Setting the Scene: Scaffolding Stories to Benefit Middle School Students Learning to Program

Jordana Kerr, Mary Chou, Reilly Ellis, Caitlin Kelleher
Dept. of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO, USA
{jhkerr,mchou,rwellis,ckelleher}@wustl.edu

***Abstract*— Research suggests that storytelling can motivate middle school students to explore computer programming. However, difficulties finding and realizing story ideas can decrease time actually spent on programming. In this paper, we present guidelines for constructing story scenes that reliably inspire ideas for novice programmers creating stories. To evaluate the impact of pre-built scenes with strategic design constraints on early programming behavior and attitudes, we conducted a between-subjects study comparing participants who used pre-built scenes and participants who crafted their own scenes. The results suggest that story starter scenes enable novice users to explore programming in the environment sooner, allow users to add and modify significantly more novel programming constructs during the length of the study, and maintain motivation for learning to program via storytelling.**

Keywords—programming for children; storytelling support

I. Introduction

Storytelling is a compelling context to foster interest in programming among middle school students, especially middle school girls [1]. Several programming environments [1, 2, 3] enable children to build a basic skillset in programming via creating animated stories. However, struggling to find a story idea can delay programming in this context; some users who struggled to find a story idea often spent a lot of time on non-programming activities, such as scene layout [1].

We hypothesize that carefully designed starting scenes for stories (see Figure 1) will enable users to 1) identify compelling storylines, 2) spend more time programming, and 3) explore more programming constructs with the extra time.

We conducted two studies: one formative and one summative. Our formative study explored how to create effective pre-built story scenes. The results of this study suggest guidelines for reliably crafting pre-built scenes that motivate story ideas. The summative study compared the programs, programming behavior and attitudes of users who programmed stories using our pre-built scenes and those who created their own scenes. We found that users who created stories using pre-built scenes spent 54% more time programming stories, spent twice as much time revising their code, and added 29% more novel programming constructs than users who created their own scenes.

This material is based upon work supported by the National Science Foundation under Grant No. 1054587.

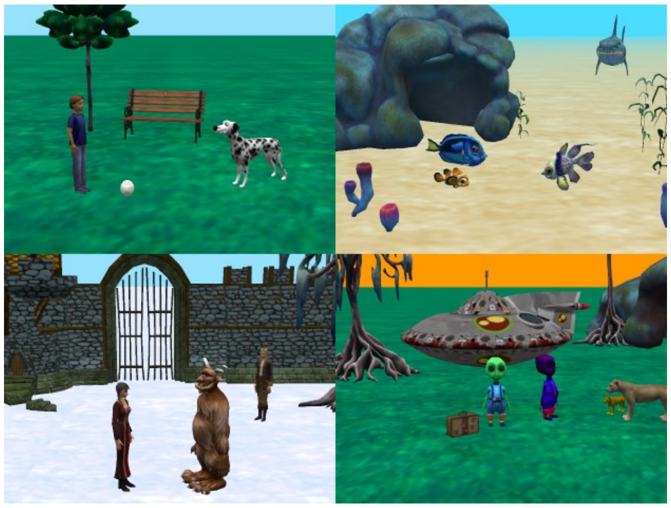
Figure 1. Examples of pre-built story starter scenes.

II. Related Work

Fundamentally, pre-built scenes are related to two areas of research: 1) tools intended to support storytelling and 2) story-based programming tools.

A. *Tools for Preparing and Creating Stories*

There are a variety of systems that attempt to support story creation. Some storytelling tools, for example, aid story construction by enforcing a structured process to help children piece together story components [4]. Other tools aim to support storytelling by providing inspiring or encouraging contexts, including immersive story environments [5, 6], interactive storytelling toys [7], and sampled playtime narratives [8]. Several storytelling tools include a set of story scenes to support storytelling [4, 5, 6, 8], but we are not aware of any research that explores how to construct these pre-built scenes to consistently spark ideas.

