Recognizing Computational Thinking Patterns

Ashok Basawapatna
University of Colorado Boulder
Department Of Computer Science
Boulder, CO. 80303
(720) 838-5838, 001
basawapa@colorado.edu

Kyu Han Koh
University of Colorado Boulder
Department of Computer Science
Boulder, CO. 80303
(303) 495-0357, 001
kohkh@colorado.edu

Alexander Repenning
University of Colorado Boulder
Department of Computer Science
Boulder, CO. 80303
(303) 492-1349, 001
ralex@cs.colorado.edu

David C. Webb
University of Colorado Boulder
School of Education
Boulder, CO. 80303
(303) 492-0306, 001
dcwebb@colorado.edu

Krista Sekeres Marshall
University of Colorado Boulder
School of Education
Boulder, CO. 80303
(303) 999-5736, 001
krista.marshall@colorado.edu

ABSTRACT
End-user game design tools are effective in motivating and exposing students with no prior programming experience to computer science. However, while there is good evidence that these environments are effective motivators, the question remains what do students actually learn? For our purposes, using AgentSheets, we would like to know if students can apply the knowledge obtained from programming games to creating science simulations. Specifically, we want to better understand if students are able to recognize *Computational Thinking Patterns* (CTP) from their game programming experience. Computational Thinking Patterns are abstract programming patterns that enable agent interactions not only in games but also in science simulations. Students and teachers who participated in a game design summer institute were administered a Computational Thinking Pattern Quiz (CTP Quiz). This quiz tested the participants' ability to recognize and understand patterns in a context removed from game programming. We found that participants, for the most part, were able to understand and recognize the patterns in a variety of contexts.

Categories and Subject Descriptors
K.3.2 Computer and Information Science Education

General Terms
Algorithms, Design, Experimentation, Human Factors.

Keywords
University Programming Education, Middle School Computer Education, Scalable Game Design, Computational Thinking, Computational Thinking Patterns, Transfer, Student Observation.

SIGCSEi11, March 9-12, 2011, Dallas, Texas, USA.

1. INTRODUCTION
Currently end-user video game design is a popular way to teach programming to students who have little or no prior programming experience [1,2,3]. The allure of using video games to introduce students to programming partly lies in students' natural interest in video games, but also, in that certain game design tools allow students to create games relatively quickly as compared to conventional programming languages [4]. To put this in perspective, a student using AgentSheets, the end user rapid game prototyping environment we employ, can make their first 'Frogger' game within 5 hours [5].

Much research has shown the teacher effectiveness of using end user game creation as a motivational tool to get students interested in Computer Science [4,6]. There has been less research, in general, on the actual knowledge/skills students acquire. For example, implicit in these studies is the idea that students learn "Computational Thinking" [7]. However, the definition of Computational Thinking at the present time is abstract at best [8]. For this method of instruction it is essential we concretely define what exactly we expect students to learn. In other words, for Computational Thinking to become a notion that is actionable, teachers require more than just abstract definitions of what Computational Thinking is or is not.

A different way to think about Computational Thinking is to move beyond its definitions towards a more pragmatic conceptualization. In one of the many schools participating in the Scalable Game Design project, one teacher came up with an interesting idea. After glancing at some of the current definitions of Computational Thinking, he indicated that he still did not quite understand what Computational Thinking really was, but he had an expectation. He would want to be able to walk up to a student participating in game design and ask:

"Now that you can make Space Invaders, can you also make a science simulation." [9]

To put it another way, the teacher's expectation is that the student should be able to use their programming knowledge to solve real world problems. The ability to create scientific simulations should

be an important benefit of thinking computationally and an indication of STEM proficiency.

For our purposes, any useful educational benefit obtained through end-user game design would have the following properties:

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.

The above specifications initially motivated us to look at the elements of game programming that enable transfer to science simulations. This led us to define Computational Thinking Patterns, which are abstracted programming patterns that are learned by students when they create games and can readily be used by students to model scientific phenomena [5]. For example, in Frogger, students must program a truck-frog collision; this situation, wherein two agents physically collide, can be used to model molecules colliding or even a car crash. The following table shows some example games and their corresponding Computational Thinking Patterns. It should be noted that these patterns are a work in progress and this table is by no means comprehensive.

