Using Scalable Game Design to Promote 3D Fluency: Assessing the AgentCubes Incremental 3D End-User Development Framework

Andri Ioannidou
AgentSheets, Inc.
andri@agentsheets.com

Alexander Repenning
AgentSheets, Inc.
alexander@agentsheets.com

David Webb
University of Colorado
dcwebb@colorado.edu

Abstract

With the IT crisis reaching alarming levels, it is more important than ever to attract K-12 students to computer science. 3D game development can be an enticing way to achieve that, but building 3D games is far from trivial. Students need to achieve a degree of 3D fluency in modeling, animation and programming to be able to create compelling 3D content. The combination of innovative end-user development tools and standards-based curriculum promoting IT fluency by shifting the pedagogical focus from programming to design, can address motivational aspects without sacrificing principled educational goals. The AgentCubes 3D game-authoring environment raises the ceiling of end-user development without raising the threshold. Our formal user study shows that with Incremental 3D, the gradual approach to transition from 2D to 3D authoring, middle school students can build sophisticated 3D games including 3D models, animations and programming.

1. Introduction: Why Incremental 3D?

K-12[1] Information Technology (IT) education fails to attract the necessary number of students to Computer Science (CS) especially at the middle school level when students make critical career decisions by judging their own aptitudes towards math and science. Fueled by bad experiences with programming, middle school IT curricula have disintegrated into keyboarding, web browsing, word processing and PowerPoint workshops with little authentic enticement foreshadowing CS careers. This is a very serious problem because despite the growing need for IT workers, the enrollment in undergraduate degree-granting CS programs in the U.S. dropped by 70% between 2000 and 2005 [1].

[1] K-12 (Kindergarten through 12th grade) is the North American designation for primary and secondary education.

The notion of IT fluency is slowly gaining momentum in education as means to train and evaluate IT skills beyond just using applications. For instance, the Fluency with Information Technology (FIT) framework by the National Academy of Sciences [2] postulates a set of skills including meta-skills such as problem solving, creativity, working in groups, and algorithmic thinking. Game design [3] and computational science [4] are gradually establishing themselves as application domains capable of balancing educational and motivational concerns of IT fluency. With the right combination of tools, curriculum and teacher training, game design can be employed effectively to teach IT to middle school students in a motivating way.

A fundamental challenge to the notion of fluency is the need to define skills, explore motivational means of promoting skills, and devise ways to assess these skills. The focus of our research is to promote the notion of *3D fluency*. People live in a 3D world; meanwhile, because of computer gaming, today's computers are highly capable of processing 3D information. Unfortunately, creating computational 3D artifacts and games can be a truly daunting task. Even end users familiar with making 2D games are likely to find the transition to 3D to be a hard one. A completely new set of tools is usually necessary to create 3D models, to animate and program them. For instance, there is very little skill transfer from 2D paint programs such as Photoshop to a 3D modeling editor such as Maya 3D. The question is if this kind of discontinuity is a conceptual consequence of 2D versus 3D with potential roots in human cognition or if it is more of an accidental consequence of computational tools that have emerged disjointedly for 2D and 3D applications.

Our goal is to promote 3D fluency through a gradual approach that we call *Incremental 3D*. We reconceptualize the universe of 2D and 3D tools and skills as a continuum rather than a dichotomy. Most tools support either 2D or 3D authoring. For example,

NetLogo[2] [5] and Scratch[3] [6] are 2D authoring environments aimed at K-12; BlueJ[4] and GreenFoot[5] are targeted for more advanced students, typically at the undergraduate level, and Macromedia Flash at professional designers. Alice[6] [7], NetLogo 3D[2], StarLogo TNG [8], DarkBASIC[7], and Macromedia Director are 3D authoring environments with varying degrees of usability for different audiences. Some 2D tools are starting to integrate 3D authoring. However, some of them have limited degree of integration with the 2D product (e.g. Swift3D is a separate component for Flash) or force the user to drop from a visual language level to a textual language with a 3D API (e.g. Game Maker[8]). AgentCubes, on the other hand, is a tool that supports 3D authoring through incremental approaches for all components of the 3D authoring process, namely *modeling*, *animation*, and *programming*. A gentle slope approach allows end users to develop 3D games by first creating a 2D version of that game and then gradually moving along well-defined stepping-stones towards a 3D version. Our hope is that this incremental process ultimately allows end users to make 3D applications just as easily as 2D applications by transferring existing skills.

