CREU 2016-2017 Final Report: Analyzing the Potential of Learning Reading and Math Skills Through Computational Thinking

Amber Wagner, Birmingham-Southern College (Formerly Kennesaw State University)
Melissa Driver, Kennesaw State University
Deja Jackson, Kennesaw State University
Erica Pantoja, Kennesaw State University
Cindi Simmons, Kennesaw State University
Kate Zelaya, Kennesaw State University

I) Goals and Purpose

Hopper's Fables is intended to help first grade students (ages 6-7) reinforce their math and reading skills while simultaneously developing their computational thinking skills. In particular, we hope to assist students with learning disabilities, but what we have learned is that if the tool is beneficial to those with learning disabilities, it will be beneficial to all students.

Our research goals address three main questions:

1. **Question:** What are the existing block programming languages, and how do they meet or not meet the pedagogical needs of students with learning disabilities?
2. **Question:** Can a programming language be created that can address the learning of reading and math in addition to computational thinking?
3. **Question:** Does the resulting programming language meet the goals set forth in the project description (i.e., Does it help students with reading and math skills? Does it change the students' attitudes towards or perceptions of CS?).

According to our investigation, there is no doubt that existing block programming languages are playing an important role in education by enhancing the learning process of students. However, most strong programming languages such as Scratch, Frozen, and App Inventor are mainly developed and used to help students gain programming skills. Because of the limitations of these languages in education, we are excited to develop a language that will address not only computational thinking but also math and reading skills. And, we are proud of our most important goal, which is to design specifically to meet the math and literacy needs of students with learning disabilities.

II) Related Work

Although there exists a movement to increase the accessibility of computer science (CS) classrooms for all students (Burgstahler, 2011), there are few resources to teach young students with learning disabilities about computational thinking and even fewer evaluations of those resources (Falcão, 2010). Teachers specializing in learning disabilities often have to adapt existing CS curriculum to meet their students' needs. Although AccessComputing (2016), DO-IT (2016), and AccessCS10K (2016) are making great strides towards increasing the accessibility of the CS classroom, there is still much work to be done. The focus of this project is to utilize an open source block programming language API (e.g., Blockly, Scratch) to

create a block programming language that would help elementary-aged students learn or reinforce reading and math skills while learning how to think computationally. Pasternak (2016) found that CS is a viable class for all students when conducting research with grades 6-10, and Apone et al. (2015) unites several universities with Code.org to assist in bringing CS to K-5. Therefore, we believe targeting elementary-aged students for this project is plausible. However, Pasternak (2016) also found that it was difficult to teach programming within a specific context. While the student researchers will be creating a programming language, the focus when working with the target audience is not to teach them how to program but to teach them how to think like a computer scientist (Wing, 2006). We also hope that the work conducted within this project will expand upon that of Pasternak (2016) indicating that it is possible to teach computational thinking through other contexts.

Scratch Jr is the inspiration for this project, but Scratch Jr is primarily used for students ages 5-7 (Scratch Jr, 2016). Similar to many existing block programming languages, we aim to build a language with a "low floor" and "high ceiling" (Papert, 1980). While we hope students ages 5-7 will use the resulting programming language, we intend to have a "higher ceiling" than Scratch Jr; thereby, allowing the language to be more appealing to older students who may need reading or math help. The language should contain simple, straightforward blocks related to reading and math skill building. Additionally, the language's environment should contain a graphical interface for output that allows the users to see the results of their programming in an exciting fashion similar to other block programming languages (e.g., Scratch, Scratch Jr, Alice).

There exists a variety of block programming languages each having their own particular focus. A few languages have resources that assist young student learners with learning disabilities (Falcao, 2010). Hopper's Fables is designed to have a focus specifically for students with learning disabilities, which does not require teachers to adapt their teaching environment specifically for these students. This language is similar to others in the idea that it does not focus on teaching students how to program, but rather think computationally. Hopper's Fables differs in that not only is it focusing on computational thinking, but also increasing the understanding of some Common Core requirements such as reading and math for the 6-7 year old age group (first grade).

Scratch Jr, Scratch, and Minecraft Hour of Code are just a few languages with similar computational focuses as Hopper's Fables. We evaluated these languages with our grading rubric along with MIT App Inventor, Alice, Snap!, Code.org's Frozen, and Code.org's Moana to determine how they stand in terms of what our research deemed a valuable and successful block language for students. Our rubric highlighted some grading elements such as "Creativity", "Extrinsic Rewards", "Clear Goal and Purpose" and "Visuality of System Status".