Table 1 Games and their corresponding Computational Thinking Patterns [5]

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

The following is a brief description of selected Computational Thinking Patterns and how they relate to game programming and STEM simulation design. For a more in-depth description of these and other patterns please see [5].

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.

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.

One possible method of evaluating Computational Thinking in the classroom, in the context of end user game programming, is evaluating the ability of teachers and students to understand and make use of these Computational Thinking Patterns. Previous research has indicated that students do indeed use these Computational Thinking Patterns to implement science simulations of their own [5,11]. Further research into automatically recognizing which patterns a student has used to implement a game/simulation yielded an analysis called the Computational Thinking Pattern Graph [10]. 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 [12]. Basically, the underlying code of a game or simulation is compared to canonical Computational Thinking Patterns coded in the same programming language. For instance, the dimension of collision indicates if there is some code exhibiting a similar pattern to a canonical implementation of two object colliding. The following figure is a Computational Thinking Pattern graph of a student's Frogger implementation as compared to the tutorial Frogger.

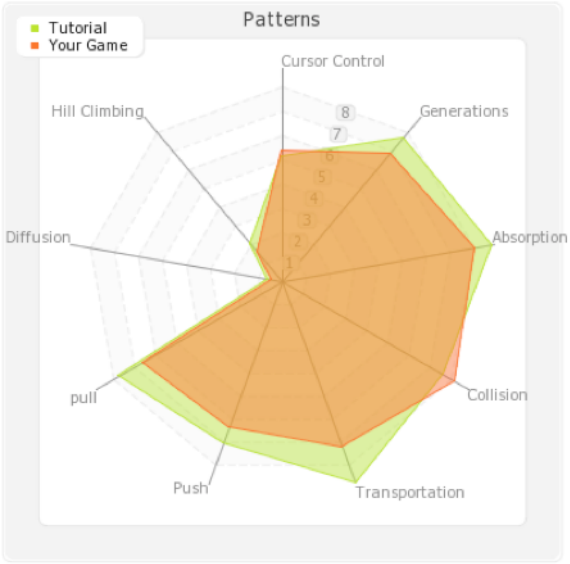

Figure 1: Computational Thinking Pattern graph. Depicts the Computational Thinking Patterns for a student's implementation of Frogger as compared to the tutorial's implementation.

This graph is one way to visually evaluate the Computational Thinking Patterns students use in a game. For example, using

Figure 1, a teacher could readily see that, to a large extent, the student was able to successfully implement all the Computational Thinking Patterns necessary for Frogger. Therefore, the graph seems to indicate the student has met a high level of proficiency in the end user game design task; this relates to point 3 of the above Educational Characteristics of Game Design. Furthermore, previous research has indicated this transfer between games and simulations seems to occur [5,11]. This relates to point 1 of the Educational Characteristics of Game Design. However, the question still remains whether the user can now recognize and re-implement these patterns in the context of a STEM simulation.

To put this another way, though the Computational Thinking Pattern graph shows that teachers can teach and students can implement these patterns, a real educational question arises as to whether the students and teachers truly understand what is being learned. In order for students to implement and teachers to teach these Computational Thinking Patterns in another context, they must first be able recognize and understand these patterns. This relates to point 2 in the above Educational Characteristics of Game Design. For example, given that a class has finished the Frogger unit, are the students able to recognize that to model a car crash simulation they would employ the same collision pattern that they used in Frogger? Specifically, the research question we are asking is "To what extent are these Computational Thinking Patterns accessible and useful to teachers teaching game design and students learning game design." To help gain better insight into this question, we developed the *Computational Thinking Pattern Quiz*.

The Computational Thinking Pattern Quiz starts with several videos that depict one or a combination of Computational Thinking Patterns. The last question of the Computational Thinking Pattern Quiz is a paragraph specification of a given science simulation, and game designers are asked to list the Computational Thinking Patterns that should be used to implement this simulation. The videos and paragraph description questions allow us to see if game designers can, and to what extent, recognize various Computational Thinking Patterns outside the context of game programming. Therefore, the Computational Thinking Pattern Quiz could be an important aspect of evaluating what students actually learn from end-user game programming as well as method of evaluating the usability of Computational Thinking Patterns themselves.

