Patterns
- Computational Thinking Pattern Quiz

What other medium is more suitable and why?

Why Games?

Educational Characteristics of Game Design:

1. Enables students to ***transfer*** their skills to science simulations and/or mathematical models
2. Is based on concepts that are easily ***recognizable*** a and ***usable*** by both instructors and students
3. Is automatically ***measurable*** for evaluation and progress tracking purposes.

Do you agree with the justification?

What are Computational Thinking Patterns?

"Computational Thinking Patterns are abstracted programming patterns that are learned by students when they create games and can readily be used by students to model scientific phenomena."

OR

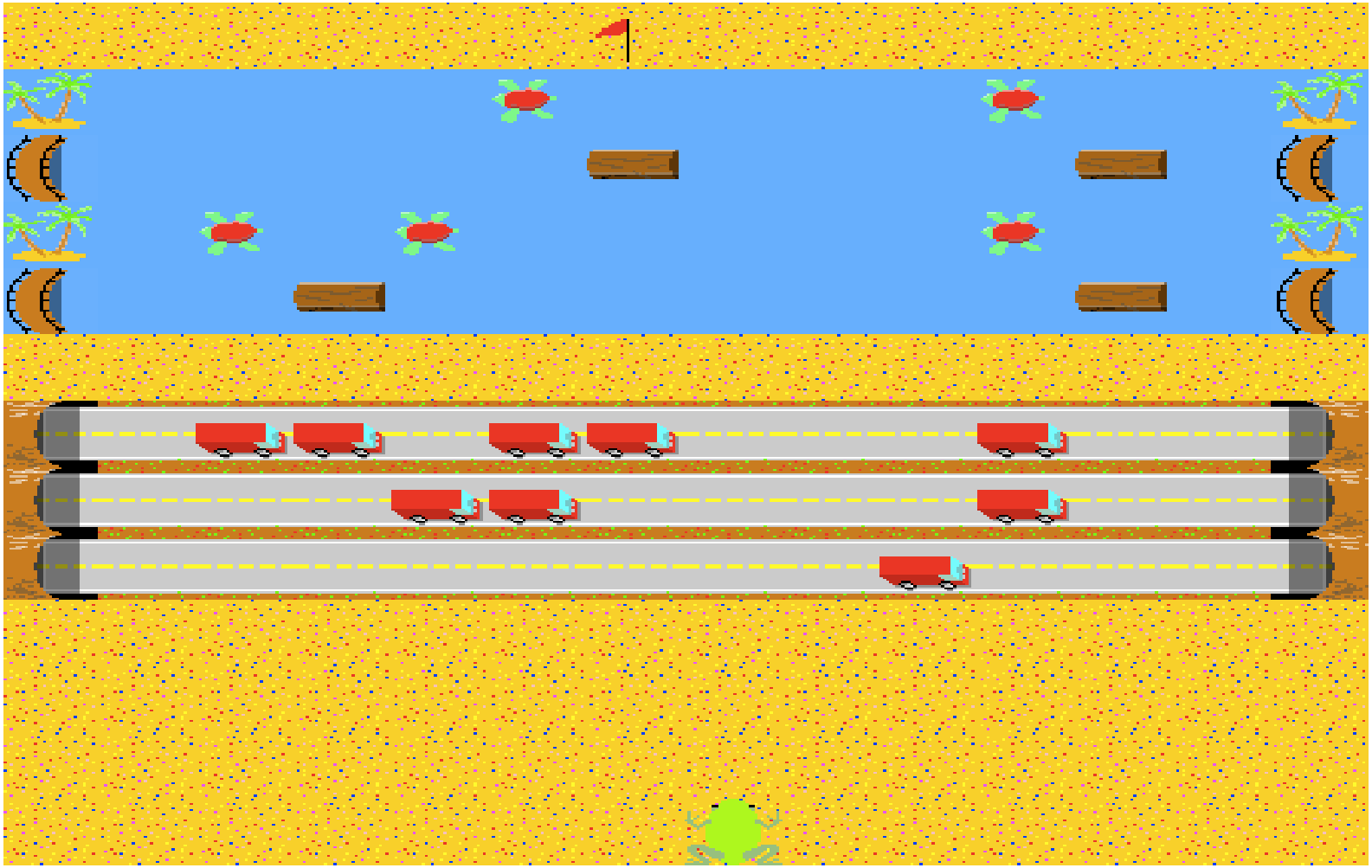
"Computational Thinking Patterns are abstract programming patterns that enable gent interactions not only in games but also in science simulations."