The rubric was based on several attributes research indicated were important. These attributes and associated articles are described below.

Learning by mistakes
The language should always keep users informed about what is going on, through appropriate feedback within reasonable time (Masip et al., 2011), but feedback is not that efficient if the language does not allow users to correct mistakes easily. In some languages, such as Frozen and Scratch, visual feedback is available all the time and the recover option "go back" is represented visually and symbolically. However, other programming languages do not have this feature; they provide feedback using code language, which is irrelevant information for students when they are not familiar with code language. Error messages should be carefully designed and prevent a problem from occurring. Either eliminate error-prone conditions or check for them and present users with a confirmation option before users commit to the action. Code errors make it difficult for students to recover from mistakes (Masip et al., 2011).

Feedback and designer/ learner models
Feedback is an extremely important feature in block programming languages as this allows students to explore and understand the goal of the program and the scripts. For instance, in Scratch, visual feedback is immediately provided to help the user to understand when scripts are triggered and how long they run. If a script encounters an error (e.g., dividing by zero), the border turns red and the block that caused the error is highlighted in red as the figure below shows.

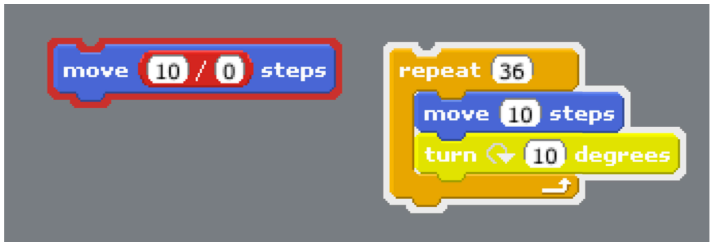


Figure 1. Error in Scratch.

Good block programming languages provide constant visible feedback for user actions (Roque, 2007). A block programming language with strong feedback is Frozen. In Frozen, the user is given feedback all the time. It provides a visible and creative area for hints and feedback available all the time. Also, it provides visible feedback with sound. For instance, if the user is missing a block it displays a message saying that there may be a missing block. There is a sound for every block used that helps the user to identify what the block does and figure out the missing one. On the other hand, languages weak in feedback can be confusing (Roque, 2007). For example, Snap! Does not provide feedback at all. If the user

drags a wrong block (i.e., a block that does not fit) it does not provide any feedback. The user must figure out by the name of the blocks, which block to use.

Interaction Flow, Symbolic Representations and Terminology

A simple and easy way to navigate block programming language is with visual representations appropriate for the assigned tasks to guarantee the language accomplishes its purpose. A nice framework enhances the capability of the language to allow the user to reach all of the features (Chang, 2012). For instance, Scratch is considered to have one of the best symbolic representations and terminology among other programming languages (Suleyman et al., 2011). In Scratch, all programming statements are well represented by text and symbols.

User Controls

User controls are very important attributes to the overall experience for the user. User controls can be described as aesthetic appeal, usability of the functions, and a clearly illustrated purpose of the game. It is important that the user has an easy experience when reading the text to correctly navigate through it. Attributes that would assist in increasing the aesthetic appeal of the text include highlighting key terms and dividing text into clear and concise paragraphs. Additionally, there should be easy navigation through the software, and images and videos should be of high quality to develop "aesthetic feeling in users " (Allen & Tanner, 2006). Click, drag, and drop attributes should also be consistent and understandable. In the document, the author states that, "education software should be stable and provide the users with a high-level technical, technological and user comfort" [1]. This is useful because if the software is difficult to maneuver through, then the goal of educating will not reach its fullest potential. In addition, user control is ranked to be one of the most important attributes that software should demonstrate.