This paper assesses the idea of Incremental 3D as an approach for end users to create 3D games and acquire 3D fluency in the process. We first describe the components of Incremental 3D, namely incremental modeling, animation, and programming, in the context of AgentCubes, then outline the steps to transform a 2D into a 3D application, and report the findings from assessing 3D fluency in two schools.

2. AgentCubes: an Incremental 3D authoring environment

AgentCubes is a 3D rapid game prototyping environment that enables even 10-year-old children to make simulations (Figure 1) and games in just a few hours. While simple compared to commercial games, these are complete, playable games. Versatility is an essential characteristic for systems to be used for Scalable Game Design [9]. They should enable students to easily create simple content, but also allow the creation of more sophisticated content. AgentSheets [9, 10], our 2D simulation and game authoring tool, has a low threshold and a relatively high ceiling, but AgentCubes raises the ceiling considerably while keeping the threshold low. Rich media such as audio, 2D images, and 3D models, a 3D environment with layers, and camera controls to switch perspectives (first person vs. bird's eye view), and sophisticated user-controlled animations enable the creation of 3D games. While 3D authoring is far from a simple task, Incremental 3D is a scaffolding mechanism that provides considerable support for *modeling*, *animation*, and *programming*.

2.1. Incremental Modeling

Incremental 3D modeling is enabled through the Inflatable Icons technology [11]. Instead of limiting end users to only using stock 3D art (including licensed characters, such as The Sims in Alice and LEGO Star Wars in Scratch) or professional 3D modeling tools with very steep learning curves, such as Maya 3D, we enable them to gradually acquire 3D fluency in modeling by creating their own 3D models. With Inflatable Icons, users draw 2D images and gradually turn them into 3D models using diffusion-based inflation techniques.

Figure 1: A traffic simulation in AgentCubes.

Early user testing in local schools illustrated that students were able to make basic inflatable icons quickly, but needed additional means for producing more sophisticated 3D models, including benchmark shapes such as bugs and cars. Selection-based inflation is one such feature. We therefore created an Adobe Photoshop-inspired set of tools that allows users to make and extend pixel selections. For instance, we included a magic wand tool to make selections based on pixel color values. Say we want to create a frog. First we use the 2D editor with the symmetry mode enabled to sketch a frog (Figure 2a). In the 3D view, the frog shows completely flat (Figure 2b). Inflating the entire frog is a good start (Figure 2c) but fails to

[2] http://ccl.northwestern.edu/netlogo/
[3] http://scratch.mit.edu/
[4] http://www.bluej.org/
[5] http://www.greenfoot.org/
[6] http://www.alice.org
[7] http://darkbasic.thegamecreators.com/
[8] http://www.gamemaker.nl/

highlight the strong legs of the frog. Using the magic wand, the frog legs get selected and inflated more (Figure 2d).

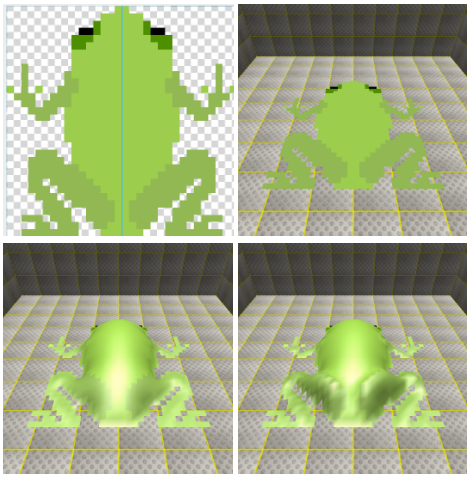

Figure 2: A frog as an Incremental 3D shape.

