A Study of Science Teachers Utilizing Visual Programming Techniques

Cheryl Denise Seals
Computer Science & Software Engineering, Auburn University

and

L. Octavia Tripp
Curriculum & Teaching, Auburn University

1. ABSTRACT

This paper presents a study of learning in Stagecast Creator to discover more about novice programmer teachers, direct manipulation techniques and exploration of methods to create interactive lessons for their classrooms. The authors performed a longitudinal guided exploration of Stagecast Creator with two middle school science teachers. The results of these evaluations help to identify implications for educational simulations for novice programmer teachers and produce a set of initial system requirements.

1. INTRODUCTION

A study of learning in Stagecast Creator was conducted to discover more about novice programmer teachers and their direct manipulation techniques. The teachers that we selected for the study were science teachers that wanted to learn to utilize more technology in their curriculum. To further refine our requirements for future research study, and become more familiar with science teachers domain of discourse about computers and programming, we performed a longitudinal guided exploration of Stagecast Creator with two middle school science teachers. The results of these evaluations help to identify implications for educational simulations for teachers and produce a set of initial system requirements, which will be utilized to refine subsequent studies.

2. COMPARATIVE ANALYSIS OF SIM ENVIRONMENT

During the preliminary stages of this research project, we identified our argument of aiming to provide software with improved usability to increase teacher adoption of new educational software technology in their classrooms. We would begin our journey by studying existing simulation creation software, select software that most closely matched our teachers' classroom needs, and then to modify existing or design and create our own software to support our teachers and their reuse strategies.

Our first initiative was to conduct a language survey and create taxonomy to narrow the field of candidates for further study. The criterion for selecting these languages was that they represent different types of software that will support educational simulation. The plan was to evaluate at least one example from each of the following categories: Visual Programming; Programming by Demonstration; Simulation/Construction Kit and Software Reuse/Visual OOP. In the visual programming environments category we were introduced to completely visual environments, and one example of this domain is LabVIEW. LabVIEW is a scientist's construction kit to build Virtual Instruments by direct manipulation. The system programming is handled by three palettes, which are used for coding, debugging, and creating. All of the programming is handled by icons in these palettes, but with a wealth of icons the programming in this environment would be very complicated for novice users with having to negotiate the cognitive overload of learning to create programs with sequences of icons as in traditional programming with sequences of text. LabVIEW would be the most useful in this category for building working educational simulations, but this selection would be very complex and cost prohibitive (i.e. greater than $1000) for the average middle science school teacher.

In the programming by demonstration category, we were introduced to environments where all the rules are created with the use of a macro recorder device and the demonstration of actions are rendered as rules. This is acceptable for simple rules, but makes the creation of more complex rules more difficult. Of this category we will give details of one example, Stagecast Creator. Stagecast Creator is a simulation micro-world building environment, where programming actions are created with programming by demonstration techniques, and rules can be modified with direct manipulation. This environment had facilities geared to support novice programmer usability and would be useful in the creation of educational simulations, and could be procured for minimal cost (i.e. less than $100). Also, ActivChemistry (i.e. Chemistry construction kit and provides user a fixed number of parts to combine to perform experiments) was a very good candidate for further study, but only supports the Chemistry content area. We are identifying software packages that can be used by all science and mathematics teachers: therefore, environments we study should be able to provide more generalized solutions.

In the simulation/construction kit category, we were introduced to environments that supported the construction kit style of rule creation. The user simply chooses the rules they would like to use and just have to assign an order for rules to fire. This is a very good technique for novice users to create educational simulations and, one example, AgentSheets already had been

used by students to create over a hundred simulations and could be procured with minimal cost (i.e. less than $100).

In the software reuse/visual object-oriented programming category we were introduced to environments that are object-oriented in their implementation and support software reuse. These environments would be very good for programmers to build simulations, but have little support for novice programmer simulation creation.