Different Computational Thinking Patterns

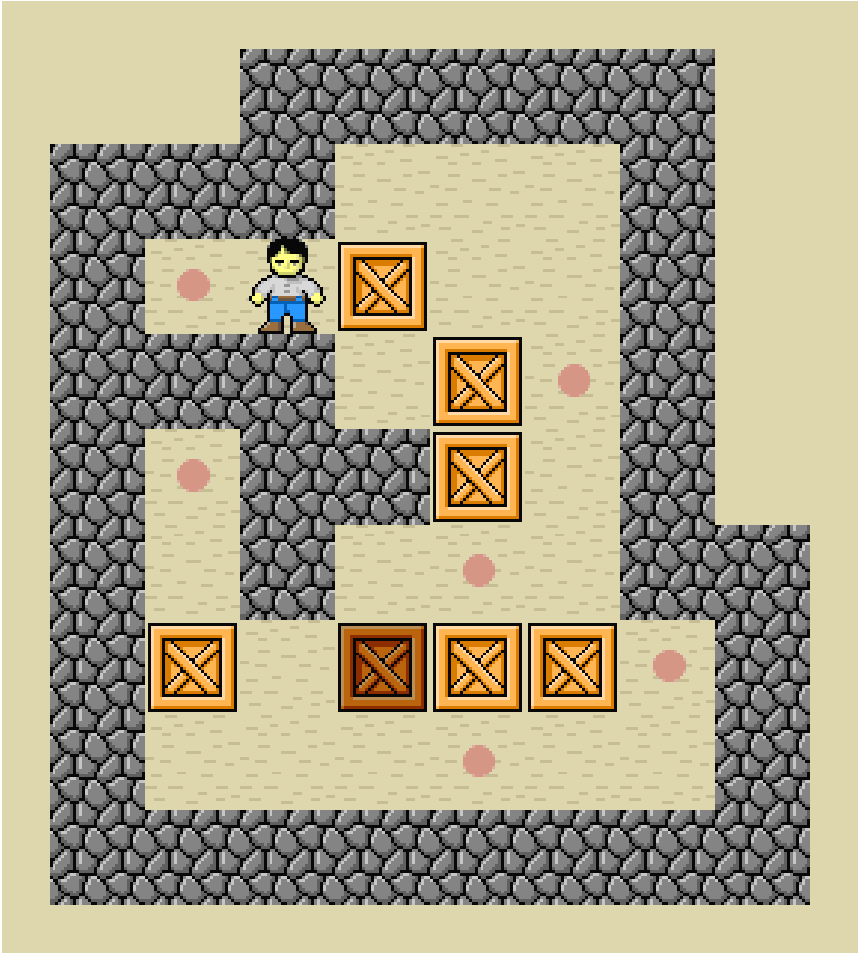
- The games used to extract the patterns are following:

