Examining Trends in Adolescents' Computational Thinking Skills within the Globaloria Educational Game Design Environment

Alex Games
Assistant Professor
Telecommunication, Information Studies, and Media
Michigan State University
428 Communication Arts and Sciences
East Lansing, MI 48824
games@msu.edu

Luke Kane
Research Assistant
Telecommunication, Information Studies, and Media
Michigan State University
258 Communication Arts and Sciences
East Lansing, MI 48824
kaneluke@msu.edu

ABSTRACT

Today there is an increasing demand by companies, governments, and society for people who know how to think computationally (i.e. think critically, logically, and solve problems in innovative ways using computational tools) (Wing, 2006[21]; National Academy of Sciences, 2010[14]), in order to be competitive in the knowledge economy. Educational video game design has shown potential in helping to prepare youth with skills germane to computational thinking, and the so-called STEM disciplines whose practices heavily rest on computation (Games, 2010[5]; Hayes and Games, 2008[13]).

The potential has been recognized by the White House's efforts (White House, 2009[20]) to support educational video game design, including national game design contests and supporting programs that teach computational thinking.

This study examined one such program, Globaloria, whose 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. Adobe Flash as the platform for developing these games. Globaloria classrooms are designed around constructionist pedagogies (Papert and

Harel and Papert, 1991[15]; Harel, 1991 [20]), 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[5]) 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. Findings suggest that a scaffolded game design-based curriculum can provide an effective context for students to develop computational thinking and deep understandings and engagement with STEM subjects, all while in forms valued within the 21st century workplace.

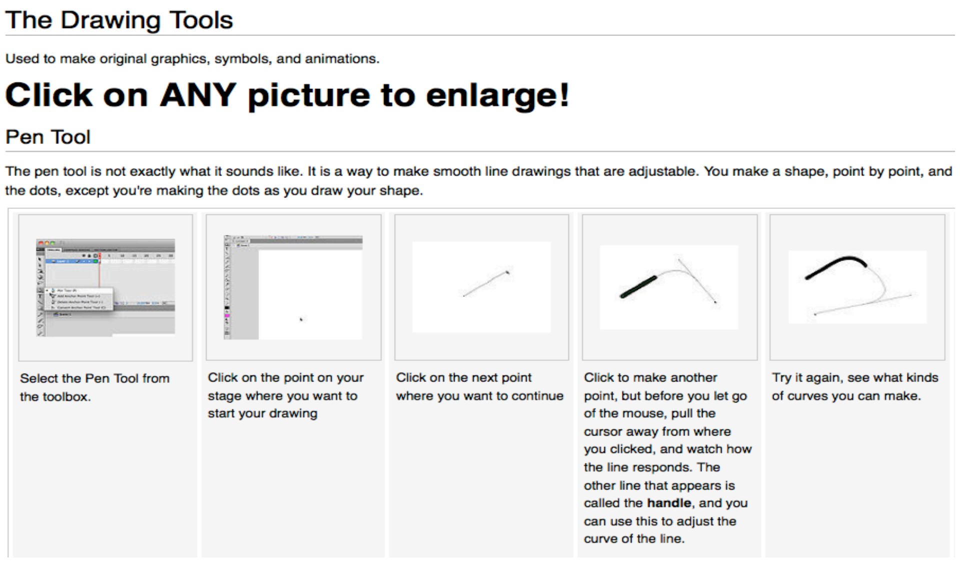
INTRODUCTION

This report concentrates on research the authors conducted for Globaloria, the first and largest social learning network where students develop STEM knowledge, digital literacy and global citizenship skills through game design. Globaloria is an education innovation invented by the World Wide Workshop Foundation (www.WorldWideWorkshop.org).

GLOBALORIA

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[11]). 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).



The Drawing Tools

Used to make original graphics, symbols, and animations.

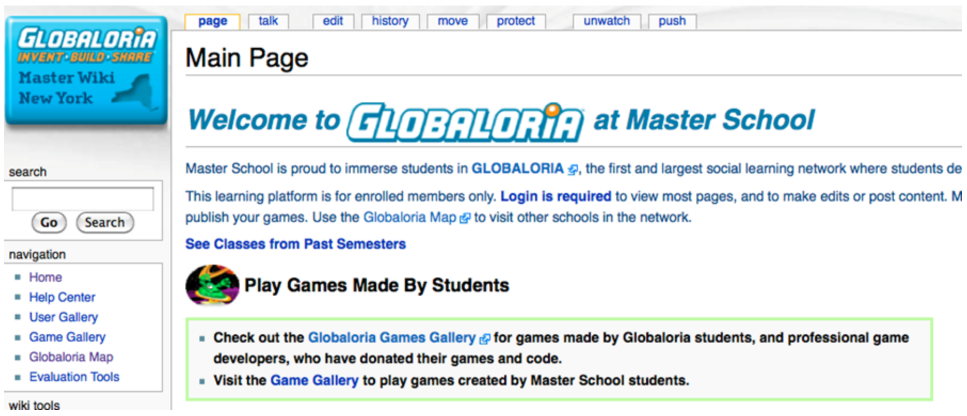
Click on ANY picture to enlarge!

Pen Tool

The pen tool is not exactly what it sounds like. It is a way to make smooth line drawings that are adjustable. You make a shape, point by point, and the dots, except you're making the dots as you draw your shape.

Select the Pen Tool from the toolbox.	Click on the point on your stage where you want to start your drawing	Click on the next point where you want to continue	Click to make another point, but before you let go of the mouse, pull the cursor away from where you clicked, and watch how the line responds. The other line that appears is called the **handle**, and you can use this to adjust the curve of the line.	Try it again, see what kinds of curves you can make.

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



Main Page

Welcome to GLOBALORIA at Master School

Master School is proud to immerse students in GLOBALORIA, the first and largest social learning network where students de

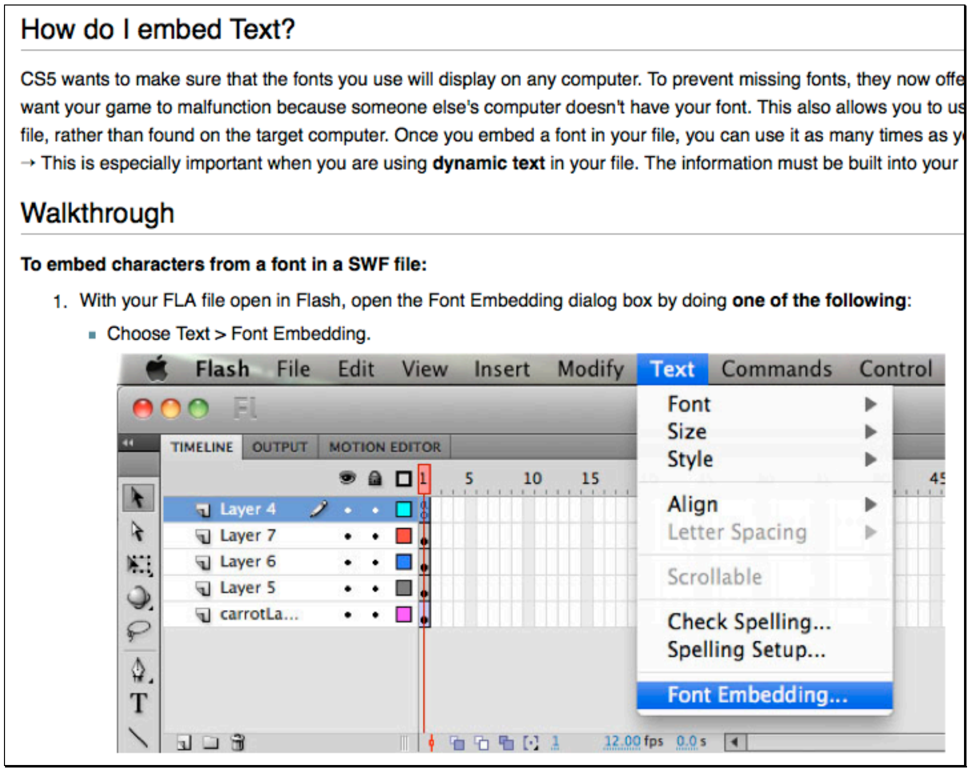
This learning platform is for enrolled members only. Login is required to view most pages, and to make edits or post content. M publish your games. Use the Globaloria Map to visit other schools in the network.

