The Zones of Proximal Flow: Guiding Students Through a Space of Computational Thinking Skills and Challenges

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

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

Kyu Han Koh
University of Colorado, Boulder
Department of Computer Science
Boulder, CO 80303
303-492-1349
kohkh@colorado.edu

Hilarie Nickerson
University of Colorado, Boulder
Department of Computer Science
Boulder, CO 80303
303-492-1349
hnickerson@colorado.edu

ABSTRACT
This paper presents a novel pedagogical framework, entitled the Zones of Proximal Flow, which integrates Vygotsky's Zone of Proximal Development theory with Csikszentmihalyi's ideas about Flow. Flow focuses on the individual– an individual is in Flow when challenges are balanced with skills. The Zone of Proximal Development, on the other hand, brings in a social learning aspect focusing on a student's ability to learn concepts with external support. From our research experiences bringing game and simulation design into middle school classrooms, we attempt to provide students with appropriate challenges using a project-first based approach that aims to keep students in Flow. The project-first approach employs inquiry based scaffolding to guide students, with appropriate support by their teachers, through Vygotsky's Zone of Proximal Development, back in to Csikszentmihalyi's state of Flow for an ideal learning experience. We call this space the Zones of Proximal Flow. Data indicate that the Zones of Proximal Flow approach works, keeping classrooms engaged in the act of game design and enabling students to advance to more complex program creations.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computers and Information Science Education

General Terms
Management, Measurement, Performance, Human Factors

Keywords
Zone of Proximal Development, Flow, Zones of Proximal Flow, Computational Thinking, Computational Thinking Patterns, Simulations, Scalable Game Design, Computer Science Education

ICER'13, August 12–14, 2013, San Diego, California, USA.

1. INTRODUCTION
At present time, much research into educational end-user programming tools focus on giving students, with little to no prior programming experience, the skills necessary to rapidly create games [1]. When compared to traditional programming languages, such as C++ or Java, these tools provide students with a means to make sophisticated graphical games immediately by employing drag and drop elements, in place of often tedious syntax that accompany traditional programming mechanics [2]. Employing graphical elements, students are quickly introduced to the logic of programming; additionally, game programming has the power to motivate students in the domain of computer science [3].

Traditional instruction, based on introducing principles-first, where students engage in multiple semesters of theory before undertaking interesting projects, is not compelling to students [21]. Just as important as end-user programming tools, is the pedagogical approach to applying these tools into the classroom curriculum. For our purposes we have found that a project-first approach, wherein students are motivated to gain the skills necessary to complete the problem, facilitates student learning of skills and the ability of classes to attempt more sophisticated projects [21]. Furthermore, scaffolding and resources provided to teachers and students allows for this project-first approach by enabling students and teachers to gain additional skills, if necessary, to face these increasingly challenging projects. While syntactic programming is necessary to overcome frustration in novice programmers, perhaps a much more important aspect of computer science education is the pedagogical approach which even has important implications for broadening participation [21].

As part of the National Science Foundation funded Scalable Game Design Project, at the University of Colorado Boulder, we employ two such end-user programming tools that enable rapid game and simulation prototyping by end-users: AgentSheets [4] and its subsequent 3D version AgentCubes [5]. One aim of these tools is to enable the concept of low-threshold high ceiling, meaning that students can quickly create games overcoming the barrier of entry into programming, but also, can make increasingly sophisticated games and simulations as they gain more programming skills. In AgentSheets, for example, students can go from no programming experience to creating their first game in around five hours; subsequently, as students gain expertise, they can create complex

high-level interactions relating to math, science, and artificial intelligence for example [6,7]. As part of this project, over 10,000 student games and simulations have been created across the United States [8].

The Scalable Game Design Project aims to broaden participation among diverse sets of students from elementary school to graduate school education. However, in addition to motivating students through games, the Scalable Game Design approach emphasizes giving students the necessary skills to investigate science [9]. Empowering students with the ability to create simulations links the Scalable Game Design process to computational thinking learning. A quote from Len Scrogan, the former director of Technology for the Boulder Valley School District, makes explicit the link between game and simulation design in terms of computational thinking. Len walked up to a student creating space invaders through the Scalable Game Design project and asked:

"Now that you've made 'Space Invaders', can you create a science simulation [9]."

Unwrapping this statement begins to reveal important educational concepts actually learned through game creation. Specifically, Mr. Scrogan is asking how students can leverage the skills they have acquired from game design and apply these skills to simulation authoring. As students learn how to create games, they are motivated to learn the programming constructs necessary to complete the agent-interactions present in these games. Furthermore, many of these interactions present in games, are also present in science simulations. Through game creation, students are actually gaining the skills necessary to create representative systems.

Creating simulations relates closely to computational thinking. Currently, computational thinking is defined to include the following: problem formulation, logically organizing and analyzing data, representing data through abstractions such as models, automating solutions through algorithmic thinking, implementing effective solutions optimally, and transferring the solution to solve a wide variety of problems [10]. Furthermore, Jeanette Wing, a major proponent of computational thinking states that finding the right abstraction, ignoring and emphasizing certain aspects of the real world underlies computational thinking [11]. When creating a simulation, students must define a problem in the real world domain. The students must abstract this problem into the representational domain creating a model that includes the aspects of the real world problem necessary to gain insight into a particular problem. Finally, students experiment on the simulation to find out information in the representational domain that can be applied to the real world domain. The concrete link between game programming and simulations is currently being established, for example, in the recent National Research Council report entitled "Learning Science Through Games and Simulations" [12].