2.2. Incremental Animation

End users that program 3D worlds appear to have higher expectations for run-time behavior. For instance, if agents move or rotate, users would like to have at least the option to have the world change in an animated way. With no animation, the agents in Figure 3a that are programmed to simply move and rotate randomly, would instantly arrive at the next frame (Figure 3c) without seeing any in-between frames. However, with animation, the agents move and rotate smoothly in a series of multiple frames such as the one shown in Figure 3b.

AgentCubes supports incremental animations. That is, initially users may not need or want to deal with animations. As they are getting ready, they can access animation parameters that are optional to language pieces such as move and rotate actions. Moreover, built-in scene awareness assisted by the notions of grids, stacks and layers (e.g. built-in gravity) significantly scaffolds 3D animation authoring for users. Finally, the Parallel Time-Jump animation approach [12] allows any number of agents to animate in parallel without the need to track object locations and the overhead of sequential animation.

2.3. Incremental Programming

To support 3D fluency, we needed a programming language that allows students to create behavior in 3D. Our conceptual starting point was our previous work with the Visual AgenTalk (VAT) programming language in AgentSheets [13]. VAT had established the usability of the rule-based approach for authoring 2D games and simulations/computational science applications in school settings [14]. For AgentCubes, we enhanced the language to include the notion of Incremental 3D, leading to *Visual AgenTalk 3D*, which includes the ability to author and run 2D projects and gradually add control over 3D aspects. VAT 3D has the following characteristics:

• **3D grid**: the AgentCubes worlds consist of layers with stacks of agents. VAT 3D features conditions and actions that orient and move agents in 3D, providing incremental support through optional parameters.

• **Camera control**: attaching cameras to agents (first person view) makes the agent the location of the camera. If the agent moves, the camera will move too. If this agent turns, the camera will turn too. This seemingly simple extension resulted in a number of cognitively interesting challenges, including the need to have conditions to test if the simulation is currently running in bird's eye or first-person view.

• **Lighting control**: end-user support for the use of light sources in sophisticated scene rendering.

• **Formula language**: The formula language allows users to express equations as functions of agent attributes using special notation to access agents via their grid locations in relative and absolute terms similar to spreadsheets. Unlike AgentSheets, which features a 2D spatial structure and operators to express computation in 2D, AgentCubes allows users to express computation in 3D.

Figure 3: (a) Cube agents programmed to move up to 4 cells and rotate randomly. (b) A snapshot of a frame in the middle of animation, showing the agents moving and rotating. (c) Agents arrive at their final positions at the end of the cycle. Without animation, the viewer would only see the first and last frame without the intermediate animation frames.

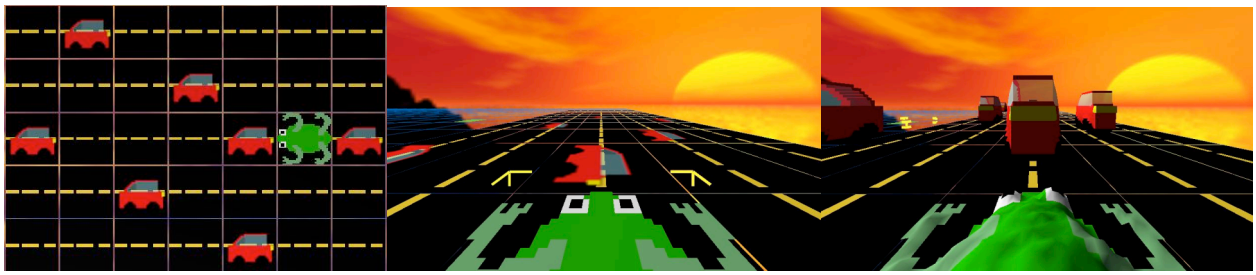

Figure 4: (a) Bird's eye view of Frogger; (b) flat frog in first person looking at flat cars; (c) 3D frog looking at 3D cars.

• **Animation support**: Optional animation parameters in movement and orientation language pieces enable the separation of logic and animation in agent behavior, thus ensuring that the logic part works without obliging the user to first define animation.