2. METHOD

In the summer of 2010 a National Science Foundation-funded 2-week Summer Institute was held at the University of Colorado, Boulder. The goal of this Institute was to present to middle school teachers and community college students methods of using game design to teach computer science and Computational Thinking in middle school classrooms across the state of Colorado, and sites in Texas, Alaska and South Dakota. Specifically, this Institute was held as part of NSF funded iDREAMS Scalable Game Design project which has, thus far, educated over 2,000 students in Colorado.

There were 2 groups that participated in the Summer Institute. Group 1 consisted of participants who had not participated in the 2009 Summer Institute. This group attended the Summer Institute for the entire 2-week session. Group 2 was made up of participants who had experienced the prior Summer Institute in 2009, therefore, they were only present for the second week session of the Summer Institute. Group 2 had completed everything Group 1 did in a previous Summer Institute and thus, by week 2, both groups were at comparable levels of expertise. Teachers in both Groups 1 and 2 came from diverse backgrounds; for example, participant teachers taught everything from computer science to Spanish to accounting. Moreover, community college students were also from diverse backgrounds such as computer science, creative writing, art, and philosophy.

The curriculum for the Summer Institute exposed all the participants to multiple Computational Thinking Patterns. In week 1, participants programmed Frogger, Sims, and Space Invaders (refer to Table 1 to see the Computational Thinking Patterns implemented in these games). In week 2, the participants implemented two science simulations and the game Pacman (which uses the patterns diffusion and hill climbing).

At the end of the Institute, all participants were given an 8 question Computational Thinking Pattern Quiz. The first 7 questions involved video of real-life phenomena relating to patterns participants programmed in previous games. The final question of the quiz was a paragraph description of a predator/prey model and asked the participants to list and describe all the Computational Thinking Patterns necessary for implementation. The following section covers these 8 questions more in-depth.

2.1 Computational Thinking Pattern Quiz

As mentioned above, the Computational Thinking Pattern quiz has 8 questions, 7 of which are video of real life phenomena that resemble in-game Computational Thinking Patterns and one of which is a specification for a predator/prey simulation. The questions were administered online and the 43 participants were given an hour to complete the quiz. The descriptions of the questions are as follows.

Question 1 is a video of 2 people sledding down a hill. At the end of the video one of the sledders collides with a person standing at the bottom of the hill. The patterns depicted in this video are 'transportation' and 'collision' and the participants were specifically asked how this related to something in the "Frogger" game. Figure 2 depicts sequential screenshots of this video as an example as to what these videos looked like. Participants were given credit if they correctly identified either one of the patterns.

Figure 2 Sequential screenshots from question 1 of the Computational Thinking Pattern Quiz wherein 2 sleds 'transport' 2 people down a hill and one of the sledders 'collides' with a person at the bottom of the hill.

Question 2 depicted a marching band coming out of a tunnel and again asked participants to say how this was similar to the "Frogger" game they programmed (i.e.: the tunnel in the "Frogger" game 'generates' trucks).

Question 3 depicts a 'collision' between two soccer players. Again participants are asked how this is similar to something they programmed in "Frogger" (i.e.: when the truck hits the frog).

Question 4 depicts a hot-dog eating contest and asks the participants how this is similar to something they programmed in "Pacman" (i.e.: the "Pacman" eats or 'absorbs' the pellets)

Question 5 depicts several football players chasing after a player with a football and the participants are asked how this relates to something they programmed in "Pacman" (ie: the ghosts 'hill climbing' of Pacman's 'diffused' scent resembles the players running after the person with the football). As for Question 1, which also depicted 2 patterns, describing either pattern yielded a correct answer.

Question 6 is a 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" (ie: Pacman's scent is diffused around the level so the ghosts can track Pacman).

Question 7 is a video that depicts marathon runners running towards the finish line and participants were asked to describe how this resembles something in the Ant Foraging simulation they used earlier in the week (ie: runners running towards the goal is visually similar to ants following a pheromone scent to food and/or taking food back to the nest).

Question 8) As mentioned above, Question 8 was 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. The specification of the simulation was as follows:

"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."

Based on what participants experienced at the Summer Institute, we would expect the following patterns to be identified as necessary to complete the simulation:

- Generation because the animals breed creating new animals.
- Absorption because the foxes eat (or absorb) rabbits and the rabbits eat grass.
- Diffusion because the rabbits diffuse a scent around the level.
- Hill Climbing because the foxes follow the rabbit's diffused scent and the rabbits seek out grass when hungry.