Creating this taxonomy of visual languages helped us to reduce our field of candidate languages based upon which languages we felt would be easily accessible to our teachers and not too cost prohibitive. We also wanted languages, which would support novice learning and reuse in the creation of visual educational simulations. We selected three simulation/construction kits (i.e. AgentSheets, Cocoa, and Hyperstudio) and one programming by demonstration system (i.e. Stagecast Creator)

Early on, it was found that our teachers had very little time to spend learning new technology and not enough time to build technology from scratch. They prefer to adapt content to make it more relevant to their lessons. Therefore, instead of using an exhaustive systems approach to learning to build simulations, minimalist self-study instruction was provided to teachers both to reduce their time in learning to use the environment [Carroll, 1990] and to explore the benefits of reuse programming in a visual simulation programming environment [Perrone, Repenning, 1998]

3. TEACHERS AS SIMULATION BUILDERS

This study pictured teachers as simulation builders who, as content matter experts in their classrooms, were the best candidates for designing educational simulations. Research here worked to reduce teacher frustration with programming. In many instances, teachers knew exactly what they wanted to model, and could create this functionality in the current AgentSheets computational model. Many teachers' domain analysis of the problem was very robust and sophisticated, but most rendered very simplified version of their models. However, since students' knowledge of the domain was limited, they would build structures that simple "look right" [Lewis et al., 1997].

With current tools, teachers might have difficulty transferring their elaborate mental models into working simulations without greater programming skill. In order to mitigate this difficulty for teachers, improved instruction with an example-based tutorial presenting different simulations of the same phenomenon could be provided that included variation in complexity and coverage. This would give teachers better understanding that a range of models can be generated from simplified problem analyses. The types of mechanisms, which are easy or difficult to model, could also be explained; however, this technique would limit exploration, which is in conflict with a minimalist hands-on approach to learning.

Even though these are simplified approaches to programming, many of the problems the teachers experienced appear to be similar to issues discussed for years in the literature on OOD and programming [Rosson, Carroll, 1996; Rosson, Carroll, Bellamy, 1990].

4. STAGECAST CREATOR STUDY

Our aim was to identify ways to motivate these novice programmers to create simulations, and continue to make their own contributions to our planned virtual learning community. In previous experiments, students were generally successful in their work with SC and reported that they enjoyed their experience. Our hope was that if teachers enjoyed building simulations they would utilized the tools and virtual community made available. Also that if their students had fun creating simulations, the students will spend more time in the environment to learn more about the content material, and with continued experience learn more about visual simulation programming.

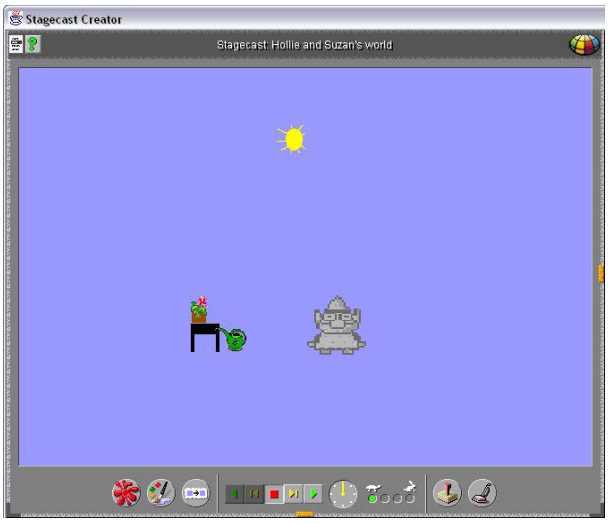
The environment that we illustrate in the study is Stagecast Creator (SC), which is based on a movie metaphor and users create a cast of characters who interact within a simulation microworld. Users create SC simulations with a macro-recorder device that allows users to program by demonstrating and example, and also by direct manipulation. To demonstrate a rule the user selects the character to be programmed and as in the Figure below a bounding box or "spotlight" highlights the character to be edited. In this example the user wants to cloud to float across the sky. They would simple drag the cloud forward and that would record a new rule for this character. This makes it very easy for novice users to begin rule creation, but there are still some semantic complexities that arise. The spatial context and visual appearance are requirements of the rule's precondition. For example if two characters are next to each other while a rule is being demonstrated, both objects would always have to satisfy the precondition or the would not be executed.

5. LONGITUDIANL STUDY OF STAGECAST

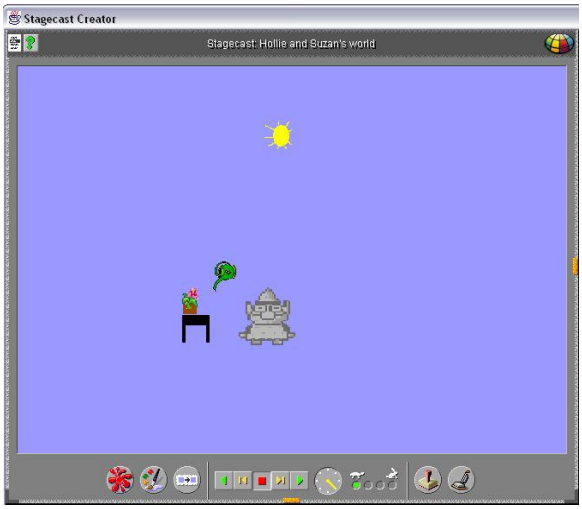
While developing a robust prototype, we discussed content material with at two middle school science teachers. Our development and studies focused on teachers in grades six through nine, and we selected two teachers who participated in the formal evaluation, and thus are already familiar with visual programming systems.