3. Incremental 3D process in Game Design

Student progression to 3D fluency is established by having a process that is gradual enough to keep students in the optimal flow of learning [15]. The following steps move students through the process of creating a 3D game starting with a 2D game:

1) Creating a 2D game: Students are guided through a game design process we call Gamelet Design to create an initially 2D version of a game. We typically use the classic arcade game of Frogger[9] because even young kids are aware of it and it seems to be gender neutral. The result is a simple but completely playable version of the first level of the game with a cursor-controlled frog trying to cross a highway with cars driving across, cars automatically being generated and absorbed at the beginning and end of the highway respectively, and dealing with the car-frog collision that results in the frog perishing and being regenerated again. The 2D version of the game (Figure 4a) does not include custom animations or 3D models at this point.

2) Creating a first person 2D game: Using incremental modeling, animation, and programming, the look and basic behavior of the 2D Frogger game gets transformed to 3D. We motivate the transition from 2D to 3D by attaching the camera onto the user-controlled character, namely the frog, and therefore changing perspectives from a world view where the user looks at the game world from a bird's eye view to a first-person view where the user sees the game world through the "eyes" of the frog (Figure 4b). After the initial "the world is flat" shock, students typically want to create 3D looking objects. Inflatable Icons are used for incremental modeling to create 3D game objects from the 2D images that the students had created during the previous step (Figure 4c). Seeing the game run and the jerky movement of the cars prompts students to change the animation parameters for the movement. To make the games seem more realistic, AgentCubes supports different animation modes (constant vs. accelerated). For cars, for instance, it makes sense to have constant animation speed, whereas for the frog it is better to have accelerated animation to simulate jumping. Moreover, simple behavior changes are incrementally implemented. With the camera attached to the frog, the students see the need to rotate the character when it changes direction, so they add rotation actions to the behavior.

3) Creating a First Person 3D Game: Modifying the look of game objects is not enough to create a 3D game. The transition from bird's eye to first person camera view also means that the coordinate system changes, which presents a conceptual perspective issue for navigation. The "absolute" right, left, up, down directions that make sense when looking at the world from a bird's eye view, no longer make sense in first-person mode (Figure 5 and Figure 6). Students expect the user-controlled character to transition seamlessly from absolute to relative coordinates. Instead, they need to implement additional navigation behavior to deal with the relative coordinate system.

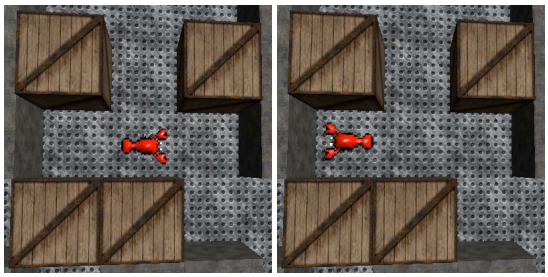

Figure 5: (a) Lobster in bird's eye view; (b) result of using the left arrow key: the lobster turns and faces to the left.

With an incremental behavior approach students are taught how to implement first-person vs. world-view navigation, extending existing code with language able to deal with 2D and 3D version of behavior. This is a fairly difficult concept that requires more than trivial programming, but at the same time presents great opportunities for learning about coordinate systems and modulo arithmetic – a concept not covered in the middle school math curriculum. Game design provides many such opportunities for learning complex concepts

[9] http://en.wikipedia.org/wiki/Frogger

on demand, rendering it an experience that synthesizes many different skills from various STEM (Science, Technology, Engineering, Math) domains, not just programming. Indicative of this was a quote from the only student who indicated he knew about modulo arithmetic in our experiment: "I knew about modulo arithmetic, I understood it, but now I know how to apply it".