Unlike Question 1 and Question 5, in Question 8 participants were specifically asked to list all the patterns they thought would be necessary to implement the predator/prey paragraph description. Therefore, Question 8 was out of 4 points.

Participant answers to all questions were given in paragraph long-answer format, and the answers were categorized and coded. As mentioned above, participants were awarded a correct answer if they named or described an intended Computational Thinking Pattern for a particular question, with the exception of Question 8 which was out of 4 correct answers.

3. RESULTS

Table 2 depicts the average scores for each question. In general, the results show that participants were able to understand and recognize Computational Thinking Patterns in a different context; the average score for all questions was over 78%, and for the first 7 questions, the average score was over 84%. This is remarkable since the participants came from diverse backgrounds and had minimal instruction on Computational Thinking Patterns before they took the quiz.

The nature of responses took two main forms. Some participants named the specific Computational Thinking Pattern that was shared between both the game and the Computational Thinking Quiz video. Other participants actually described the interaction in the video that was similar to the patterns. Both were deemed correct answers. The specific results are as follows.

The first three question refer to how the videos (described above) resemble something the participants had seen in Frogger.

Question 1 depicted two people sledding down a hill and with a sledder collision at the bottom. Surprisingly, every participant got one of the two Computational Thinking Patterns leading to every participant getting the question right. Some participants even wrote both Computational Thinking Patterns though it was specifically not required by the question. This is not too surprising since collision and transport are both taught in Frogger, the first game everyone learns (and the one that is most taught by participants in their respective schools). The following answer is indicative of a correct answer to both patterns:

"The people are being transported by the tubes and the announcer is hit (collision) like the frog and the truck."

Question 2 was a video of a marching band coming out of a tunnel. As with Question 1, the participants correctly identified this pattern as the average score was 93%. The following correct answer is representative of how many participants described the game/video similarity:

"Generation of trucks, logs, turtles is similar (sic) to the tunnel generating people so to speak."

Question 3 depicted a scene from the 2004 World Cup where one player infamously head-butts another player. 88% of the participants correctly identified the Computational Thinking Pattern. Though many people described this pattern, fewer people were able to correctly name or describe this Computational Thinking Pattern than in prior questions. The reason for this might lie in the fact that in the video one player head-butts another player in the context of a soccer game which could lead to participants adding information from their own experiences or knowledge of, in this case, soccer to answer the question. For example, a typical incorrect answer looked as follows.

Table 2 Average score of participants for each question in the Computational Thinking Quiz. The numbers in parenthesis denote the total possible points for a given question

	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

"It could be either the guy getting a red flag . . . or a third agent keeping track of a loss."

The participant refers to aspects of a soccer game (red card cheating vs. cheating in Frogger and referee scoring) that are not explicitly shown in the video. Everything this participant alluded to is correct, but is not the simple 'collision' pattern we were looking for in this case. Ambiguity brought about by the implied video context is a shortcoming of the Computational Thinking Pattern Quiz. The typical correct answer resembled the following.

"There is a collision with two different team members just as the car collides with the frog. . ."

Questions 4 through 6 deal with how the videos are similar to the game Pacman. The answers to this portion of the quiz were for the most part all correct; this might be due to the fact that Pacman was one of the most recent games the participants had programmed so it was fresh in their minds.

Question 4 depicted a hot dog eating contest. This question was answered correctly by 95% of the participants; most people seemed to understand that hotdogs being eaten by a person was similar to how Pacman absorbs the pellets. This is one of the main patterns in Pacman. Furthermore, absorb is a simple pattern that is also shared in other games such as Frogger when a truck is absorbed by a tunnel at the end of the road; thus, participants were very familiar with this particular pattern. A typical answer for Question 4 looked as follows:

"PacMan eats pellets and they erase, just like the hot dogs erase when they are eaten."

Question 5 was a video of a football player being chased by other football players. This question had 2 answers, if the participants answered or described hill climbing or diffusion they were awarded a correct answer. Similar to Question 4, everybody basically answered Question 5 correctly (98%). An indicative correct answer given by participants is as follows.