We have done much research into the relationship between game design and simulation creation in educational contexts. This research has led to the development of a construct entitled Computational Thinking Patterns [13]. Computational Thinking Patterns are agent interactions students initially learn in game design, but then, transfer to the creation of science simulations. Examples of Computational Thinking Patterns include programming one agent to collide with another agent, one agent creating another agent, one agent tracking another agent, and one agent transporting another agent among many others. For example, in the game Pacman a student might program ghost agents tracking a pacman agent; similarly in a predator/prey simulation, a student might implement the predator agent tracking a prey agent. A more exhaustive list and description of Computational Thinking Patterns can be found in [7].

Previous research has shown that users can identify the necessary Computational Thinking Patterns necessary to implement an interaction outside of the game context and use the Computational Thinking Patterns construct to directly create simulations [14,15]. By combining Computational Thinking Patterns, users can begin to piece together and experiment on simulations, such as epidemiology simulations and predator/prey simulations that relate to their classroom curriculum.

In addition as acting as the units of transfer between games and simulations, we have also started analyzing games and simulations by their constituent Computational Thinking Patterns. This method is called Computational Thinking Pattern Analysis (CTPA) and currently, we have the ability to view the constituent patterns students have programmed when creating a given game or simulation [13]. Computational Thinking Pattern Analysis allows us to compare what skills students have acquired through game programming and identify corresponding simulations that students can create given the mastery of these pattern combinations [8,13]. Furthermore, Computational Thinking Pattern Analysis can be used as an assessment tool that enables us to match student skill with appropriate challenges within the domain of past work.

The idea of matching skills and challenges has a rich history in terms of pedagogy research. For example, Vygotsky's theory regarding the Zone of Proximal Development describes the difference between what a learner can do with and without external help [16]. Similarly, Csikszentmihalyi's theory of Flow states that people are in a completely motivated and engaged state when the skills they have garnered match well with the challenges they are currently undertaking [17]. For our research purposes, implementing game and simulation projects in school classroom curricula, the strategy we employ combines both of these theories into an emerging space we call the Zones of Proximal Flow. In short, our Zones of Proximal Flow strategy combines a project-first gentle-slope approach, involving games and simulations of increasing complexity, matched with student skill, plus added external scaffolding to guide anxious students back into the Flow. The remainder of this paper will describe the synthesis of the Zones of Proximal Flow, the implementation of this strategy in the classroom, and the results of this approach exposing students to computational thinking through game and simulation design.

2. THE ZONES OF PROXIMAL FLOW: CSIKSZENTMIHALYI MEETS VYGOTSKY

Educational theories regarding learning and development are often described as if they are independent of other processes. In reality, both classrooms and individual learners are complex entities in which it is possible to observe evidence that supports multiple educational theories simultaneously. During Scalable Game Design activities, students can exhibit behaviors that are characteristic of both Csikszentmihalyi's notion of Flow [17], which relates primarily to specific moments in time, and Vygotsky's theory of the Zone of Proximal Development [16], which describes a student's ability to move beyond independent problem solving through scaffolding over extended periods of time. Considering these two ideas led us to develop the Zones of Proximal Flow framework to account for what we were observing. In this section, we examine both theories in detail, explain their

compatibility, and view Scalable Game Design through the lens of this combined framework.

The concept of Flow has its origins in Csikszentmihalyi's study of intrinsically motivated activities [18]. A high degree of immersion in such an activity can result in an optimal experience, which refers to feelings of firm control and coping ability during goal pursuit while performing at the limits of one's abilities. As an example, intense participation in a demanding creative or athletic endeavor can generate an optimal experience regardless of whether the outcome is aesthetically pleasing or the activity results in physical discomfort. Optimal experience is also associated with a state of deep concentration; the related distortion of the passing of time as the experience unfolds; and a notable feeling of enjoyment, altered consciousness, or extraordinary experience. Together, these experiential characteristics identify the state known as Flow. An important precondition for flow is that an individual's level of skill must correspond well to the level of challenge. Overly easy challenges result in relaxation or boredom, depending on the degree of mismatch [19,20]. Similarly, challenges that are too difficult prompt vigilance or anxiety. Figure 1 depicts Csikszentmihalyi's early flow diagram.

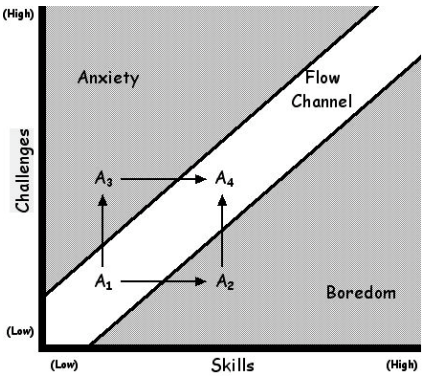

Figure 1. Csikszentmihalyi's depiction of Flow, the region between boredom and anxiety wherein skills align with challenges [17]

While Flow has been examined in the context of both short and long timescales, it has most often been used to describe 'in the moment' states of an individual [17,20]. In the short term, staying in a Flow state is contingent on continuing motivation to prolong the activity underway, which is determined by assessing the individual–environment interaction occurring at a given moment rather than by consulting an established structure. Concomitant incremental improvements in performance can be observed. Because Flow is a pleasurable state, people desire to repeat experiences that produce it, leading to growth over the long term both in skill and in the level of challenge to be faced. Note that either skill or challenge increases can take the lead role at any given point in this progression, with the other factor subsequently rising to complement it. In formal learning environments, it has been found that active learning promotes short-term flow and that flow experiences predict greater persistence and achievement in the associated activity over the long term [19].