Symmetry Between System and Real-World

Matching between the system and the real-world is making the game easily connected to the real world as well as being age appropriate. This can be described as real world situations or problems in the program can be logically be translated into the real-world. Allen and Tanner (2006) state that "software content provides the themes for educational projects, problem tasks and real-life examples." For example, the vocabulary words first grade students understand can be acted out. This is very important because the game will not only appeal to the grade level of the users but also assist them in furthering their understanding by providing an enjoyable resource expanding upon classroom material. According to Allen and Tanner (2006), "texts [should be] stylistically and professionally adjusted to the age of users in the target group and develop their communication skills." This again reiterates the need for the terminology and other attributes to align with what the students are actually learning in the classroom. The rubric focuses on the vocabulary and its appropriateness to the intended users, the use of real-life examples, and the use of problem-solving situations that would need to be evaluated

in order to determine the amount of connectedness to the real-world a particular software exhibits.

Visible Progress

It is also an important attribute for the user to be able to visually witness their progress and level completion status reference; therefore, progress status should be visually appealing and easily noticeable. For example, a bar filled to a certain amount in order to express progress. In addition, the visual representation should be automatically updated once progress or regression has been made. This is very important because it provides immediate feedback to the user and gives them the ability to be constantly updated on their success throughout the game.

Attractive Text/Graphics/Video Sound, Navigation

Diemand-Yauman et al. (2011) found a correlation between the effort it took to read text and the ability of users to remember that information for later testing. Surprisingly, information presented in a "harder-to-read" font, (i.e., Comic Sans) was better remembered than the same information in “easier-to-read” type (Diemand-Yauman et al., 2011). One theory is that making the users work harder to read text forces them to focus on the text more acutely, engaging deeper parts of their brains than if they could simply breeze through it.

Visual programming environments create an ease of developing/creating programs in a perspective that is fun and non-threatening. The hopes of an environment such as this is that students will no longer feel anxiety and low self-esteem when encountering CS; this way they will be more open to hopefully pursue a degree in CS. Meerbaum-Salant, Armoni, and Ben-Ari mention how girls using Storytelling Alice “demonstrated greater motivation and willingness to spend extra time on the computer, when compared with those using generic Alice.” (Diemand-Yauman et al., 2011).

However, an interesting raised by Meerbaum-Salant et al., (2013) was that they “de-emphasize aspects of appearance such as the costumes, sounds and visual effects.” Students tend to get distracted by these extraneous features. As an example, in the study performed by Meerbaum-Salant, Armoni, and Ben-Ari, the authors found that media manipulation was the focus for 21% of the students (Meerbaum-Salant et al., 2013). While creativity is one of the major foundations in getting children in CS, it is understandable how it can take away from learning, especially from the younger students.

Help users recognize, diagnose, and recover from errors - Hints

If students are afraid of mistakes, then they are afraid of trying something new, of being creative, of thinking in a different way. Studies in a secondary school (Mason et al., 2016) have shown that when students are taught about growth mindsets and that the brain is malleable, their motivation to learn dramatically increases.

Therefore, it is important that users of a tool are presented with an opportunity to grow from their mistakes. The environment should make the user feel that it is acceptable to make mistakes.

Deep Shareability - Computer Clubhouses

The ability to share one's work is important because it allows the student to become more digitally fluent. To share projects with one another, it shows that they are designers and creators using digital technology (Resnick, 2001). In addition, for students to share their projects with others, they are able to exchange knowledge. In other words, students must be able to "make things" with a computer language and if one student does not know how to do a particular task, by sharing projects that student can see how the other student completed it (Resnick, 2001). The goal of this deep shareability is to "not to teach basic skills, but to help young people learn to express themselves and gain confidence as learners" (Resnick, 2001). The students also learn how to work with one another and how to view someone else's project through the eyes of the creator.

This information aided in creating the rubrics used in the appendices to evaluate eight existing block languages. The rubric is presented below.

Block Language Evaluation Rubric

The purpose of this rubric is to evaluate block programming languages from a user's perspective. Each language evaluated will receive a number of points for each feature determined by how well the language meets the requirements of each point value.