1. Frogger
2. Sokoban
3. Centipede
4. Space Invaders
5. Sims

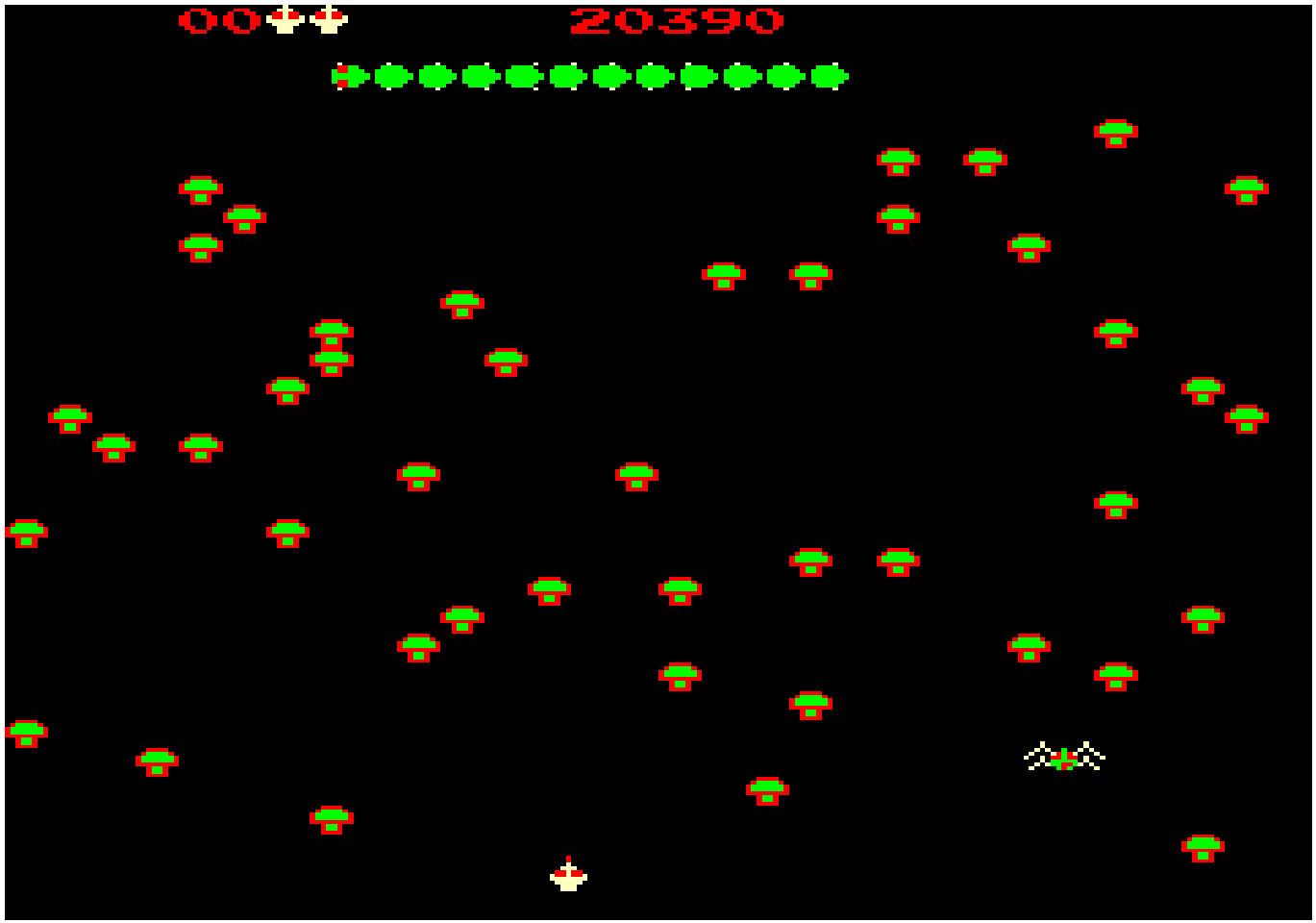
Frogger



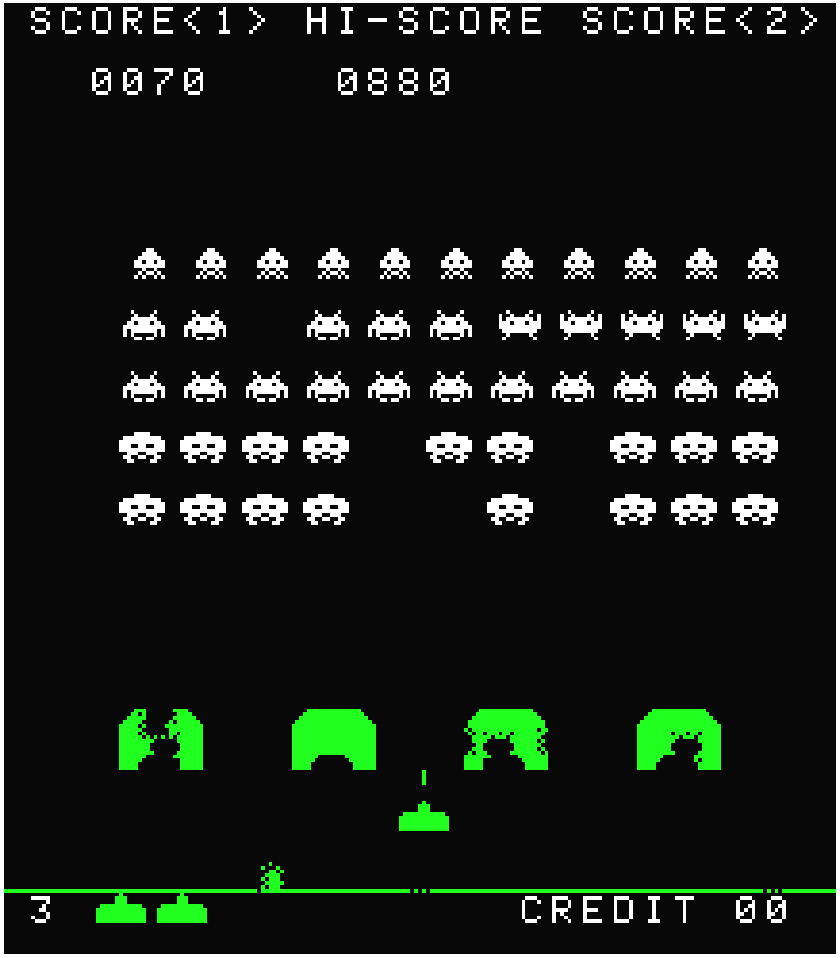
Sokoban



Centipede



Space Invaders



Sims

Computational Thinking Patterns

- ***Generation:*** To satisfy this pattern, an agent is required to create another agent; in real life, for example, raindrops emanate from clouds. Analogously, in predator/prey science simulations, animals breed to create new animals. Conversely, the ***Absorb*** pattern is when one agent deletes another agent.
- ***Collision:*** The collision pattern occurs when two agents physically collide. In real life, a car crashing into another car is an example of a collision. In science simulations atoms can collide with other atoms to make new elements.
- ***Transportation:*** In the transportation pattern, one agent carries another agent. In real life a car transports a person. In science simulations red blood cells transport oxygen molecules to parts of the body.

Computational Thinking Patterns

- ***Diffusion:*** Diffusion allows for the "scent" of an agent to be dispersed around a level. In real life, the scent of freshly baked bread originating from the kitchen is present in other rooms. In a science simulation diffusion can be used to depict how heat is transferred from one side of a heated metal bar to the other side.
- ***Hill Climbing:*** An agent employing a hill-climbing algorithm looks at neighboring values of interest and moves towards the one with the largest value. These values could be, for example, the "scent" of another agent. In real life, mosquitoes hill climb the smell given off by humans.

What other patterns can be added as far as game designing is concerned?
Are these Programming Patters?