Vygotsky's theory of the Zone of Proximal Development (ZPD) arose as a way of explaining how learning and development in children are related [16]. Rejecting existing theories that development must precede or coincide with learning, Vygotsky instead proposed that learning itself was necessary to promote development. From this viewpoint, the developmental level of an individual can be measured not just in the traditional manner, by examining current ability, but also by determining what that individual is ready to master given appropriate assistance. The ZPD is the area between these two levels, which "defines those functions that… are in the process of maturation [16]." Learning experiences aimed within the ZPD are the most useful. According to Vygotsky, there is no developmental advantage to providing instruction in areas of existing mastery, nor is it possible for children to learn material that is beyond their ZPD. Figure 2 depicts the Zone of Proximal Development.

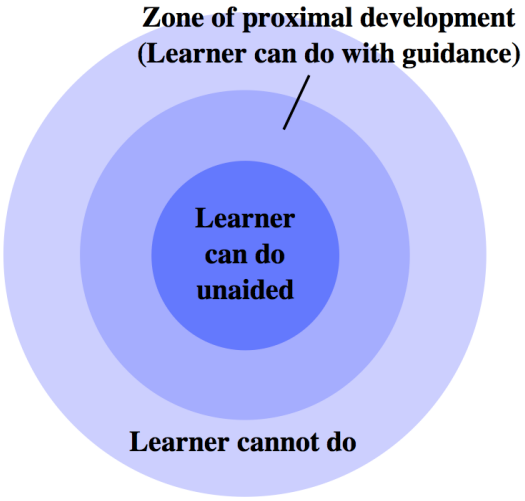

Figure 2. Zone Of Proximal Development which represents what the learner can do with guidance, used with permission under © CC0 1.0 Universal Public Domain Dedication

The sociocultural context in which learning occurs is a key part of Vygotsky's theory. Children naturally begin learning from their parents at birth, spurring their initial development. School brings both a more systematic approach to instruction and more opportunities to learn from and with peers—children working together in a group may be able to accomplish what an individual child could manage only with a teacher's guidance. In either case, what starts out as an interpersonal experience promotes development that leads to subsequent independent accomplishment, thus shifting the ZPD. The theory's major themes are summed up in this brief passage from Vygotsky's writings:

"We propose that an essential feature of learning is that it creates the zone of proximal development; that is, learning awakens a variety of internal developmental processes that are able to operate only when the child is interacting with people in his environment and in cooperation with his peers. Once these processes are internalized, they become part of the child's independent developmental achievement [16]."

Through our research approaches in the Scalable Game Design Project, we have found that the concepts of Flow and the ZPD are complimentary. Csikszentmihalyi emphasized the individual's skills establishing a hard boundary between states of Flow and non-Flow. Furthermore, the concept of Flow does not have any notion of social learning, or learning outside of the individual. However, we have found that students can be in Flow, wherein the game or simulation authoring challenges match their skill level, but also, students might be in a state where they need outside help in order to meet the program challenges they face. This outside help takes the form of focused in-class instruction, online tutorials, and peer learning among classmates. When

students make use of this outside help, because of the scaffolding that exists, they are not in a state of anxiety necessarily, but rather, in a state more akin to the Zone of Proximal Development.

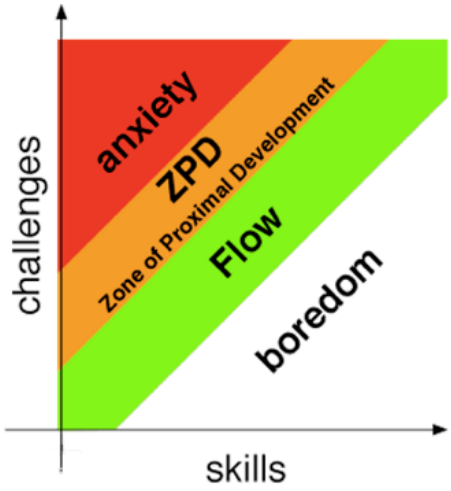

Figure 3. Zones of Proximal Flow wherein ZPD is located in between regions of Flow and anxiety

Combining Flow and ZPD we arrive at space overlaying a social learning element to Flow. Figure 3 depicts this combined space entitled the Zones of Proximal Flow wherein the Zone of Proximal Development is lies between the regions of Flow and anxiety. The question still remains how do we apply this construct to facilitate engaged student learning trajectories? The following section describes how the Scalable Game Design Project implementation relates to the Zones of Proximal Flow.