See Classes from Past Semesters

Play Games Made By Students

- Check out the Globaloria Games Gallery for games made by Globaloria students, and professional game developers, who have donated their games and code.
- Visit the Game Gallery to play games created by Master School students.

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 offe
want your game to malfunction because someone else's computer doesn't have your font. This also allows you to us
file, rather than found on the target computer. Once you embed a font in your file, you can use it as many times as y
→ This is especially important when you are using **dynamic text** in your file. The information must be built into your

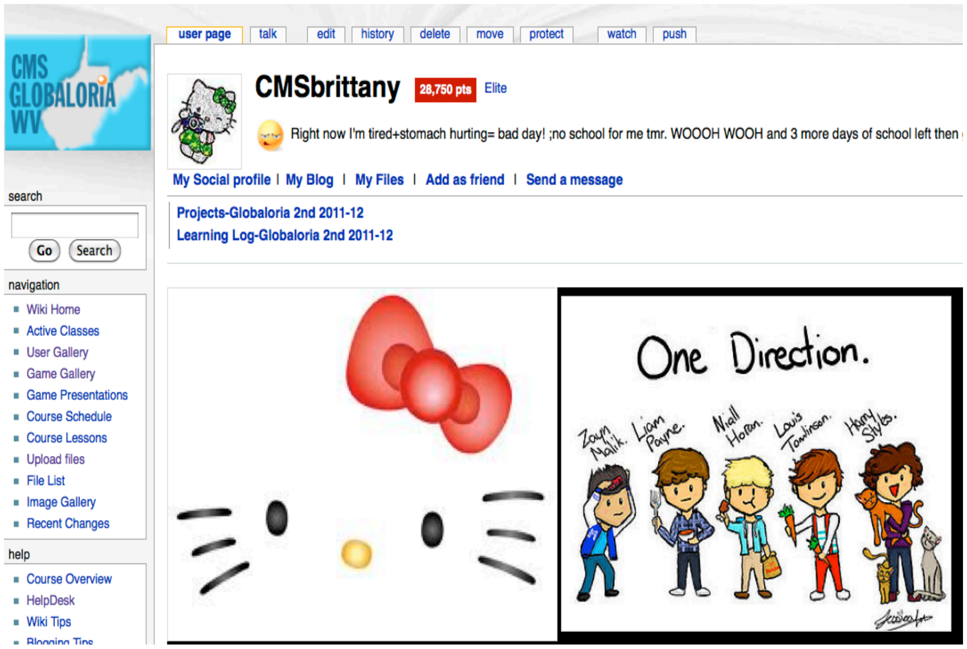
Walkthrough

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).

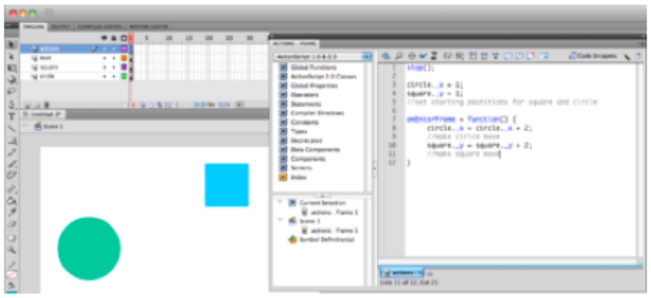


Right now I'm tired+stomach hurting= bad day! ;no school for me tmr. WOOOH WOOH and 3 more days of school left then

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 (inks 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).

```
onEnterFrame = function() {
        circle._x = circle._x + 2;
            //make circle move
        square._y = square._y + 2;
            //make square move
```



```
stop();

circle._x = 1;
square._y = 1;
//set starting positions for square and circle

onEnterFrame = function() {
        circle._x = circle._x + 2;
        //make circle move
        square._y = square._y + 2;
        //make square move
}
```

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 our 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, assets needed to be created, 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[6]; National Academy of Sciences, 2010[14]). 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[22]) 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[8]). 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

Computational thinking is a way of thinking and solving problems effectively using the logical, mathematical, and representational tools that computers make available work our way through data, and transform it into actionable information. Over the years, multiple frameworks of computational thinking have been proposed by scholars, each emphasizing different aspects of the construct, from the more abstract and logical operations necessary to construct a software algorithm, to the more social aspects involved in solving a complex computational problem collaboratively (National Academy of Science, 2010[14]). In order to operationalize the construct for this particular study, we turn to a framework recently proposed by Google (2010)[12], which synthesizes most of these perspectives according to the actual practices of software professionals today. 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.
2. **Pattern Recognition** is the ability to see or identify recurring systemic connections between objects, as well as recurring subsystems in systems.
3. **Pattern Generalization and Abstraction** is the ability to recall previously encountered patterns and use them to aid in the solving applicable problems in different contexts; a form of near transfer that is characterized by students' ability to abstract.
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.
5. **Data Analysis, Modeling, and Visualization** 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 be one as well.

We chose these five dimensions given that they present two advantages over those presented in other frameworks: 1) They 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, 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[26]).

GAMES' THREE DIALOG FRAMEWORK OF GAME DESIGN AND CT

In a series of studies of children designing their own videogames, Games (2008[4]; 2010[5]) 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 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 materials 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. Rooted in Discourse theory (Gee, 2005; Games, 2008), the framework sees these dimensions as dialogic in nature, as a conversation between a designer and one or more players, mediated by the game. Dialog is a process of iterative progression toward a common construction of meaning between 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 involving 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, 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, Denner, et. al., 2011).

For this to take place, the environment must encourage and support learner self-regulation and empowerment as keys to success. A key part of this process is to provide learners with models of effective practices in the CT domain of interest, in this case, with good models of game design, so they can begin to analyze and ultimately appropriate those strategies later on, a process that Wiggins and McTighe (2005) have called understanding by design. As shown in previous studies (Games, 2008; 2010; Repenning, 2009), when this is the case, learners' design decisions, language for describing those decisions, and the tools and artifacts they involve in enacting them, increase in sophistication and depth (for example, shifting from an unplanned, in the moment strategy of creation, to a planned and deliberate one). In this study then, the authors' goal was to examine the ways in which the ecosystem of learners and learning environment shape learners' decisions, words, and tool use to reflect five dimensions of computational thinking over time.

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 questions:

1. In what ways does students' game design discourse evolve over time within the Globaloria context?
2. How do skills and practices of computational thinking evolve as students' discourse evolves within Globaloria?

To answer these questions, we used a qualitative methodology of case studies (Stake, 1995[16]) and -consistently with the three dialog framework-, multimodal discourse analyses (Gee, 2005[9]) 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[10]) 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[7]; Vygotsky, 1978[18]; Wertsch, 1998[19]; Engstrom, 2005[3]), 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 through 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 (e.g., board, scorecard)
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.

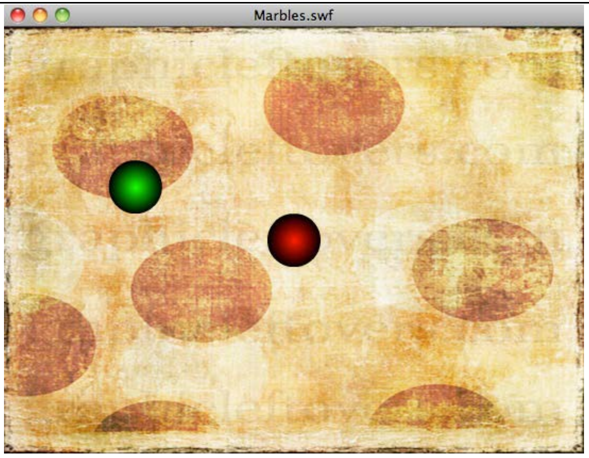
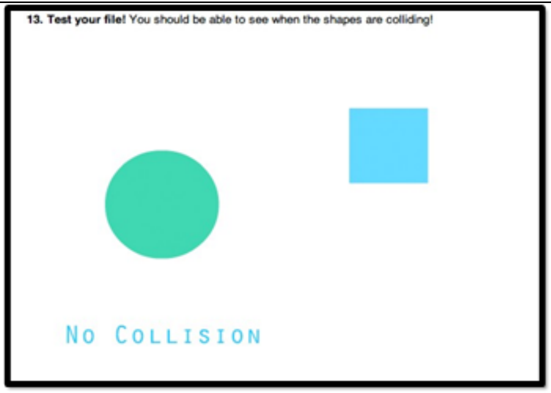
Data were collected from pre- and post-assessment protocols conformed that consisted of individual interactive interviews with Globaloria students, as well as from observations of their design activities during Globaloria class over a period of 4 months.

