

Understanding the Evolution of Adolescents' Computational Thinking Skills within the Globaloria Educational Game Design Environment

Alex Games
Adjunct Associate Professor
Telecommunication, Information Studies and Media
Michigan State University
games@msu.edu

ABSTRACT

This document is a continuation of a study that between 2011 and 2012 examined the evolution of adolescents' computational thinking in the context of Globaloria, an innovative learning platform centered on game creation invented by the World Wide Workshop.

Since Jeannette Wing's 2006 treatise on computational thinking, researchers at the intersection of the fields of education, the learning sciences, and computational thinking have focused on developing curricula that help youth to develop these highly valued skills. These skills are 21st-century problem-solving abilities and built on the foundation of critical thinking, logical thinking, and using computation to solve complex problems (Wing, 2006; National Academy of Sciences, 2011). Design and project based learning have already been recognized as strong methods of developing these skills (Harel and Papert, 1991; Kolodner, 2003), however, game design has recently emerged at the forefront of this research (Basawapatna, Koh, Repenning, et al., 2011; Games, 2008, 2010; Stolee and Fristoe, 2011), seeking to capitalize on the enormous popularity of video games, with even the White House recognizing and supporting efforts to teach computational thinking via game design (White House, 2009).

Globaloria's goal is to foster computational thinking and STEM skills and concepts in middle school and high school students by immersing them in a social network for learning through game design and programming, using a robust curriculum and digital platform that leverages industry-standard tools to conceive, plan, program and publish web-based games focused on educational and social issues. Globaloria classrooms are designed around constructionist pedagogies (Papert and Harel, 1991; Harel, 1991), and feature a project-based curriculum supported by a framework of Web 2.0 technologies, and an online community of school classrooms, educators, and professional game

designers.

Using a theoretical framework developed by Games (2010) to study thinking in the context of game design, and case study methodology supported by multimodal content and discourse analyses, the study examined the evolution of 30 students' computational thinking and STEM learning and literacy as a function of their changes in language use, design strategies, and game artifact production (10 of these students were also interviewed last year, providing year-to-year longitudinal data on the evolution of their skills in the Globaloria classroom). Findings suggest that over time and within a scaffolded game design-based curriculum, students can develop computational thinking skills and deep understandings and engagement with STEM subjects. Results also show that a shift towards a design-based approach to the curriculum can positively affect students' design thinking, resulting in more engaging and thoughtful games.

INTRODUCTION

The World Wide Workshop invented and launched the Globaloria Platform in 2006, and has been the provider and operator of the Globaloria Network nationwide and worldwide in the past 6 years (see www.WorldWideWorkshop.org/map). Globaloria is constituted by a social learning network, within which is embedded a well structured, game-making curriculum that allows students to "create educational games and interactive simulations, for their own personal and professional development, and for the social and economic benefit of their communities" (World Wide Workshop, 2010). At the core of Globaloria are five interconnected digital platforms where students and educators invent, build and share their learning with one another and Globaloria staff. The platforms leverage industry standard tools such as a learning platform WikiMedia, Google Tools, Adobe Flash, Skype,

and others to support the year-long learning and collaboration following a 100+ hour curriculum and over 100 game-design tutorials. (see Figures 1a, 1b, and 1c).

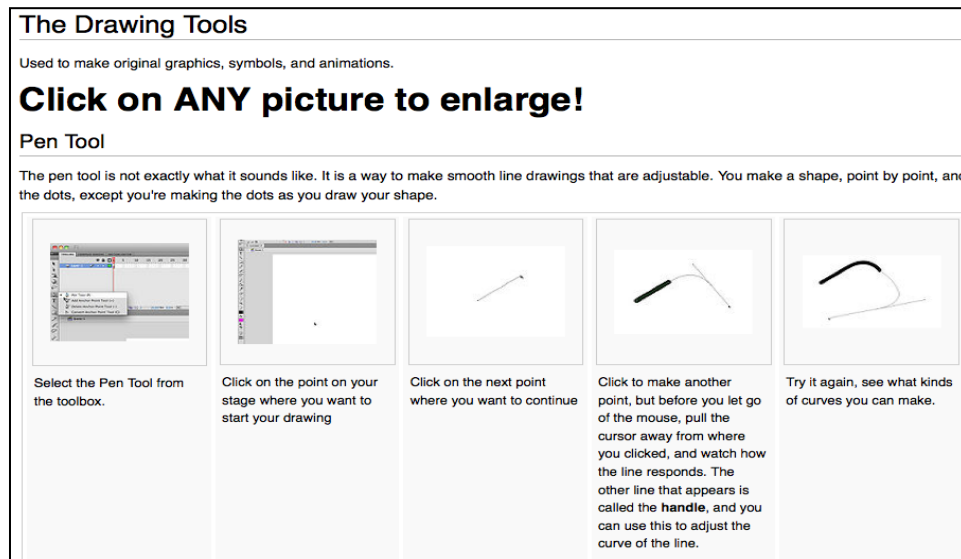


Figure 1a: An Example of a step-by-step “drawing tutorial” on the Globaloria Platform.

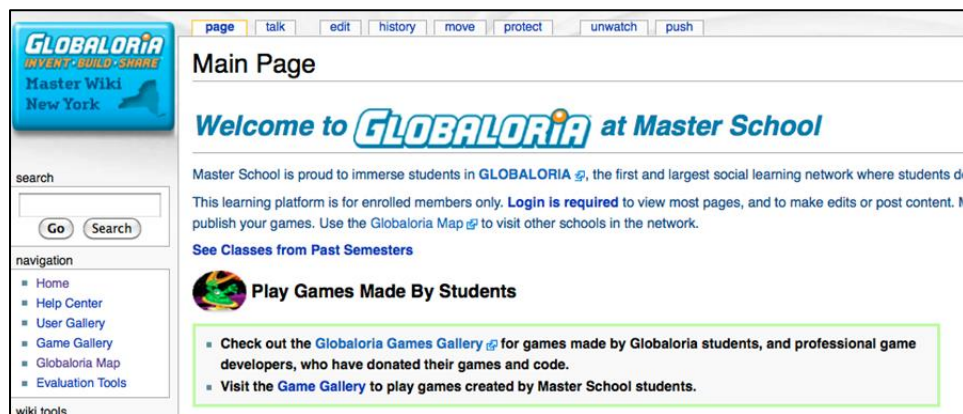


Figure 1b: Globaloria provides each participating class or group with a unique space customized with the school’s name, the class and teacher name, and unique user gallery and course syllabus.

How do I embed Text?

CS5 wants to make sure that the fonts you use will display on any computer. To prevent missing fonts, they now offer a way to embed fonts in your SWF file. This also allows you to use a font that is not found on the target computer. Once you embed a font in your file, you can use it as many times as you want. This is especially important when you are using **dynamic text** in your file. The information must be built into your file.

Walkthrough

To embed characters from a font in a SWF file:

1. With your FLA file open in Flash, open the Font Embedding dialog box by doing **one of the following**:
 - Choose Text > Font Embedding.

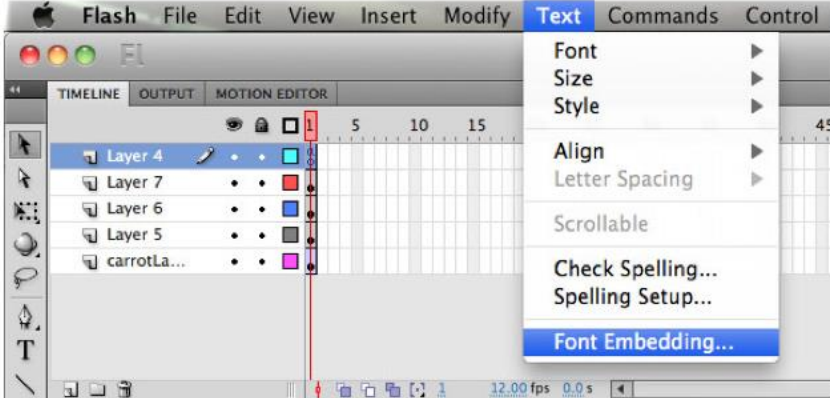


Figure 1c: An Example of a tutorial for “Embedding Text” in Flash Actionscript is easy to find on the Globaloria Platform. Over 50 tutorials are available for students to use as-needed, during their learning process of game design and programming.

Students also have their own personal learning platform wiki pages in the Globaloria Learning Platform. To these they can post their blogs, notes, assignments, favorite games and animations, and information about themselves (non-private information, of course - favorite movie, actor, video game, food, etc.) (see Figure 2).



Figure 2: An Example of a student’s “Profile Page” on the Globaloria Platform. Each participating class gets a User Gallery, and Course Syllabus, as well as 100-150 hour-long Game Design Curriculum and Shared Resources (links are on the left-nav).

The students' learning platform wikis are also connected to the school-learning platform so that they are able to find and access help from the lessons and tutorials, as well as communicate with each other. Alongside the community and personal learning platform wiki is Adobe Flash (while explaining Flash and how it works is important, that explanation is not within the scope of this paper. Explanations can be found at these links: [Format](#), [Actionscript](#), and [User Experience](#)). As mentioned before, students in Globaloria learn to create educational games, simulations, and animations, all of which is done in Flash. Like many programming languages, Flash is a relatively complex language to learn, especially for middle school students. However, similarly to the libraries of codes that professional programmers may use, Globaloria provides libraries of commonly used codes on its school learning platform, as well as tutorials and examples of how and when to use those scripts. For example, pieces of code that make objects move and follow other objects can

be found on the school learning platform, and the students simply have to find the appropriate code for the game or animation that they are working on, copy and paste that code into their Actionscript, and update the instance names. Alternatively, students can find and download .fla files (similarly to how Word documents are identified as .doc or .docx files, Flash files are identified as .fla files) from the school learning platform and replace the assets with their own, or to see how the Globaloria staff created that particular animation or game so that they can replicate it on their own (see Figure 3).

