Sugar Goes Spelunking: An Engine to Build Your Own Text Adventure

By

Aviva Hope Rutkin

****

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Neuroscience

UNION COLLEGE

June 2012

ABSTRACT

RUTKIN, AVIVA HOPE   Sugar Goes Exploring: An Engine to Build Your Own Text Adventure. Department of Neuroscience, June 2012.

ADVISOR: Kristina Striegnitz

Computer games can be effective educational tools. Studies show that classroom computers increase students' motivation to learn, promote collaboration among peers, and have the potential to improve classroom performance. Text adventures, a form of technological storytelling, particularly align with many of the qualities of successful learning games. To that end, I developed an activity for Sugar, One Laptop Per Child's free and open-source desktop environment, in which children can create and play their own text adventures. My activity, *Spelunk*, will launch on Sugar in the spring of this year.

"When I played *Zork*, I felt like I was in that world, and the computer screen disappeared, and it was like I was reading a book, only even cooler than reading a book."

Paul O'Brian, interactive fiction author

1 Introduction

1.1 Classroom computers

The idea of using computer games as educational tools is by no means new. The combination goes as far back as 1971's *Oregon Trail*, and has been studied intensely by computer scientists, educators, psychologists, and media scholars alike. While some researchers were initially wary of computers' potential for distraction, many have discovered a variety of benefits to technology into the classroom.

First, computers increase students' motivation to learn. They have been linked to higher levels of student engagement, better classroom discipline, higher attendance records, and increased self-reported interest in learning. There are positive effects for teachers as well; teachers who have computers in their classroom have an easier time planning and delivering lessons, and they make an active effort to include computers in instruction more often. [1][2] Computers have even been found to improve the attitudes of parents to their children's classroom. [3]

Second, computers promote collaboration between students. Some educational psychology theories, such as cognitivism and social development theory, argue that this is crucial for a good learning environment. Numerous studies show that computers do in fact lead to increased teamwork in the classroom, often indirectly encouraging students to work together in search of a solution. [3][4] Games are particularly effective in this regard. As psychologist Dr. Janette Hill elegantly states,

> Research into psychological and sociological benefits of play also revealed that games support intrinsic motivation as well as opportunities for imitation and learning by providing feedback, fantasy, and challenges. By creating virtual worlds, games integrate knowing and doing. Games bring together ways of knowing, ways of doing,

ways of being, and ways of caring: the situated understanding, effective social practices, powerful identities, and shared values that make someone an expert.

When students work together, they help each other learn new skills and concepts more quickly. By doing so, they also take the pressure off of a potentially overwhelmed teacher in a crowded classroom. [5]

Studies are divided over whether classroom computers have a definitively positive or merely a neutral impact in the classroom. However, the benefits that some research has unearthed are not insignificant; classroom computers have been linked to increased nonverbal IQ scores, enhanced reading skills, and improved memory performance. One study of thousands of American fourth graders and eighth graders concluded that computer use was positively correlated with academic achievement in mathematics and the social environment of the school. [1] Another assessment, a large-scale randomized evaluation of a laptop program in rural Peru, suggested that "the estimated impact [of laptop presence] on the verbal fluency measure represents the progression expected in six months for a child." [6]

While a few continue to question the strength of these connections, there is little research suggesting that the machines have an opposite, negative impact. The more likely explanation for the disparate conclusions is that there are effective and ineffective ways to integrate computers into the classroom. With respect to games, those that are simplistic, repetitive, or patronizing are usually found to be educationally unproductive. [7] In an influential paper published in 1980, video game design scholar Thomas Malone identified a number of key characteristics for success with computer games, including:

- An easily understood goal
- Uncertain outcome due to variable difficulty, multiple goals, hidden information, or randomness

- Fantasy fulfillment
- Concrete feedback to the player
- A broad range of challenges [8]

Meeting all of these requirements can, of course, be a formidable challenge, and they only go so far as to tell us how to make games engaging, not educational. Game designers must find a way to incorporate all of these characteristics and still more if they wish to find success with a particular endeavor.