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:

Computational Thinking Dimension	**Interview Task Sample**
Decomposition: Students were shown a simple game made in Flash (see Figure a) and asked to recreate it. The goal of this exercise was to establish whether or not the students could look at a game in Flash and break it down into its core components. Successful completion of the task was not reliant upon the actual recreation of the marble game, but how deeply they could deconstruct the given system.	 *Figure a. The Marble game*
Pattern Recognition: Students were instructed to find the "Collision Detection" tutorial on the curriculum and watch the brief collision detection animation, which shows two objects moving perpendicular to each other and passing through one another without any sign of a collision taking place (Figure b). The students then played a short, three-level flash game depicting a seal trying to escape sharks moving across the	 **13. Test your file!** You should be able to see when the shapes are colliding! *Figure b. The collision detection tutorial*

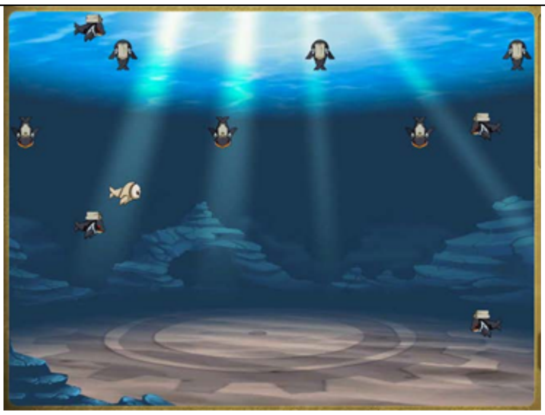
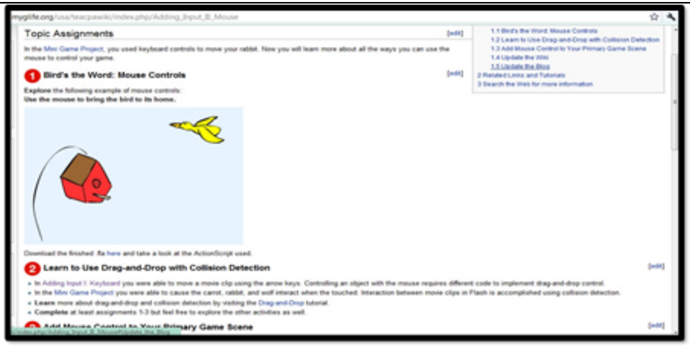
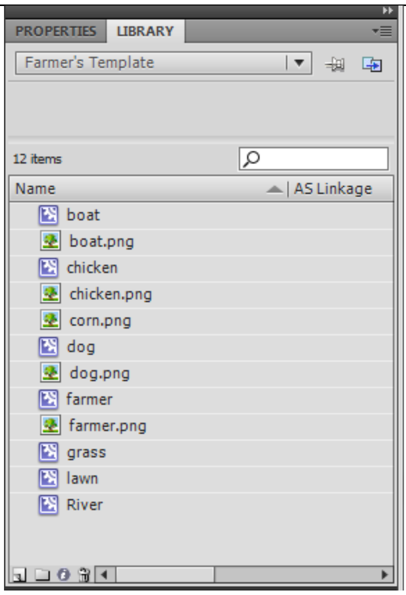
screen in random patterns (Figure c). One of the levels had no collision detection, and thus the sharks would pass through the seal without bouncing off of it or damaging it. The other two levels implemented collision detection, and the seal would be damaged and eventually killed by every shark touching it, an event which would then bounce the shark in the opposite direction. Students who were able to successfully complete this task were able to correctly identify which level without collisions as most accurately resembling the same pattern in the tutorial animation, and explain why.	 *Figure c. The flash game*
Pattern Generalization and Abstraction: Students were directed to the "Adding Input II: Mouse" tutorial on their learning platform wiki pages. Once there, the students were played a simple interactive system wherein a bird can be dragged to a birdhouse (using mouse input) (see Figure 3). The students were then asked to build a system using the drag and drop pattern, using money as the theme for the simulation.	 myglife.org/usa/teacpawiki/index.php/Adding_Input_II_Mouse Topic Assignments [edit] In the Mini Game Project, you used keyboard controls to move your rabbit. Now you will learn more about all the ways you can use the mouse to control your game. 1 Bird's the Word: Mouse Controls [edit] Explore the following example of mouse controls: Use the mouse to bring the bird to its home. Download the finished .fla here and take a look at the ActionScript used. 2 Learn to Use Drag-and-Drop with Collision Detection [edit] • In Adding Input I: Keyboard you were able to move a movie clip using the arrow keys. Controlling an object with the mouse requires different code to implement drag-and-drop control. • In the Mini Game Project you were able to cause the carrot, rabbit, and wolf interact when the touched. Interaction between movie clips in Flash is accomplished using collision detection. • Learn more about drag-and-drop and collision detection by visiting the Drag-and-Drop tutorial. • Complete at least assignments 1-3 but feel free to explore the other activities as well. 3 Add Mouse Control to Your Primary Game Scene [edit] 1.1 Bird's the Word: Mouse Controls 1.2 Learn to Use Drag-and-Drop with Collision Detection 1.3 Add Mouse Control to Your Primary Game Scene 1.4 Update the Wiki 1.5 Update the Blog 2 Related Links and Tutorials 3 Search the Web for more information *Figure d. The drag and drop lesson in the learning platform wiki*
Algorithm Design: The task given to the students to demonstrate Algorithm Design was to animate in flash the solution to the classic logic puzzle, the Farmer's Dilemma (Kordemsky, cite). To complete the task effectively, students were given a library of flash symbols depicting the core elements of the puzzle (i.e. a farmer, a dog, corn, river, and so on, see Figure e) and were asked to create a flash movie that would show the solution to the puzzle. If done properly, this would require them to describe the logical steps to solve the problem (it's algorithm), before or during the implementation of the design.	 PROPERTIES LIBRARY Farmer's Template 12 items Name AS Linkage boat boat.png chicken chicken.png corn.png dog dog.png farmer farmer.png grass lawn River

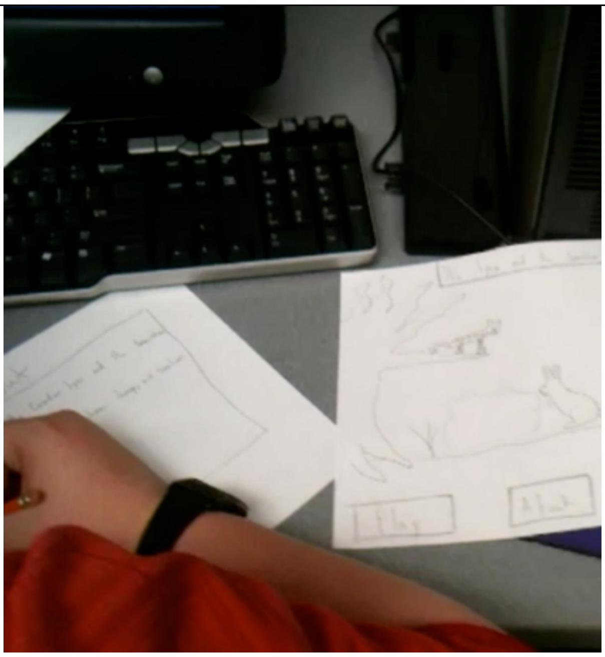
	Figure e. The farmer's dilemma flash library
Data Analysis and Visualization: This task consisted of the students reading a passage from a textbook explaining the predator-prey relationship between the populations of Canadian lynx and snowshoe hare, including tables displaying the number of population members for each animal, measured in the wild by scientists on a monthly basis during a 12 month period. Using this information and any other online resources of their chose, the task asked them to creating a game depicting the key ideas in the passage using Flash. This was tested both in an individual format during the interviews, as well as in a group format, during a one hour Globaloria classroom session.	 Figure f. The lynx and snowshoe hare task

Table 3. The individual interview protocol

Data were collected from middle schools in Austin, Texas and Lewisburg, West Virginia. Due to the number of schools running Globaloria, site selection was random in that state. However, the East Austin College Preparatory Academy (EAPrep) 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.