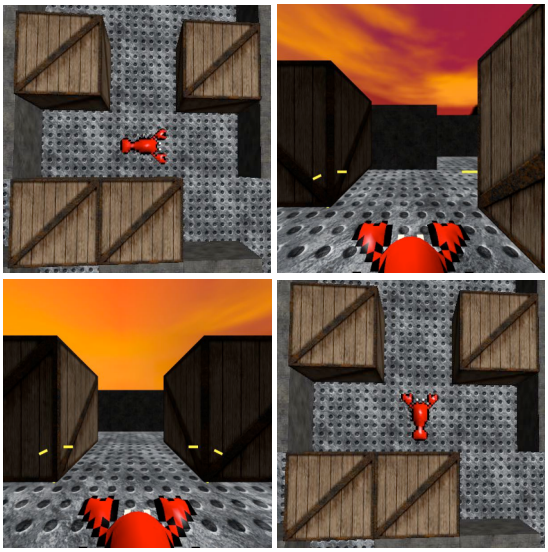

Figure 6: (a) Lobster in bird's eye view; (b) lobster in first person view; (c) result of using the left arrow key: the lobster turns to *its* left; (d) result viewed from birds' eye view: lobster is actually facing up in the absolute coordinate system.

At this point, students have a simple but complete 3D game.

4) Constructing a 3D world: As a final step, we introduce students to a truly 3D world. Not only are the objects of the game 3D, but there is movement in all three dimensions using layers in the 3D grid. This 3D environment enables students to first navigate a ready-made 3D maze and then construct their own mazes by directing the movement and rotation of a spaceship drilling holes in a solid cube. Indicators of 3D fluency in this activity are specific design aspects of the mazes students create (e.g. toggling between bird's eye and first-person views, toggling between visible and invisible walls to evaluate the maze structure, rotating the world to view the possible routes in the maze) and the usage of orientation and visualization tools to verify if the maze satisfies the given design criteria.

4. Assessment

We formally evaluated the effectiveness of the Incremental 3D approach as a way to achieve 3D and IT fluency at the middle school level.

4.1. Study Design

Collaborating with educational researchers from the University of Colorado's School of Education with experience in working with students in technology-intensive instructional environments, as well as expertise in conducting classroom-based research in K-12 settings, we designed an evaluation study. The study was designed to document the impact of student use of AgentCubes on identifiable learning objectives with respect to the development of student IT and 3D fluency, mainly following the FIT framework. Given the scope of the feasibility study, we focused on a subset of FIT and 3D fluency elements that included IT Skills such as using a graphics package to create illustrations, IT Concepts such as algorithmic thinking and programming, and Intellectual Capabilities such as managing complexity, engaging in sustained reasoning, and managing problems in faulty situations.

Instruction followed the Incremental 3D steps mentioned above. In addition to formative evaluations during instruction, upon completion, to measure fluency, we designed problem-solving situations in which students were asked to troubleshoot authentic programming scenarios. Instead of traditional pre/post tests, we opted to perform an authentic assessment that would require students to draw upon what they had learned about game design and programming agent behavior to identify and solve troubleshooting scenarios in an intentionally defective version of a 3D Frogger game given to them. Within a 45-minute period students had to figure out at least five things that were wrong with the game and re-program the agents' behavior to fix those problems. These included issues with movement in world and first-person view, missing behavior, and defective generation rates.

The debugging scenarios were challenging since students were neither told what the problems were nor how to locate the problematic procedure within the AgentCubes environment. They needed to identify the problem, locate the problematic agent and its behavior, locate the exact problematic procedure in the code, and correct the program for the agent. The troubleshooting tasks were unfamiliar situations to students and were not discussed in previous sessions. Students were required to complete the activity on their own and could only ask the instructors questions of clarification.

We recognized that offering students an opportunity to engage in troubleshooting was an authentic experience familiar to any computer programmer. It required managing problems in faulty situations in addition to sustained engagement in reasoning and application of programming skills. Our eagerness with

presenting such tasks to students was tempered by uncertainty regarding students' ability to identify the problems, student insight in locating the problematic procedures for a given agent, and knowing how to resolve the problems. However, using the troubleshooting assessment to gather evidence of student FITness was rewarded by the intensity of student engagement throughout the assessment and what students were able to accomplish, which is discussed in the findings section.