1.2   Text adventures

A text adventure is a computer game in which the player explores a virtual world through the written word.

> *Forest Path*
> This is a path winding through a dimly lit forest. The path heads north-south here. One particularly large tree with some low branches stands at the edge of the path.
>
> > `go north`
>
> *Clearing*
> You are in a clearing, with a forest surrounding you on all sides. A path leads south. On the ground is a pile of leaves.
>
> > `kick leaves`
>
> Kicking the pile of leaves isn't notably helpful.
>
> > `move leaves`
>
> In disturbing the pile of leaves, a grating is revealed.
>
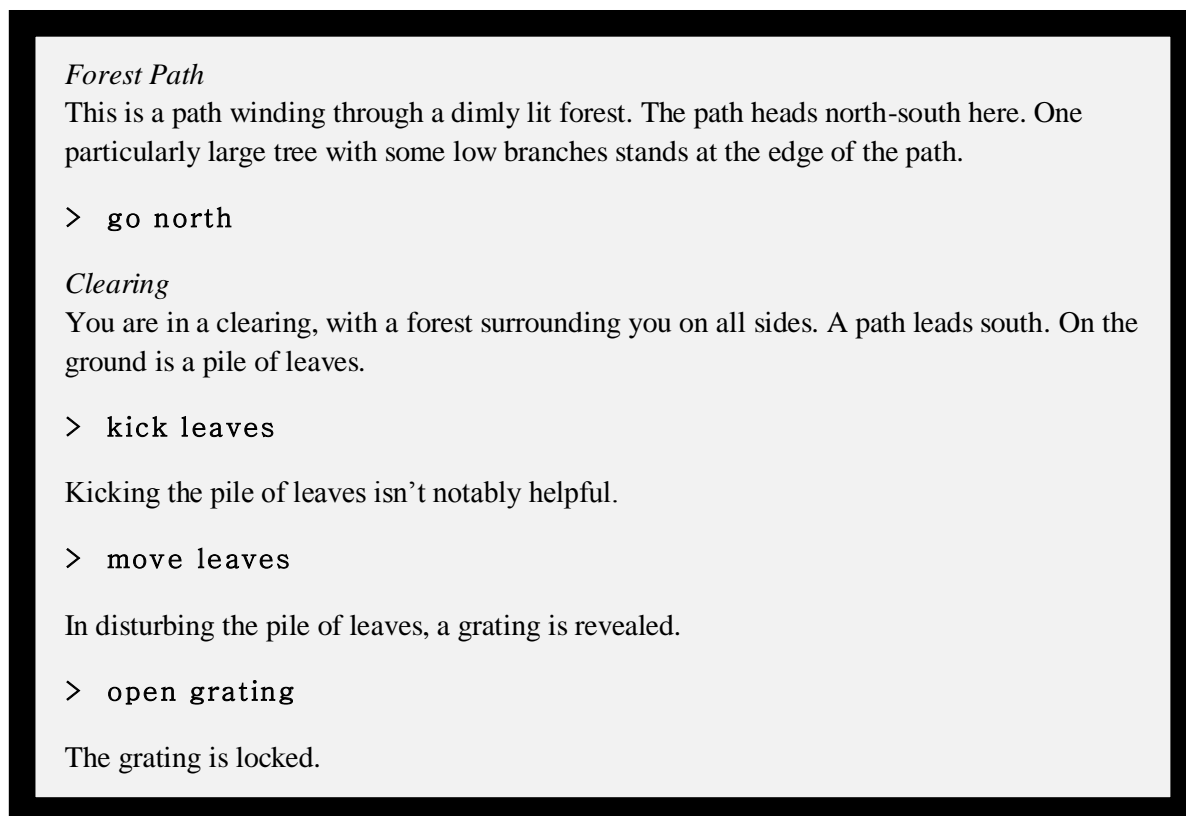> > `open grating`
>
> The grating is locked.