B. *Storytelling Tools for Introducing Programming*

Some applications [1, 2, 9] connect storytelling and programming while providing suggestions about story content. For example, in Storytelling Alice [1] a “go crazy” animation associated with a specific story character sparked stories explaining why the character went crazy. However, these starters and prompts are a fixed resource; over the long-term users would have to look to other sources for novel inspiration.

III. Background

We implemented pre-built scenes with story prompts in Looking Glass [3], a programming environment for middle school students that is designed to support storytelling.

A. *Looking Glass*

Looking Glass has two views: the Scene Editor (see Figure 2A) and the Action Editor (see Figure 2B). In the Scene Editor, users start with a blank scene in which they can add characters and props provided in the model gallery. The scene models include traditional story characters and props—such as fairies and ogres, castles and pirate ships—and everyday objects found at school and home. Users can adjust the scene by resizing and repositioning models. In the Action Editor, users can drag and drop programming blocks to animate stories. Users can organize blocks with basic control structures to animate the scene.

B. *Pre-built Scenes*

To support users in getting started programming stories in Looking Glass quickly, we added support for pre-built scenes. For the purposes of this paper, we define a pre-built scene as a 3D environment with scenery and characters supplemented with a textual story prompt. A pre-built scene might, for example, consist of a knight, princess and Ogre with a prompt like, "How will the knight rescue the princess from the ogre?" The story prompts spur possible storylines, but do not attempt to direct how the user should program the story.

When users open Looking Glass, the starting screen presents a list of pre-built scenes (see Figure 2C). Users can browse featured and popular scenes, or access bookmarked scenes from the Looking Glass community site [3].

IV. Formative Evaluation

We conducted a formative study to explore how to create pre-built scenes that effectively enable users to quickly identify compelling storylines.

A. *Participants*

We observed 30 participants (17 male, 13 female) between 10 and 18 (average age=11.6, sd=2.12) create stories using pre-built scenes in Looking Glass. We recruited our participants for this study from the visitors to the Saint Louis Science Center. Participants were compensated with a $5 museum gift certificate in acknowledgment of their participation.

B. *Materials*

Over the course of the formative study, we designed and tested 28 unique pre-built scenes. The scenes spanned a variety of themes including home and school life, fairytale settings, underwater exploration, and alien invasions. We included scenes and prompts with a range of complexities.

C. *Methods*

We conducted our formative evaluation over a series of single-participant sessions. Each session lasted for 20 minutes. Participants watched a three minute video tutorial to learn about the basics of selecting pre-built scenes and creating stories within Looking Glass. We then gave participants up to eight scenes to choose from, and asked them to create an animated story within the selected scene.

D. *Data*

For each session, we made notes on the participant's ability to form a story. We occasionally prompted participants with neutral questions such as "What's going to happen next?" Their answers helped us to gauge how quickly they formed solid plot ideas to program. At the end of each session, we asked participants what was hard, frustrating, or easy about implementing their story, why they chose the story they did, and whether or not they liked their story or would make a different one if they could. We also saved and analyzed participants' Looking Glass programs.

E. *Lessons Learned*

The following section presents the results of our observations in the form of guidelines for building scenes to effectively generate story ideas.

1) Use Position to Suggest Relationships

In the beginning of our formative study, many users felt compelled to move characters around in the scene, often bringing one or two forward as a focal point to center the scene around. This particularly happened in scenes with characters positioned side by side and not implying involvement with the scene. We found that placing the characters in strategic positions within the scene could suggest story roles. Grouping "clusters" of characters can imply "us" versus "them." Several successful scenes provided three to four characters in these arrangements (see Figure 3). Focal characters—highlighted by either being pulled forward or placed apart from a group—can suggest who the protagonist should be in the story.