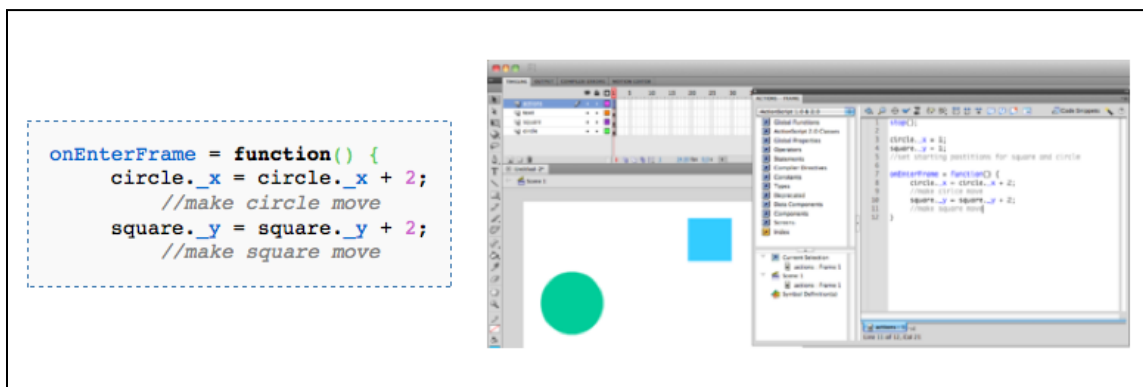


Figure 3: An Example of a step-by-step tutorial on how to program “Collision Detection” in Flash Actionscript on the Globaloria Platform.

In terms of the structure of Globaloria, it is and functions like a full class, either as a stand-alone elective or integrated into a subject-specific class, such as history or math. In all implementation models, students meet once a day and receive assignments and homework. However, due to the more creative nature of Globaloria, classroom time is structured more like a design-based class - students are given an assignment and allowed to use whatever tools they need to complete that assignment, and guidance and support are given by teachers in the classroom and virtually by Workshop staff and volunteer experts when necessary, creating a blended learning environment. Unique from the traditional educational format of a teacher lecturing at students with little to no meaningful

interactions, this structure often combines multiple disciplines when completing assignments and contains constant teacher-student interaction and feedback.

During the assessment, for example, we broke the classes into groups and had the groups build a game about the predator-prey relationship between the Canadian lynx and the snowshoe hare, which was given to them in the form of a short passage. At a minimum, students needed to create original assets, research needed to be done, and the appropriate code had to be found in order to complete this assignment, and each member of each team contributed to this process. Our observations of the classrooms also confirmed that the teachers are constantly providing feedback and assistance to students.

Evidence that computer games and simulations can be used for learning in STEM literacy has become increasingly available as research in learning sciences, educational psychology, and media studies has become available (Games and Squire, 2011; National Academy of Sciences, 2010). Modern videogames are complex sociotechnical systems where players constantly engage in cycles of play interactions with rule-bound computer simulations built by teams of designers, computer programmers, and digital artists. To participate in the production of a videogame, learners must be able to solve a series of complex problems using computer technologies, and think in terms of the computer tools they will use to enable or limit them from doing so.

The body of knowledge and practice skills necessary to solve problems effectively using computing tools has been termed *Computational Thinking* (CT) by several scholars, and as Wing (2010) has argued, these skills and knowledge will be fundamental for countries like the U.S. to maintain competitive workforces in the 21st century. The primary reason for this is that the advent of modern computational technologies (Computers and other ICT's) have fundamentally transformed the social, economic and even demographic

mechanisms of a globalized world (Gee, Hull, and Lankshear, 1996). Today's workplaces are rapidly becoming sophisticated sociotechnical systems, where everything from serving a hamburger to producing the newest space exploration technology requires people capable of understanding the connections between software, hardware, information, and the way they mediate human activity. Social media and mobile technologies have made our social, entertainment, and civic lives exist in a sea of information, where effective (and safe) participation, require citizens capable of understanding and taking advantage of the powerful computational tools that now lie at their fingertips.

OPERATIONALIZING COMPUTATIONAL THINKING IN GAME CREATION

Computational thinking is a construct that encompasses a constellation of forms of thought and practice that professionals can use to solve problems effectively using the logical, mathematical, and representational tools that computers make available to work through data, and transform it into actionable information.

Given the nascent nature of CT as a field of study, no widely accepted theoretical framework yet exists that characterizes it fully, given the overlapping nature of computation across fields such as computer science, engineering, and mathematics, each one of which is interested on very specific aspects within it.

Wing's work has time and again posited that computational thinking is a set of skills broadly applicable in general life. However, at the point of writing the vast majority of the CT frameworks proposed by scholars examining it's emergence within games have been firmly and narrowly situated in the discourse and practices of computer science, by and large giving preference to different subsets of problems within that field as the ad-hoc standards against which CT evidence is measured. For example, Repenning and his

colleagues have used Computational Thinking Patterns in much of their work (Repenning, et al., 2010, 2011; Basawapatna and Ioannidou, 2009, 2011) with an overt emphasis on how those patterns appear in software code in a 2d game creation platform called agentsheets. The specific interest CT focus of the study was to observe the code implementations of specific algorithms (e.g. hill climbing, collisions, diffusion) as students attempted to replicate preexistent game designs. Similar frameworks have been proposed in Berland and Lee's work on board games (2011) and Stolee and Fristoe's (2011) work with Kodu Gamelab.

We argue such narrow focus is problematic to understanding computational thinking in game creation, as it tends to put a single discipline lens in an activity that is inherently multidisciplinary. Creating games is creating sociotechnical systems, as for the game creator, solving the problem of how players will perceive and experience the play created by the game is just as fundamental as solving the problem of making the game software work to enable such play. Hence, we required a framework capable of examining CT in game creation not only at the technical level of software production, but also at the cognitive/psychological, and social/aesthetic levels inherent to the process of making games.

Toward this end, we decided to use a framework for analysis that synthesizes two others, described in the following sections.

Part 1. Computational Thinking

First, In order to operationalize CT for this particular study, we built on the Exploring Computational Thinking Framework proposed by Google, (2012). The framework characterizes CT according to five distinct dimensions that encompass habits of mind and practice germane to solving problems using computational tools. These are:

1. **Decomposition** is the breaking down, or ability to break down, a problem into its core components. For example, when we give directions to someone for how to get to our house, we are *decomposing* the process of getting from one place to another. In game design, we can decompose a game into its subsequent parts, such as its assets or coding.
2. **Pattern Recognition** is the ability to see or identify recurring systemic connections between objects, as well as recurring subsystems in systems. Google uses the example of using patterns in stock prices to help us decide when to buy and sell. In video games, patterns help the player to identify character behavior, which allows the player to make better decisions on how to play the game.
3. **Pattern Generalization and Abstraction** is the ability to recall previously encountered patterns and use them to aid in the solving of applicable problems in different contexts; a form of near transfer that is characterized by students' ability to abstract. One example from Google is from mathematics, where we write equations in terms of variables instead of numbers, which allows us to solve problems using different values. An example in game design is creating an item-crafting mechanic, which allows players to simulate the experience of crafting various items, such as hammers or pickaxes. While the real-life experience does not directly translate with the virtual experience, it does *abstract* the experience, giving the player a similar experience.
4. **Algorithm Design** is the construction of a step-by-step process towards the solution of a design problem. The more sophisticated this skill, the more *efficient* the algorithm will be. One example of an algorithm is a play in football that a coach may design for his players to follow during a game. In game design, programmers follow algorithms, or step-by-step instructions, for writing code that will make the game function properly.
5. **Information Modeling** refers to the students' ability to extract data from sources relevant to a phenomenon and to use it to construct a model that can be communicated to others. As an example, examining a table of daily ozone values in a metropolitan area, and using them to construct a histogram to share with others would be a form of modeling and visualization, but constructing a game with rules and mechanics based on the same values in a way that others could use them would also be an example. In the item-crafting example, the designer would analyze the core experience of crafting an item. The resulting gameplay mechanic is a model, or simulation, of the actual process of crafting such items.

Recent reviews of the field have recognized that across the board, most of the aspects of this framework are commonly acknowledged whether explicitly or implicitly in other studies. The use of abstractions to model world phenomena, the ability to think of these phenomena in terms of systemic interactions, the use of symbol systems to represent and manipulate those abstractions, the use of algorithms to structure and control the flow of information in modelling processes and problem solutions, the use of flow control techniques such as conditional logic and iteration to optimize those algorithms, and the use of iterative refinement, among the key thinking skills that must be put into practice for effective computation in real world activities beyond computer science. (Grover and Pea, 2013).

Another advantage of this framing of CT is that it's dimensions 1) are measurable, meaning that there are concrete and perceptible processes and products that can be captured by systematic research either through observation or other means; 2) They are applicable to a broad range of professions and activities involving the use of modern computing tools beyond computer science, which in our view is a necessary prerequisite to differentiate computational thinking from other forms of logical and mathematical abstract thought; 3) they incorporate or encompass a broad swath of dimensions of computational thinking that have been discussed as relevant to K-12 instruction (Barr and Stephenson, 2011).

In our analysis we operationalize the term "patterns" as organizations of elements into

recognizable forms. These could be patterns made of abstractions alone, software code alone, visual representations alone, or as in most cases with modern videogames, combinations of these and other components. In gaming and game creation practice recognizing these patterns provide players with “tools to think with” (Games, 2010) that facilitate strategic problem solving. For example collection of characters in a video game may appear to be moving randomly, but upon further inspection, experienced players will observe that the characters always move from the top to the bottom of the screen, and their movements are staggered at regular intervals. Noticing this pattern of behavior will facilitate the players thinking of strategies to safely navigate the level.

Our framework and analyses also have a relative focus on pattern abstraction, which distinguishes our analyses from other work that focuses more on pattern recognition and computational processes (debugging, Boolean logic). We choose to focus on abstraction and generalization because we feel that it is the most important dimension of CT; while pattern identification is important, it is pattern abstraction and generalization that allows someone to make use of those patterns in other contexts. Video games do an exceptionally good job of forcing their players to perform pattern abstraction. In *Batman: Arkham City*, for example, most of the bad guys in the game are unarmed and can be subdued rather simply. Within a single group, there is little or nothing to distinguish one of these thugs from another. As the player progresses, s/he will encounter thugs armed with knives, guns, and shields. Aside from the actual weapon, these bad guys are always dressed in a different color from the normal, unarmed thugs (red, blue, or orange vs. gray). Taking down these thugs requires more complex action from the player. Identification of the pattern (i.e., gray, gray, red) and appropriate action are critical for the player to succeed in the game. Pattern abstraction is also critical in intermediate to higher

level mathematics, and is the basis for algebra and geometry. The Pythagorean theorem requires students to be able to identify and apply a specific equation to a specific set of problems, which can all be identified by specific patterns. In game design, a student may often recognize the pattern of a moving object, but not necessarily be able to understand the underlying code. As the student gains skill and knowledge, s/he may start to understand that when an object moves, there is a specific piece of code that causes the movement. Once they can identify and understand this, *abstraction* allows the student to use that code in another game that might require the same or a similar type of movement. We would like to note that in the fifth dimension, we have substituted the original google term "visualization" for the term "modeling". Although *visualization* is an adequate enough term in its broadest sense (games are media that encode information in a predominantly visual form), we feel that *modeling* more accurately portrays our own thought processes with respect to the interview tasks, which asked the students to construct interactive models of a predator-prey relationship.