Figure 1. A sample exchange from *ZORK I: The Great Underground Empire*, released in 1981. [A]

Text adventures are a type of storytelling called interactive fiction; they require written action on the part of the player in order to move forward in the story. "The tensions at work in a cybertext, while not incompatible with those of narrative desire, are also something more, a struggle not merely for interpretive insight but also for narrative control: 'I want this text to tell *my* story; the story that *could not be* without me,'" explains video game scholar Espen J. Aarseth. [9] The player exercises autonomy within the confines of the text adventure's novelistic script by typing simple sentences such as 'go east,' 'take lantern,' and 'kill thief.' He or she must also solve a series of small logic puzzles scattered throughout to reach the end of the story and win the game.

The first text adventure was, in fact, named *Adventure*. It was written in 1975 by a programmer named William Crowther as a detailed simulation of the Flint Mammoth Cave System in Edmonson, Kentucky. The game remained remarkably faithful to real-world geography, and included fantastical treasures as incentives for the player to continue exploring the caves.

Although the game was originally intended only for his two young daughters, Crowther uploaded *Adventure* to the Internet precursor ARPANET, where it eventually piqued the curiosity of graduate student Don Woods. "I realized that it was unlike anything I had encountered up to that point," Woods said. [10] He wrote to Crowther asking for permission to improve the original code; upon receiving it, Woods debugged the program, added a scoring system, and stocked the cave with additional magical elements. The resulting work, renamed *Colossal Cave*, gained more widespread popularity through ARPANET and the user group DECUS, and it is widely credited for kick starting a decade of impassioned text adventuring.

Pursuant text adventures run the gamut of styles, including labyrinthine games like the famous *Zork* series; whodunit mysteries such as *Deadline*, in which the player must apprehend a murderer; digitized walkthroughs of famous novels, both fanatically canonical and unexpectedly derivative; sexualized storylines like *Leather Goddesses of Phobos*, which can be played on several different 'naughtiness' levels; and unconventional moralistic tales, most notably *A Mind Forever Voyaging*.[1] Games were occasionally sold with 'feelies'—fake documents, evidence, or merchandise—which added an element of realism to the gaming experience. [11]

The majority of text adventures were produced by Infocom, a software company founded by a group of MIT students in 1979. Infocom patterned their games off of the original *Adventure* epic, with an added emphasis on the need for problem solving. "Although our games are interactive fiction," read one edition of the company newsletter, *The New Zork Times*,

> they're more than just stories: they are also a series of puzzles. It is these puzzles that transform our text from an hour's worth of reading to many, many hours' worth of thinking. It is these puzzles that cause a player to suddenly leap out of bed in the middle of the night and run to his computer because he just thought of a possible solution to a problem. The value of our games is that they will provide many hours of stimulating mental exercise. [12]

Many puzzles require the player to experiment with language. For those who are playing without any guidance, this task can prove quite difficult. For example, *Adventureland* required players at one point to enter the unusual phrase 'unlight

---

[1] *A Mind Forever Voyaging* is set in the future in a disintegrating United States at risk of nuclear war. The game, which was not commercially successful, has been analyzed as a critique of right-wing politics, particularly Reaganomics.

lamp.' Another puzzle in *A Hitchhiker's Guide to the Galaxy* was so challenging that it inspired its own line of T-shirts.[2] [13]

Some games were also predicated on the complex spatial challenge of navigating one's way through a maze. You might leave one room going north, but find yourself in a different room when you backtrack south. Lethal pitfalls such as bottomless holes lurked in the corners of early versions of *Zork.*[3] *Colossal Cave* contained a set of underground rooms that appeared to be completely identical, making navigation especially confusing.[4] It was essentially impossible to solve these games without drawing a very detailed map. This problem could also be circumvented by leaving a trail of different inventory items behind you as you went.