"Both are seeking - the football players are seeking the player with the ball and the ghosts are seeking Pacman."

Question 6 examines the similarity between one liquid diffusion into another liquid and how Pacman's scent is diffused across a level. Unlike the liquid, Pacman's diffused scent is invisible making the contextual leap more challenging. However, participants still uniformly understood the connection between this Computational Thinking Pattern of diffusion in Pacman and the diffusion of one liquid into another as 95% of participants got this question right. The following is a common participant answer we came across

"This shows the diffusion of the dye which represents the scent we assigned to pacman."

Question 7 referred to an Ant foraging simulation that the participants modified.

Question 7 depicted marathon runners running on a path towards the finish line much like the ants in the ant simulation followed pheromones to food and followed an implicit path back to the nest once finding food. Of the first 7 questions this is the one that participants had most trouble with as only 85% of participants got it right. As with Question 3, this question might have suffered from ambiguity relating to the video as participants may not have understood the significance of a person travelling on a path (something they see in everyday life and possibly deem as not notable). For example, the following incorrect answer was given by one participant:

"This is similar to the ant simulation because they both have a large number of similar "agents" moving around quickly."

This answer completely disregards any notion of agents moving on a path in favor of how the video matches the overall aesthetic of the ant simulation. In contrast, the correct answers all noticed the trail aspect of the video. A common correct answer participants provided is as follows:

"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."

Note that the second answer is somewhat similar to the first answer but focuses on the idea of seeking which involves the diffusion and hill climbing Computational Thinking Patterns.

Question 8 deals with the specification of a predator/prey science simulation. As mentioned above, the patterns needed to correctly implement this simulation are generate, absorb, hill climbing, and diffusion. Of all the Computational Thinking Quiz questions, we thought Question 8 would be the most challenging and the most indicative of whether participants could actively recognize and transfer the Computational Thinking Patterns they learned in prior game development to a science simulation. Surprisingly, the results were positive. 20 of the 43 participants were able to name or describe all 4 computational thinking patterns. An excerpt representative answer from this group is as follows

"You would use collaborative diffusion, the fox will hill climb to find a rabbit, the fox will then absorb the rabbit. . . When one (of the) foxes are stacked on another fox it will generate a new fox."

An additional 13 participants were able to correctly identify 2-3 correct patterns. Many of these participants missed the less complex patterns such as absorb and generate while correctly pointing out the more sophisticated patterns of hill climbing and diffusion. A typical answer from this group is as follows:

"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 when they're with another one of their species, also found by either diffusion or random movement. I would make the rabbits/foxes give off a low level of scent for "heat" . . ."

It is hard to say why this would be the case. A possible reason might be that since absorb and generate are simpler patterns, they are easier to overlook. 9 participants were able to name one Computational Thinking Patterns and only 1 participant was not able to name any. Given the performance on prior quiz questions, it could be that these participants were able to correctly recognize the patterns in different contexts visually but were not yet at the level where they could recognize the patterns from a more abstract English description.

Though participants did well in the Computational Thinking Pattern Quiz, there are a few issues with using this as a method of evaluation for Computational Thinking Patterns. The first, as referred to in Question 3 and Question 7, is the possible video ambiguity as to the specific Computational Thinking Pattern they would use to program a specific phenomenon. Sometimes participants added their own knowledge to the context of the video. Other times participants referred to shared actions that happened both in the video and the previous game they

programmed; however, these were not at the generalized level of Computational Thinking Patterns, but rather, at a more specific implementation level or more of an aesthetic similarity between the video and the previously programmed game. In Question 8, wherein participants were given a specification and asked to list the Computational Thinking Patterns they would use for implementation, participants sometimes overlooked some of the more obvious patterns to describe the more complex patterns involved. It could be that if these participants actually programmed the predator/prey simulation they would, in fact, readily recognize and implement all the patterns correctly as the missing patterns would be made more explicit (i.e.: grass would not disappear when eaten by rabbits leading them to implement the absorb pattern). Even with these problems, the 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.