Part 2. Thinking in Game Design

Game creation is a complex, interdisciplinary field that has as its first and foremost goal to create sociotechnical systems that engage human beings in play (Gee, 2007, Crawford, 2004, Schell, 2009). Modern videogames are seldom created by individuals, because their complexity requires that different professionals have focus on specific aspects of the system (e.g., software engineering, gameplay, art, production, business and marketing strategy, and so on), and then use artifacts from software to design documents to paper prototypes, to overcome their specific *horizon of observation* (Hutchins, 1993) –i.e. their ability to see how their actions affect other teams and how the actions of other teams affect them- to help the game studio as a whole act as a learning organization where

knowledge of the overall sociotechnical system is distributed among employees.

In educational game creation, the games students create tend to have a much smaller scope, and thus it is possible for them to grasp this diversity of domains to a sufficient degree that they can produce a functional and playable system. In a series of studies of children designing their own videogames, Games (2008; 2010) characterize the sociotechnical practices and ways of thinking that students develop while creating games in the classroom, and proposed a framework for analyzing and assessing learning in the context of game production as a function of the acquisition of key constructs central to the discourse of professional game designers (the role of designers in most professional studios is to be the keepers of game design documents which establish the functional, aesthetic, and technological specifications of the system, thus allowing them to serve as central mediators between teams). The framework examines increasing sophistication in this discourse as a function of the degree with which learners' decisions, language, and tool use reflect an increasing understanding of the nature of games as sociotechnical systems, through an awareness of a) the affordances of the software code, materials, assets and tools available to them in creating games, b) the abstractions necessary for an idealized player to play a game with these materials (e.g. rules, mechanics, goals and so on), and c) the probable ways in which real players would interpret and understand these materials and abstractions during play. In addition, in framing game design thinking as increasingly dialogic, the framework acknowledges the central importance of systematic iterative refinement in the improvement not only of game software (i.e. debugging), but also of the user experience created through it, and thus extends this dimension of computational thinking to encompass the overall sociotechnical system being created. Rooted in Discourse theory (Gee, 2005; Games, 2008), the framework sees these

dimensions as dialogic in nature, as ongoing conversations between one or more designers and one or more players, mediated by the game artifact. Gameplay and game refinement are acts of iterative progression toward a common construction of meaning between the two parties, a common understanding. As learners become more adept at thinking in terms of design, in this framework, their use of dialog to understand design as a process becomes more apparent. Table 1 describes the scope and nature of the evidence that would make each dialog overt.

Dialog	Description
Material Dialog Perspective	Refers to language and practices that reflect that students have an understanding and use of the techniques necessary to construct a game system. Akin to DiSessa's notion of material intelligence (2002) For example, a designer could not make a quality game using Flash, unless they understood both the programming principles of Actionscript, as well as their connections to its' vector graphics system.
Ideal Player Dialog Perspective	Refers to language and practices that reflect an understanding of the abstractions that need to be encoded in the tools to transform a system of materials into a play system, including the use of the specialist language that game designers use to describe the possible actions that these abstractions enable and limits for players (game rules, mechanics, etc). For example, discussing the way that the rules in chess would define the possible ways in which a player could move a pawn.
Real Player Dialog Perspective	Refers to language and practices denoting an understanding of how to use the game system built from the materials to encode knowledge and metaphors that make it clear and overt to real players how the game is to be played. This dialog also involves an understanding of the encoding of knowledge extraneous to the game in order to support player decision making during the game. An example of a failure to understand this would be a learner designing a game for young children containing a lot of fast-scrolling text, with the expectation that they would need to read this to play.

Table 1: Games' Three Dialog Framework.

As Games has shown (2008; 2010; 2011), becoming fluent in the three levels of dialog gives them the tools to think of the creation of computer games as the process of

systematic problem solving where computation at the technical, psychological, and social levels play key roles toward the construction sociotechnical systems of play (Games, 2010). This involves learning to see these systems as constituted by three different but mutually necessary models, which are 1) models of automated interactions between software objects (material dialog), 2) models of human-computer interactions bound by abstract rule systems (ideal player dialog), and 3) models of expression and communication of meaning defined by cultural conventions of fun and play (real player dialog). As scholars have argued, it is precisely in the process of reconciling these three levels of modeling, abstraction, and problem solving, that creating games provides learners with powerful tools to support computational thinking (Lee, Martin, and Denner, et al., 2011).

Due to the dialogic nature of learning and thinking through game design, evidence of students' progress is only visible in an analysis of learners' ways of talking and doing during the cycles of iterative refinement and continuous formative feedback that are fundamental to game design. Once positioned as a dialog, an analysis of learning evidence must necessarily include not only how a learner attempts to construct meaning with the tools to think with at his/her disposal, but also the ways in which the learning environment is supporting or failing to support such construction. When tools and curricula support the dialogic process, students are gradually empowered to drive their own learning, and are repositioned from an understanding of themselves as players (users of games), to creators (owners of games), a principle that has been observed across the board in other learning environments that support CT (Lee, Martin, and Denner, et al., 2011).

RESEARCH QUESTIONS AND METHODS

Given the assumptions given by the above theoretical framework, this research aimed to document the evolution of children's computational thinking in the context of their participation in the Globaloria curriculum, by answering the following research question:

1. In what ways does students' computational thinking in game design evolve over time within the Globaloria context, as reflected by their discourse?

To answer the question, the authors used a qualitative methodology of case studies (Stake, 1995) and, consistent with the three dialog framework, multimodal discourse analyses (Gee, 2005) of learners' games, design decisions, and tool use over a period of six months of participation in Globaloria. The goal of the study was to produce a "thick description" (Geertz, 1973) of the Globaloria learning ecology and its impact on computational thinking over time.

In line with Globaloria's constructionist philosophy this study examined children's learning from a socio-culturally situated perspective. In this perspective, language, action, and thought are seen as integrally interconnected (Gee, 1992; Vygotsky, 1978; Wertsch, 1998; Engstrom, 2005), and evidence of changes in thought emerges from triangulating evidence of changes in students' ways of enacting the solutions to design problems. Such evidence was collected by coding video data for *design decisions*, *talk* about those decisions (both captured through screen recordings), and the *computational artifacts* resulting from and supporting those decisions (game software, paper game designs stored in the Globaloria curriculum learning platform wiki) over time.

Data observations were coded according to categories generated from the intersection of the three levels of thought in the three dialog framework, and how the five dimensions of computational thinking would be applied in them. Table 2 summarizes

these codes.

	Decomposition	Pattern Recognition	Pattern Generalization and Abstraction	Algorithm Design	Data Analysis and Visualization
Material Perspective	Breaking a problem into its software components.	Recognizing patterns for their software components.	Applying patterns made of software abstractions.	Designing problem solutions based on the affordances of software abstractions.	Surfacing models for algorithms through software visualizations and programming.
Ideal Perspective	Breaking a problem into its play abstractions.	Recognizing patterns for their play abstractions.	Applying patterns of play abstractions.	Designing problem solutions based on specific play abstractions.	Surfacing models of algorithms through play mechanics and play elements.
Real Perspective	Breaking a problem into its cultural interpretations.	Recognizing patterns for their sociocultural interpretations (e.g., genres).	Applying patterns for their sociocultural abstractions.	Designing problem solutions through sociocultural abstractions.	Surfacing models of algorithms through cultural imagery.

Table 2: Codes used for the analysis of computational thinking in game design.

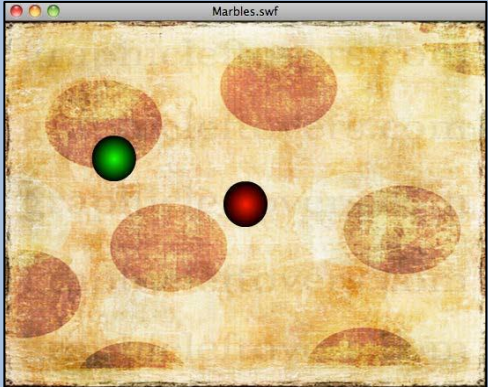

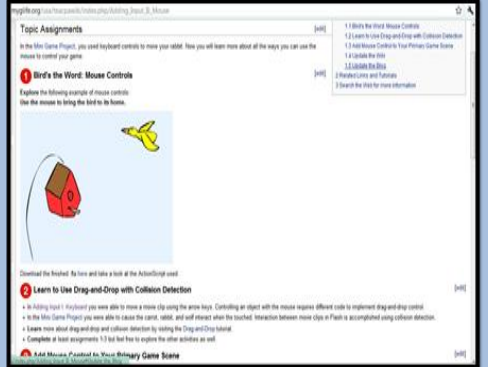
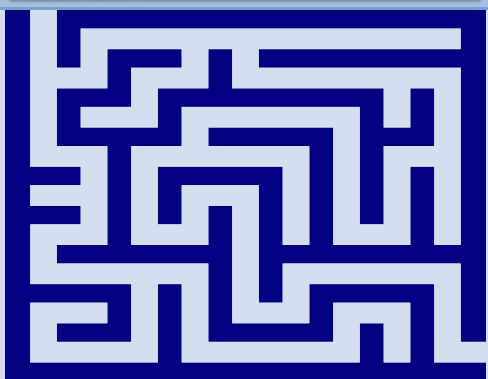
Evidence of increases in sophistication in each one of the cells in the above matrix was assessed through a rubric consisting of 4 levels. Level 1 displays an absence of any evidence whether in their design decisions or their language, of that particular dimension of computational thinking in game creation. Level 2 was displayed when students made statements about their design process pointing to some knowledge to the particular dimension, either using the specialist language of games, flash, programming, or so on. Level 3 was displayed when students used statements and or concepts in the Globaloria curriculum to give verbal explanations that detailed how that dimension had affected their design decisions. Level 4 was displayed when students gave explanations using specialist language or content, as well as demonstrated the use of the concepts applied in their design decisions during the task.