Site	Students	Pre	Post
EAPrep, Austin, TX	15	1/24/11 - 1/25/11	5/2/11 - 5/3/11
Eastern Greenbriar Middle School, Lewisburg, WV	15	2/7/11 - 2/8/11	5/24/11 - 5/26/11

Table 4: Sample and dates

RESULTS

Decomposition

By and large, students displayed an increase in sophistication on decomposition strategies between pre and post-test. At pre-test, none of the students had been able to clearly identify in discrete ways either the abstractions or the physical elements that they would use in the process of reproducing the clip. At post-test however, two-thirds of the students (22 out of 30) were able to identify marbles, the background, and code to move the marbles as necessary components of the game during the post-assessment, a clear improvement in their understanding of the materials necessary to begin reproducing the animation.

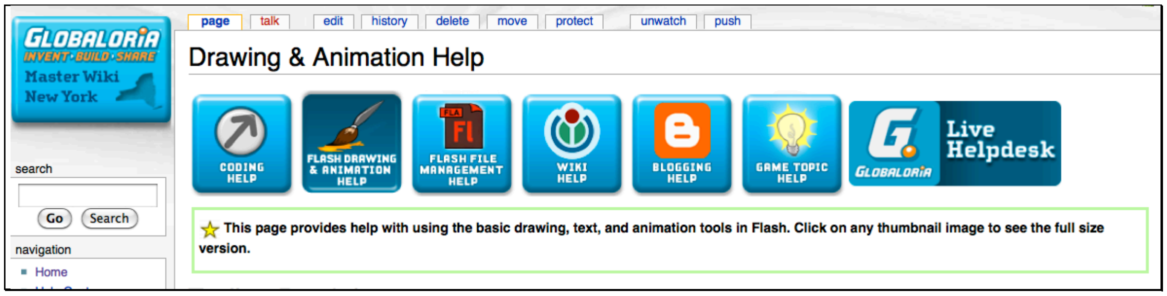
At the ideal level, their discourse showed that a key change in their decomposition skill was driven by their appropriation of the specialist language of interactive media embedded in the Flash authoring system. An analysis of the discourse they used in their description of the process they would follow to recreate the clip at pre-assessment was represented by phrases such as *"I would have to put the marbles and make them move"*. In contrast, by post-assessment, the same statement took the form *"I need to create two marble movieclips on the stage, and then use the codes to program them to move"*. Other terms such as *frames, scenes, in addition to the stage, codes and movieclips* also became very common in their lexicon during the interviews and during class observations.

This appropriation is important, for as Games (2010), and other scholars (Gee, 2007; Lakoff, cite) have previously argued an expanded lexicon within a specialist discourse such as game design gives learners "tools to think with", that help them organize their mind and understand the abstractions and nuances used by experts in those fields, in addition to enabling them to communicate and learn from discourse with more experienced others. In this case, such language allowed learners to more deeply discuss and understand the abstract relationships between the visible elements of the game (the marbles on the stage), and the systemic interactions between them represented by the software.

This was particularly evident in their design decisions and use of software tools at post-test, and most interestingly, also revealed two very different strategies. The first one, observed predominantly among students in West Virginia, was to rely on the use of direct timeline animation utilizing movieclips and motion blend techniques to reproduce the marble collision animation directly on the timeline. This was in general a more sophisticated approach at decomposing the problem than what the same students generally showed at pre-test, and demonstrated their understanding of some important abstractions, such as the use of vector graphics with interpolated offset positions using Flash motion blends as a way to break down the problem (a similar technique to that used by animators with flipbooks in the days prior to computer graphics), but was more limited than the second approach in that it would limit their exposure to the interactive functionality in the marble game.

By contrast, students in Texas tended to strongly approach the decomposition of the problem by considering Actionscript programming a key component of the problem-solving strategy. The largest increase in decomposition sophistication observed was at the level of code tied directly to changes in visual elements.

The use of vector graphics circles turned into movieclips was a nearly universal step used by students as a way to start designing a solution. Similarly, the need to create a background, either by drawing it or using an image the researcher provided to them was also widely understood. With these elements in place, their understanding of the connection between them and code was strongest in their use of the _x and _y properties of Flash movieclips to store offset values that would make the marbles translate along their respective Cartesian axes, every time a new frame was painted on screen by Flash. The most common approach to doing this was by navigating to a lesson in the curriculum on the learning platform specifically focused on animating objects within game control loop, and then copying and pasting the Actionscript code indicated as necessary to create motion on a stage object into their own code before modifying it to fit the nomenclature given to their own clips (see figure x below).



Drawing & Animation Help

This page provides help with using the basic drawing, text, and animation tools in Flash. Click on any thumbnail image to see the full size version.

Figure 4: An Example of the Globaloria Platform and Curriculum Help Desk Resources on Drawing and Animation

The above process has important implications for understanding the evolution of students' computational thinking in Globaloria, highlighting that pedagogical effectiveness is contingent, at least partly, on the students' ability to draw connections between the code samples in the learning platform lessons and their final course projects.

In both the post assessment interviews as well as the projects uploaded to the learning platform, it was evident that students relied very strongly on learning platform searches and a process of copying code, pasting it, and then observing the result by executing the movies they had designed. However, a clear limitation in their strategies by that point was that while they seemed to have memorized a set of cause consequence chains (e.g. x code has y effect), their recall was more akin to rote memorization than understanding. When asked to elaborate on why they thought a specific code would do what it did, they were seldom able to explain its general function, and for code with more abstract systemic implications they usually had no idea of its effects. Clear examples of this were the functions *hitTest(object)*, which generates an event when an object has collided with another on the stage, and "gotoAndStop(*scene#*), which sends the movie to a specific scene, removing any elements existent on the stage and drawing new ones. In both of these cases, the functions were widely used in the students' final projects, however, during the interviews only half of the students knew where the collision detection code would be necessary (16 out of 30), and only 4 students knew when the switch scene function would be useful.

Interestingly, between pre-test and post-test, none of the students were able to identify interactivity (a player pressing on a marble to start its motion), as a core piece of the problem, let alone use Actionscript code to make this happen. This indicates that at a real player level, the Globaloria curriculum was not as strong at connecting individual students to the notion of an audience for their designs as it could have been. In some ways this may explain the strong similarities between most final projects students created at the end of the course.

As a whole, these findings suggest a general trend toward increasing sophistication in computational thinking, albeit not one following a uniform path. In line with the notion of epistemological diversity (Papert and Harel, cite), this points to an opportunity in the Globaloria curriculum to provide students with opportunities to discover concepts that one path might provide while the other one might not and thus complement students' learning.

On the other hand, while the trend by students to memorize the location of pieces of code with certain functionality definitely points to a powerful mechanism at work in the curriculum for students to appropriate new knowledge, it is also clear that its scope may be limited in terms of driving student understanding of the systemic nature of software and games. However, it is possible that if within the learning platform and curriculum multiple exercises involving the use of the functions mastered in different contexts were provided (perhaps in a following academic year), such understanding would be more likely to emerge.