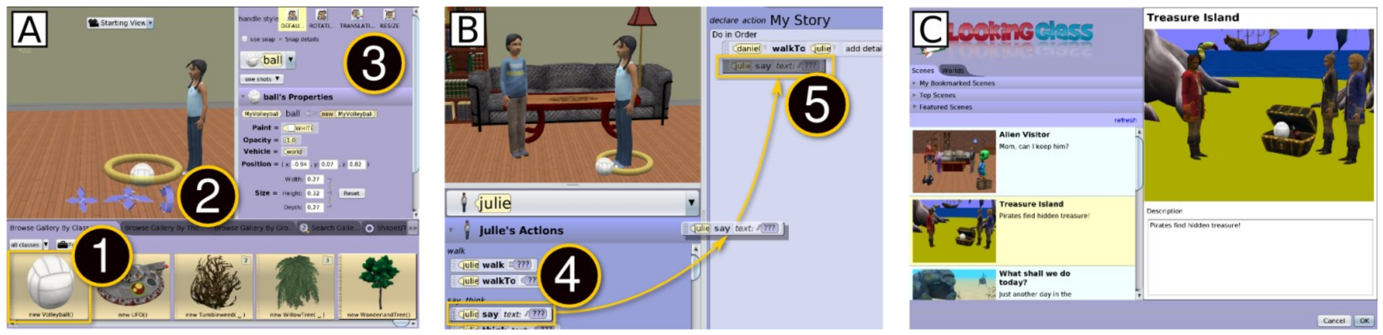

Figure 2. Views in Looking Glass. The Scene Editor in Figure 2A allows users to (1) select a prop or character, (2) add it to the scene, and (3) adjust parameters such as size, orientation and color. The Action Editor in Figure 2B allows users to (4) select a story action, and (5) drag it into the code pane. Figure 2C shows pre-built scenes previewed in the Looking Glass opening dialog.

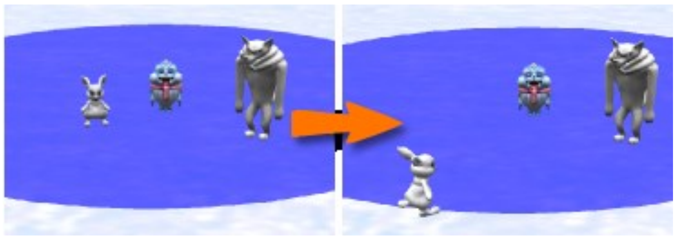
Figure 3. Example of clustered social groups.

2) Choose Inspiring Characters

The types of characters we chose to include also had an effect on stories. Themed characters such as aliens and monsters often provided participants with preconceived notions regarding their roles in stories. The wolf character, for example, was repeatedly labeled as the bad guy whenever he appeared in a scene, sparking stories centered on conflict.

3) Provide Relatable Themes

When asked about why they chose a particular pre-built scene over others, participants often pointed out something in the scene that they identified with, such as familiar themes like pets and school. Many participants gravitated towards the more socially themed stories, such as relationship stories, which has been observed in previous work [1]. We found that scenes suggesting scenarios of life at home, forming relationships and fitting in provide grounds for relatable stories.

4) Include Unexpected Singularities

Participants generally succeeded in coming up with story ideas in scenes that included something out of the ordinary, or eye-catching. For example, in a scene depicting a dog show, one participant eventually noticed that one of the dogs was, in fact, a lion, and exclaimed "That's a *lion*! But this is a dog show!" The participant then assembled a story justifying the lion's right to compete in dog shows. Our observations during formative testing suggest that unexpected elements in the scene, such as the alien in the school cafeteria (see Figure 4), can be used to help users focus their stories.

5) Provide Prompts as Secondary Support

We found that prompts can serve as additional story support by providing hints and ideas about what could happen in the scene, though their success varied. Some prompts were vague, such as "Lunchtime at school", but accompanying scenes were still successful when our previous guidelines were applied. Other prompts were detailed enough to provide specific plot ideas, which were sometimes followed and other times ignored or altered. In our study, prompts never negatively impacted storytelling and they provided valuable support for some participants.

V. Summative Evaluation

We applied the lessons learned during formative evaluation to a new set of scenes, and conducted a between-subjects study to compare the performance and attitudes of users given pre-built scenes with prompts (experimental) and users required to set up a custom scene (control).

A. Materials

We provided participants with video tutorials and versions of Looking Glass that varied by condition.