3. ZONES OF PROXIMAL FLOW AND THE SCALABLE GAME DESIGN APPROACH

The Scalable Game Design curriculum attempts to bring computer science education into public schools through game and simulation design. The notion of the Zones of Proximal flow is a framework that guides students through the challenge skills space in a low threshold high ceiling path of increasingly sophisticated projects. This gentle slope trajectory starts with a simple game, such as Frogger, wherein students learn beginning Computational Thinking Patterns like one agent colliding another agent, one agent creating another agent, and one agent changing another agent, and one agent tracking another agent. After students master these patterns, students move onto more sophisticated games like Sokoban, where they learn how to make one agent push another agent; Centipede, where they learn how to make a group of agents move in concert with one another; and the Sims, wherein they learn how to make one agent track other agents in the world [6]. At any point, if game challenges do not match the student's skills, scaffolding is provided to support student development. These include class instruction, tutorials that are readily available on the Scalable Game Design wiki, in-class peer student learning, assessment instruments that make explicit student skills obtained through programming, and the ability to download fellow classmates' projects. This scaffolding is an attempt to take students who are in the Zone of Proximal Development and bring them back into Flow. Figure 4 depicts how the Zones of Proximal Flow relates to the Scalable Game Design Project.

In Figure 4, the horizontal axis represents students' computational thinking skills, as measured by Computational Thinking Pattern Analysis. The CTPA captures a single aggregate value between 0% Computational Thinking Pattern coverage, i.e., a student not having been exposed to any of the patterns in the inventory, and 100% Computational Thinking Pattern coverage, i.e., a student exposed to all Computational Thinking Patterns, presumably through building a sequence of projects. The vertical axis represents the level of the design challenge that would be intrinsic to a certain game or STEM simulation.

Computational Thinking Patterns begin to make explicit the skills necessary for students to create STEM simulations. The Scalable Game Design project starts with simple games students can create quickly. As students progress they learn the constituent Computational Thinking Patterns necessary to create associated simulations. For example, when a student completes the Frogger game, this student has learned how to make one agent change another agent; i.e. when the frog agent gets hit by a truck it changes from a living frog agent into a dead frog agent. At this point, the student has gained the skill necessary to change a healthy tree into a tree that is on fire as in the forest fire simulation depicted in Figure 4; i.e. when a healthy tree agent is adjacent to a tree agent that is on fire, the healthy tree agent changes into a tree agent that is on fire with a given percent chance associated with susceptibility of the healthy tree agent to the fire.

For our pedagogical approach purposes in the Scalable Game Design project, detecting the presence of these patterns in student-created games and simulations can enable us to integrate the principles of Csikszentmihalyi's Flow and Vygotsky's Zone of Proximal Development to keep students engaged in the act of creating games and science simulations. We employ a professional development program based on about 35 contact hours in which we train teachers to have students build their first playable game from scratch in about a week (e.g., 5 lessons x 45 minutes). The ability to create a playable game is essential if students are to reach a profound, personally changing, "wow, I can do this" realization. In general, the Scalable Game Design project takes a project-first approach with just in time skill acquisition to motivate students to garner the skills necessary to create games simulations.

The fundamental idea of the project-first, just-in-time principles approach can be illustrated through what we call the *Zones of Proximal Flow* (Figure 4), which combines Csikszentmihalyi's notion of Flow [18] with Vygotsky's notion of the Zone of Proximal Development (ZPD) [16. Flow is an ideal condition for learning, and illustrates the importance of attending to the affordances and limitations of particular forms of mediation, as well as the need for social and technical scaffolding to advance learning. Mindful of Chaiklin's [22] instructive discussion of the misunderstandings of the ZPD, and Griffin and Cole's observation about the limitations of a narrow view of the ZPD as a space of productive adult-centered scaffolding [23], the ZPD can be understood as a socially-mediated accomplishment, involving the orchestration of participation in a rich set of carefully designed practices where forms of assistance and tool use are strategically employed.

In middle schools we have found that the project-first path is significantly more effective than paths that rely on learning many principles first [21] without the presence of a concrete and interesting challenge. The Project-First path helps to engender Zones of Proximal Development where, with proper support, they quickly learn relevant CT concepts.