Pattern Recognition

Pattern recognition was a dimension for which students showed substantial changes in sophistication between pre and post assessment. A majority of the students that we interviewed went from being unable to articulate the similarities between the game levels and tutorial, to being able to do so (23 out of 30 by post-test). Their discourse indicated that from a material perspective, they were able to recognize at a visual level the pattern of interaction between system components at the level of their movement along a Cartesian axis. From an ideal play perspective, their discourse suggested that a key driver of their pattern recognition at this level was having a mental model mapping player actions to the movements in the game. In this way, they were able to map their control of the seal in the Flash game to the circle in the tutorial, and the square to the shark. However, most students (18 out of 30 by post-test) failed to identify the lack of a collision event (signaled by a visual and auditory signal showing the seal being damaged when touching a shark, and a change in its direction as a result of this event) as the fundamental differentiator between the level with the most similar pattern (level one, just recognized by 5 students) and the rest of the levels.

This was surprising considering that when asked during the interviews, most students had recognized collision detection (and the code in the learning platform to make it happen), as necessary to replicate the marble game. However, in this task, most students were able to identify a lack of Actionscript code as a common pattern between the tutorial and the game, which once more points to an increasing sophistication in their material ability to tie the use of code in making games, but also reinforces the previous tasks' evidence that students tended to acquire a narrow interpretation of the uses of the curriculum and learning platform constructs by developing constrained cause-consequence pattern recognition. This might support the notion that their understanding of the underlying principles of a pattern like collision detection might be limited.

Pattern Generalization and Abstraction

As with the first two dimensions, the majority of students showed an increase in sophistication in taking the drag and drop pattern and applying it to a different context, and in the same manner, the sophistication grew most strongly in the use of concrete materials and visual representations than in their understanding of the underlying abstractions tied to their code.

By post-test, all students consistently were able to use the Flash authoring system to construct vector graphics representations that by and large resembled money and a piggybank, money and a bank or something very close to that. From both material and ideal play perspectives it was clear that the students were able to use analogical reasoning to transfer the drag and drop pattern presented in the bird and birdhouse tutorial to the new context, through an understanding of the pattern as the basis of a coherent interactive system modeled on the relationship between money and a piggybank. This was consistent with their approach to replicating the Marble game, which often started with replicating the visual assets. Interestingly at pre-test, when the time came to use code to recreate the drag and drop interaction for a user, most students relied on the copy paste technique, and used the Keyboard Input code in the Globaloria learning platform, which they were learning about at that moment, rather than the mouse input used by the bird tutorial.

An analysis of both their discourse and tool use process suggests that like in the previous dimensions, students tended to rely on their previous knowledge of cause-consequence chains between code and visual elements to think about the relationship of code and interactivity with visual objects.

By post-assessment however, their sophistication in this task had increased, and their discourse and design decisions showed that their increased experience with code allowed them to see the pattern beyond relationships in the objects moving on screen, and seeing a system where underlying code enabled players to interact with the objects in a specific way using the mouse.

Evidence of this improvement is supported by an analysis of their tool use. That is, navigating the learning platform to the drag and drop lesson, finding and downloading the bird and birdhouse .fla file, examining its Actionscript code, and copying and pasting the code into their own games.

For this pattern, it was evident that generalization involved thinking about the system not only at a material level, but at an ideal and real play level as well, something that can be confirmed by the fact that many student final projects posted on the learning platform relied on a similar drag and drop mechanism to work.

By successfully taking and applying the code from the tutorial to their new game, the students were showing an understanding of the difference between the two games as aesthetic, with the same underlying code functionality, a skill fundamental in areas like computer science and data analysis.

Algorithm Design

The analysis of the data collected suggests that at this level, students displayed the least amount of change in sophistication over time. In general, the students in this sample struggled with figuring out the Farmer's Dilemma. Most remarkably, neither at pre- nor post-assessment did a substantial number of children articulate through their discourse or their design activities a plan or set of discrete steps for solving the puzzle before beginning to construct the system.

Only two of the students were able to articulate the solution in either session, and they indicated that the reason was that they had encountered a similar problem in class before. This point is important, as it also points to the need for the Globaloria curriculum to develop ways that enhance learners' familiarity with a broader range of system patterns, an aspect central to design instruction that may help positively impact students algorithmic thinking more broadly.

As a general rule between pre- and post-assessment, the students still displayed a strong bend toward thinking in terms of material representations, often disconnected from an underlying model of systemic interactions between components. In both pre- and post-assessments, a third of the students tried to animate the solution to the puzzle (a possible solution to the problem, but not an optimal one), ignoring the fact that creating a simple drag and drop system would allow a player to solve the puzzle by hand. The rest of the students attempted an interactive solution, and were split between using the Keyboard Input and using the Mouse Input to solve the problem, once more pointing to their generalization of well-known patterns, but without a clearly planned out approach to applying those patterns to an overall solution.

Information Analysis and Visualization

Consistent with the observations in the previous dimensions, students relied heavily on visual representations and previously used patterns taken directly from the curriculum both in the individual and the group formats. In the individual format, there were substantial differences between pre- and post-assessment. By post-assessment, it was clear from an analysis of the first two levels of play (material and ideal) that students had moved from barely being able to begin solving the problem without prompting, to using key points in the data to visually represent the relationship in the passage using Flash symbols on the stage.

They also understood the need to use Actionscript to communicate to a player the systemic relationships between these elements (through the assets, Actionscript, and the gameplay). Consistently with the observations in the previous tasks, their use of software and interaction patterns was still somewhat narrow and rote, which limited their ability to use them to use Flash to depict some of the more nuanced aspects of the passage (e.g. that the population measurements show the population of lynx trailing that of hares both in growth and decline throughout the year).

The result, was that just as with most of their final projects, the games created individually tended to "re-skin" (a term used to refer to using a different visual element connected to the same piece of software code) the software patterns directly collected from the learning platform, from a simple tutorial using a wolf, a bunny, and a carrot (figure 5a), to the same tutorial replacing the wolf for a lynx, and the bunny for the snowshoe hare (figure 5b), without any consideration for the food sources.

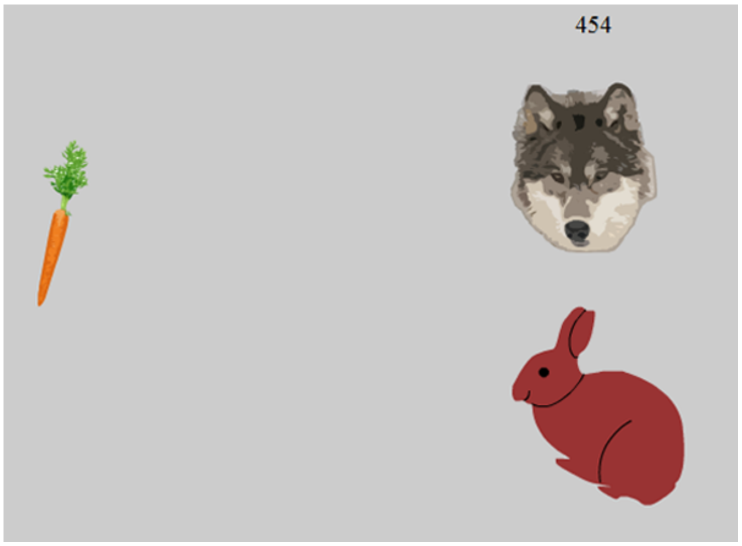


Figure 5a: The Adding Input: Keyboard mini game from the Globaloria learning platform wiki