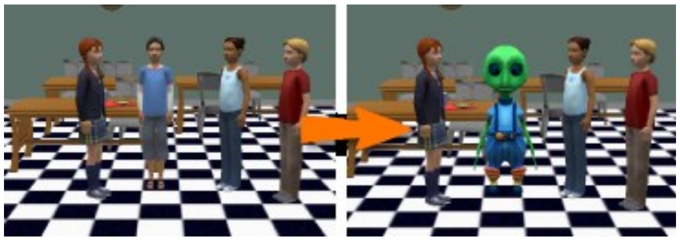
Figure 4. Example of an unexpected singularity.

1) Video Tutorials

Both the control and experimental versions of the tutorial video demonstrated how to program a story about kicking a ball. Additionally, the control tutorial demonstrated how to lay out the scene using the Scene Editor. The experimental tutorial demonstrated how to select and open a pre-built scene.

2) Looking Glass

Control participants used the standard version of Looking Glass. To create a program, control participants needed to create a scene. Participants in the experimental condition used a version of Looking Glass that included pre-built scenes and disabled access to the Scene Editor, but participants in the experimental group could still alter character and prop starting positions by dragging them in the preview pane. We provided eight scenes that followed our guidelines for participants in the experimental condition to choose from.

B. Participants

We recruited 46 participants through the Science Academy of Saint Louis. Of the 46, we excluded 9 participants due to either late arrival or a network failure. The remaining 37 participants (18 female, 19 male) were between ages 10 and 15 (average=12.1, sd=1.63). The study consisted of eight two-hour sessions with up to six participants in each session. Each participant received a $10 Amazon gift certificate.

C. Methods

We randomly assigned users to either the control or experimental Looking Glass environments. Participants first completed a demographic survey. Next, they watched their assigned video tutorial and spent up to 60 minutes creating one or more stories in their assigned version of Looking Glass. Afterwards, participants completed an attitude survey.

D. Data

We collected a brief demographic survey, participants' programs, Looking Glass logs, and an attitude survey.

1) Looking Glass Programs and Logs

We collected participants' story programs. Participants also wrote a short textual description for their stories. Looking Glass logged participants' actions for both the Scene Editor and the Action Editor throughout the session.

2) Attitude Survey

We examined user experience through the Intrinsic Motivation Inventory's (IMI) Task Evaluation Questionnaire (TEQ) [10]. The TEQ includes 22 Likert scale items divided into 4 subscales: interest/enjoyment, perceived competence, perceived choice and pressure/tension. The Cronbach's alpha values for interest/enjoyment (α=0.941) and perceived competence (α=0.847) were acceptable. The Cronbach's alpha for perceived choice (α=0.618) and pressure/tension (α=0.485) were below the standard accepted level of reliability (0.7), so we have chosen not to analyze the results from these subscales. 6 out of 37 participants did not complete the entire survey; we have chosen not to include uncompleted surveys in our results.

VI. Summative Results

We hypothesized that story scaffolds within the programming environment will enable users to spend less time preparing and more time programming stories. We also hypothesized that this extra programming time will prompt users to spend more time exploring novel code concepts. To provide insight on the impacts of story scaffolds in Looking Glass, we examine two kinds of data: 1) behavioral differences between participants during the study and 2) how participants rated their experiences and attitudes about their assigned version of Looking Glass.

A. *How Participants Spent Their Time*

Since pre-built scenes remove the need for scene creation, experimental participants spent more time programming. To provide insight into the programming both groups did, we explored what participants did during the study time.

1) *Story Preparation*

The control condition spent an average of 22 (sd=8.5) out of 60 minutes interacting in the Scene Editor. Half of this time (avg=10min52sec, sd=8min38sec) was during story creation. Users in the control condition idled on average 51% longer than users in the experimental condition (p<.001), noted when logs did not record any story edits in either the Action Editor or the Scene Editor for more than two standard deviations from a user's mean idle time. Control users spent on average 58% (sd=15.76) of that idle time in the Scene Editor, which could possibly be due to extensive browsing in the model gallery.