While the overarching goal of any given text adventure is always the same, each individual session of the game is different depending upon how the player chooses to act. Even after a game has been solved, there are often still new things to discover. "Once we have determined how a game is supposed to be read," writes game design professor Julian Kücklich in *Literary Theory and Digital Games*, "we can start thinking about whether there are other ways of reading it, and why such an alternative reading might or might not be more meaningful." [14] For many, pushing against the boundaries of the game language was the real source of joy in
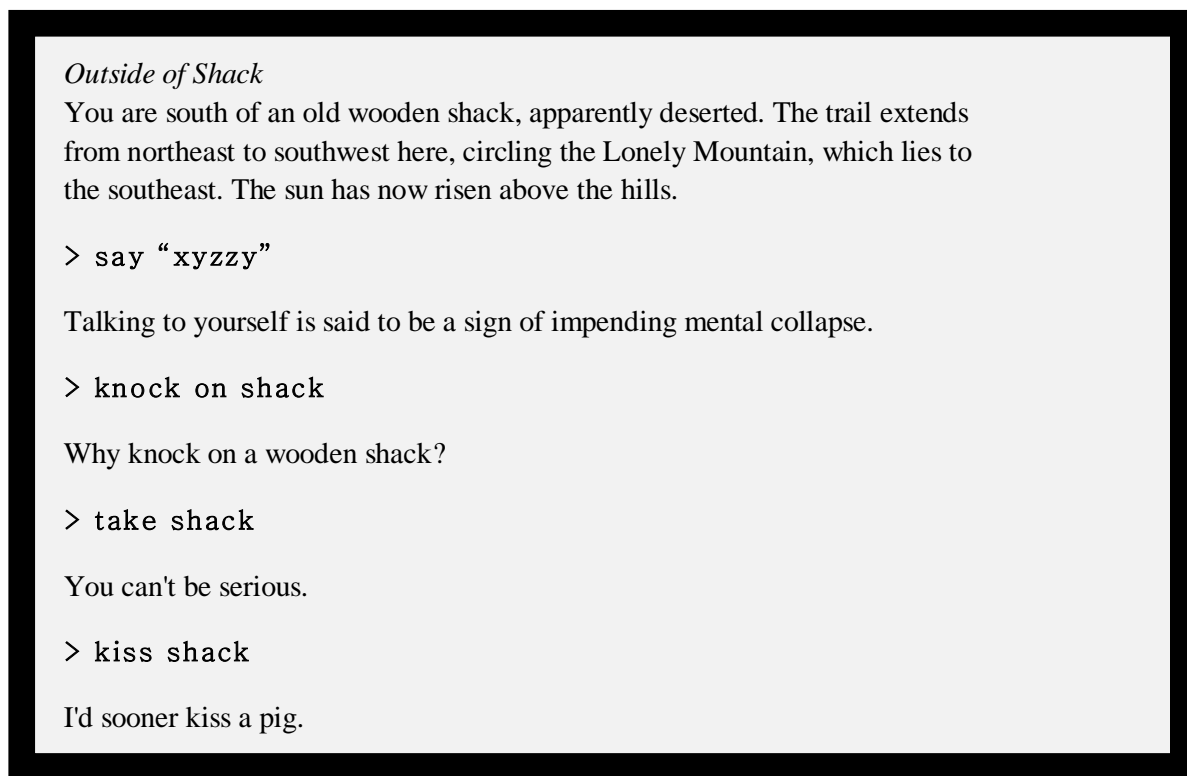
---

[2] In short, the player needed to retrieve a linguistically talented creature called a Babel Fish from a dispensing machine by doing four things: hanging a dressing gown on a hook; placing a towel on a grate on the floor; and putting a satchel and then a book in front of a panel. Due to its difficulty, this task was nearly removed from the game altogether after initial testing. However, *Hitchhiker's Guide* author Douglas Adams insisted that it be left in.

[3] These holes would disappear when the player turned on his lantern. However, many players complained about the physical impossibility of there being a bottomless hole in the attic of a house, so later versions of *Zork* replaced the holes with menacing creatures called grues.