The Computational Thinking Quiz results imply that in 1 to 2 weeks time the diverse participants in the Summer Institute were able to recognize Computational Thinking Patterns in a variety of different contexts. The fact that middle school teachers and community college students could relatively easily pick up on these patterns helps to support point 2 in the Educational Characteristics of Game Design above wherein we ask if the educational elements are readily recognizable and understandable by teachers to students. This coupled with prior research showing that Computational Thinking Patterns can be used for automatic evaluation and that students tend to transfer these patterns when creating simulations, indicates that Computational Thinking Patterns are one way to measure the educational benefit of end-user game design.

4. CONCLUSION

Claiming an educational benefit through end-user game design necessitates the ability for students to gain tangible and measurable Computational Thinking skills in order to be useful in the classroom. For our purposes, our expectation of whether Computational Thinking has occurred is based on whether students are able to transfer the knowledge they gained from game programming to science simulations. We define Computational Thinking Patterns as the specific units of transfer between games and science simulations. In this paper, we show that Computational Thinking Patterns are readily recognizable and understandable by teachers and community college students across different contexts, which is an important step in showing the usefulness of Computational Thinking Patterns in the classroom.

Future research will look at other ways to make Computational Thinking Patterns more explicit in the programming process, identify more useful Computational Thinking Patterns, and develop further evaluations of student learning and methods of evaluating the usefulness of patterns themselves. We will administer the Computational Thinking Quiz to over 2000 middle school students during the 2010-2011 school year. Results from the quiz will further inform this research.

5. ACKNOWLEDGMENTS

This material is based in part upon work supported by the National Science Foundation under Grant Numbers No. 0833612 and DMI-0712571. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. A special thanks to Andri Ioannidou for all her help and invaluable advice.

6. REFERENCES

[1] Cooper, S., Dann, W., Pausch, R., Cooper, S., Dann, W., and Pausch, R., Teaching Objects-first In Introductory Computer Science, In Proc. SIGCSE 2003, Reno, Nevada, USA, 2003

[2] Peppler, K. and Kafai, Y. B., Collaboration, Computation, and Creativity: Media Arts Practices in Urban Youth Culture. In C. Hmelo- Silver & A. O'Donnell (Eds.), In Proc. Computer Supported Collaborative Learning, New Brunswick, NJ, USA, 2007

[3] Repenning, A., Excuse me, I need better AI! Employing Collaborative Diffusion to make Game AI Child's Play. In Proc. ACM SIGGRAPH Video Game Symposium, Boston, MA, USA, ACM Press, 2006.

[4] Sturtevant, N. R., Hoover, H. J., Schaeffer, J., Gouglas, S., Bowling, M. H., Southey, F., Bouchard, M., and Zabaneh, G. 2008. Multidisciplinary students and instructors: a second-year games course. In proc 39th SIGCSE Technical Symposium on Computer Science Education, Portland, OR, USA, 2008.

[5] Basawapatna, A. R., Koh, K., and Repenning, A. 2010. Using scalable game design to teach computer science from middle school to graduate school. In Proceedings of ITiCSE '10. ACM, Bilkent, Turkey 2010.

[6] Squire, K., Video games in education. International Journal of Intelligent Simulations and Gaming, (2) 1. 2003

[7] Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., and Saulters, C. 2010. Teaching computational thinking through musical live coding in scratch. In Proceedings of the 41st ACM Technical Symposium on Computer Science Education Milwaukee, Wisconsin, USA, March 10 - 13, 2010. SIGCSE '10.

[8] National Academy of Sciences on Computational Thinking. Report of a Workshop on The Scope and Nature Computational Thinking, National Academies Press, 2010

[9] Repenning, A., Webb, D., and Ioannidou, A., "Scalable Game Design and the Development of a Checklist for Getting Computational Thinking into Public Schools", Proc. SIGCSE' 10, ACM Press, WI, USA, 2010

[10] Koh, K., Basawapatna, A., Bennet, V., and Repenning, A., "Towards the Automatic Recognition of Computational Thinking For Adaptive Visual Language Learning," to appear in Proceedings of the 2010 Conference on Visual Languages and Human Centric Computing (VL/HCC 2010), IEEE Computer, Madrid, Spain.

[11] Basawapatna, A., and Repenning, A., "Visualizing Student Game Design Project Similarities", In Proc. Diagrams 2010, Portland, Oregon, USA 2010.

[12] Landauer, T. K., Foltz, P. W., and Laham, D., Introduction to Latent Semantic Analysis. Discourse Processes, 25, 1998, 259-284