Feature	**0 Points**	**1 Point**	**2 Points**	**3 Points**
Clear Goal/Purpose of Application	There is no clear goal or purpose of application.	The purpose of the application is confusing and unclear.	The purpose of the application is understood, but it is not intuitive.	The purpose of the application is clear and users can easily understand the assigned tasks.
Creativity	Creativity is not permitted. (The user cannot design their character, decorate, change color, etc.)	One to two creative options are given, but there are limited attributes.	Few creative options are given, but there are a variety of attributes.	The software allows for user creativity with many attributes.
Extrinsic Rewards	There are not any extrinsic rewards. (No rewards for continual success, bonus games)	One or two extrinsic rewards are given. (Extra points)	Several extrinsic rewards are given.	The software promotes the use of extrinsic rewards throughout. (Winning prizes, bonus games, extra points)

Positive Feedback	No feedback is provided. (recognition of success)	General feedback is given but only in a limited about (i.e. only at the end of the level)	General feedback is given throughout the game. (i.e. "good job")	Detailed/specific feedback is given to the user. (i.e. "That's right 2+2 = 4)
Appropriate Terminology and consistant (Related to age level as well as correlates to programming terminology)	Applicable terminology is not used throughout the application. (age appropriate language depending on grade level based off of further research)	Terminology correlates to programming terminology but is not applicable to the age group..	Language/vocabulary is appropriate to the intended user but does not easily translate to programming languages.	Both the language/vocabulary match the intended user and correlate to programming languages.
Appropriate Symbolism (i.e., container blocks look like containers)	Appropriate Symbolism is not used. (loop blocks look like such, visual representations of items are accurate)	Several symbols are used but are not graphically appealing to the user. (too large or small given the assigned tasks)	Limited graphics are used, but all are graphically appealing.	The usage of visual representations is diverse and appropriate for the assigned tasks.
Attractive Text/Aesthetics/Visually Consistent	Not consistent, and a lot of irrelevant information is shown.	Inconsistencies in the design. Some irrelevant information displayed.	Some inconsistencies in the design, overall aesthetically pleasing. No irrelevant information displayed.	Simple, clean, and consistent; The design that is aesthetically pleasing and contains no irrelevant information.
Constructive Feedback/ Help users recognize, diagnose, and recover from errors (problem hints)	The software does not give problem hints or assist users in discovering the correct answer.	Easy to find help, but the hints are inadequate. Do not tell the user the mistake.	Easy to find help, but the hints are adequate. Tells the user the mistake but no hints or explanation.	The software provides detailed problem hints for users, and assists them in recognizing, diagnosing and recovering from their errors. Provides suggestions on how to fix the mistake. Provides an explanation.
Learn by mistakes/Accessibility to return to previous lessons	No instructions are given to go back to previous steps. No way to correct mistakes.	Unclear instructions to go back to previous steps. Difficult to correct mistakes.	Able to navigate through previously learned material. Able to correct mistakes.	Easy to go back to the previously learned material. Easy to correct mistake.

Visibility/Audibility of System Status	System status within the software is not shown or heard by user.	The user obtains updates on their status but is not consistently aware of where they stand.	A visual status bar is displayed making the user aware of status, but audio cue is used to notify the user of status.	The user is consistently made aware of their status and progress within the software through the use of a visible status bar or audio reinforcement.
Instruction Manual/Documentation (usage hints - block doesn't fit)	No manual is provided. No usage hints for users.	Instructor is provided documentation that is not clear or easily navigable.	Instructor is provided documentation that is clear but not easily navigable. The user is given occasional usage hints.	The Instructor is provided with adequate documentation on the software. The user is given usage hints. (they are told when blocks do not fit.)
Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	The document does not allow for easy shareability.	The software allows for shareability with a limited number of applications but requires external software, or it not easily accessible.	The software allows for shareability with a limited number of applications but easily accessible.	The software allows for easy shareability in a variety of formats and manners.

References

AccessComputing. (2016). http://www.washington.edu/accesscomputing/.

AccessCS10K. (2016). http://www.washington.edu/accesscomputing/accesscs10k.

Allen, D. and Tanner, K. (2006). Approaches to Biology Teaching and Learning. *Rubrics: Tools for Making Learning Goals and Evaluation Criteria Explicit for Both Teachers and Learner,* CBE-Life Sciences Education*.* 2006. Vol 5, 197-203

Apone, K., Marina, B., Brennan, K., Franklin, D., Israel, M., and Yongpradit, P. (2015). Bringing grades K-5 to the mainstream of computer science education. *Proceedings of the 46th ACM Technical Symposium on Computing Science Education (SIGCSE)*, March 2015, Kansas City, MO, pp. 671-672.

Burgstahler, S. (2011). Universal design: Implications for computing education. *ACM Transactions on Computing Education (TOCE)*, vol. 11, 3, October 2011.