[4] *You are in a maze of twisty little passages, all different* might be followed by *You are in a twisting maze of little passages, all different* and then *You are in a twisty maze of little passages, all different*. See?

text adventures. For example, take the following transcript of one session of *Enchanter*:

> *Outside of Shack*
> You are south of an old wooden shack, apparently deserted. The trail extends from northeast to southwest here, circling the Lonely Mountain, which lies to the southeast. The sun has now risen above the hills.
>
> `> say "xyzzy"`
>
> Talking to yourself is said to be a sign of impending mental collapse.
>
> `> knock on shack`
>
> Why knock on a wooden shack?
>
> `> take shack`
>
> You can't be serious.
>
> `> kiss shack`
>
> I'd sooner kiss a pig.

Figure 2. A sample exchange from *Enchanter*, released in 1983. [A]

This kind of experimentation often uncovered Easter eggs, surprising storylines, and snarky asides, all of which were extra thrills for the player. It also gave programmers the added challenged of trying to anticipate the different ways that players would use language in their game. [15][16]

1.3   Motivation

My goal was to develop an engine in which students could create and play their own text adventures. Although text adventures have not traditionally been used in an educational setting, I theorized that they could make the transition.

They are, after all, a reimagining of the familiar short story format. Additionally, text adventures possess many of the aforementioned good game qualities, including an understandable structure, immediate player feedback, a wide variety of difficulty levels, and the possible fulfillment of personal fantasies. It follows that text adventures are a potentially valuable way for students to practice their reading and critical thinking skills.

A handful of engines, such as ADVENT and Quest, already exist online for free. There is even a programming language called Inform that allows you to write and interpret your own interactive fiction stories. However, these options are generally complex and unintuitive; even a college student would need to take their time acclimating to these programs. Such patience cannot be expected from younger students. Instead, my engine needed to be extremely easy for new users to understand.

I decided to program my engine in Python so I could later convert it into a Sugar activity. Sugar is a free, open-source desktop environment that is specifically designed to be kid-friendly and encourage collaboration. It is the default interface for One Laptop Per Child, a nonprofit dedicated to the dissemination of computers in the developing world.[5] Applications for Sugar (called 'activities') can be downloaded online, and they range anywhere from study tools to word processing applications to chat clients to games, both educational and otherwise. By designing my engine with Sugar in mind, I had the potential to reach an audience beyond Union's gates.

---

[5] OLPC uses the XO laptop, which has been designed for their particular use. Highly inexpensive and power-efficient, these laptops can be found in over 40 countries around the world, including Peru, Uruguay, Rwanda, Ghana, Brazil, Afghanistan, Mongolia, and the United States. [17]

2   Engine

My activity is named *Spelunk* in homage to Crowther's original text adventure. *Spelunk* was programmed twice: first, as a game that could be played entirely within the Python editor, and then again with Sugar's chosen GUI library, PyGTK.[6] I did this in order to work out particular programming obstacles and language difficulties before preparing for Sugar.

Designing *Spelunk* primarily involved the satisfaction of two opposing desires. I wanted to provide users with the freedom to create games that were unique. They should be able to use language in ways that I could not anticipate, and to create a puzzle on the same order of magnitude as, say, Adams's Babel fish

Figure 3. The activity icon for Sugar.

dispenser. However, I also needed to balance that functionality with an interface that children could easily understand. To that end, certain structures are already in place for the user when he or she starts the engine, such as a default grid-like **map** of 25 empty locations, also known as **rooms**. (See Figure 4.)
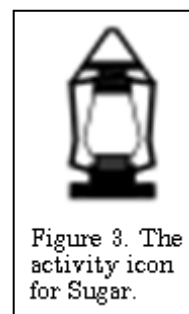
To create a room, one simply needs to click on one of these empty spots and give it a name and a description. The first room created is the default location that the player starts in at the beginning of the game, as indicated by an asterisk. (This can be changed later after multiple rooms have been created.) Rooms that are adjacent to one another are automatically connected, allowing the player to travel from one location to the next. These connections, indicated by the = and —— buttons, can then be broken by clicking them and adding a **barrier** like a *locked door.*