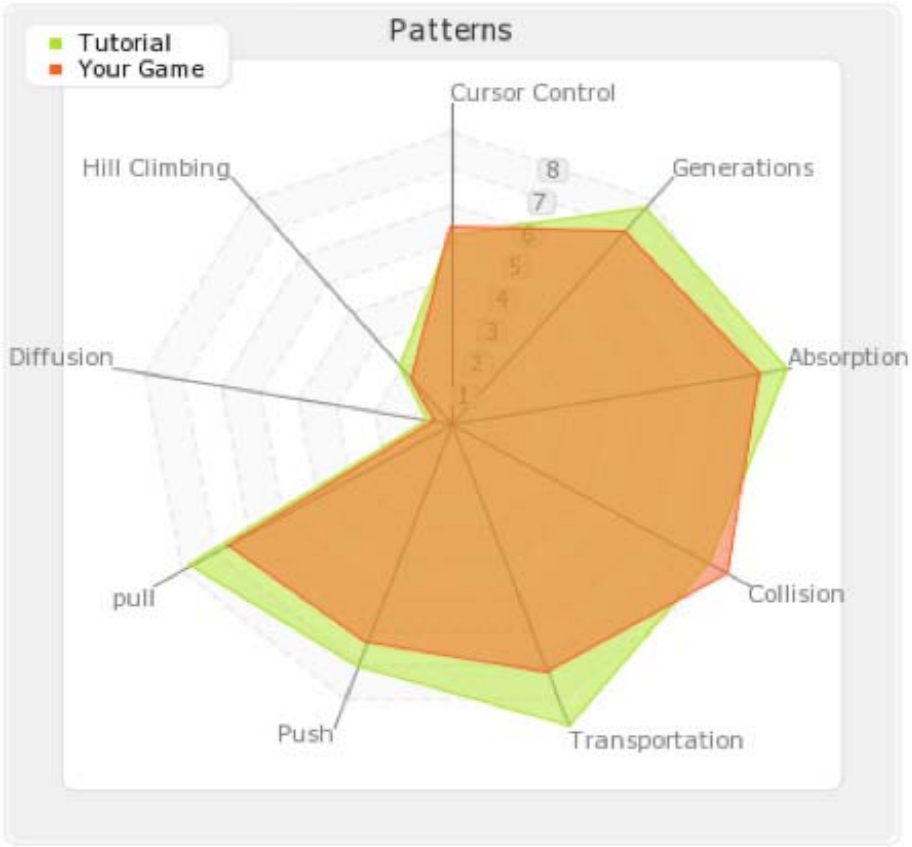
Important Note: These CT patterns are associated with game designing not generic Computation Thinking

Computational Thinking Pattern associated with each game

Games	Computational Thinking Patterns
Frogger	Generation, Absorption, Collision, Transportation
Sokoban	Push, Pull
Centipede	Generation, Absorption, Collision, Push, Pull
Space Invaders	Generation, Absorption, Collision
Sims	Diffusion, Hill Climbing

Why Sokoban does not have Transportation? Is agent not transporting boxes?

Computational Thinking Pattern Graph



The Computational Thinking Pattern Graph employs an approach similar to Latent Semantic Analysis to create a graph that depicts the Computational Thinking Patterns used to program a given game.

Computational Thinking Pattern graph. Depicts the Computational Thinking Patterns for a student's implementation of Frogger as compared to the tutorial's implementation

Computational Thinking Pattern Quiz.

Set of 8 questions, asked to identify the Computational Thinking Pattern involve in the activity:

1. Collision of two people sledding down the hill in following picture when compared with Frogger:

2. Marching band coming out of tunnel compared with Frogger.
3. Collision of two soccer players.

Computational Thinking Pattern Quiz.

4. Hot-dog eating contest compared with Pacman
5. Several football players chasing after a player with a football, Pacman
6. Video depicting one type of liquid being 'diffused' in another type of liquid and participants were asked to state how this was similar to Pacman
7. Video that depicts marathon runners running towards the finish line, compared with a specific simulation
8. A written paragraph that described a predator/prey simulation, and participants were asked to talk about all the computational Thinking Patterns they would use to create this simulation.

"This simulation involves the Predator Prey relationship between the Fox and the Rabbit. The Foxes find and eat Rabbits when they are hungry. Otherwise, Foxes will breed with other Foxes to create new Foxes. The Rabbits also breed with other Rabbits to create new Rabbits. Finally Rabbits, when hungry, seek out and eat grass."