Chang, T. (2012). Using graphical representation of user interfaces as visual references. Jun. 2012, Massachusetts Institute of Technology, pp. 1-133.

CS For All. (2016). https://www.whitehouse.gov/blog/2016/01/30/computer-science-all.

Diemand-Yauman, C., Oppenheimer, D. M., and Vaughan, E. B. (2011). Fortune favors the (): Effects of disfluency on educational outcomes. Cognition, 118(1), pp. 111-115.

DO-IT. (2016). http://www.washington.edu/doit/.

Falcão, T. P. (2010). The role of tangible technologies for special education. *Extended Abstracts on Human Factors in Computing Systems (SIGCHI)*, April 2010, Atlanta, GA, pp. 2911-2914.

Masip, L l., Granollers, T., and Oliva, M. (2011). A heuristic evaluation experiment to validate the new set of usability heuristics. *Proceedings of 8th International Conference on Information Technology: New Generations*. IEEE Computer Society, Washington, DC, USA.

Mason, A., Yerushalmi, E., Cohen, E., and Singh, C. (2016). Learning from mistakes: The effect of students' written self-diagnoses on subsequent problem solving. The Physics Teacher, 54(2), pp. 87-90.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. Computer Science Education, 23(3), pp. 239-264.

Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.

Pasternak, A. (2016). Contextualized teaching in the lower secondary education long-term evaluation of a CS course from grade 6 to 10. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE)*, March 2016, Memphis, TN, pp. 657-662.

Resnick, M. (2001). Revolutionizing learning in the digital age. The internet and the university: Forum, pp. 45-64.

Roque, R. (2007). *OpenBlocks: an extendable framework for graphical block programming systems*. Diss. Massachusetts Institute of Technology, 2007.

Scratch Jr. (2016). https://www.scratchjr.org.

Suleyman, U., Karakus, M., and Turner, S. W. (2011). Implementing IT0/CS0 with scratch, app inventor forandroid, and lego mindstorms. *Proceedings of the 2011 conference on Information technology education*. ACM, 2011.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, vol. 49(3), pp. 33-35.

III) **Process**

The preliminary step in our research process required an understanding what our research is about. This meant answering many questions about our research, such as:

1. What do we want to create?
2. Who do we want to impact?
3. What will our contribution for world betterment be?
4. What do want our research to make a difference in?
5. As computer scientist, how can we use our skills to help others?

After answering these preliminary questions, the scope of the project was important to consider. The goal of the research project was to create a block programming language for students with learning disabilities. It was important to determine if anything existed that already accomplished this same goal. Fortunately, it was concluded that nothing on the Internet exists in helping students with learning disabilities how to program. Additionally, the main goal of existing block programming languages is to teach students how to program. In order to create a unique block programming language, the goal to help students' confidence in their reading and math skills, skills many students struggle with, would be the primary goal while the block programming language discretely teaches students programming principles.

The team then started researching and evaluating other block programming languages, such as Scratch and Alice. Each team member played and interacted with the block languages and made a list of likes/dislikes. From this step, the rubric for Hopper's Fables was created and sent to the College of Education to revise, ensuring it was realistic for students with learning disabilities as well as to meet pedagogical standards.

While performing this research and constructing the rubric, the team studied HTML/CSS, Javascript, and Blockly's API in preparation of creating the actual block language, Hopper's Fables.

Once approved by the College of Education, evaluations of existing block languages began as well as construction of Hopper's Fables. The first step was to create a GitHub repository for the team to better enable collaboration. The next step was to inject Blockly into our workspace and create our own blocks. Each of us made a block to get a feel for it and learn how to make custom blocks to match how we wanted them to look.

Currently, we are creating the graphics for Hopper's Fables and the interface it will go on as well as creating more blocks with working functionalities using JavaScript. Once these last steps are completed, we will have created a basic setup of Hopper's Fables and will continue to develop it.

IV) **Results and Discussion**

Originally, this project aimed to produce two types of results: 1. Evaluations of existing block programming languages and 2. User data based on a custom-built language. However, as we tackled the various milestones of this project, it became evident that there was additional information needing to be collected including terminology data. As a result, we constructed a terminology study, which will be performed this summer at a Girls, Inc. summer camp. The primary results we have to date are the evaluations of eight different block programming language, which helped in forming the requirements needed to develop a custom block language.