Similar to participatory design methods we wanted to get requirements for simulations from representatives of our potential user group, and get their perceptions on what they expected from a software curricula aid and ideas for simulations. We planned to perform 10 sessions with our two teachers. During our meetings we introduced them to a visual programming environment, gave them instruction to bring help them to bridge the Zone of Proximal development [Vygotsky]. There was considerable distance between what the users knew initially and their potential for knowing and creating in the area of visual programming. Our aim was to increase their level of competence in this area by coordination with the proficient or a more knowledgeable other [McMahon 1996].

In order to get more familiar with our teacher participants and their comfortability with computers and novice visual programming of simulations, we had an opportunity to perform a longitudinal study with two middle school science teachers scheduled for 10 weeks. The tool that they were to investigate was Stagecast Creator. We began their experience as a typical tutorial with guided exploration cards to guide their self-study. Teachers worked individually, during their learning session. It was suggested that their first learning session be structured as a minimalist tutorial in the form of guided exploration cards. The tutorial guides the teachers through starting up the system, opening a micro-world example, interacting with that example and examining the rules and creating new rules. For their second and third sessions the teachers explored some of the system supplied and facilitator supplied examples. Now that the teachers are more familiar with the environment and some simulation examples that can be created, we asked them to bring in a set of requirements. In specific we asked the teachers what topics they would cover during the year and of these topics, which they think, would be good candidates for educational simulations. For their forth session we began by reviewing the following suggested simulation topics for middle school physical science: Scientific Method, Physical Properties, Atomic Theory, and Chemical Properties.



A.



B.

FIGURE 1 TEACHER CREATED PHOTOSYNTHESIS

And the second simulation scenario "Raisins in ginger ale" was an illustration of scientific inquiry as well. The second simulation experiment was set up with ginger ale and raisins in a glass. The scenario is that you pour ginger ale into a glass and next the student adds raisins to the glass. The effect of carbonation in the water will cause the raisins to bounce to the top of glass propelled by carbon dioxide bubbles. Once the bubble reaches the surface it will burst and the raisin will fall back to the bottom of the glass. This cycle will repeat until the level of carbonation is to low and finally all the raisins will finally settle at the bottom of the glass. This scientific method experiment is illustrated in Figure 2.

To give the teachers a tangible task for their next four sessions, their task was to create two simulations that they would use in their first month of science classes. Both teachers agreed that their first in class experiment illustrated the scientific inquiry. And to illustrate the scientific inquiry the teachers would illustrate natural phenomena with a small experiment, ask their students to explain their observations, and finally the teacher will explain the scientific phenomena. The first set of simulations created by our teachers dealt with the area of scientific inquiry and their first two classroom experiments were to illustrate "cause and effect" scenarios. Scenario one was "If you add water and sunlight to a plant, then the plant will grow" as an illustration of photosynthesis. And the second simulation scenario was an illustration of scientific inquiry as well.

The first simulation created by our teachers was an illustration of photosynthesis and was described as follows: You begin the experiment with the sun shining and a flower in a pot. With the sun shining, as the gardener waters the flower it grows. This phenomenon is caused by photosynthesis, when the chlorophyll in the plant receives sunlight and water; this is transformed into energy and causes the plant to grow. In the simulation the gardener walks toward the watering pot, waters the flower, and the flower grows in a three-stage animation. The photosynthesis scientific method experiment is illustrated in Figure 1.

6. TEACHER INTEREACTION DISCUSSION

When our teachers began this project their first question was "Do we have to draw a glass?" My answer was no we want to investigate you simulation creation techniques not your drawing ability so if we can find usable images we will use those and the ones we can't find we will have to create. Their next question was "Where can we get a big glass to put the raisins in?" the answer was "Let's go to the web and do a Google search."