Answers

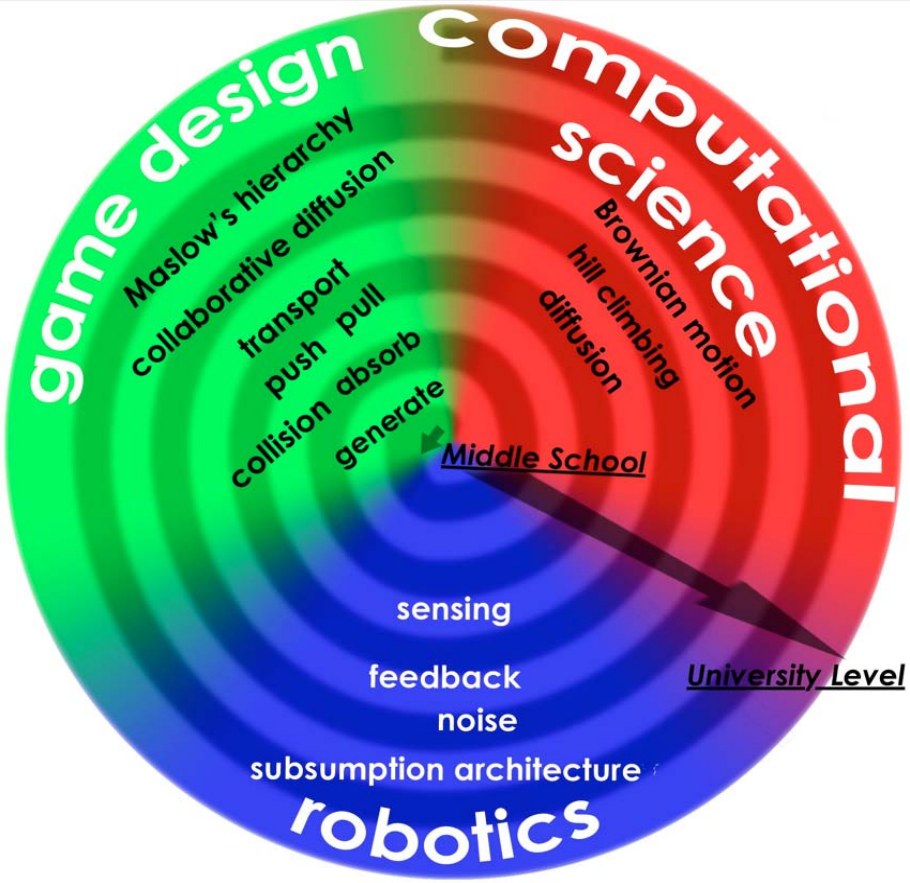
1. “The people are being transported by the tubes and the announcer is hit (collision) like the frog and the truck.”
2. “Generation of trucks, logs, turtles is similar to the tunnel generating people so to speak.”
3. “There is a collision with two different team members just as the car collides with the frog. . .”
4. “PacMan eats pellets and they erase, just like the hot dogs erase when they are eaten.”
5. “Both are seeking - the football players are seeking the player with the ball and the ghosts are seeking Pacman.”
6. “This shows the diffusion of the dye which represents the scent we assigned to pacman.”
7. “The runners are behaving like the ants after they have located some food. They are all heading in the same general direction as fast as they can.”
8. "Foxes and Rabbits use DIFFUSION/HILL CLIMBING in order to find their food sources, or you can have it be based on random movement. Foxes and rabbits will GENERATE new versions of themselves."

Results

	Q1 (1)	Q2 (1)	Q3 (1)	Q4 (1)	Q5 (1)	Q6 (1)	Q7 (1)	Q8 (4)
Participants	1	0.929	.881	.952	.976	.951	0.846	3.14

- "Computational Thinking Pattern Quiz is a good first step towards evaluating if students recognize what they learn from game programming as well as validating the usefulness of Computational Thinking Patterns themselves." Is it?
- Computational Thinking has occurred is based on whether students are able to transfer the knowledge they gained from game programming to science simulations?
- Use of Computational Thinking Patterns as the specific units of transfer between games and science simulations?

Computational Thinking Pattern Spiral

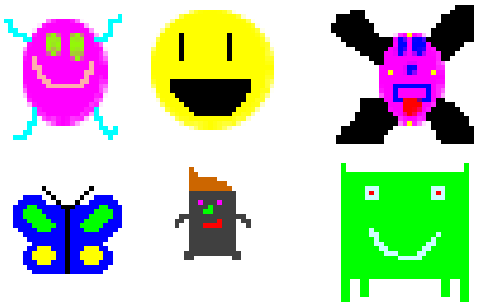


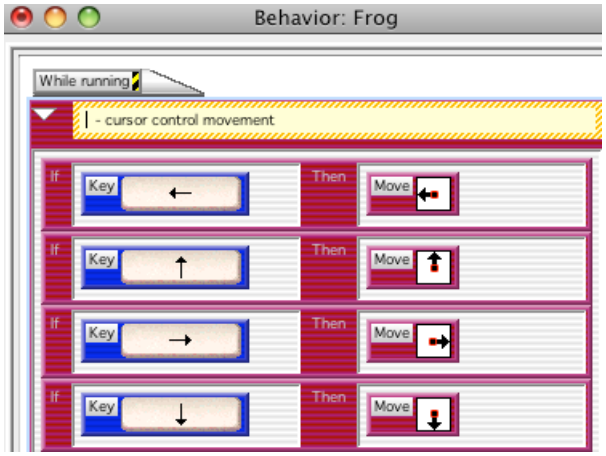
Is it not possible to learn transportation without learning push pull?

- A collection of computational thinking patterns specifying common object interaction that can be found in a number of domains including game design, computational science and robotics.
- Iterative approach to introduce and connect these concepts. For instance, random movement in game design is conceptually similar to Brownian movement in physics.
- Computational thinking patterns such as the collision of objects to highly advanced ones such as Maslow's hierarchy of needs
- Implies increased connectivity among the three computer science areas of robotics, computational science and game design.

Automatic Semantic Evaluation Tool

- AgentSheets programs consist of user created "agents," which are the game characters.
- All behaviors in AgentSheets are implemented using "If/Then" conditional statements. AgentSheets enables the use of 16 different conditions and 23 different actions, in combination, to create behaviors for any given agent.
- With the 23 conditions and 16 actions, it is possible to represent each game as a vector of length 39, wherein each element of the vector represents how many of each individual conditions and actions are used to implement a given game. Using these vectors, any game created in AgentSheets can be compared to any other game through a high dimensional cosine calculation for similarity.