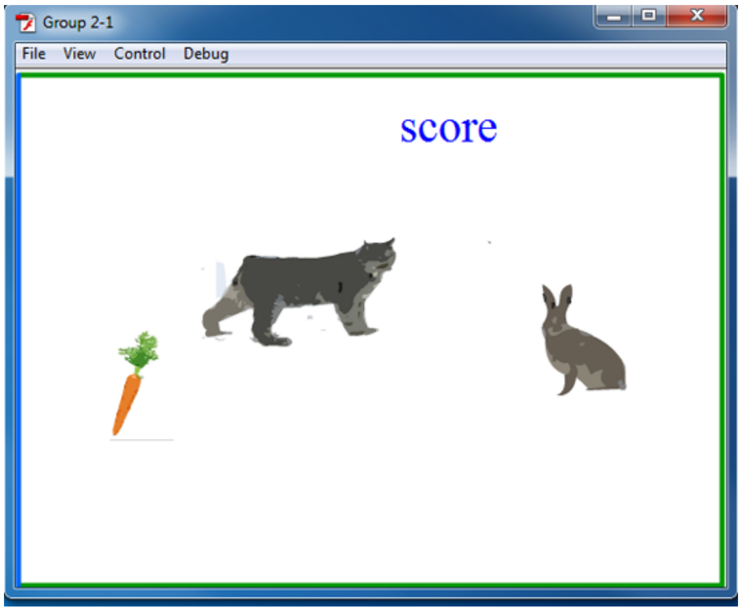
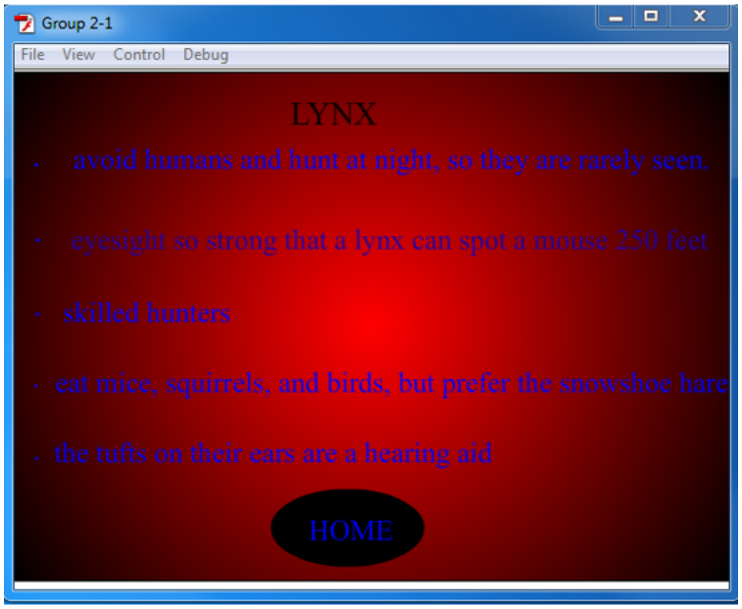


Figure 5b: A typical student representation of the Snowshoe Hare and Canadian

Considering the decontextualized and rote application of patterns suggested by the other levels, the similarity in the two games makes sense. This marks an opportunity for Globaloria, as it suggests that opportunities for students to use contexts outside of the curriculum could help them think more broadly, systematically, and creatively about what they have learned in the learning platform. Seen from an ideal play perspective, their code and the relationships between visual elements were directly analogous to the tutorial, and neither their discourse nor their design process reflected any consideration given to how someone other than themselves might understand the key points they were trying to make. Their general process for creating the games followed these steps: Identifying a pattern in the task, matching that pattern to something on the learning platform, downloading the Flash source file from it, and generalizing the pattern on the new context by changing the instance names regardless of its appropriateness. Clearly, these are very important steps in their computational thinking development, and show the positive impact that Globaloria can have on students. However, they also show the limitations of this form of learning from the constructivist/constructionist aspirations of a 21st century curriculum like Globaloria, as this process showed very little initiative on the part of students to drive their own learning outside what they already know. As a result, the process they displayed time and again would only work as long as the task does not require the students to apply their knowledge of Flash to interactive systems outside of those defined in the learning platform where self-regulated learning would be paramount to success.

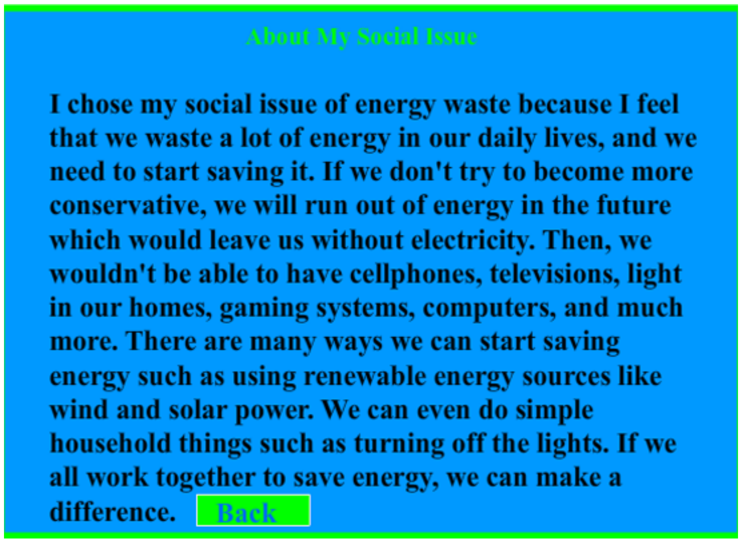
From a game-based learning perspective, this also has important implications for the potential that the Globaloria curriculum has of impacting students' learning beyond Flash. The lynx and snowshoe hare is an exercise on constructing systems from information, a modeling activity that is fundamental to most 21st century knowledge work, particularly in the STEM disciplines. The research on games and learning has shown time and again that one of the most fundamental ways in which games drive learning is by actively giving players input and manipulation over the relationships between system components in computer simulations (Rieber, 1996; Gee, 2003; Games, 2008: Shaffer,2006;). To take advantage of the game format most effectively to communicate the information in the passage for this task, players would have to attempt to integrate it in a software simulation of those relationships, before building play on that simulation. However, the games made both in the Globaloria final projects in the learning platform, as well as in this task, tended to use the information in the passage as facts and declarative knowledge interspersed as static messages and screens separate from the simulation (Figure 6a and 6b), a fact that confirms the opportunity the curriculum has to further students' learning much more broadly through deep computational thinking skill development.



LYNX

- avoid humans and hunt at night, so they are rarely seen.
- eyesight so strong that a lynx can spot a mouse 250 feet
- skilled hunters
- eat mice, squirrels, and birds, but prefer the snowshoe hare
- the tufts on their ears are a hearing aid

Figure 6a: A separate scene containing facts about the Canadian lynx (post-assessment).



About My Social Issue

I chose my social issue of energy waste because I feel that we waste a lot of energy in our daily lives, and we need to start saving it. If we don't try to become more conservative, we will run out of energy in the future which would leave us without electricity. Then, we wouldn't be able to have cellphones, televisions, light in our homes, gaming systems, computers, and much more. There are many ways we can start saving energy such as using renewable energy sources like wind and solar power. We can even do simple household things such as turning off the lights. If we all work together to save energy, we can make a difference.

Figure 6b: A separate scene in a final project about energy conservation.

CONCLUSION

We revealed many interesting findings in this study, all of which suggest that the Globaloria platform and its curricular structure show much promise in positively impacting students' computational thinking skills across the board of our research model's five dimensions. In general, our analysis suggests positive trends in terms of students' growing sophistication in each of the computational-thinking dimensions demonstrated through the pre- and post-assessment. In many ways, the purposes of our methodology and analysis are not so much to examine what students learn about the tools they have used, as much as to understand how students learn to think and act with the tools provided by Globaloria in the rich problem solving space of game creation, which shares with other similar contexts the need to develop skills to effectively participate in a world where computational technologies mediate almost every aspect of life. **Here are 10 main conclusions regarding how a group of students' computational thinking skills evolved during their first year of Globaloria participation.**

1. The analysis of the evidence collected from the individual protocols suggests that while working individually during the tests, students' showed substantial changes in their ability to **Decompose a Design Problem** into parts, but with a strong material and visual bend. The fact that most of the West Virginia students attempted to solve the problem using keyframes (in Flash) rather than coding supports this view. Similar, albeit not so pronounced trends were seen in the **Pattern Recognition task**, where most students identified similar patterns at the level of visual objects but not so much of code abstractions. We believe this to be because the task given for pattern recognition required them to examine an interactive system with a pattern that did not directly map to a specific Globaloria lesson, and evidence of this insight is supported by the fact that for most students, Pattern Generalization showed a marked increase, but only to the degree that students were able to directly identify a pattern given to them in the learning platform, and then copying and pasting the analogous code as a starting point for their own designs; a process they repeated virtually in every task.