4.2. Contexts

The evaluation study was administered in collaboration with Science Discovery, the University of Colorado's science outreach program, and was conducted in the context of 4 after-school classes in two middle schools, one in Boulder and one in Aurora, Colorado. Forty students attended the initial session. The race and ethnic background of students recruited for the AgentCubes course was a close approximation of the respective background of students found at the participating schools, with the majority of participants at the Boulder school reporting a Caucasian background and the majority of participants at the Aurora school reporting a Hispanic background (Table 1). Participation was on a voluntary basis. A large number of students was recruited by researchers and teachers. School administration and teachers whittled the recruitment group down to the 40 students we could accommodate in the experiment. The requirements included having two groups of all-female students and a participant sample that represented the school population. It is also interesting to note that Boulder is a technology hub region whereas Aurora is an inner-city, low-income area.

Table 1: Study participants from Aurora (top) and Boulder (bottom) schools

	Male	Female	Total	AgentCubes	School
African-Am	4	2	6	30%	17%
Asian-Am	0	1	1	5%	3%
Caucasian	1	0	1	5%	11%
Hispanic	4	6	10	50%	68%
Multi-Eth	0	1	1	5%	nr
Native-Am	1	0	1	5%	1%
	10	10	20		

	Male	Female	Total	AgentCubes	School
African-Am	1	0	1	5%	1%
Asian-Am	0	0	0	0%	4%
Caucasian	7	8	15	75%	84%
Hispanic	0	1	1	5%	11%
Multi-Eth	1	2	3	15%	nr
Native-Am	0	0	0	0%	< 1%
	9	11	20		

4.3. Findings

The findings resulting from the overall evaluation study are grouped in three categories: 1) technology; 2) curriculum; and 3) broadening participation.

4.3.1. Technology: For the technology category, the criterion to measure success was whether students can build a simple game from scratch, including 3D models and behavior programming in a short period of time (< five hours). The technology findings (TF) are:

TF1) All students were able to create a working 3D game in less than five hours: All students made at least one game. Multiple kids went beyond what was expected in class and created extra games. It is interesting to note that it was mostly boys from the Aurora school that did that.

TF2) All students were able to create sophisticated 3D models from scratch using Inflatable Icons: The Inflatable Icons technology turned out to be highly accessible to all students. Inflatable Icons were able to cover the spectrum from rough and ready abstract looking 3D model drafts all the way to sophisticated 3D models. It is interesting to note that, on average, girls spend more time and paid more attention to detail in creating their 3D models than the boys.

TF3) All students were able to add animations to their games incrementally and customize animation parameters: Students were able to enable and disable animations as well as customize them. Customization allowed students to control the animation timing and acceleration parameters. The incremental nature of the animation approach built into AgentCubes allowed students first to build a game and then, when necessary, add the animations after they had developed the main game mechanics.

TF4) Most students (85%) were able to program their own character control in 1st person and bird's eye view successfully: This was a very challenging task: it included understanding and application of modulo arithmetic, a concept that is unfamiliar to middle school students. Even so, students were able to follow instruction and 85% of them were able to complete the implementation of the challenging first-person navigation. Also, 75% of them were able to fix the intentionally defective version of first-person navigation in the unassisted troubleshooting session.

4.3.2. Curriculum: The criterion to evaluate curriculum was based on achievements towards FITness goals. The curriculum findings (CF) are:

CF1: Most students (75%) were able to solve most issues (60% or more) in the troubleshooting activity.

Almost all students demonstrated sustained engagement and persistence in resolving these problems. All students were able to identify at least three of the problems and attempted to resolve the problem by reprogramming agent behavior. As a matter of fact, 75% of students solved the majority of the issues (3 or more).

Table 2 summarizes the percentages of students able to troubleshoot each scenario. Please note that out of the 40 original participants, a subset of 24 students participated on the day the troubleshooting activity was administered. In addition to overall results, data is disaggregated by school, gender, and ethnicity. It is worth noting that female students and students at the Boulder School were more successful in resolving car movement and generation issues. Male students and students at the Aurora school were more successful in resolving the scenarios related to frog movement.