Program Behavior Similarity

Equation to calculate similarity:

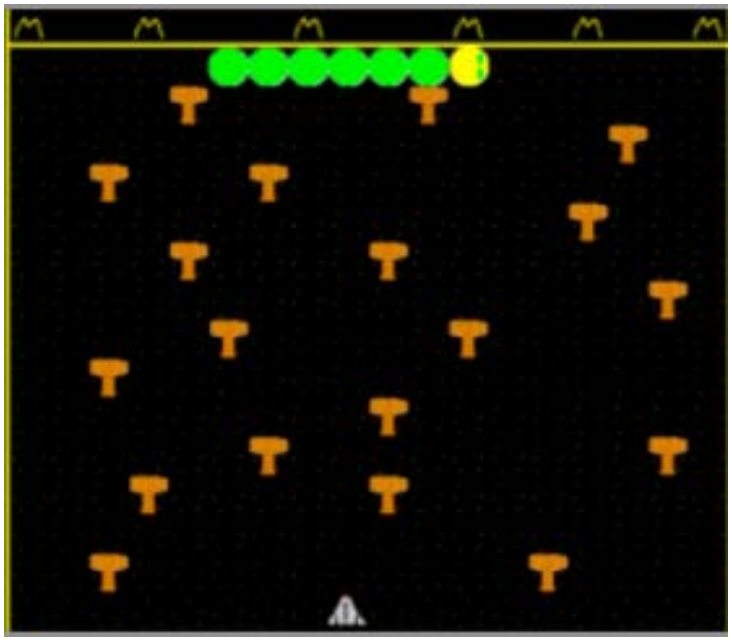
Only advantage of this technique, automat-ability?

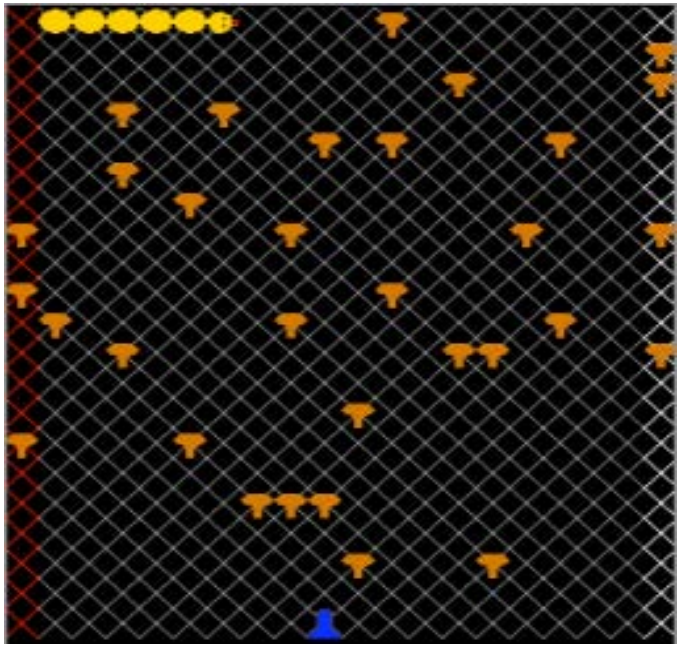
$$PBS(u,v) = \frac{\sum_{i=1}^{i=n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \cdot \sqrt{\sum_{i=1}^{n} v_i^2}}$$

The Program Behavior Similarity (PBS) is obtained by finding the angle in-between of two n-dimensional vectors, u and v.

The high dimensional cosine similarity comparison of games is robust to two games having the same proportion of rules, but having these rules in differing numbers. In such cases, a syntactic analysis would categorize the games as different.