Data were collected from pre- and post-assessment protocols that consisted of individual interactive interviews with Globaloria students, as well as from observations of their design activities during Globaloria class over a period of 4 months. The interactive interview consisted of a sequence of questions that began by requiring students to solve various problems in game creation with Flash, while verbally articulating an explanation of their design decisions during the solution process. Parallel video recordings of participants' and their computer screens were captured, with the goal of documenting participants' design decisions, choices for tool use, and language describing their thinking and practices during each stage of the protocol.

Each component of the interview consisted of a task involving the design and construction of games or parts of games using flash and the Globaloria learning platform. Each task was specifically designed to emphasize one of the individual dimensions of the CT model in this study. The tasks were designed to be incremental, so that tasks later in the protocol would require applying the dimensions previously tested, giving us more opportunities to sample each dimension. These overlaps are noted where relevant.

The assessment was not designed to get at declarative knowledge, but rather to assess how well students were able to recall, construct, and apply their own knowledge in the process of solving design problems, in line with the constructionist philosophy of the Globaloria curriculum.

The five dimensions, displayed by students' through design decisions, tool choices, and verbal explanations of their solution strategies during each task, were the basis for coding our observations for discourse analysis (Gee, 1999; 2005; 2007), as described in Table 3:

Dimension	Description	Heuristics for Identification	Example
<i>Decomposition</i>	Students were shown a simple game made in Flash (see Figure a) and asked to recreate it.	<ul style="list-style-type: none"> Background Marbles At least two different pieces of Actionscript code. 	
<i>Pattern Recognition</i>	Students were asked to compare and contrast two systems that highlight the concept of collision detection	<ul style="list-style-type: none"> Points of collision in both the game and the learning platform Events when collision detection is <i>not</i> present. 	
<i>Pattern Generalization</i>	Students were asked to apply a similar underlying coding scheme to a similar design and mechanical pattern.	<ul style="list-style-type: none"> Assets (piggybank and money) Coding required to make the money move. 	
<i>Algorithm Design</i>	Students were asked to write out a discrete set of steps for solving a maze.	<ul style="list-style-type: none"> Steps should be written from the first-person perspective, as directed. 	


<i>Information Modeling</i>	<p>Students were given a short story about predator-prey relationships and asked to model that relationship in a game.</p>	<ul style="list-style-type: none"> • The predator • The prey • Interdependent nature of that relationship (as lynx go up, hare go down; as hare go down, lynx go down; etc.) 	
-----------------------------	--	---	--

Table 3: The individual interview protocol, including the observation coding criteria.

Data were collected from middle schools in Austin, Texas and Elkins, West Virginia. Due to the large number of schools running Globaloria in West Virginia, site selection was random in that state. However, the East Austin College Preparatory Academy in Texas was selected because it was the only school in Texas running Globaloria at the time. 15 students were interviewed in the Globaloria class in each site.

RESULTS

The data for this study were collected from two different sites over the two-year period of the study. The goal was not to make comparisons across sites, but rather to get a broader sample for longitudinal analysis. Table 4 gives a breakdown of the number of students at each site, and the dates for the pre and post-assessments for each site. Participants were all in middle school, ranging from 6th grade to 8th grade.

Site	Number of students	Pre	Post
East Austin Prep, Austin, Texas	15	2/19/12 – 2/22/12	5/21/12 - 5/22/12
Tygarts Valley Middle School, Elkins, WV	15	3/7/12 – 3/9/12	5/10/12 - 5/11/12

Table 4: Sample and dates.

Changes in Teaching Practice

One of the most fundamental changes observed in the overall Globaloria learning ecosystem this year vs. last was a strongly increased emphasis on the part of instructors to move away from instructor-centered teaching practices to learner-centered ones. A particularly evident change was reflected in their desire to place emphasis on helping students develop design oriented thinking and practices more closely aligned with Globaloria’s constructionist learning epistemology and pedagogical underpinnings, as opposed to trying to cover all content in the platform in a linear fashion. This was evident in both the changes to their discourse as well as their practices this year. In terms of discourse, this pivot to a more design-centered form of instruction was exemplified by typical teachers’ interview responses to why they felt design was important in their classroom, with statements indicating that learning to make learning games was *“like teaching a child to write me an essay without knowing what an essay is. You have to give them a scaffold so they can know game design”*, or decrying former years’ lack of quality games available that would integrate content and gameplay, stating that *“there was simply a lack of quality games out there that students made. I can’t even think of a game where they would be learning something through the actual game.”*

In framing game creation as an analog to writing or focusing on the importance of the

learning that would take place in actual play, they were recognized the inherently learner-centered nature of game creation, which in turn translated into important shifts in classroom practice.

The ways in which these new practice orientation manifested itself in the classroom this year were several. First, In terms of practices, while in past years the use of lectures and individual work had been somewhat prevalent among teachers and students (especially at the Texas site), this year peer collaboration in both the research and creation of game content were much more prevalent. Second, there was a stronger emphasis within the curriculum to provide learners with more and more diverse experiences with game types and play patterns.

The result at post assessment phase this year was a collection of games with much greater diversity and sophistication than in years past, where the quiz format had been prevalent (see figures 7 and 8 for examples). In particular, evidence of deeper computational thinking in these games can be seen in the thoughtful and deliberate way in which they incorporated the learning content they intended to expose learners to within the game mechanics of play, which for most games are the key activities players will repeatedly perform, and where most of their attention will reside during play.

How did these changes take place and what role did computational thinking play in them? The answer resides along the multiple dimensions of game design and CT examined in our protocol, outlined in the following sections.

A more Dialogic Mindset, Debugging and Iteration

One of the central differences observed between this year and past years in both student classroom practices and interviews was a much more pervasive use of systematic,

iterative refinement in all students' processes of game creation. This mimics the observations previously conducted by Games in other design centered technology learning environments (Games, 2008; Games and Kane, 2011). For Games, iterative refinement is so central a pillar to the development of effective design thinking, that the concept of the three levels of understanding necessary to develop good games as *three dialogs*, reflects a key change that must take place in learners' habits of mind and practice before they can effectively learn from game creation. This iterative refinement perspective transcends the three levels and binds them together as a mutually reinforcing system of communication (Games, 2010). In computer science learning research, debugging is the practice most representative of such dialogic perspective, and has been shown to be a fundamental skill necessary for effective computational thinking (Berland & Lee, 2011).

In the case of Globaloria, the student's move to a dialogic approach to their project development was present in the form of both *debugging*, -i.e. observing undesired functionality in a piece of actionscript code, hypothesizing and coding it's solution, and testing it again- and *game refinement* -i.e. observing undesired *play* behavior in the system at a sociotechnical level, and then changing its rules, mechanics, or representations to make it better-.

Debugging was typically evident within both the marble game and piggybank game tasks, and by pre-test, 100% of the students showed at least one instance of debugging behavior in their interviews. A typical example in the marble game was the students copying code to make the marbles move directly from the platform into their game, noticing that the code only allowed the marble to move along one axis, and then changing it to allow diagonal movement (for details about this process, see last year's report). In the piggybank task this year, a student would copy and paste the arrow key movement code

(and only the arrow key movement code) from the learning platform to his Actionscript. The student would then test his game, and observe that the movement the code would create was not what they wanted it to be. More often than not, students forgot to use the *stop()* instruction at the end of the sequence of motion instructions, and at test time their game would rapidly flash back and forth between the game scenes simply not move at all. 50% of the students would eventually figure out which code was missing by post-test, up from less than 20% last year. The dialogic perspective also permeated students' discourse, indicating its impact on their computational thinking habits. A typical example of this took place when a West Virginia student found his game was not working. When the researcher asked him to speculate on why that might be, he replied, "I'm probably missing something...like a curly bracket, or something", which in turn provided him with a starting point to begin a new cycle of testing and refinement.

Game refinement, on the other hand, was most typically present in the predator/prey task, and in the marble game task. In the first, since students had the ability to co-design their game with peers, the most evident form of game refinement would come from exchanges where peers would observe an initial design in action, and provide feedback to those who had implemented the design.

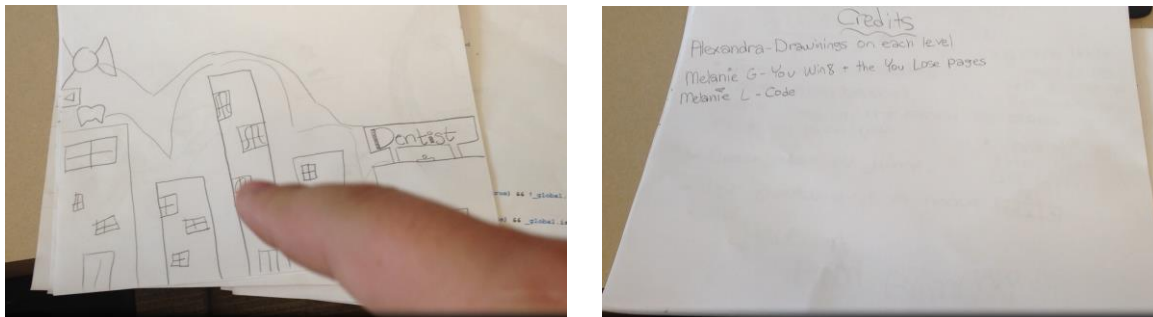


Figure 4. Discussing a game design and team responsibilities for a game design

This would be followed by the team engaging in a discussion about how the game could

be improved followed by a new round of implementation, critique, and so on.

In the marbles task, on the other hand, evidence of iterative refinement at the level of game design was also often present, particularly during the process of breaking down a game into its components and assembling a new one, which would often lead to conversations about those design elements that would make the game play better. This is discussed in detail in the following section.

Decomposition

In line with the findings of previous studies, the evolution of students' thinking towards increasingly sophisticated computational thinking and practices was evident this time around as well. While during the first year only a fraction of the students sampled had been able to identify the basic components of the marbles minigame, this time around all of the students in both their pre and post-assessments were able to identify marbles, background, and code as necessary components of the game. The students' overall confidence and competency with Flash was noticeably higher in the second year, with most of the students appropriating into their discourse the abstract relationship between visual assets, needed symbols, and their specific function within the software, and applying them to the successful creation of their own game.