2. Developing the above set of skills is important, as it is the basis upon which children can learn to **search for code** with a certain functionality in a body of documentation, and **apply a base pattern of interactivity** created by that code across multiple contexts, which is an activity professional developers usually enact when using programming language application programming interfaces (API's). However, our observations also suggest that at least during the first year of Globaloria exposure, the students' use of patterns is limited to contexts of application closely similar to the problems presented in the learning platform lessons (near transfer), and that students' understanding of the general principles of computation regulating how these patterns work to affect visual representations in the Flash stage is also limited to these contexts. In many ways this is to be expected given that the students observed in this study were first year Globaloria students without previous background in computer science theory or the use of Flash for coding.

3. The **Algorithm Design task** was designed to build on previous dimensions of computational thinking. The findings from this task strongly suggest that the limited number of algorithmic patterns known by students in their first year hampered their ability to work backward from an intended solution to the interaction design task, thus limiting the overall sophistication of their solutions. However, a positive finding shows that students who were successful at solving the puzzle had encountered puzzles following the farmer's dilemma pattern before, and this supports our view that *exposing students to a variety of working game and interactive systems designs throughout the course would be highly beneficial to their taking full advantage of those aspects of computation*, even during their first year. This too points to a need to instruct learners in **Systems Design Principles** together with the emphasis on tool use in the current curriculum.

4. The close relationship between the students' familiarity with the Flash examples given in the learning platform lessons and their post-assessment performance was also observed in the games that the beginner-level students produced over the course of the semester. Many of the final games seen on the learning platform had predominant interactive quiz components, using the mouse, keyboard and drag and drop patterns with very little variation to how they existed in the original lessons, and incorporating the content knowledge intended for players to learn as static text screens separate from the simulations they constructed, as opposed to *integrated into the play activity*, as is the point of games for learning.

5. While this evidence indeed suggests that the Globaloria platform and learning ecosystem can be effective at teaching students to apply the knowledge learned in their Flash lessons, it also suggests that developing additional modules in the platform that reinforce the students' *understanding of best practices to incorporate content knowledge in the form of game systems* would be especially beneficial for students' learning in areas beyond programming, such as math and science, which have strong requirements of systemic thinking (Wilensky, 1999). This also holds for learning civics and other social science domains through game design, which Globaloria students do. The Globaloria platform and curriculum take advantage of game-based learning format, and harnesses students' enthusiasm around gaming, an activity that a majority of children engage in today, in order to help them understand the complex and systemic relationships of certain knowledge domains that are often so problematic for them in later grades.

6. Globaloria has a strong constructionist basis. It is designed for encouraging student self-reliance, self-initiative, personal interest and curiosity as the fundamental in driving the learning of game design (it's based on Harel 1991; Harel and Papert, 1991). The findings from this study suggest that the program (at least among first-year students) faces an important challenge in placing constructionist expectations on participating students. It is important to note that all the other courses and activities in the Globaloria schools are organized around more rigid, teacher-driven, traditional instructional models. We observed that the students were not constructionist in solving the problems we presented to them in the pre- post-assessments. In particular, we observed their tendency to narrowly use what they had learned in previous lessons to attempt to tackle the design decomposition tasks, rather than to take advantage of the information at hand, more effectively and creatively. We observed more "School-style thinking" as opposed to the "Globaloria constructionist-style thinking" in the student's tendency to stay within the information available to them immediately, rather than to seek new information, even when resources like the Internet were available to them during every task.

7. This points to a need to create mechanisms within the Globaloria learning environment that bolster self-driven, constructionist habits of learning. For example, we believe that if the body of game design patterns taught to students were to be expanded, it would give learners both more flexibility to think more deeply and creatively about the goals of interactivity in games or other software systems, and harness the strong engagement that gaming tends to elicit in students within the Globaloria demographic more effectively towards self-regulated learning. These patterns could be introduced in the form of other genres of games, or even other forms of interactive projects, provided as choices for the teams, thus allowing the students an increased degree of ownership of their project overall. This would allow learners to use a game format in its most effective form to learn knowledge relevant in other courses and contexts of interest to them, both aspects that should positively impact their motivation throughout the course (Wigfield and Eccles, 2000).

8. From the perspective of the "**three dialogs**," this study's results continue to point at the view that the Globaloria platform encourages computational thinking more strongly at a "material dialog" level (learning to use the tools), than it does at the ideal of "real player dialog" level (what kinds of activities or user needs would the systems created with those tools be for). Designing a game is a very complex task that requires as much creativity and artistic skill as it takes scientific approaches to computation. We propose that for Globaloria to effectively use game creation to introduce learners to STEM content, all of the domains involved in game making must be presented explicitly in the platform. At least in our observations, self-regulation and creativity among students were still emergent, and we believe more design-oriented instruction could bolster these skills, even in their first year. It may be interesting for us to conduct the same study with students who complete more than one year of Globaloria.

9. In an age where information is available at our fingertips with a few keystrokes, developing students' ability to *learn how to learn* about multiple domains is as important as learning a body of domain-specific patterns and practices (Papert 1980, 1993; Lee, Martin, Denner, et. al, 2011) and we believe Globaloria is a learning environment and digital teaching and learning framework that can be optimized to effectively help learners learn how to learn, how to produce digitally, and extend their computational thinking skills to support their effective learning of both the art and science of game creation on educational topics.

10. Finally, this **study's methodology** shows the promise of using assessments based on the "matrix of computational thinking" and the "three dialog" frameworks for explaining the evolution of learners' thinking and problem-solving practices in 21st century pedagogies, such as the ones advocated by Globaloria. While the scope of this study was limited by the resources, time frame between pre- and pot-assessments, and the number of participants we studied, it nevertheless showed how assessing the "different dimensions of computational thinking" provided convergent evidence showing positive trends in how students' computational thinking skills evolve within Globaloria over time. In the future, similar assessment methodologies and analyses could be applied to comparative studies of individual versus group work, of students who go through a refined version of the Globaloria curriculum that incorporates extended design patterns and game design lessons, or contrasts between students at different levels of achievement over time.

REFERENCES

1. Brown, A.L., & Campione, J.C. (1996). Innovations in learning: New environments for education. In L. Schauble, & R. Glaser (Editors), *Psychological Theory and the Design of Innovative Learning Environments: On Procedures, Principles, and Systems* (pp. 289-325). Hillsdale, N.J, England: Lawrence Erlbaum Associates.

2. 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, & G.A. Milton (Eds.), *Social Facilitation and Imitative Behavior* (pp. 91-110). New York: Holt, Rinehart & Winston.

3. Engstrom, M.E., and Jewett, D. (2005). Collaborative Learning the learning platform wiki Way. *TechTrends*,

49(6), 12-15.

4. 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.

5. 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.

6. 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 (Editors), *Computer Games and Instruction* (pp. 17-46). Charlotte, N.C.: Information Age Publishing.

7. Gee, J.P. (1992). The social mind: Ideology and social practice. New York, Bergin & Garvey.

8. Gee, J.P., Hull, G.A., Lankshear, C. (1996). *The New Work Order: Behind the language of new capitalism*. Sydney, Australia: Allen & Unwin.

9. Gee, J.P. (2005). *An Introduction to Discourse Analysis: Theory and Method*. New York, NY: Routledge.

10. 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.

11. Globaloria (2011). *What is Globaloria?* Retrieved from http://www.globaloria.org/intro#WhatIsGlobaloria

12. Google. (2010). What is Computational Thinking? Retrieved from http://www.google.com/edu/computational-thinking/what-is-ct.html

13. Hayes, E.R., & Games, I.A. (2008). Making Computer Games and Design Thinking. *Games and Culture, 3(3-4)*, 309-332.

14. National Academy of Sciences (2010). Report of a Workshop on the Scope and Nature of Computational Thinking, Washington, D.C.: National Academies Press.

15. Harel, I., & Papert, S. (1991). Situating Constructionism. In I. Harel and S. Papert (Eds.),

Constructionism (pp. 1-12). Norwood, NJ: Ablex.