Furthermore, 25% of the students went beyond the scope of the activity and improved the program in other ways, such as using the graphics tools to change or inflate game components such as cars and turtles so they would be easier to see in first-person view.

Table 2: Percentage of students identifying and completing troubleshooting tasks: 1) one type of cars not moving; 2) other type of car stacking up; 3) 2D navigation not functioning correctly; 4) 3D navigation not functioning correctly; 5) not enough turtles to make game playable.

		Troubleshooting Tasks					
Groups	**N**	**1**	**2**	**3**	**4**	**5**	**Average**
All Students	24	67%	88%	79%	75%	42%	70%
Schools							
Boulder	14	71%	93%	64%	64%	50%	69%
Aurora	10	60%	80%	100%	90%	30%	72%
Gender							
Male	16	63%	81%	88%	88%	50%	74%
Female	8	75%	100%	63%	50%	25%	63%
Ethnicity							
Caucasian	13	69%	92%	69%	62%	46%	68%
Hispanic	5	60%	80%	100%	100%	20%	72%
Afr-Am	3	67%	100%	100%	100%	33%	80%
Other	3	67%	67%	67%	67%	67%	67%

CF2: Scalable Game Design is a feasible strategy to create FIT-oriented curriculum using AgentCubes: Data from our observation protocols was analyzed in terms of opportunities to address the five elements of FIT mentioned above over the five class sessions. A hierarchical rating scheme was developed to distinguish potential opportunities from observed opportunities with and without guidance. Every session included opportunities to address multiple goals, but what distinguished the latter sessions from the earlier ones were the opportunities for students to demonstrate their achievement of FIT goals apart from instruction. From the results of the assessment activities (troubleshooting and 3D world construction) we concluded that the instruction using AgentCubes did result in the development of student fluency in 3D and IT across several elements, in particular algorithmic thinking programming and managing faulty situations.

CF3) Students have capacity for visualization and representing 3D objects as illustrated by their ability to navigate 3D mazes and create their own: All students were able to navigate and create their own 3D mazes with varying degrees of complexity. With AgentCubes students could create a 3D maze by drilling holes in 3 dimensions into a solid large cube following specific design criteria and with the expressed goal of constructing a maze that would offer sufficient challenge to their classmates.

4.3.3. Broadening participation: The criterion we used to evaluate this category was whether the technology and curriculum could be used across ethnicity and gender, both in technology hub areas and inner city school cultures. The broadening participation findings (BPF) are:

BPF1) The idea of Game Design is compelling to middle school girls. We were able to easily recruit more than 50% girls. The percentage of female students involved at both schools was greater than 50%. Organizing the weekly sessions by gender may have had some influence on the ability to recruit a higher percentage of female students to agree to participate in these sessions. This was influenced by earlier experiences in recruiting female students in after-school STEM courses offered by Science Discovery. Student attendance over the five sessions experienced some attrition, with the most significant attrition occurring among the Aurora school female group. Based on follow-up discussions with teachers and students, it appears that there were various reasons for this attrition such as overlapping family commitments or other after school commitments. It is not uncommon to find high attrition rates in voluntary after-school programs. Strategies to address this would be to either offer the AgentCubes sessions during the school day (i.e., as part of a computer education course) or provide a more compact after-school course over the course of one week (five half-day sessions) rather than organized as two-hour sessions each week over a period of five weeks.

BPF2: The under-privileged school did better than the privileged school in authentic assessment (but difference is not statistically significant). The troubleshooting performance of students at both schools was essentially the same. The Aurora students outperformed the Boulder students on the challenging frog movement tasks.

BPF3: There was no major difference between the ethnicity groups in troubleshooting performance. From Table 2, we see that African American students on average completed 80% of the troubleshooting tasks during the authentic assessment activities. Hispanic students on average completed 72% of the tasks. Caucasian students on average completed 68% of the tasks. Other Ethnicity students on average completed 67% of the tasks. Note that both, the African American and the other ethnicity groups were small (n=3).