---

[6] I used two online manuals, *Make Your Own Sugar Activities* [18] and *PyGTK 2.0 Tutorial* [19], to develop the graphical user interface and adjust the code to Sugar's specifications.
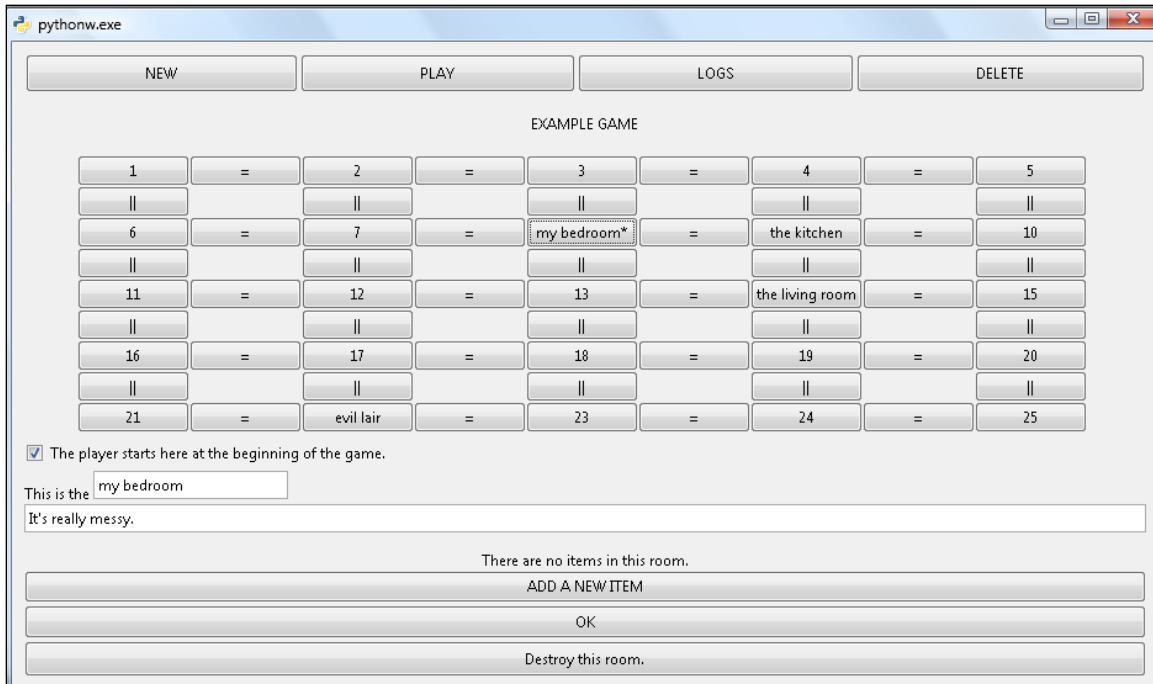
Figure 4. A screenshot of *Spelunk*'s room editor. The space below the map changes according to what the user is trying to do; the map then updates accordingly. The top buttons remain static.
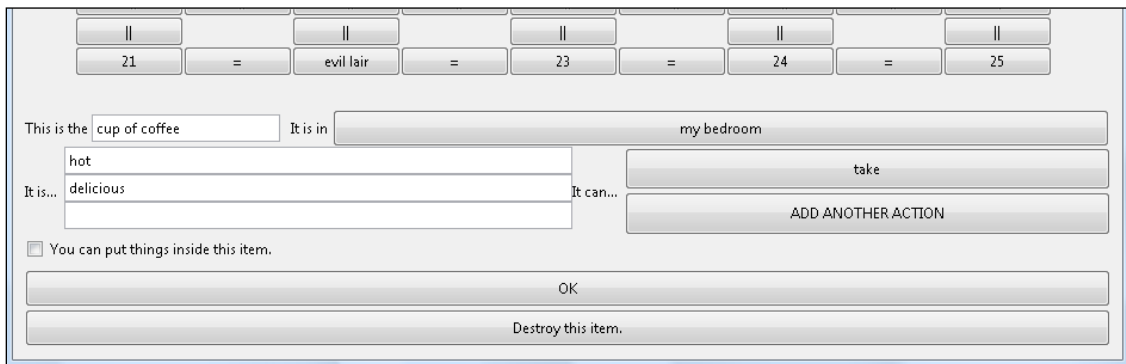

Figure 5. A screenshot of *Spelunk*'s item editor.