This was displayed in three key changes within students' discourse when describing their decomposition of the game into its main parts:

First, nearly all of the students incorporated the *object* and *symbol* abstraction into their descriptions, especially at the moment of using code to change the state of visual elements. Statements such as a student needing of "add code to move the object during the game", and "I have to convert the marbles into symbols in order to be able to add code

to them" were typically part of a majority of descriptions by post-test, as was the application of these elements to recreating the game, as Figure 5, which shows a student's typical use of objects to switch between scenes in the game, suggests. These are of central importance to students' computational thinking from a material perspective, since they strongly indicate their appropriation of the underlying object-oriented mode behind actionscript in their thinking about games.

Second, the sophistication of this appropriation was evident in their ability to differentiate between different visual assets and the data types that would most effectively represent their functionality in code. By post assessment, most students showed widespread knowledge of the differences between the function of "button" symbols and "movieclip" symbols. Typically, this was evident in their discourse as well as their design practices. An example of this took place during a students' process of reconstruction of his Marble game. He began thinking out loud, and mentioned that he needed "to convert the start button to a button, which is different from the marbles, which need to be movie clips." When asked how he knew the difference, he said that he learned it in class, stating specifically that "buttons do different things only when you click on them, but movie clips make [more] things happen". These statements stood in stark contrast with the previous year's analyses of the game by students, where the start button for the game had by and large been ignored as a key component.

Third, a move toward a more sophisticated level in students' computational thinking this year was also evident in their use of systemic abstract patterns to describe key game components in this task. Their discourse incorporated a nearly pervasive use of *functional subsystems* (rather than just single assets or objects) to describe key components of the game in their decomposition task. For 76% of the students at preassessment, and 93% at

post assessment, descriptions involving the use of actionscript code to “make the marbles move”, “switch scenes”, or incorporate “collision detection” were elements identified as key components. This stood in stark contrast with the year before, when nearly all students on the same stages only used visual and superficial game components such as the marbles, text, or background, as the components identified as key for decomposition.

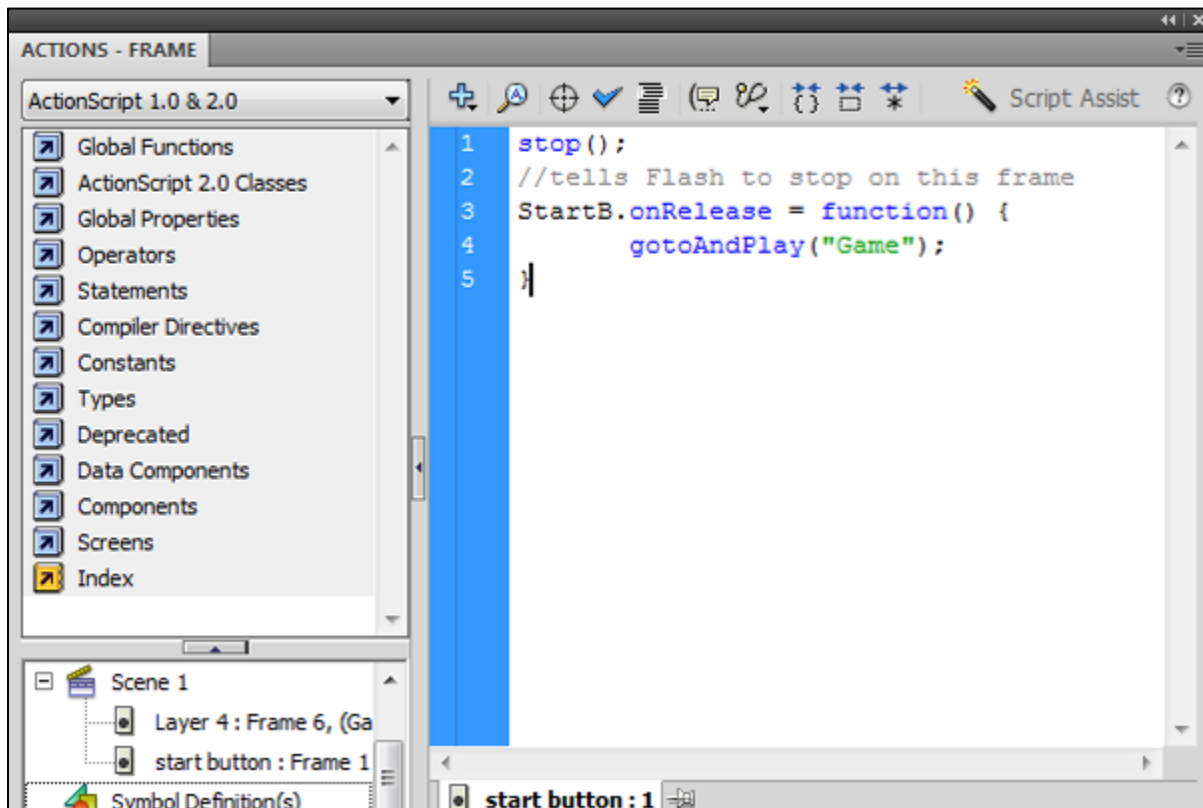


Figure 5: The scene-switching code that a student used during their recreation of the Marble game.

The above evidence also indicates that this year their evolution as computational thinkers was supported by a more robust set of thinking patterns focused on the ideal play level. As Games has pointed out (2008), this level of thought is characterized by the prominence it gives to abstractions not of software function, but of gameplay, that is, of games as play systems. A central characteristic of this form of thinking is the prominence it gives to the

role that “the player” (i.e. an idealized player) and the potential play experience he will have with the game system have on framing the design of the formal rules and the core mechanics (the key player interactions with the game) that will make the game work.

This increased accounting for an ideal player as part of the play system they intended to recreate was evident in the marked number of students in both Texas (from 5 in the year 1 to 12 in year 2) and West Virginia, Texas (none in year 1 and 8 in year 2) who by post-assessment identified “Instructions” as a necessary part of the game, as they would enable players to understand the rule system and be able to play.

Their discourse and design decisions also showed how a stronger emphasis on a target player audience for their game facilitated more sophisticated decomposition. During their process of reproducing the game at post-test, all students incorporated statements focused on providing players more input and mechanisms for direct interaction with the game. Statements such as “I’d make it where the green [marble] could be moved around by the player”, “I would set different goals for the marbles to get to” , “I would add enemies and obstacles for the marble to navigate” (87% of students reported this at post-test) were typical of at this stage, reflecting the appropriation of fundamental game design abstractions such as *game goals* and *game mechanics*, and *player challenge* (Games, 2008, 2010; Salen and Zimmerman, 2007) into their thinking processes during game creation. In practice, this resulted in overall Globaloria game projects with much more diversity and play sophistication than in previous years, with the student-made games looking more like professional games (i.e., platformers, puzzle games, narrative games, etc.; see Figure 7).

Pattern Recognition

This year, all students interviewed recognized the visual elements of the collision detection pattern in the marbles and seal games (see Table 3). However, probing deeper showed an uneven understanding across students of the underlying code logic governing the pattern's functionality. Revisiting the Marble task, 76% of students were able to identify general purpose codes (scene switching, collision detection, movement via keyboard or mouse input) as fundamental components constituting the pattern during the pre-assessment, but increased to 93% by post-assessment. By and large, their explanations held the intuition that collision detection represented a software event where a system effect would take place where two objects overlapped, as in the case of the bullets and the seal in the task, or the square and the circle in the learning platform tutorial (see Figure 6). However, an interesting misconception remained for 15% of the students, who claimed that collision detection in the first level of the seal and shark game was present when in fact it wasn't, justifying their claim on the fact that the seal running through the sharks would most closely resemble the visual effect of the demo in the platform. Such misconception is important, as it highlights a need (similar to last year's with the overall game patterns), to clarify the underlying principles that define a general collision event in code logic through multiple examples, versus its visual effect counterparts.

wiki tools

- File Uploader
- File List
- Image Gallery
- Recent Changes

unit 1: getting started

- Course Overview
- Create Your Profile
- Explore Globaloria
- Create Your Blog
- Join the Community

unit 2: game making intro:


- Play to Learn Part 1
- Choose Learning Topic
- Plan Game Scene
- Make Paper Prototype Part 1
- Draw Background

Related Tutorials

- Mini Game code.

1 Collision Detection Tutorial

Follow along in Flash and create a collision detector!



COLLISION DETECTED

1. Open a new Flash AS2 file. *Click any image to enlarge.*

Figure 6: The animation on the learning platform caused students to form an incorrect conceptualization of collision detection.

Pattern Generalization and Abstraction

This year, students showed substantial improvement in the tasks within this dimension. 86% showed increased proficiency with Flash from pre-assessment to post-assessment, as they successfully converted a drag-and-drop game into a keyboard-based game, when in previous years nearly half had not even been able to begin using Flash for the task and limited themselves to paper. In addition, the gains in proficiency between pre and post assessment in this task translated into increased efficiency in the application of the code and mechanics pattern to designing a new flash mini-game, since task completion times went from a median of 9 minutes to a median of 7, in addition to substantial qualitative

difference in the completeness of their content, where typical aspects included change from partial to full interactivity with the game mechanics (money and piggybank on the screen, vs. player being able to move the money to the piggybank), and increased accuracy in the physical model being simulated (money disappearing into the piggybank when placed there) became common. This increased proficiency is consistent with observations made through the previous two dimensions, and marks an important shift in students' thinking with abstractions this year, as where last year many were able to translate the play pattern at the level visual elements and mechanics using exactly the same underlying code (mainly copying and pasting from the platform, giving emphasis to a mainly material way of thinking), this year's observations showed their ability to recognize these same elements and transport them into a new game, but use a different underlying code base and still make the game work (showing a more balanced emphasis on material, ideal, and real play) .

Evidence of Pattern Generalization and Abstraction also became evident in the predator-prey relationships task, but re-emphasized last year's observation that in a collaborative setting, students computational thinking seems to be amplified and scaffolded by peers and teachers. When asked individually, all students were able to describe the main themes and components of the passage (i.e. that lynx prey on snowshoe hare, that hare consume vegetation and reproduce in large numbers, and that they both live in the Canadian forests), but only 17% could independently identify the dynamic relationship between the populations of the lynx and hare described in the theoretical statements at the end of the passage. However when the time came to create a game model, 93% of the students used the wolf and bunny mini-game as a template for illustrating the lynx and hare passage in Flash. While this attests to a strong evolution in

students' pattern generalization development, it also highlighted some opportunities for future development, as the section on modelling discusses.