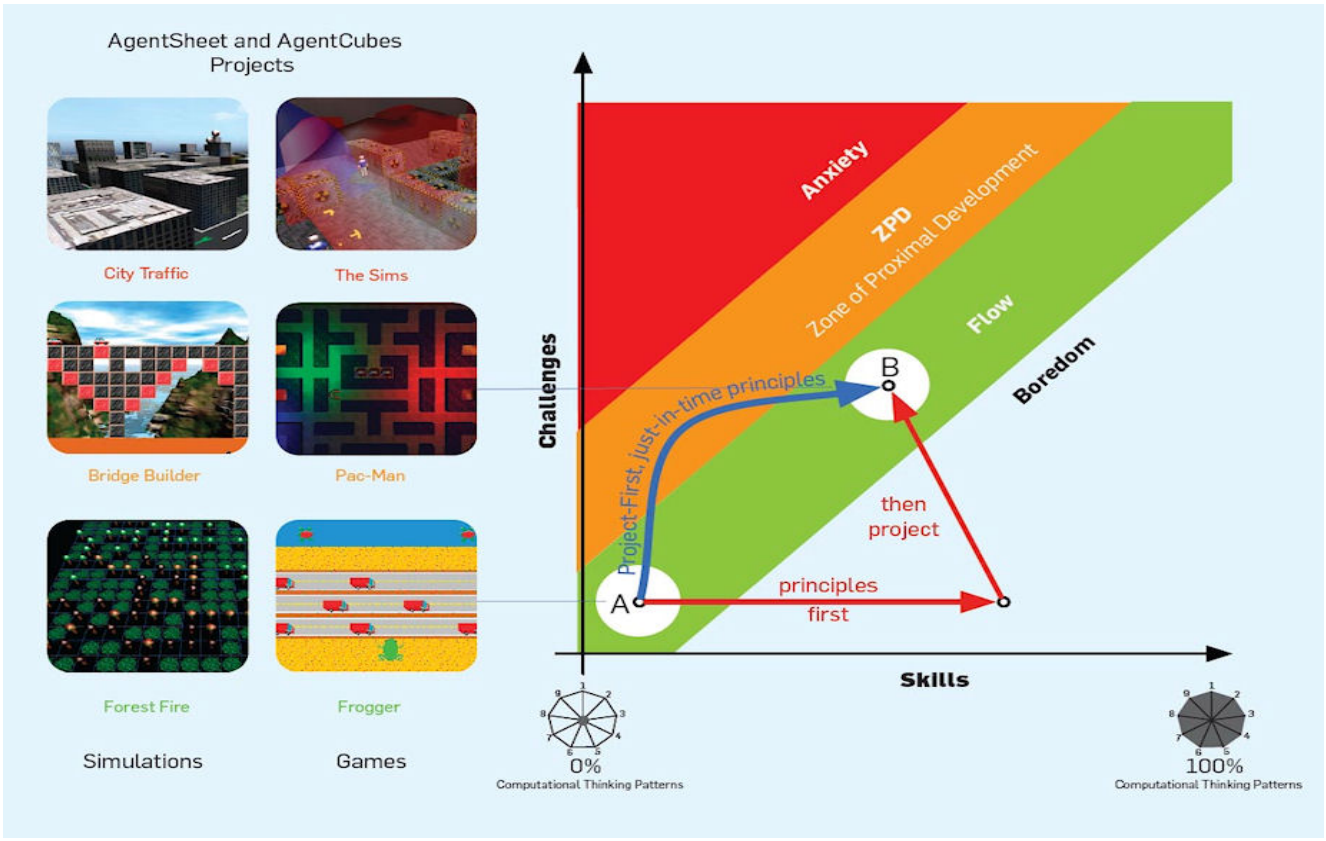

Figure 4: Zones of Proximal Flow relating to the Scalable Game Design Project. Project-First approach motivates students to gain skills in the form of Computational Thinking Patterns allowing them to meet game and simulation design challenges

Looking at Csikszentmihalyi's flow diagram, we see that this skills first approach risks putting students into a state of boredom. The project-first approach, on the other hand, risks students falling into a state of "anxiety" if the project is outside their given skill level. Our approach combines a gentle-slope of increasingly complex projects with added scaffolding to guide anxious students back into the flow.

The project-first versus the principles-first approaches can now be described as instructional trajectories connecting a skill/challenge starting point (A) with destination point (B) in Figure 4. In many traditional CS education models, a *principles-first* approach would expose Jasmine (a hypothetical student) to a number of concepts such as AI search algorithms (A*, LPA*) that may or may not be relevant for future projects. At some later stage, Jasmine receives the challenge of making a related project such as a Pacman-like game. The acquisition of skills without the context of concrete challenges is not a bad pedagogical model, especially at the undergraduate CS level, but it runs the risk of seeming irrelevant, hence boring, for a broader audience of younger students if it does not go hand-in-hand with project-based approaches [21]. This assertion is consistent with the Flow model and with our own observations in classrooms. Instead of decoupling the acquisition of principles and the application of these principles to a project, the project-first approach combines *just-in-time CT skill acquisition* with *application* in the production of a tangible artifact. Employing this approach, we are witnessing truly inspired math performance of students implementing sophisticated artificial intelligence [6].

As mentioned above, Flow focuses on the ability of a person to accomplish a task in a discrete moment of time. In the Scalable Game Design Project, we equip students with the necessary skills, such as the mastery of various Computational Thinking Patterns, such that when presented with a new task such as the creation of a related simulation, the student can complete the task keeping that student in flow. However, there are many situations where the student might not be in Flow, for example, when a student initially is tasked with their first game. When students create Frogger for example, some students might have enough know-how to create the agent interactions necessary to complete the game, but some students might not. These students are no longer in Flow, but rather, going towards a state of anxiety wherein the challenges do not necessarily match up with the skills they have garnered.

The Scalable Game Design project approach deals with this by providing external scaffolding instruction constructs present in Vygotsky's Zone of Proximal Development. The Zone of Proximal Development focuses on how students have the means and motivation increase their individual acuity gaining the know-how necessary to complete a given task over time. Unlike Flow, which focuses on the skills an individual student has in a given moment, the Zone Of Proximal Development is a region where students can learn from external sources over time. From this we can outline the general strategy of the Scalable Game Design Project, namely, using this gentle-slope project-first trajectory providing students with the skills necessary to be in Flow, and, if a given student is not in flow, providing the scaffolding necessary to create a region akin to the Zone of Proximal Development, to

guide the student back into Flow avoiding student-anxiety altogether. This is the essence of the Zones of Proximal Flow diagram depicted in Figure 4.