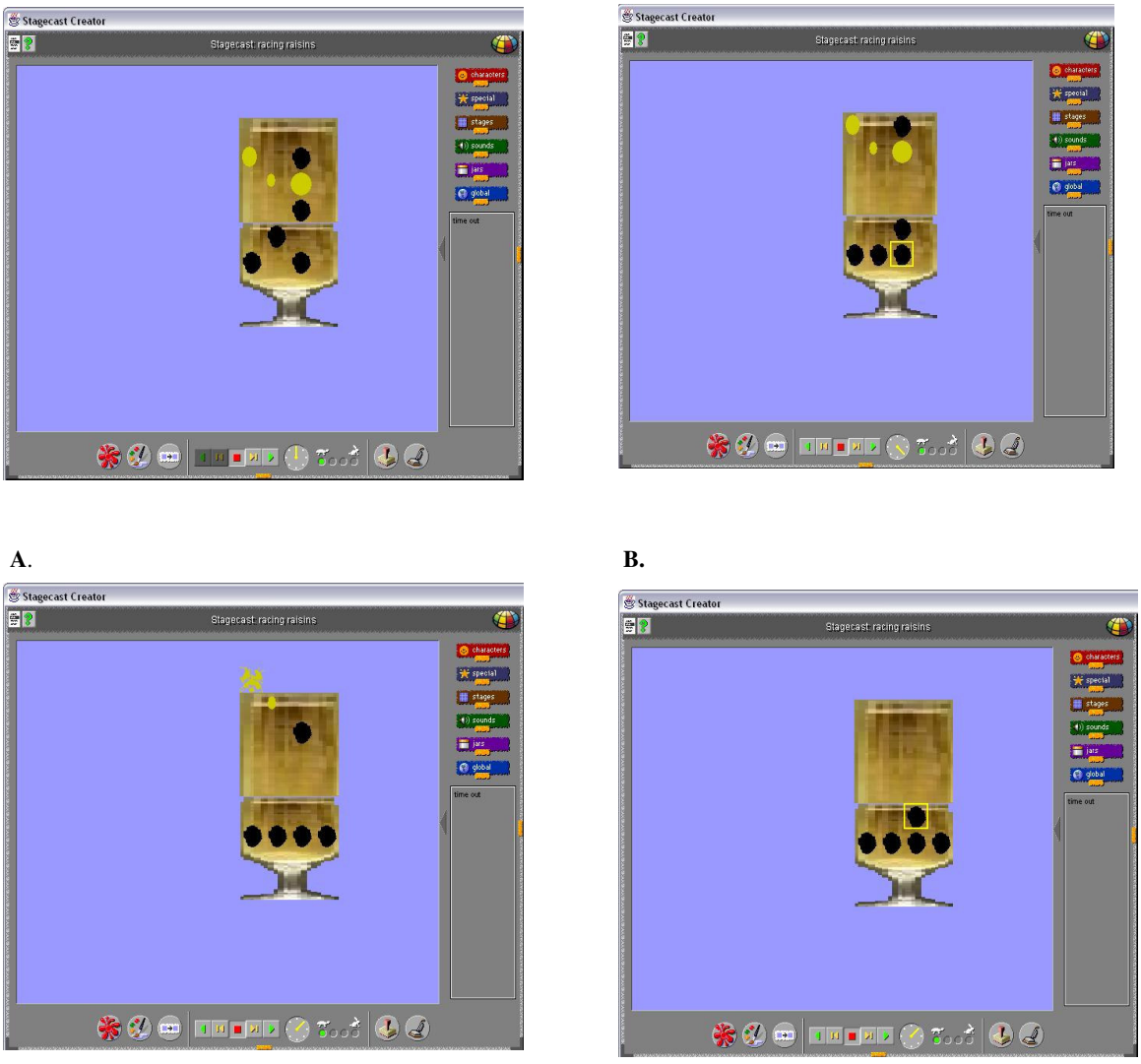


FIGURE 2 TEACHER CREATED CARBON DIOXIDE EFFECTS ON RAISINS IN GINGER ALE

We completed six sessions and gleaned a myriad of information from these sessions. We found that teachers were comfortable with the guided exploration style of our minimalist tutorial and were able to learn to create simulations with self-study of our tutorial materials. From these sessions, we also found that teachers could identify good candidates for simulation topics. This was also helpful in exploring motivational issues for our teachers. They were motivated enough to complete two simulations but were frustrated with having to creating their own objects and preferred to have a handy library of objects to reuse. The facilitators helped the users import some images from the web and import them into their simulation, but created all the rules for their working simulations. The is a great argument that teachers are motivated to perform activities that are useful for their classrooms, and with minimal training our teachers created working educational simulations that would be useful in supporting their science class topics.