Algorithm Design

The task for Algorithm Design was changed from last year's design to get more explicitly at how students will write down a discrete set of directions. In this case, they were asked to write the steps for solving a simple maze, as if they were standing in the maze itself. The solutions varied widely in terms of form and sophistication, with the simplest being one-word steps – “left, right, straight” – and the most complex being complete sentences – “Turn left at the first intersection.” Further analysis of their instructions gives some insight into their thought processes, particularly regarding how they perceive themselves in relation to the maze.

As the instructions were stated, the correct solution to the maze should start out looking similar to this: walk straight, turn left at the first intersection, follow path and turn right at second intersection, follow path and turn right at second intersection, etc. Following these directions would correctly get the maze-solver to the end of the maze. A majority of the students, 80%, used this point of view in both the pre and post-assessments. The remaining 20%, however, used a top-down point of view for solving the maze, in which typical student solutions used statements such as “go down, turn left, go up, go right”, and so on. This, in essence, inverted the directions that one should follow: go down (not forward), turn right (not left), follow path, and turn down (not right). These inverted directions had very important ramifications for the rest of their design process, as they framed their perspective of what the solution would be, with regards to the students' ability to place themselves within the maze. Their decision (or instinct) to solve the maze

from their own point of view showed a displacement, or disconnection, between themselves and the maze itself, and in turn provides a very important insight into how the “correct” solution to a problem from an algorithmic perspective can be substantially shifted depending on their initial perspective. The most important implication of this aspect comes at the level of assessment, as it implies that the dialogic perspective students are developing would also require a dialogic form of assessment to be effectively evaluated.

Students' abilities to design algorithms, however, were not limited to the maze task. Further evidence for this dimension was most notably observed during the Marble and the piggybank tasks. During the Marble task, the students were asked to list all of the components of that game need to recreate the Marble game that they had just played. They were then asked to go ahead and try to recreate, to the best of their ability, that game. In the piggybank task, the students were asked to make a game where the player used the keyboard to move money into a piggybank. It was in these moments, where students displayed their ability to identify a discrete set of steps that would lead to a functioning game. These sets of steps are the same general set of steps that all of the students utilized during both the pre and post-assessments.

1. Marble task:

- a. Students' first step was to either use the assets provided to them, or create their own, and then convert these assets to symbols, so that they could be manipulated on the timeline and via Actionscript.
- b. The second step was to rename the layers on which the assets were placed (background, marbles, actions).
- c. A third step, which was not performed by some of the students, was to create separate scenes containing the title scene and the gameplay scene.

For the students who included the title screen, this impacted their fourth step.

- d. The fourth step was to search the learning platform for code that could be used to make the marbles move and collide.
 - i. This step was observed to be the least clearly executed step, as some of the students did not know any of the codes that should be used. However, students who *did* know some of the codes (scene switching, collision detection, linear movement) were not always able to find them with ease.

2. Piggybank task:

- a. The first step in this task was always to create new assets (in contrast with the Marble task, no assets were provided). The sample was split almost evenly between using the shape tools and the freeform drawing tool to create their piggybank and money. In all interviews, both pre and post-assessment, this was the longest step.
- b. The second step was to convert these assets to symbols. Notably, two students in West Virginia and one student in Texas forgot to convert their assets to symbols, which created problems when they went to test their games. Again, the debugging skill was observed here, as all of them were quickly able to figure out what went wrong.
- c. After the students created their assets, their next step was always to start searching their learning platform for the necessary code. At the beginning of this step, students must be able to recognize the code that they would need, and then match that to something that they have seen in their experiences.

This allows them to quickly search for the correct lesson on the learning platform.

- d. The final step was to copy and paste the code, and then replace the instance names with those that they created. Once this was done, the students tested their games.

Examining these steps shows that the students had, in essence, a protocol for creating Flash games: create assets, find code on the learning platform, copy and paste that code into their own games, replace the instance names, and finally, test the game. Any errors that occurred would be corrected, and then the game would be playtested and iterated at a gameplay level. This process in game and software design is commonly called a *pipeline*, and is a form of computational thinking that transcends not only computer science and games, but touches virtually every project where complexity must be harnessed into a process of scalable creation.

All of the students this year were heavily reliant upon this set of steps, but future work on this dimension should focus on including additional complexities in the problem spaces, so that observations can be made on how well students are able to adapt this protocol to new situations. Such observations could provide rich data on near-transfer of these computational thinking skills.

Information Analysis and Modelling

Observations of this dimension stayed consistent with the previous dimensions with regard to the students' surface-level pattern abstractions. As noted earlier, all of the students could easily explain the main points of the passage, which was that it was about the snowshoe hare and the Canadian lynx. However, very few of the students (17%)

could independently explain the dynamic relationship between the two species' populations. This had important consequences on the design decisions for their games.

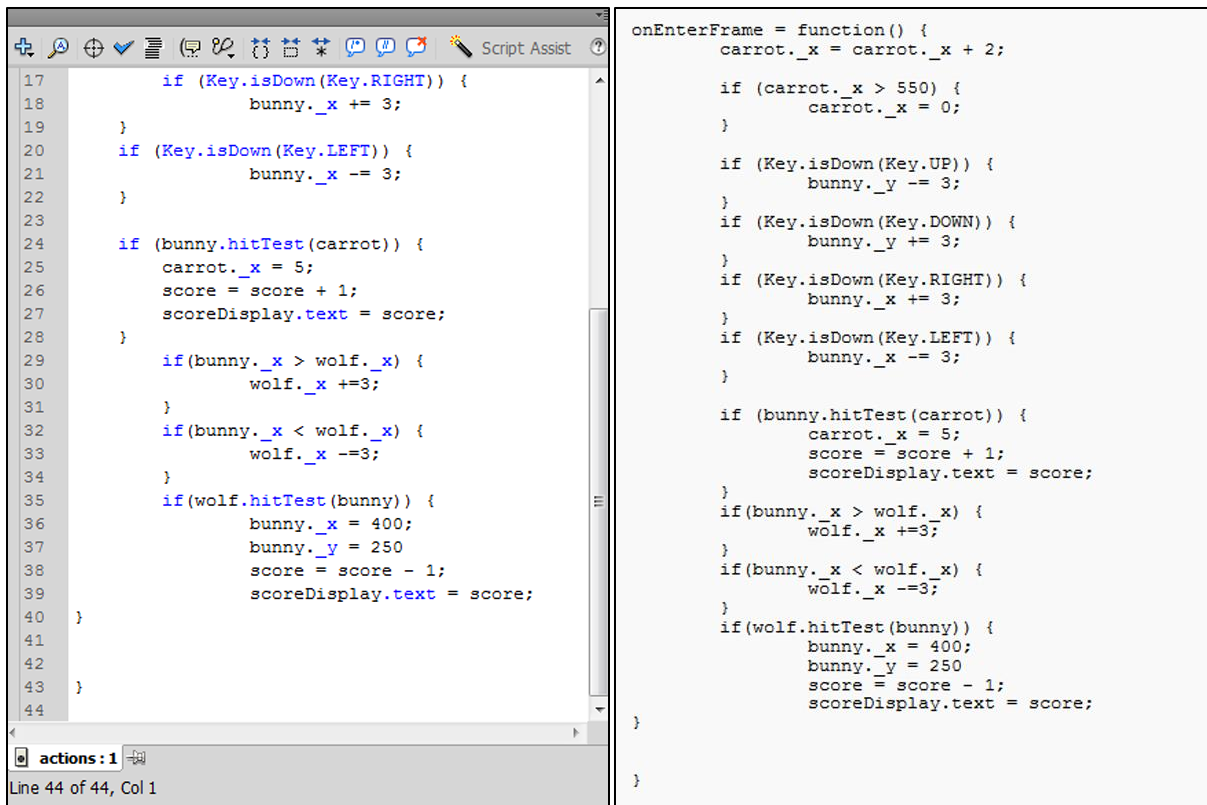


Figure 7: Side-by-side comparison of a student's code from the IAV task and the code from the wolf and bunny minigame.

Similar to last year, many of the students based their games directly on the wolf and bunny minigame that many of them had completed earlier in the year or in previous years (see Figure 7). This minigame is the first form of player input that students learn, and it makes sense that they would match this game format with the lynx and hare passage, as they are both about predator and prey. Unlike last year, the limited scope of designs observed from the students stands in contrast with the previous dimensions, where more creative thinking and higher proficiency with Flash were observed. Classroom observations of student work and game samples from the student showcase also contradict the observations made during this task.

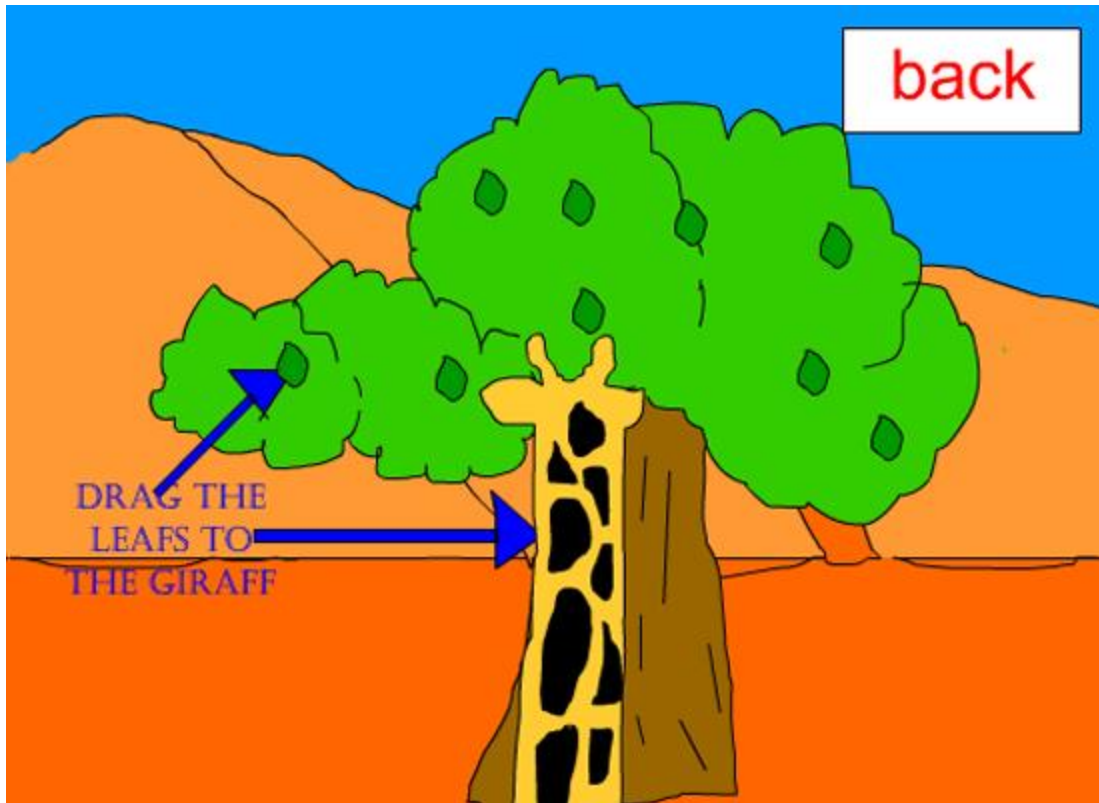


Figure 7a: A game from 2011 where the player simply drags leaves to a giraffe. The primary content of the game is on a separate "About" screen, which is accessed from the title screen.