Users can also add **items** to a room. The initial item creator requires only a name, a list of descriptive adjectives, and an indication of whether players can 'take' the item and carry it around with them. (A player, for example, could conceivably take a *train ticket*, but not the *train* itself.)

The item can then be modified in greater detail. One option is to place other items inside of it, and then place constraints on when those inner items can be accessed. For example, a *chest* might be filled with *treasure*, but the player can only take that treasure once the chest is open. Items can also be given additional **actions**. Actions define what the item can do, how the player can interact with the item, and how items might interact with one another. At their most basic level, actions only require a name (e.g., *write* or *break*) to function. However, they can become much more complex by adding a limit, a precondition that must be met, or an after-effect. For example, if your game included the items *knife* and *bread*, you might want players to be able to pick up a loaf of bread, slice it, and then eat it. A *knife* is able to slice an infinite number of times provided there is something around that can be sliced. The *bread* can be sliced once, provided there is something around that can slice it; once that happens, its name changes to *sliced bread* and a new action called *eat* is added.

Some common text adventure items and actions are available in the engine in a premade inventory.[7] These serve as good examples of the different options available to the user, and are a stress-free way for novices to make their game more elaborate. Additionally, a small sample text adventure is included with the activity.

Games are saved periodically during the editing process. Users can stop at any point to play their game in a new window. A **log** of each of these sessions is saved to a folder, giving users the opportunity to adjust their game according to whatever difficulties the players are having. Additionally, users might mine new ideas for their game from the different commands that players try to use.

---

[7] The item inventory includes an axe, backpack, bottle, loaf of bread, chest, flashlight, hammer, hat, key, lamp, money, paper, pen, screwdriver, sword, table, treasure, water, and wrench. These items each have at least one action already built in. The premade actions include cut, drink, eat, hit, read, spend, take, turn on, turn off, wear, and write.

3   Future

There are, of course, many ways that I or another programmer could improve upon the existing activity.

The preprogrammed map is fairly restrictive, especially for users who want to create more complicated geographical relationships. During the development of *Spelunk*, I worried that more freedom in this area would make the engine incomprehensible to young users. Additionally, more options would result in more opportunities for programming errors on my part. These concerns ultimately led me to the confined grid structure. However, there is the potential for an 'advanced' engine setting; for example, one could permit users to preselect the size of the map grid, or make connections to move up, down, northeast, southwest, etc. Furthermore, the grid could also be scrapped entirely in favor of a more free-form map editor with geographically unnatural relationships between rooms.

Many other, more complex pieces could be incorporated into the activity. Other engines offer non-player characters for the player to interact with, or randomized events such as the appearance of a dangerous troll. Additionally, many games award points for the successful completion of a given action, thus allowing players to win the game once they have accrued enough points. Again, *Spelunk* was designed in favor of simplicity of the part of both the user and the programmer. I would imagine that these functions could be built into the existing structure, and recorded in the map in the same manner that items and actions are recorded in rooms.

The activity can also be altered for an international audience. Sugar is used in dozens of languages around the world, and though the engine is currently in English, I strove to make the code adaptable this regard. If someone wanted to

translate *Spelunk* into their native language, it should require a fairly limited amount of work. (As far as I know, the different features of the game do not rely upon the particular idiosyncrasies of the English language.)

Spelunk will be uploaded to Sugar at the end of the year, and will be available for download online at activities.sugarlabs.org. Ideally, the engine should be tested with real children for feedback, allowing us to pinpoint which features were most and least successful. I originally intended to pilot *Spelunk* with local students at the Union College Kenney Center, but unfortunately the engine was not completed in time. However, users can leave comments on the activity online, and the door always remains open for future improvements.

References

[1] Jurich, Sonia. "Computers in the Classroom: How Effective???"*TechKnowLogia* (2009) 31:34. Accessed online at http://tinyurl.com/techknowlogia.