Gathering these requirements of topics reaffirms that our teachers would find a software curricula aid very helpful in the classroom. We referenced these topics during our research and to complete our system. We plan to provide technical support of the prototype system, and the aim of this experiment was to ascertain whether the system will be robust enough for teachers to incorporate simulations into their science teaching, what if any impact it has on their students, and what impact they believe simulations and this environment will have on science education.

7. SYSTEM REQUIREMENTS

The end-user population studied in this sample was domain experts in the area of physical science, biological science, and physics. The age groups that teachers designed simulations for were middle and high school students. The goals of the requirements analysis phase were to narrow the scope of work and to begin preparing a structure for the design and development of the system. The system incorporated an environment for drawing, behavior creation and reuse of simulation microworlds and their components. System requirements are highlighted; giving a rational that substantiates requirements for general environment, robust support of reuse and others.

Table 1 Initial Requirements

I. General environment
• Providing a set of rich drawing tools which allow user to draw and modify objects
• Supports simple undo and incremental testing
• Clear recognizable common sense icons
• Easy creation/execution of graphical simulations.
• Satisfactory level of usability for novice programmers.
• Platform independent implementation (i.e. Java or Smalltalk)
II. Robust support of reuse
• A base set of generic template objects that can be reused
• Opening of multiple worlds to facilitate copy/past and reuse learning.
• Support user creation of object behaviors/interactions with rule templates or toolkit
• A library of reusable objects and simulation projects
III. Some secondary requirements
• Importing graphics
• Importing background graphics
• Support for multimedia and interactive simulations
• Ability to have macro recorder to record actions and create rules.

8. SIMULATION BUILDING IMPLICATIONS

Tools for Teachers

The aim was to have teachers able to create real simulations quickly; relying on their domain expertise and the new skills they have learned about simulation creation. The rationale for building simulations as educational material is practical. Kuyper states that simulations are independent of time and place, which makes them more readily available for real experience. He also states that simulations can provide a better conceptual model of a situation, and can be used to create virtual environments [Kuyper, 1998].

With this research, the following general problems with visual languages were identified: environmental, drawing tools, and with rule creation. A problem was also encountered with the level abstraction. Most of the environments studied were based on a grid-based concept and objects, when layered, did not operate as anticipated (e.g., rain on top of a flower in one case would evaluate only the object on top). Minimalist instruction was used by creating tutorial materials for analysis. The tutorials created were very helpful to users, got them started quickly, and aided them in completing the exercises with salient results. Ideas were also explored for the building of reuse libraries by providing categories for types of problems. Since this new environment was created for science teachers

and simulations were categorized based on content areas like, for example, Mathematics, English, Biology, Social Science, etc. The first category to be built into the library was Physical Science. Within each category, the plan is for simulations to be organized in alphabetical order. The identification of general problems with visual languages in this study helped to build our knowledge of visual languages.

9. REFERENCES

[1] J. M. Carroll, **The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill.** Cambridge, Massachusetts: The MIT Press, 1991.

[2] M. Kuyper, **Knowledge engineering for usability: Model-mediated interaction design of Authoring Instructional Simulation. University**. University of Amsterdam, Department of Psychology, 1998.

[3] C. Lewis, C., C. Brand, G. Cherry, & C. Rader, **Adapting user interface design methods to the design of educational activities.** Proceedings of Human Factors in Computing Systems. Los Angeles, CA, 1998).

[4] C. Perrone, and A. Repenning, **Graphical Rewrite Rule Analogies: Avoiding the Inherit or Copy & Paste Dilemma**. Proceedings of the IEEE Symposium of Visual Languages, pp. 40-46. Nova Scotia, Canada: Computer Society, 1998.

[5] M.B. Rosson, and J. M. Carroll, **Usability Engineering: Scenario: Scenario-Based Development of Human Computer Interaction.** Morgan Kaufmann Publishers. San Francisco CA: Academic Press, 2002.

[6] M.B. Rosson, and J. M. Carroll., C. D. Seals, and T. L. Lewis, **Usability Engineering: Scenario: Scenario-Based Development of Human Computer Interaction.** Morgan Kaufmann Publishers. San Francisco CA: Academic Press, 2002.

[7] L. Vygotsky, **Thought and Language.** Cambridge, Massachusetts: The MIT Press, 1986.