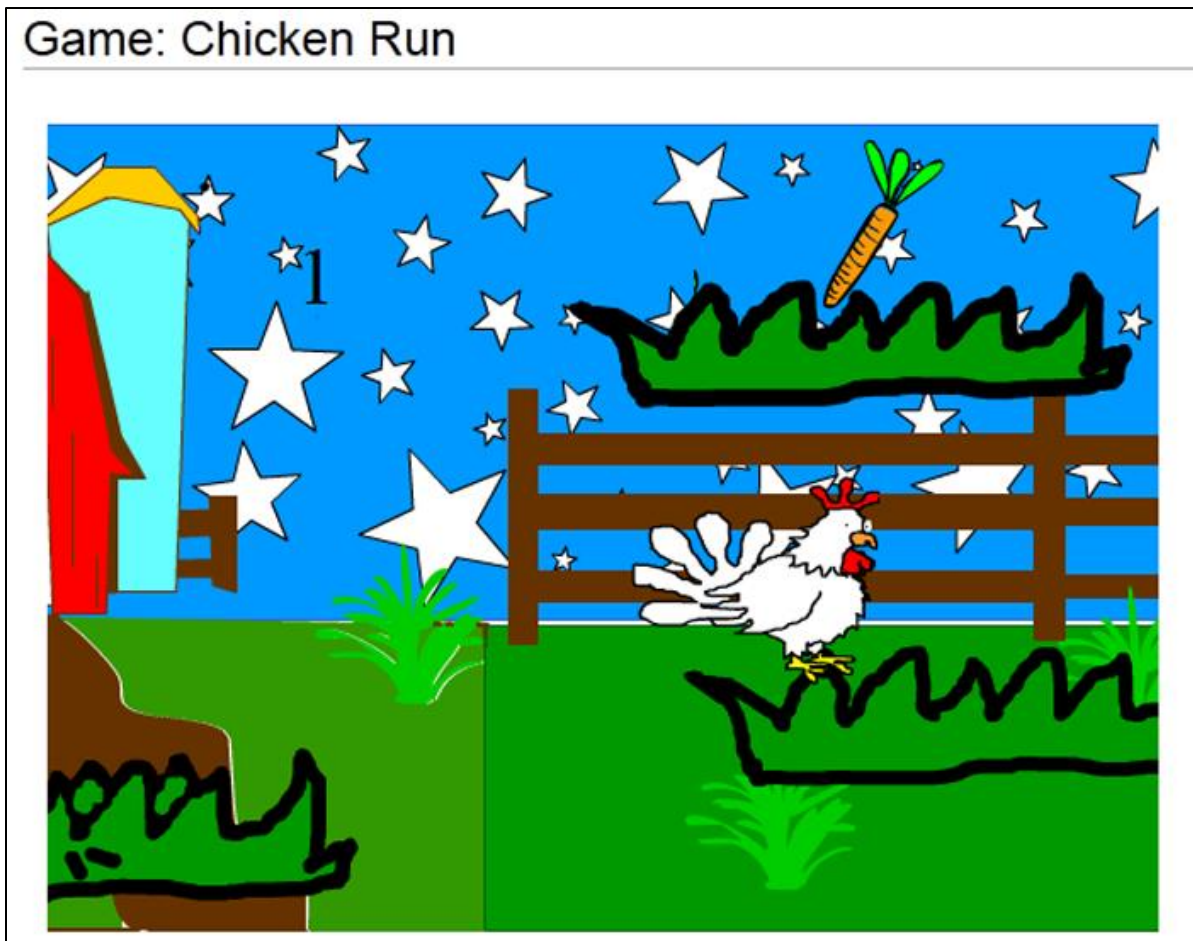


Figure 7b: A platform game that showcases the new variety of gameplay mechanics that students worked on during the 2011-2012 school year. The chicken is controlled with the Arrow keys, and can jump with the Spacebar.

It is clear that the students *do* understand how to design a wider variety of games and gameplay types in Flash (see Figures 7a and 7b), but in terms of presenting and illustrating data from a written passage, they did not show the same ability to design diverse experiences. It is still important to note that, however, that the students were able to draw comparisons between two extremely similar patterns and produce working games from these abstractions.

The observations made directly from the task show that when the option is available, students will tend to create simulations based on previous experiences or lessons. The evidence also suggests that left to their own resources, students still found it

challenging to independently extract and represent abstract relationships within grade-level text from other disciplines and translate them into the flash game format. Of all the tasks in our protocol, this one is the most difficult to accomplish effectively as it requires a higher order synthesis of all the previous CT dimensions. As a consequence, it often requires more scaffolding from instructors than do the other tasks. Regardless, the fact that most students could approach the task by relying on their previous experiences suggests that the background knowledge and skills being developed in Globaloria can effectively put students in a path that prepares them for future, more complex computational learning and transfer experiences (Bransford and Schwartz, 1999), and sheds light into further opportunities for the curriculum to provide experiences that more efficiently get students there through scaffolding and extended experiences transferring content into game form.

CONCLUSION

This study builds on the previous year's study, and continues to reveal very positive trends in the Globaloria curriculum, particularly related to last year's key conclusion points, as well as some areas in which improvement can be made. As the findings and associated tables show, the evidence from this study suggests that the Globaloria platform and its curricular structure appear to positively impact students' computational thinking skill across the board of this research model's five dimensions. In general, the improvements were seen across both knowledge and process levels of students' thinking. **11 key conclusions from our observations regarding how students' computational thinking skills evolved over this past year, and the methodology used to evaluate them are outlined below.**

1. **The overall learning ecosystem of Globaloria teaching and learning practices moved into a more strongly design-centered focus.** For both students and teachers, a stronger focus on design brought forth ways of thinking of learning and practicing computational thinking that were more centered on the learner's activity rather than the teacher's. For students, this had the particular effect of enabling more iteration and systematic exploration as a central learning strategy, that both game design and other disciplines heavy on computation widely recognize as key to success. The result are better, more creative and sophisticated game projects this year.
2. **The increase in overall proficiency with Flash can lead to increased efficacy by students on the recognition task. They also expressed more confidence in their ability to solve the problem.** Similar to learning any other tool or skill, increased time with Flash has resulted in the students knowing how to take the Marble game further

than they could last year. This higher proficiency was also observed in the Pattern Abstraction task, where more students were able to complete the piggybank game. Research on self-efficacy in domains akin to computational problem solving has shown that this increase can help students not only become better problem solvers, but keep them interested in pursuing careers in that field (Pajares & Miller, 1994; Britner and Pajares, 2001). This has particularly important implications for girls in the program, who have historically been underrepresented in STEM (Bandura and Barbaranelli, 2003; Griffith, 2010).

3. **The students appeared to be more adept at recalling and identifying patterns that they have encountered. However, the students' discourse revealed the potential for important misconceptions as to the underlying principles behind those patterns.** In the pattern recognition task students knew what collision detection was, and could identify it, but were incorrectly defining the wrong pattern due to simple text. In other words, although they knew that a collision detection pattern was present, an important minority of students described changes in visual aspects (the overlapping of shapes) as the reasons for the pattern, rather than the underlying logic of collision events in actionscript. This is an important observation, as it suggests that the computational principles behind recognition may not be as clear to students as the overt feature changes driving such recognition. Pattern recognition also necessarily took place throughout the interview, and could be observed whenever the student searched for code on their learning platform. In line with last year's observations students showed that they recognize the code or behavior that is necessary to make patterns like collision detection work, but were indexing that with code that they had previously encountered, potentially obscuring their understanding of its underlying

principles.

4. **Pattern Generalization and Abstraction were more evident this year across almost all of the tasks. In particular, their generation of simple heuristics applied to iterative refinement shows an opportunity for the curriculum to develop their abstraction skills more robustly.** The students' discourse and creation practices this year displayed the appropriation of stronger abstract computational aspects of flash into their thinking during design problems. This was particularly evident in their increased use of iterative refinement to make code they copied from the platform to their projects work specifically as intended within those projects while only a minority remained just replicating the original behavior in the new pattern. Such refinement then allowed them to develop some basic heuristics to analyze, test, and refine aspects of their projects systematically beyond the simple copy and paste tasks observed last year. Given that systems analysis and design techniques are fundamental to computer science, this points at an opportunity for the Globaloria platform to incorporate instructional modules that could take advantage of student projects to contextualize powerful computational thinking constructs.
5. All student this year demonstrated a stronger level of performance designing algorithms, however an important insight this time around was that for some students their initial perception of the problem framed the rest of their algorithm design in way that led them to a fundamentally different solution. Rather than frame this as correct or incorrect, **the important insight here is that the move to a dialogic, design centered paradigm of instruction, requires equally dialogic methods of assessment**, presenting a potential challenge to Globaloria in a climate where school districts still rely heavily on standardized assessment for high stakes decisions. **Based**

on the evidence, it can be concluded that these students' time in Globaloria has given them the ability to construct a solution with clear steps from beginning to end, and to create an effective production pipeline, a skill highly valued in a world of pervasive computer-mediated work. However, their individual ability to incorporate patterns that were not necessarily similar to those in the platform was noticeably lower than their overall ability to construct a solution. This speaks once more to the need to introduce more and more diverse examples of interactive systems using different underlying design patterns that students can then use later in their computational learning life.