16. Stake, R. (1995). The art of case research. Newbury Park, CA: Sage Publications.

17. Vygotsky, L.S. (1962). Thought and language. Cambridge, Mass: MIT Press.

18. Vygotsky, L.S. (1978). Interaction between learning and development. In *Mind and Society* (pp.79-91). Cambridge, MA: Harvard University Press.

19. Wertsch, J.V. (1998). Mind as Action. New York, NY: Oxford University Press.

20. Harel, I. (1991) Children Designers: Interdisciplinary constructions for knowing and learning mathematics in a computer-rich classroom. New Jersey: Ablex Publishing.

20. 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

21. Wing, J.M. (2006). Computational thinking. *Communications of the ACM, 49(3),* 33-35.

22. Wing, J.M. (2010). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society, 366*, 3717-3725.

23. Wiggins, G. & McTighe, J. (2005) *Understanding by Design.* Washington DC: Association for

Curriculum Development.

24. Pea, R. (1997) Practices of Destributed Intelligence and Designs for Education. In G. Salomon (Ed) Distributed Cognitions: Psychological and Educational Considerations. Cambridge, UK: Cambridge University Press.

25. Lee , I., Martin, F., Denner, J., et.al. (2011) Computational Thinking for Youth in Practice. ACM Inroads 2 (1). 32-37

26. Barr, V. & 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.

27. Barr, D; Harrison, J; & Connery, L. (2011) Computational Thinking: A Digital Age Skill for Everyone. Leading and Learning with Technology. (March/April)

Appendix A

This appendix contains selected examples of students that best represented the overall trends observed during the assessment interviews.

<u>*Thomas*</u>

Starting with Thomas' pre-assessment, the interviewer, Patrick, began the interview by showing the student the Marble game and how it is played. He gave the student an opportunity to play the game for himself, which hardly took any time at all, due to the simplicity of the game, and once the student had a chance to see the Marble game in action, the interviewer asked Thomas how he would start making the game. In much the same way that Brown and Campione laid out their curriculum, Fostering Communities of Learners (1996[1]), which built on Vygotsky's zone of proximal development (1962[22]), the interviewers asked questions so as to elicit information that the students presumably knew, yet had a hard time articulating on their own.

1 *P: What do you think are the first steps that you would do to start creating this thing?*

2 *T: Make the background.*

3 *P: Ok, so you'd make the background. And then what would you do?*

4 *T: Draw the marbles.*

5 *P: Ok, so make the background and then the marbles; then what's the next step?*

6 *T: Add all the coding.*

7 *P: Add all the coding. Ok, so what is involved in adding all the coding?*

8 *T: Trying to figure out how to make the **green ball** hit the **red ball**.*

Note in line 8 Thomas uses the terms "green ball" and "red ball" instead of green and red "marble". Although easy to overlook, his use of differing terms here shows a lower level of connection to the cultural interpretation of the system that he sees (the Real Player Dialog). In terms of decomposition, Thomas is breaking down the game into shapes, not objects. However, later in the interview, he did start to use the term "marbles". Going past that, Thomas was not able to break the system down any further, and while searching the learning platform wiki , he was not able to find any code that he could apply to the game. To summarize Thomas' pre- assessment, he was able to decompose the system in both its first and second levels, which we have defined as the marbles and background, and some sort of code.

Moving forward to the post-assessment, Thomas ended up being one of four students who were able to identify scene-switching as a necessary component in the Marble game. In terms of decomposition, this shows us that Thomas has learned more about how Flash works. By further breaking down the system to deeper level, Thomas is showing that he knows that getting the game to switch from the introduction scene to the scene where the game actually takes place is a critical part of recreating the Marble game. In terms of the actual execution, Thomas was not actually able to make the game switch scenes; he ultimately gave up on that task and moved on to dealing with the problem of how to make the marbles move, which was the point at which he got stuck during his pre-assessment. One thing to note during this process, although it is not strictly related to the process of decomposition or computational thinking, is the fact that Thomas organized the initial layers, which he used for the introduction screen and the Start button, into a folder on the timeline and then created new layers for the gameplay scene, which he then organized into its own folder. While this was entirely unnecessary, he has developed a habit that professional game designers and animators frequently exhibit to keep their work organized, which is a positive trend. Returning to the process of making the marbles move, Thomas was still unable to work that out. Patrick, the interviewer, moved on to the next activity in the interview after it became apparent that the process of making the marbles move was outside of his zone of proximal development, and Thomas would not be able to complete it, regardless of how much guidance he could be given. *Elena*

Similarly to Thomas, Elena was asked if she thought that she could recreate the marble game on her own and identify what the main parts of the game were.

9 *P: What do you think are the main parts of this...if you had remake this?*

10 E: Um, the background colors.

11 P: The background colors...then what would you do?

12 E: Um, the shapes.

13 P: The shapes.

14 E: Um, the writing?

15 P: Mmhmm, the text...

16 E: The movements.

While the questions are essentially the same, the answers are quite different. Elena uses the term "background *colors*", the "shapes", and the "movements", whereas Thomas used the terms "background", "marbles", and "coding". Elena's answers show how focused she is on the visuals and the graphics; Thomas' answers indicate that he is thinking more about how the system actually works (the authors acknowledge that although we have pointed out that

Thomas used incorrect terms (*ball* instead of *marble*), he was still using those terms in the context of trying to figure out how the system works). Elena's focus on aesthetics became apparent as she began to build the Marble game. Instead of using the given assets (see figure 5), Elena created her own assets, which consisted of a light-blue background, with purple squares and stars. This was a lengthy process, taking just over 5 minutes to complete. In terms of Games' Three Dialog framework, this aligns with the Real Player Perspective, and shows us that, similar to Thomas and the ball/marble error, she is displaying a lower cultural connection to the Marble game. As an analogy, this would be similar to tasking a student with drawing a dog and getting a picture of a cat because it also has four legs and a tail. This demonstrates that this student was particularly focused on the aesthetics of the game, which, in terms of decomposition, shows a different interpretation of that system.

Before moving on to discuss the post-assessment, it is worth noting that Elena also typed the word "instructions" on her background. While the specific term, "instructions", is not found anywhere in the Marble game, this indicates to us that she is applying a pattern to this game that she has seen in other games. It is a good rule of thumb for a game designer to provide instructions to the player if it will not be inherently obvious how to play the game. In doing this, Elena is "behaving like a professional", similar to the way in which Thomas organized his layers into folders. These habits are habits of professionals and are most likely a product of the Globaloria curriculum. In support of this point, it makes sense that Elena would at least attempt to add instructions to her game, as all of the Globaloria students' final games had a button for instructions built into their games.

During the post-assessment, little changed, in terms of her focus on the assets. She was again much more interested in making sure that she made good looking assets and less focused on getting the marbles to actually move. As an example of this, she highlighted the marbles in the library (which should be noted, are already converted to symbols), and instead of dragging them onto the stage, she used them to figure out what color they should be. Using the pre-drawn and pre-converted assets as a template, she found the correct colors and used the oval/circle tool to create her own marbles, which ended up being ovals of various sizes. With regards to decomposition, she is aware that the marbles are a necessary component of the game, which covers the first level of decomposition outlined above.

Paul

Paul displayed very fidgety behavior during the interview. He seemed to stall for time, even though it was made clear that there was not a hard time limit or a grade for his performance; after creating his assets, it became clear that he did not know what the next step was. Once directed to the learning platform wiki , it was apparent that he did not know what he was looking for or what he would do if or when he found it. However, it is possible, and likely, that this behavior is a result of evaluation apprehension (Cottrell, 1968[2]), which was present in more of the students in West Virginia (10 out of 15) than in Texas (5 out of 15). After further review of his interviews, it was observed that he is rather insecure in his problem-solving ability due to the fact that he does not self-identify as a gamer, stating, "I don't play games too much and sometimes, I'm not like the kind of person that can figure out stuff very, very quickly. But gamers, they can figure something out like that." It is very likely that this lack of self-efficacy was a contributing factor to his underperformance during the post-assessment.