The Scalable Game Design project supports class instruction through summer teacher training at the Scalable Game Design Summer Institute. Our project has taught this Institute over the last 4 years at the University of Colorado Boulder. The institute consists of 5-7 days wherein middle and high school teachers from across Colorado and the United States learn how to create games and simulations using AgentSheets/AgentCubes as well as teaching strategies that better equip teachers with effective methods of integrating computational thinking into their classroom environments. In addition to this, teachers create and share classroom materials using the Scalable Game Design Wiki. This openly accessible information makes tutorials and lesson plans readily available. Finally the Scalable Game Design Arcade allows students to download over 10,000 games that other students have created. The Scalable Game Design Arcade is a vibrant social community that enables student comments on games and game ratings. In downloading other students games one can look at how other students may have implemented different patterns and also give peer feedback on aspects of the game itself. Previous research has shown the effectiveness of these peer-learning mechanisms [7]. All of these resources enable students, who might be having difficulty with a particular project, to garner the skills necessary to arrive back in Flow wherein their skills are well suited for the project based challenge presented.

4. FINDINGS

Through informal observation of thousands of students in the Scalable Game Design Project, we started to develop the notion of the Zones of Proximal Flow. Furthermore, scaffolding computational thinking through a project-first approach with student summer camp, computer workshops, and after-school programs has been successful [7]. Through our experiences, we wondered if there was any possible quantitative evidence indicating this Zones of Proximal Flow existence in computational thinking education, and if this evidence exists, what might it look like?

One piece of evidence that might indicate the existence of the Zones of Proximal Flow in the Scalable Game Design project is if, given the scaffolding provided by the project game design tutorials, students are able to advance to more sophisticated game and simulation challenges. To validate a student's intrinsic motivation in CT education, the program should show a sustainability of learning over a period of time; otherwise the student's educational intrinsic motivation would end after one experience with the program. Through our project, we have witnessed many project schools that were able to extend and transfer their students' learning abilities and problem solving skills to the next level of problem domains. In other words, through our project, students' learning abilities were increased to meet higher level of challenges incrementally.

Another piece of evidence that could show the existence of the Zones of Proximal Flow more directly would be to actually calculate the Computational Thinking Pattern present in student game or simulation artifacts. By employing the aforementioned Computational Thinking Pattern Analysis, we can actually see which skills students had previously gained to meet a given project challenge. From this, we can begin to see if our project challenges vs. skills graph resembles something akin to the Zones of Proximal Flow.

In the following section, we illustrate two kinds of evidence of the Zones of Proximal Flow: project sustainability and student programming abilities matched to the Zones of Proximal Flow graph.

4.1 Sustainability: Probability to Advance

One way to verify the Zones of Proximal Flow framework is to explore its sustainability. If indeed there is a notion of a gentle slope of game design and simulation authoring projects presenting gradually increasing challenges one would expect to see a high percentage of teachers and students advancing from one kind of project to a more sophisticated one.

Over the last 3 years, 72 different types of games and simulations have been collected from 46 participating schools. As part of the project, teachers were required to do at least one in-class game or simulation unit and were given compensation when they completed this unit; teachers were encouraged to do more than one unit but no compensation was given for additional units and additional in-class game or simulation design units were not required in any way. All 46 schools submitted at least one project (game or simulation) to the Scalable Game Design Arcade; of those 46 schools, 37 schools submitted two projects or more. Also, 30 schools and 23 schools submitted 3 and 4 projects or more, respectively (Figure 5). Interestingly, these numbers show that approximately 80% of classrooms moved forward from the previous project (or an attrition rate of 20% for each project): 81% of the schools that submitted at least one project, submitted two projects or more, and 80% of this second group submitted three projects or more.

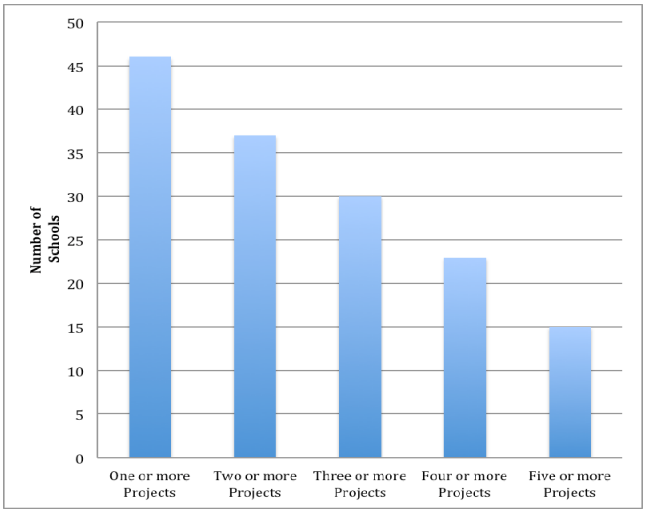

Figure 5. Probability to Advance: A high degree of sustainability is suggested by a large rate of advancement. Over 80% of schools advance to create a second project, of these 80% advance to a third project, of these 80% advance to a fourth project.

Considering teachers' short training timeframe and the lack of financial support after the first module implementation, the 80% success rate can be considered quite high, implying the existence of sustainability. Also, this result possibly indicates that many project schools have successfully helped students follow the Zones of Proximal Flow, spanning students' problem solving skills over multiple problem domains. Coupled with previous research that shows that teachers and students actively use the scaffolding provided by the project, this data indicates that the scaffolding is effective in overcoming mismatched challenges and

skills allowing students to move onto subsequent projects. It should be noted that this in no way proves the existence of the Zones of Proximal Flow, however, it does indicate that students, for the most part, had adequate skills to meet their programming challenges.