In comparison, participants in the experimental condition averaged less than 1 minute to select an initial scene. However, three users switched to a different scene after less than 5 minutes. Two of the three switched upon learning that they could not add new characters to the scene. This may indicate that these users had a story in mind, but could not implement it without additional scene items.

2) *Code Exploration*

Users in the experimental condition spent 54% more time programming, which aligns with our hypothesis. During this time, experimental users spent on average 14 minutes and 9 seconds modifying existing code, versus an average of 6 minutes and 22 seconds for control users, a significant difference (p<.001). Experimental users also explored an average of 12.7 (sd=5.47) novel code elements, as compared to 9 (sd=4.59) for control participants. We define a novel code element as a first time use of an action, construct or parameter input. This difference is significant (p<.05), suggesting that participants in the experimental condition tended to explore more elements of the programming interface than participants in the control condition.

B. *Attitudes*

We examined the two IMI subscales with acceptable reliability: interest/enjoyment and perceived competence. Although it was not significant, participants in the experimental condition tended to report higher levels of agreement with the interest/enjoyment scale (F[1,29]=3.643, p=0.085). There were no statistically significant differences between the control and experimental groups for the perceived competence scale (F[1,29]=2.89, p=0.156).

VII. Conclusion

We hypothesized that support in finding a story idea might lead to additional programming gains by focusing users' attention in the programming space, and not on story idea generation. Participants in the experimental group spent more time programming. This resulted in experimental participants exploring 29% more constructs and methods than control participants, and also spending more than twice as long modifying their program code. Particularly in cases where there is limited time to introduce programming to young people, the use of pre-built scenes is potentially valuable.

Acknowledgment

This work was funded by NSF Grant No. 1054587 and by the NSF Graduate Research Fellowship under Grant No. DGE-1143954. We would like to thank the Academy of Science in Saint Louis and the Saint Louis Science Center for their help with participation recruitment.

References

[1] C. Kelleher, R. Pausch, and S. Kiesler, "Storytelling Alice motivates middle school girls to learn computer programming," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2007, pp. 1455.

[2] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick, "Scratch: a sneak preview," in Proceedings of the Second International Conference on Creating, Connecting and Collaborating through Computing, 2004, pp. 104–109.

[3] "Looking Glass." [Online]. Available: https://lookingglass.wustl.edu/.

[4] A. Russell, "ToonTastic: a global storytelling network for kids, by kids," Conf. on Tangible, Embedded and Embodied Interaction, pp. 271, Jan. 2010.

[5] J. Robertson and J. Good, "Using a Collaborative Virtual Role-Play Environment to Foster Characterisation in Stories," Journal of Interactive Learning Research, vol. 14, no. 1, pp. 5–29, 2003.

[6] H. Alborzi et al., "Designing StoryRooms: interactive storytelling spaces for children," in Proceedings of Designing Interactive Systems, 2000, pp. 95–104.

[7] M. Umaschi, "Soft toys with computer hearts: building personal storytelling environments," in CHI '97 Extended Abstracts on Human Factors in Computing Systems, Atlanta, Georgia, 1997, pp. 20.

[8] J. Cassell and K. Ryokai, "Making Space for Voice: Technologies to Support Children's Fantasy and Storytelling," Personal and Ubiquitous Computing, vol. 5, no. 3, pp. 169–190, Aug. 2001.

[9] L. Sherman, A. Druin, J. Montemayor, A. Farber, M. Platner, S. Simms, J. Porteous, H. Alborzi, J. Best, J. Hammer, A. Kruskal, J. Matthews, E. Rhodes, C. Cosans, and A. Lal, "StoryKit: Tools for children to build room-sized interactive experiences," in CHI '01 Extended Abstracts on Human Factors in Computing Systems, Seattle, Washington, 2001, pp. 197–198.

[10] "Intrinsic Motivation Inventory." [Online]. Available: http://www.selfdeterminationtheory.org/questionnaires/10-questionnaires/50.