There are several block programming languages that we evaluated against the rubric: Scratch, three selections from Code.org, Snap, Alice, Scratch, Jr., and MIT App Inventor. The purpose of evaluating existing block programming language was to determine the strengths and weaknesses that we could either implement into our language or change in order to improve the overall quality.

Based on the evaluation of Scratch (Appendix A), it is a well-designed language for student creativity. The language is not necessarily in the format of a game so it does not provide extrinsic rewards or feedback. In fact, the language allows for students to create games themselves. Scratch exhibits the following characteristics: clear goal, creativity, and aesthetics. The users are able to select from tutorials of what they would like to create and are able to develop anything they would like. Appropriate terminology is used as well; words such as turn, move, and glide are easy to understand for the intended audience. If users have questions, they can refer to the documentation. Scratch has sharing capabilities, which allows users to collaborate and share their progress.

We studied three selections from Code.org. The first one is Minecraft (Appendix B); it is well designed and helpful for students who are learning how to code; however, the game lacks in creativity, extrinsic rewards, and sharing ability because the game was premade. While it has a status bar, it does not remind the user of how far they have progressed. On the other hand, the goal is clearly established with each mission. Also, some positive attributes are that the game provides feedback both positive and constructive. For example, a congratulations window is prompted when a mission is completed and an option to view the code is also accessible to the users. In addition, most of the terms are appropriate for the audience, and instructions are provided. Some of the terminology is more in line with the game

rather than with coding. Some of the terms such as “spear” or “birch” may not be appropriate for the audience as they may not know these terms. Overall, Coge.org’s Minecraft rated extremely well against the rubric.

The next Code.org Hour of Code instance evaluated was Frozen (Appendix C), which ranked extremely high compared to the other languages. During the evaluation, there were very few faults that were found in the language. The goal and purpose is clear throughout the duration of the game; there are creative attributes and extrinsic rewards where users win prizes, bonus games, and extra points. It also provides positive feedback by saying good job when a level is completely successfully as well as constructive feedback with detailed hints for the users to recover. The terminology and symbolism are also appropriate for Code.org’s intended audience and correlate with programming languages. The overall appeal of the game is positive and attractive to users.

The last Code.org Hour of Code instance that was evaluated was Moana (Appendix D). Overall, this game ranked very well compared to the other block programming languages. Attributes such as establishing a clear goal, using programming terminology, and providing feedback were all included in the game. The users are told at every level what the purpose and goal was and how to be successful. The terminology was simple and consistent which allowed the users to easily understand what they are able to do, and if a mistake was there, detailed feedback was provided that navigated the users to completing the level. Furthermore, the language was aesthetically appealing both visually and audibly. Also, users have many ways of gaining help whether it was through the user of the manual or reading the hints that are provided. In conclusion, the Code.org programming languages are well designed and have a lot of features that we would like our language to have.

The next languages that were evaluated were Snap! (Appendix E) and Alice (Appendix F). There were some positive attributes in this language such as the use of appropriate terminology that both correlated with the intended user and to programming languages; the usage of visual representations is diverse and relates to the assigned tasks; and the aesthetic and audible appeal of both languages. Snap! provides increased creativity as the levels progress and allows for shareability, but it does not provide extrinsic rewards, positive or constructive feedback, or illustrate a clear purpose of the language. In a similar manner, Alice did not perform well against the rubric. The language is very different to understand and navigate and is not appropriate for younger children trying to learn how to programming. It nearly had none of the attributes in the rubric; however, there is a high level of creativity, which contributed to the detriment of the language because it took away the purpose of the software. Alice did provide many avenues of sharing one’s code, but in all, it is not a good example of the goal of our language.

Overall, Scratch, Jr. (Appendix G) performed better against the rubric than MIT App Inventor (Appendix H). Scratch, Jr.'s block terminology matches the intended users and its simplicity made it very easy to understand. In addition, the aesthetics were simple, clean, and consistent, which were found to be very helpful considering the target audience is younger children. One of the characteristics of MIT App Inventor that performed well against the rubric was shareability; the language allows users to easily share their project in a variety of formats. Scratch Jr. provided another form of structure that this research group took into consideration and had to make a decision on before starting implementation: block alignment. The alignment of blocks fit horizontally rather than vertically in Scratch, Jr. Other languages such as MIT App Inventor and the Code.org selections have their block alignment vertically, which illustrates how actual lines of code during programming would look. This posed a conflict to the development of Hopper's Fables because to foster English and sentence structure, horizontal blocks may be more aesthetically appealing, but in order to foster programming structure, vertical alignment may be more useful. In the end, the research team decided to maintain the consistency of other languages and impose the vertical alignment, which is also how Blockly languages are constructed. As a result, the research team was able to gain some important knowledge and narrow the focus of our language.