4.2 Zones of Proximal Flow in a Classroom

One might wonder if there was a more direct method to indicate the existence of the Zones of Proximal Flow. An initial attempt at this might involve finding a way to graph student challenges vs. student skills. Then, we could compare the student trajectories over time, in this challenge vs. skills space, to see if there is any resemblance to the Zones of Proximal Flow graph depicted in Figure 4. In other words, can we illustrate student skills and challenges that progress overtime through Scalable Game Design curriculum on the graph of Zones of Proximal Flow?

We have a method, developed by Koh et al [13], that allows us to calculate the "skills" a user shows in a given game creation task by looking at the existence and amount of individual Computational Thinking Patterns present in that game. This method is called Computational Thinking Pattern Analysis (CTPA). Computational Thinking Pattern Analysis measures the semantic meaning of the games/simulations programmed with AgentSheets or AgentCubes [13]. Through CTPA process, a student-programmed project can be converted into a holistic number per computational thinking pattern to assess student-learning abilities.

The Latent Semantic Analysis technique, as applied to CTPA, analyzes the implemented computational thinking patterns (CTP) in a given AgentSheets/AgentCubes project. CTPA compares a specific game/simulation with pre-defined canonical Computational Thinking Patterns using an LSA inspired technique. To perform CTPA, a given AgentSheets/AgentCubes project should be converted and expressed as a vector. The interpreted AgentSheets/AgentCubes project vectors are calculated with the Equation 1 to show its semantic meaning [13].

$$\text{CTPA}(m) = \left[\frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2} \sqrt{\sum_{i=1}^{n} v_i^2}} \right]_1^m$$

Equation 1. Computational Thinking Pattern Analysis

In this equation, u and v refer to a given project and one canonical computational thinking pattern respectively. N corresponds to the vector size of a project or Computational Thinking Pattern, and m refers to the number of Computational Thinking Patterns that are applied to CTPA. The calculated result of CTPA through CTPA (1) to CTPA (*m*) could be represented as an *m* length vector [13].

The CTPA calculated values, each element in *m* length vector, indicate student-learning abilities in each computational thinking pattern programming level. The length (norm) of this *m* length vector, a numerical value, is interpreted as the student's skill to design/implement a game/simulation. Also, this Skill Score equation is used to measure the challenges that students face (programming complexity) to program a game/simulation in the same manner to assess student-programming abilities.

$$\text{Skill Score} = \frac{\sqrt{\sum_{i=1}^{n} (P_i)^2}}{\sqrt{n}}$$

Equation 2. Skill Score

In the Scalable Game Design project-first approach, students are given a challenge, such as a simulation or game, which motivates them to gain the skills necessary to accomplish that challenge. Given that we can calculate student skills, by looking for the presence of Computational Thinking Patterns, we can begin to fashion what this challenges vs. skills graph might look like in a classroom context. Figure 6 depicts this initial attempt at this challenges vs. skills graph.

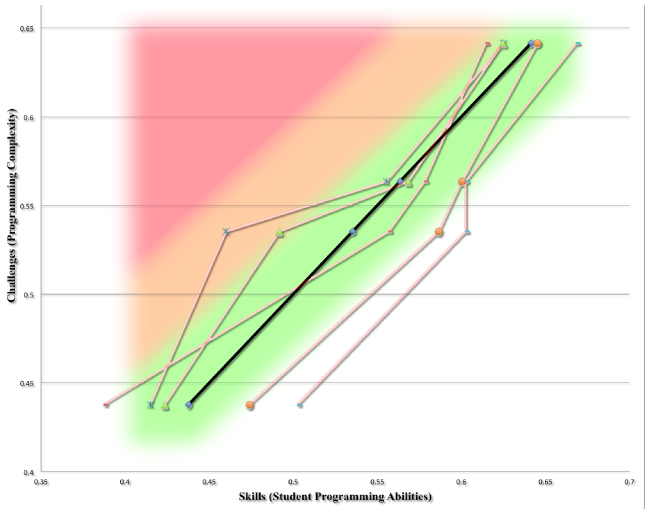

Figure 6: Student trajectory data resembling the Zones of Proximal Flow

The colored lines in Figure 6 represent five individual student trajectories over the course of four games. These five trajectories were chosen not because they are indicative of this particular class, but rather, they give an initial picture as to how different students advanced through the class. The black line in Figure 6 represents the game students would create if they followed the tutorial perfectly. The overlaid colored regions are meant to be a very informal representation of Flow, ZPD, and anxiety, and by no means imply that we know where the thresholds to these zones might exist.

To the right of the black line, we see a zone wherein students have additional Computational Thinking Patterns included in their project indicating a possibly more sophisticated game or simulation. To the left of the black line we see a zone wherein student project do not have all the Computational Thinking Patterns necessary to create that particular game or simulation present. This area indicates that student did not display all the skills necessary to meet the game or simulation authoring challenge. The black line, in essence, can be thought of as 'Flow' where the challenges and the student abilities are exactly matched.

We currently have no way yet of knowing how wide or narrow the Flow zone or the Zone of Proximal Development is in Figure 6. We currently speculate that if a graph of a student goes to the left side of the Flow line, then the student goes into the Zone of Proximal Development or Anxiety zone wherein the student might

need outside instruction to complete the challenge. Conversely, if a graph of a student goes to the right side of the Flow line, then the students has more expertise as compared to the challenge. Finally, if a given students progression line is too far to the left of the Flow line, we surmise that this might indicate student anxiety. If a given student's line is too far to the right of the Flow line, it may indicate student boredom.