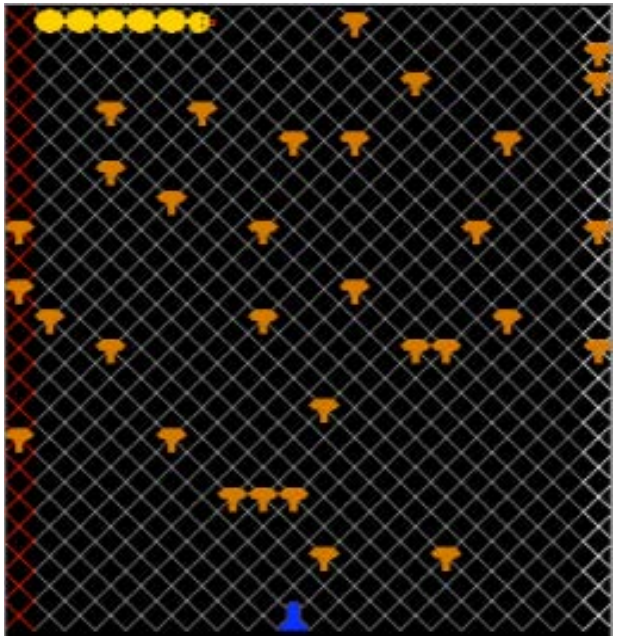
Similar Games

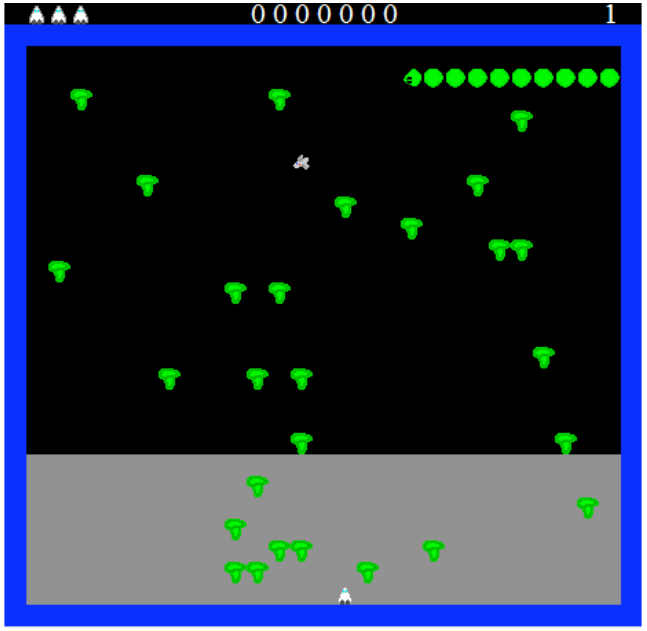




Two similar Centipede Games with a similarity score of 0.89

Dissimilar Games

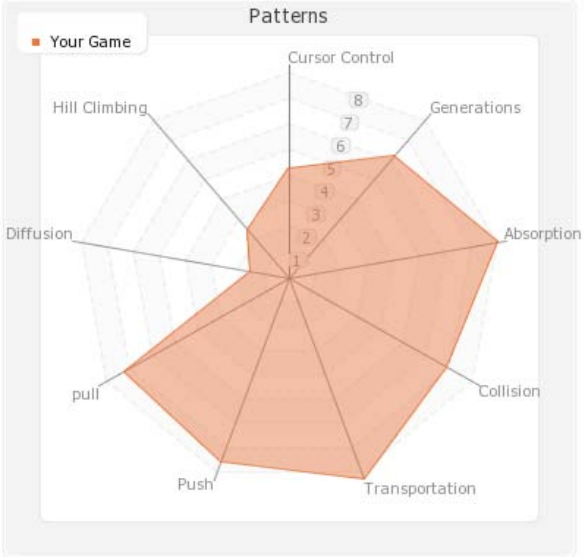


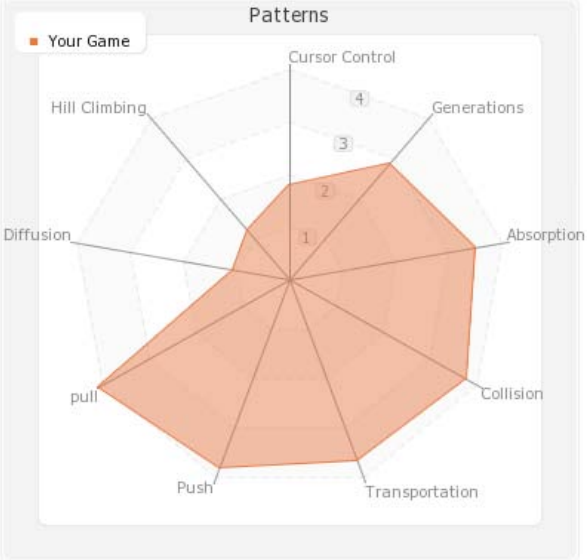


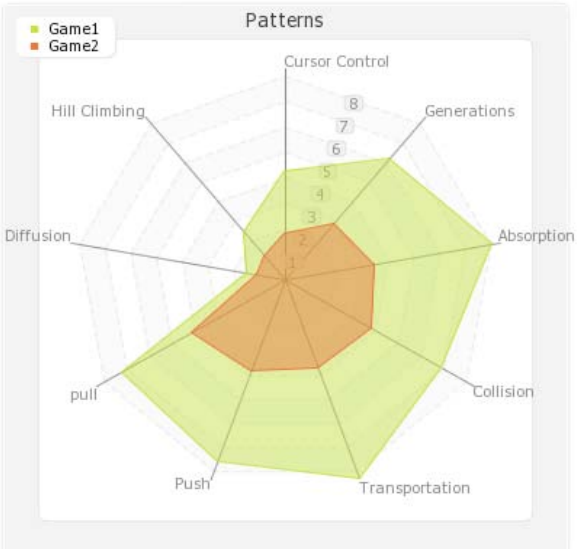
Two Centipede Games with a low similarity score of 0.43 (Centipede A: Left, Centipede B: Right)

Structure of Centipede A and B

	Centipede A	Centipede B
Number of Agent Classes	8	19
Number of Depictions	13	35
Number of Methods	26	38
Number of Rules	107	129







Calculating Divergence to measure Creativity

$$Divergence(x) = \frac{\sqrt{\sum_{i=1}^{n}(u_i - v_i)^2}}{\sqrt{n}}$$

Is it an appropriate way to measure creativity?

Vector A = (0.525, 0.557, 0.432, 0.641, 0, 0.687, 0.721, 0, 0.197)

Vector B = (0.373, 0.499, 0.679, 0.623, 0.096, 0.455, 0.51, 0, 0.106)

divergence score of the given game = 0.15

The student-submitted game and the tutorial can be represented as nine dimensional vectors respectively (0.525, 0.557, 0.432, 0.641, 0, 0.687, 0.721, 0, 0.197) and (0.373, 0.499, 0.679, 0.623, 0.096, 0.455, 0.51, 0, 0.106). The difference of those two vectors is (0.152, 0.058, -0.247, 0.018, -0.096, 0.232, 0.211, 0, 0.091). The normalized (divided by the value of rooted n) length value of that vector is 0.15, and this is the value of divergence score of the given game.