6. As the decomposition findings show, interviews with Globaloria teachers revealed that they are invested in the wide range of incoming experiences that students have with video games. This carries quite a bit of weight, with regards to computational thinking, as the students' own experiences and knowledge with video games can be leaned on to make it easier to show connections between their own experiences and what is happening in the classroom. Due to the immense and increasing popularity of video games, it is important that a curriculum be flexible enough to include the incoming student experiences, which only serves to enrich the overall learning environment. However, as with the collision detection task, it is important for instructors and/or platform to address potential misconceptions that such background knowledge can bring. Knowing that teachers actively work to support the learning of game design constructs, students can feel more comfortable in calling upon their own experiences.
7. **In terms of Decomposition, the general-purpose level of decomposition that was observed has important ramifications for the students' pattern recognition and pattern generalization and abstraction, as it provides them with a starting point**

in problem solving. Later, when they are confronted with similar problems, they can use these starting patterns to identify more specific lines of code, similar to when a math student may know that a problem should use the equation, $y = mx + b$, they can then start to fill in the missing parts of the equation. Knowing that a game should have collision detection, as most games do, a Globaloria student can then search their online learning platform for the lesson on collision detection and find the appropriate code. In other words, the students were able to identify, simply through visual observation, that the system contained collision detection and that there was some sort of code that would make the marbles move. This level of decomposition is viewed through the Ideal perspective, which allows the students to identify the play abstractions within the system. More deeply and within the Material perspective, the students would also need code that would make the game switch scenes from the title scene to the gameplay scene. The ability to identify the specific pieces of code requires a higher level of mastery over the tool within which the system was created. The fact that more students were able to identify more pieces of specific codes indicates that their proficiency in Flash, and its impact on computational thinking has increased from this point last year.

8. The Globaloria curriculum is shifting towards using a more design-centered lens that balances both top-down design practices with bottom up student creativity. This is especially evident in the diversity and quality of observations, students' final games, and in the teacher interviews. Student work that was observed in the classrooms and on the showcase have provided evidence that they are now **integrating content and gameplay**, instead of using direct facts interspersed with limited gameplay to relay messages. This point is important as it addresses one of the main conclusions from

last year, which was that students' problem-solving ability was limited, thinking in terms of very rote and declarative facts. In terms of computational thinking, this would provide little evidence that the students would be able to recognize and apply patterns to novel problems.

9. In terms of the three dialogs, the evidence from this study show that students are shifting from thinking strongly at the material dialog to thinking more towards a combination of material and real player dialogs. Again, the student work that was observed in both the classrooms and in the student showcase provides evidence of this shift. This addresses another key point from last year's study, which was that the Globaloria platform might want to consider shifting towards a more design-centered lens (Figure 7b).

10. Last year's study emphasized the need for Globaloria to shift away from an instructionist driven classroom to a more design-centered one. This would provide an enhanced opportunity to develop computational thinking skills, and increase students' self-efficacy and attitudes towards computer science and STEM topics. This year's study found data that Globaloria has indeed begun to make that shift. Students are now integrating content into the gameplay, they are (and had been) performing research independently and as part of a team, they are creating a wider range of gameplay experiences, and they are showing deeper levels of computational thinking. The primary recommendation for Globaloria is to increase this focus, as well as diversify the toolset and the types of media to which students are exposed. Learning to design the same core game in different tools (Java, C++, Unity, etc.), as well as in different forms (video, music, etc.) will only enhance the learning experience. This can also better prepare students for careers in the STEM

fields.

11. One of the key observations in this study was that while the analysis methodology continues to show much promise in bringing to light important ways in which Globaloria is shaping students' computational thinking practices and skills, the study is limited by our ability to generalize those findings to most students in the program due to the small student sample sizes afforded by the qualitative nature of the study. In order to address this, we believe more robust and generalizable findings could be obtained by taking advantage of the data available in the Globaloria platform backend, and by implementing the tasks in our instruments in more automated way within the platform so they can reach a larger number of students at a time. In this way, we could then rely on currently available analytics technologies and methods to automatically code and interpret patterns in students' Globaloria progress in a more scalable and generalizable way.

REFERENCES

1. Bandura, A., Barbaranelli, C., Caprara, G.V., and Concetta Pastorelli. Self-Efficacy Beliefs as Shapers of Children's Aspirations and Career Trajectories. *Child Development*. 72(1), 187-206.
2. Barr, D., Harrison, J, and Connery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Leading and Learning with Technology*. (March/April).
3. Barr, V. and Stephenson, C. (2011). Bringing Computational Thinking to K-12: What's Involved, and What is the Role of the Computer Science Education Community? *ACM Inroads*, 2(1), 48-54.
4. Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, 1(2), 65.
5. Brown, A. L. and Campione, J. C. (1996). Innovations in learning: New environments for education. In L. Schauble, & R. Glaser (Eds), *Psychological Theory and the Design of Innovative Learning Environments: On Procedures, Principles, and Systems*, (pp. 289-325). Hillsdale, N.J, England: Lawrence Erlbaum Associates.
6. Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *Review of research in education*, 24, 61-100.
7. Cottrell, N. B. (1968). Performance in the presence of other human beings: Mere presence, audience, and affiliation effects. In E. C. Simmel, R. A. Hoppe, and G. A. Milton (Eds.), *Social Facilitation and Imitative Behavior* (pp. 91-110). New York, NY: Holt, Rinehart, and Winston.
8. Engstrom, M. E. and Jewett, D. (2005). Collaborative learning the wiki way. *TechTrends*, 49(6), 12-15.

9. Games, I. A. (2008). Three Dialogs: a framework for the analysis and assessment of twenty- first-century literacy practices, and its use in the context of game design within *Gamestar Mechanic*. *E-Learning and Digital Media*, 5(4), 396-417.
10. Games, I. A. (2010). Gamestar Mechanic: Learning a designer mindset through communicational competence with the language of games. *Learning, Media and Technology*, 35(1), 31-52.
11. Games, I. A. and Squire, K. D. (2011). Searching for the Fun in Learning: A Historical Perspective on the Evolution of Educational Video Games. In S. Tobias and J. D. Fletcher (Eds.), *Computer Games and Instruction* (pp. 17-46). Charlotte, N.C.: Information Age Publishing.
12. Gee, J.P. (1992). *The social mind: Ideology and social practice*. New York, NY: Bergin & Garvey.
13. Gee, J.P. (2003). *What Video Games Have to Teach Us About Learning and Literacy*. New York, NY: Palgrave Macmillan.
14. Gee, J. P. (2005). *An Introduction to Discourse Analysis: Theory and Method*. New York, NY: Routledge.
15. Gee, J.P., Hull, G.A., and Lankshear, C. (1996). *The New Work Order: Behind the language of new capitalism*. Sydney, Australia: Allen & Unwin.
16. Geertz, C. (1973). Thick description: toward an interpretive theory of culture. In *The Interpretation of Cultures: Selected Essays* (pp. 3-30). New York: Basic Books.
17. Globaloria (2011). *What is Globaloria?* Retrieved from <http://www.globaloria.org/intro#WhatIsGlobaloria>
18. Google (2010). What is Computational Thinking? Retrieved from <http://www.google.com/edu/computational-thinking/what-is-ct.html>

19. Griffith, A. (2010). Persistence of women and minorities in STEM field majors: Is it the school that matters? *Economics of Education Review*, 29(6), 911-922.
20. Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, 42(1), 38-43.
21. Harel, I. (1991). *Children Designers: Interdisciplinary constructions for knowing and learning mathematics in a computer-rich classroom*. Norwood, NJ: Ablex Publishing.

22. Harel, I., & Papert, S. (1991). Situating Constructionism. In I. Harel and S. Papert (Eds.), *Constructionism* (pp. 1-12). Norwood, NJ: Ablex Publishing.
23. Hayes, E. R. and Games, I. A. (2008). Making Computer Games and Design Thinking. *Games and Culture*, 3(3-4), 309-332.
24. Lakoff, G. and Johnson, M. (1980). *Metaphors to Live By*. Chicago: University of Chicago Press.
25. Lee, I., Martin, F., and Denner, J., et.al. (2011). Computational Thinking for Youth in Practice. *ACM Inroads*, 2(1), 32-37.
26. National Academy of Sciences (2010). Report of a Workshop on the Scope and Nature of Computational Thinking, Washington, D.C.: National Academies Press.
27. Pajares, F. and Miller, M.D. (1994). Role of self-efficacy and self-concept beliefs in mathematical problem solving: A path analysis. *Journal of Educational Psychology*, 86(2) 193-203.
28. Pea, R. (1997). Practices of Distributed Intelligence and Designs for Education. In G. Salomon (Ed). (1997). *Distributed Cognitions: Psychological and Educational Considerations*. Cambridge, UK: Cambridge University Press.
29. Rieber, L. P. (1996). Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research and Development*, 44(2), 43-58.
30. Shaffer, D. W. (1996). *Escher's World: learning mathematics and design in a digital studio*. Cambridge, Mass: MIT Press.
31. Stake, R. (1995). *The art of case research*. Newbury Park, CA: Sage Publications.
32. Vygotsky, L. S. (1962). *Thought and language*. Cambridge, Mass: MIT Press.
33. Vygotsky, L. S. (1978). Interaction between learning and development. In *Mind and*

- Society* (pp.79-91). Cambridge, MA: Harvard University Press.
34. Wertsch, J. V. (1998). *Mind as Action*. New York, NY: Oxford University Press.
 35. White House. (2009). President Obama launches “Educate to Innovate” campaign for excellence in science, technology, engineering & math (STEM) education. Retrieved from the White House website: <http://www.whitehouse.gov/the-press-office/president-obama-launches-educate-innovate-campaign-excellence-science-technology-en>.
 36. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
 37. Wing, J. M. (2010). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society*, 366, 3717-3725.
 38. Wiggins, G. and McTighe, J. (2005). *Understanding by Design*. Washington, D. C.: Association for Curriculum Development.
 39. Wilensky, U. and Resnick, M. (1999). Thinking in Levels: A Dynamic Systems Approach to Making Sense of the World. *Journal of Science Education and Technology*, 8(1), 3-19.