5. DISCUSSION AND CONCLUSION

The Zones of Proximal Flow graph depicted in Figure 3 combines two pedagogical concepts to arrive at a strategy for keeping students engaged in classroom activities. This paper is an initial attempt to explain a strategy that applies the Zones of Proximal Flow theory and begins to present data possibly showing its existence. To that end, Figure 5 shows that classes readily tried more sophisticated projects indicating classroom engagement.

Figure 6 is an initial attempt at displaying student trajectories on a challenges vs. skills graph. In Figure 6 we see student progressions that begin to resemble the graph shown in Figure 3. This initial data is relevant to the Zones of Proximal Flow existence, however much more research must be done to support this theory. Further research needs to investigate where the thresholds for Flow, the Zone of Proximal Development, anxiety, and boredom lie in this space. Furthermore, this theory should be applied to other areas of classroom instruction, possibly providing teachers with a means of evaluating themselves or their classes giving them cues as to when outside instruction and scaffolding are effective and better

The Zones of Proximal Flow, a combined concept of Flow and the Zone of Proximal Development, is designed to promote student intrinsic motivation and leverage student learning experience. In our previous research, we have described the Zones of Proximal Flow as our project approach to broaden participation of minority and female students [ref], but it was just a theory. In this paper, we have shown early stage empirical data that begins to indicate its existence and possible effectiveness. It is our hope that future research into the Zones of Proximal Flow will paint clearer picture of this space, and be useful not only in the context of end-user programming, but have applications and facilitate engaging student curricula in a variety of different educational domains.

6. REFERENCES

[1] Squire, K. D. (2003). Video games in education. Int. J. Intell. Games & Simulation, 2(1), 49-62.

[2] Repenning, A., & Ambach, J. (1996, September). Tactile programming: A unified manipulation paradigm supporting program comprehension, composition and sharing. In Proc. Visual Languages, 1996. (pp. 102-109). IEEE.

[3] Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Computing Surveys (CSUR), 37(2), 83-137.

[4] Repenning, A. (2000). AgentSheets®: An interactive simulation environment with end-user programmable agents. Interaction.

[5] Ioannidou, A., Repenning, A., & Webb, D. C. (2009). AgentCubes: Incremental 3D end-user development. Journal of Visual Languages & Computing, 20(4), 236-251.

[6] Repenning, A. (2006, July). Excuse me, I need better AI!: employing collaborative diffusion to make game AI child's play. In Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames (pp. 169-178). ACM.

[7] Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010, June). Using scalable game design to teach computer science from middle school to graduate school. In Proc. of ITICSE '10 (pp. 224-228). ACM.

[8] Koh, K.H., Repenning, A, Nickerson, H., Endo, Y., & Motter, P., 2013. Will it stick?: exploring the sustainability of computational thinking education through game design. In Proc. of SIGCSE '13 (pp. 597-602) ACM.

[9] Repenning, A., Webb, D., & Ioannidou, A. (2010, March). Scalable game design and the development of a checklist for getting computational thinking into public schools. In Proc. Of SIGCSE '10 (pp. 265-269). ACM.

[10] Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. Learning & Leading with Technology, 38(6), 20-23.

[11] Wing, J. M. (2008). Computational thinking and thinking about computing.Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.

[12] "Front Matter." Learning Science Through Computer Games and Simulations. Washington, DC: The National Academies Press, 2011.

[13] Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010, September). Towards the automatic recognition of computational thinking for adaptive visual language learning. (VL/HCC), 2010 IEEE Symposium on (pp. 59-66). IEEE.

[14] Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011, March). Recognizing computational thinking patterns. In Proc. Of SIGCSE '11 (pp. 245-250). ACM.

[15] Basawapatna, A., Repenning, A., & Lewis. C., 2013. The simulation creation toolkit: an initial exploration into making programming accessible while preserving computational thinking. In Proc. Of SIGCSE '13 (pp. 501-506). ACM.

[16] Vygotsky, L. (1987). Zone of proximal development. Mind in society: The development of higher psychological processes, 52-91, Cambridge, MA, Harvard University Press

[17] Csikszentmihalyi, M., & Csikszentmihalyi, I. (1975). Beyond boredom and anxiety: The experience of play in work and games. San Francisco: Jossey-Bass.

[18] Csikszentmihalyi, M. (1997). Finding flow: The psychology of engagement with everyday life. Basic Books.

[19] Nakamura, J., & Csikszentmihalyi, M. (2009). Flow theory and research.Handbook of positive psychology, 195-206.

[20] Nakamura, J., & Csikszentmihalyi, M. (2002). The concept of flow. Handbook of positive psychology, 89-105.

[21] Webb, D., Repenning, A. and Koh, K. Toward an Emergent Theory of Broadening Participation in Computer Science Education. In Proc. Of SIGCSE '12 (pp. 173-178). ACM.

[22] Chaiklin, S. The zone of proximal development in Vygotsky's analysis of learning and instruction. Cambridge University Press, Cambridge, England, 2003.

[23] Griffin, P. and Cole, M. Current Activity for the Future: The Zo-ped. Jossey-Bass, San Francisco, 1984