Other results of our work can be seen in
Appendix I: Blocks required for Hopper's Fables,
Appendix J: Common Core Math for First Grade,
Appendix K: Common Core Vocabulary for First Grade, and
Appendix L: First Day of School Storyline.

Two of the graphics for Hopper's Fables are pictured below: Admiral Grace Hopper (Figure 2) and the introductory screen (Figure 3). The language is built around a student's first day of school (Appendix L).

Figure 2. Admiral Grace Hopper

Figure 3. Introduction Screen

V) Future Work

As we progress with the project over the summer, we will advance into further development, testing, and delivery. The development stages involve developing all the graphics, developing the blocks, and the Blockly API and functionality. Development also includes ensuring an understanding of the Common Core requirements and the knowledge of our user base, first grade students. Testing of the software involves both testing the actual code and acceptance testing. Testing the code will ensure it works as intended for a variety of scenarios and cases. Acceptance testing involves testing the language with users to ensure it meets standards and is understood as intended. These users will be elementary school students who would most likely use this software, and studies will be coordinated with Girls, Inc. of Greater Atlanta. The last phase is delivery, which involves dispersing the software, which we will do by making the block language available online. This also includes written up formal documentation and papers on the overall research experience and our results.

VI) Web Links

All of our work is being stored on GitHub with a wiki page about the project:
https://github.com/awagne30/ksu-creu-2016/wiki

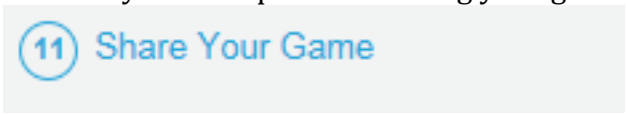
The GitHub page with our code is:
https://github.com/awagne30/ksu-creu-2016

We wanted a more creative interface for the project so we created a wordpress site:
https://hoppersfables.wordpress.com

VII) **Presentations and Publications**

ACM Mid Southeast 2016: Hopper's Fables, Abstract with Presentation
ACM SIGCSE 2017: Hopper's Fables: A Mathematical Storytelling Adventure, Poster Presentation

Appendix A: Scratch Evaluation

- **Clear Goal:** 3
    - Very clear goal, you can select from tutorials of what you'd like to create
- **Creativity:** 3
    - Scratch allows the user to create virtually anything they'd like
- **Extrinsic Rewards: 0**
- **Positive Feedback: 0**
- **Appropriate Terminology: 3**
    - Words such as turn, move and glide are easy to understand for the intended audience.
    - If they are not understood, they can go into the documentation and access the information on blocks in there.
- **Appropriate Symbolism: 3**
- **Attractive Text/Aesthetics: 3**
    - Very simple, white and light gray design.
    - Since the language has its own design features. It ensures that your design will not be affected by the system's design.
- **Constructive Feedback: 0**
- **Learn by Mistakes: 0**
- **Visibility/Audibility of System Status: 0**
    - Not included, nor necessary. There isn't a status option because it is created for you to do on your own time.
- **Instruction Manual/Documentation: 3**
    - In create mode, you are able to select a step-by-step directions for a limited list of projects.
    - There is also documentation on how to and about the blocks themselves
- **Shareability:** 3
    - It literally has an option of sharing your game displayed
    - 

- **My evaluation description:**
    - Scratch is a well-designed language for student creativity. The language is not necessarily in the format of a game so it does not provide extrinsic rewards of feedback. In fact, the language allows for students to create games themselves.