5. Conclusions and Future Work

Our preliminary experiences and findings with Scalable Game Design, our low-threshold/high-ceiling framework supporting skills beyond programming, ranging from theoretical design skills to concrete development skills, lead us to believe that we can establish IT fluency and broaden participation in computer science with game design activities. Preliminary results from the studies indicate that it is educationally effective to use AgentCubes as a low-threshold game design environment featuring Incremental 3D for teaching IT skills to middle school students. To have a systemic impact, we will scale up research and development along different dimensions:

• *technology*: provide more scaffolding techniques especially for incremental programming.

• *content and curriculum*: develop longer modules offered as part of the curriculum for comprehensive coverage of IT standards.

• *teacher training*: a systematic approach to teacher training is essential for technology adoption in schools.

• *social factors*: explore the factors leading to the somewhat disappointing attrition rates for girls, given their interest in game design and ability of achieving the level of fluency required to create their own games.

6. Acknowledgements

This work is supported by the National Science Foundation under Grant Number No. IIP 0712571. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

7. References

[1] Computing Research Association, "CRA Bulletin: Enrollments and Degree Production at US CS Departments Drop Further in 2006/2007," 2008, (available at http://www.cra.org/wp/index.php?p=139).

[2] Committee on Information Technology Literacy, National Research Council, Being Fluent with Information Technology. Washington, D.C.: National Academy Press, 1999.

[3] K. Salen and E. Zimmerman, Rules of Play: Game Design Fundamentals. Cambridge, MA: The MIT Press, 2003.

[4] President's Information Technology Advisory Committee (PITAC), "Report to the President: Computational Science: Ensuring America's Competitiveness," June 2005.

[5] U. Wilensky and W. Stroup, "Learning through Participatory Simulations: Network-Based Design for Systems Learning in Classrooms," in Computer Supported Collaborative Learning (CSCL '99), Stanford University, CA, 1999.

[6] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick, "Scratch: A Sneak Preview," in Second International Conference on Creating, Connecting, and Collaborating through Computing, Kyoto, Japan, 2004, pp. 104-109.

[7] M. Barbara, L. Deborah, and C. Stephen, "Evaluating the effectiveness of a new instructional approach," in Proceedings of the 35th SIGCSE technical symposium on Computer science education Norfolk, Virginia, USA: ACM, 2004.

[8] E. Klopfer and S. Yoon, "Developing Games and Simulations for Today and Tomorrow's Tech Savvy Youth," TechTrends, vol. 49(3), pp. 33-41, 2005.

[9] A. Repenning and A. Ioannidou, "Broadening Participation through Scalable Game Design," in Proceedings of the ACM Special Interest Group on Computer Science Education Conference, (SIGCSE 2008), Portland, Oregon USA, 2008.

[10] A. Repenning and A. Ioannidou, "Agent-Based End-User Development," Communications of the ACM, vol. 47(9), pp. 43-46, 2004.

[11] A. Repenning, "Inflatable Icons: Diffusion-based Interactive Extrusion of 2D Images into 3D Models," The Journal of Graphical Tools, vol. 10(1), pp. 1-15, 2005.

[12] A. Repenning and A. Ioannidou, "AgentCubes: Raising the Ceiling of End-User Development in Education through Incremental 3D," in IEEE Symposium on Visual Languages and Human-Centric Computing, 2006, Brighton, United Kingdom, 2006, pp. 27- 34.

[13] A. Repenning and A. Ioannidou, "Behavior Processors: Layers between End-Users and Java Virtual Machines," in Proceedings of the 1997 IEEE Symposium of Visual Languages, Capri, Italy, 1997, pp. 402-409.

[14] A. Ioannidou, A. Repenning, C. Lewis, G. Cherry, and C. Rader, "Making Constructionism Work in the Classroom," International Journal of Computers for Mathematical Learning, vol. 8(1), pp. 63-108, 2003.

[15] M. Csikszentmihalyi, Flow: The Psychology of Optimal Experience. New York: Harper Collins Publishers, 1990.