Game Dimensions and Creativity

- Agents

The characters or agents in AgentSheets [14], make up the entire game worksheet.

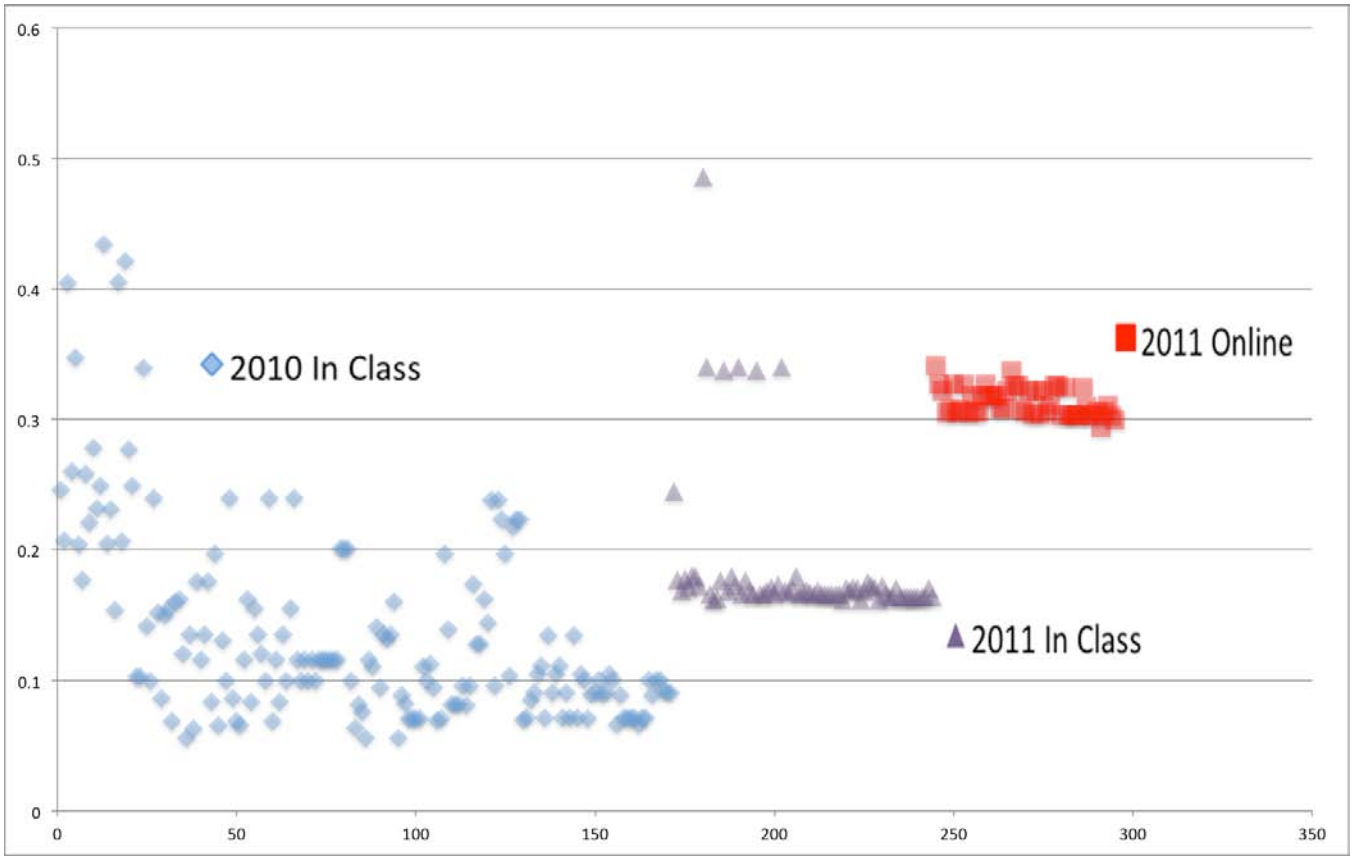
- Levels

The game level sequence, as well as the difficulty of the levels can show a students' creativity or divergence from the tutorial "norm."

- Behavior

The programming that students create, determines the behavior of characters, and is the most complex aspect of the game-design process.

Three different classes



Using Scalable Game Design Arcade (SGDA)

Scattered Divergence Calculation Graph: X-axis represents time by order of submission. Y-axis represents Divergence Score. Each dot means individual submission. 296 Frogger games are displayed in this graph.

Divergence Calculation Score in Each Class

Divergence Score	Standard Deviation	Average
In Class 2010	0.074	0.135
In Class 2011	0.057	0.186
Online 2011	0.011	0.314

"We conjecture that not only is the revised tutorial a significant factor in the represented divergence between class conditions, but that the in-class/online condition comparison also appears to be a significant factor affecting the divergence calculation for at least two class conditions, effecting calculated creativity."

DISCUSSION

- Is creativity measureable?
- If yes, what other approaches we can use to measure the creativity?
- Can we automate Computational Thinking assessment?
- What other ideas do you have?
- Is it possible to use this approach in classroom?

Thank You