[2] Kulik, J.A. "Meta-Analytic Studies of Findings on Computer-based Instruction." *Technology Assessment in Education and Training*. Ed. E.L. Baker and H.F. O'Neil, Jr. Hillsdale, NJ: Lawrence Erlbaum, 1994.

[3] Grimes D. and Mark Warschauer. "Learning with laptops: a multi-method case study." J. Educational Computing Research, Vol. 38(3) 305-332. Baywood Publishing Co. Inc., 2008.

[4] Gee, J. *Situated Language and Learning*. New York, NY: Routledge, 2004.

[5] Hill, Janette. *Impacts of Playing Video Games on Learning in Children*. Rep. University of Georgia, 29 Mar. 2006. Accessed online at http://tinyurl.com/janettehill.

[6] Crisita, J., Pablo Ibarrarán, Santiago Cueto, and Ana Santiago. *Technology and Child Development: Evidence from the One Laptop per Child Program*. February 2012. Accessed online at http://tinyurl.com/olpcreport.

[7] Kirriemuir, J. and Angela McFarlane. *Literature Review in Games and Learning*. Rep. futurelab. 2004. Web. Accessed online at ¡http://tinyurl.com/gameslearning¿.

[8] Malone, T.W. Toward a theory of intrinsically motivating instruction, *Cognitive Science*, 1981, *4*, 333-370.

[9] Aarseth, Espen J. *Cybertext: Perspectives on Ergodic Literature*. USA: The John Hopkins University Press, 1997.

[10] *GET LAMP: A Documentary About Adventures in Text*. Dir. Jason Scott. Accessed online at http://tinyurl.com/getlampvideo.

[11] Montfort, Nick. *Twisty Little Passages: An Approach to Interactive Fiction*. USA: The MIT Press, 2005.

[12] Infocom, Inc. "Frank Answers to the Ten Most Frequently Asked Questions." *The New Zork Times*. 3(2)(Spring 1984): 1. Accessed online at http://tinyurl.com/newzorkfrank.

[13] Adams, Douglas and Steve Meretzky. Interview by M.J. Simpson. British Broadcasting Company. Accessed online at http://tinyurl.com/bbcadmer.

[14] Kücklich, Julian. "Literary Theory and Digital Games." *Understanding Digital Games*. Ed. Joe Bryce and Jason Rutter. London: SAGE Publications, 2006.

[15] Gutman, Dan. "Shoot Your Own Men! And Other Weird Ways To Play." *Computer Games* December/January 1984. Accessed online at http://tinyurl.com/gamesmuseum.

[16] Kidder, Tracy. *The Soul of a New Machine*. USA: Back Bay Books, 1981.

[17] "One Laptop per Child interactive map." *One Laptop per Child*, 2012. Accessed online at http://one.laptop.org.

[18] Cameron, J. and Simmons, J. *Make Your Own Sugar Activities*. Accessed online at http://en.flossmanuals.net.

[19] Finlay, J. *PyGTK 2.0 Tutorial*. April 13, 2005. Accessed online at www.pygtk.org/pygtk2tutorial/index.html.

[A] Pot, Martin. "Play Infocom Adventures Online." *Play Infocom Adventures Online*. Accessed online at http://pot.home.xs4all.nl/infocom/.

Acknowledgements

This project was made possible by many people, particularly Union College's inimitable staff and faculty. The neuroscience department allowed me the freedom to pursue an unusual senior project; the computer science department welcomed me with open arms. Kristina Striegnitz, my terrific thesis advisor, gets the lion's share of my gratitude. The bookstore employees who resurrected my laptop after I spilled tea on it get most of the rest.

My friends were also a great help throughout the last thirty weeks. Mike Corti was an enthusiastic debugger. Ajay Major lent me the font files and a patient ear. Aleena Paul, Sara Block, and Rachel Baker provided boundless spiritual support. They are all saints for putting up with me every day.

Last but not least, I would like to thank my parents, Dr. Maura Kates Rutkin and Mr. Alan S. Rutkin '80. In a way, this is all their fault.