Appendix B: Code.org Minecraft Evaluation

- **Clear Goal: 3**
  - The game tells you what the goal is to complete in each mission**.**
- **Creativity: 1**
  - Since the game is premade, there is little room for creativity because there are missions to be completed. However, you are able to choose between 2 premade characters.
- **Extrinsic Rewards: 0**
  - There are no awards or points.
- **Positive Feedback: 3**
  - Prompted with a congratulations window and the option to view the code you have written.
- **Appropriate Terminology: 2**
  - I have already evaluated the terms used. These may include words such as congrats, turn, etc. Most of the terms are appropriate for the audience.
  - In my opinion, some of the terms such as spear or "birch" may not be appropriate for the audience as they may not know these terms.
- **Appropriate Symbolism: 3**
  - The game appears like the usual Minecraft game with abstract features as everything is represented as a block.
- **Attractive Text/Aesthetics: 3**
  - Nice and neat gray design. The game appears like any normal Minecraft game.
  - Not distracting.
- **Constructive Feedback: 3**
  - Example: "Not quite. You have to use a block you aren't using yet".
- **Learn by Mistakes: 3**
  - When you have not completed an assignment, you are prompted with a message that you haven't entirely completed the mission.
  - You can then click, reset in order to reset the mission. Or can just add on the code you missed.
- **Visibility/Audibility of System Status: 2**
  - At the top of the window, there is a status bar that lets you know what section you are working on.
- **Instruction Manual/Documentation: 3**
  - The game offers a need help section on each page in which you can click in order to view videos and see helpful hints.
- **Shareability: 0**
  - The software cannot be shared, however, you may get a certificate saying you have completed an hour of code for this game.
- **My evaluation description:**
  - Minecraft Hour of code game is well designed and helpful for students who are learning how to code. The game lacks in the way of creativity

and while it has a status bad it does not remind the user of how far they have progressed.

Appendix C: Code.org Frozen Evaluation

Feature	Rating	Explanation/Comments
Clear Goal/Purpose of Application	5	Excellent! The goal of the game is clear.
Creativity	5	Excellent! Very creative language, it has many fun attributes.
Extrinsic Rewards	5	Excellent! The software promotes the use of extrinsic rewards throughout. (Winning prizes, bonus games, extra points) • Hour of code diploma • Bonus
Positive Feedback	5	Excellent! • Hints all the time • Hints throughout videos and sounds • Visible area for hints
Appropriate Terminology and consistant (Related to age level as well as correlates to programming terminology)	5	Excellent! Simple, clear, (age appropriate) Both, the language and vocabulary match the intended user and correlate to programming languages
Appropriate Symbolism (i.e., container blocks look like containers)	5	Excellent! Very appropriate symbolism for the assigned tasks The representation of programming statements is great.
Attractive Text/Aesthetics/Visually Consistent	5	Excellent! Simple, clean, consistent, and organized. The design is very attractive and easy to navigate it. And, contains no irrelevant information
Constructive Feedback/ Help users recognize, diagnose, and	5	Excellent!

recover from errors (problem hints)		The software provides detailed problem hints for users, and assists them in recognizing, diagnosing and recovering from their errors. (by using videos, sounds and examples) Provides suggestions on how to fix the mistake. Provides an explanation.
Learn by mistakes/Accessibility to return to previous lessons	5	Excellent! Very easy to go back to the previously learned material, and easy to correct mistake.
Visibility/Audibility of System Status	5	Excellent! • Nice and visible status bar. • The user is consistently made aware of their status and progress within the software. • Audio is used to notify user status
Instruction Manual/Documentation (usage hints - block doesn't fit)	5	Excellent! The user is given usage hints. (they are told when there are missing blocks) Hints are given though sounds, videos, and other documentation.
Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	5	Excellent! Progress and current work is saved automatically

Appendix D: Code.org Moana Evaluation

CODE.ORG MOANA

Feature	**Rating**	**Explanation/Comments**
Clear Goal and Purpose of Application	3	• 28 seconds: Introduction Video - "Our me and our team, the Village of Coders, in discovering and navigating the new world of coding to help defeat the Kakamora on our latest journey! • Based on the movie where she has to defeat them things as well; shows a clip from the movie explaining • Importance of coding, advertising Hour of Code • Each level gives an explanation of what the purpose of this level is for
Creativity	1	• Not very creative • At level 11, the user is able to choose one of two characters to go on ship with
Extrinsic Rewards	0	• No extrinsic rewards are given
Positive Feedback	2	• "Well Done" after level completion • Celebratory music is played • Graphics are appealing
Appropriate Terminology and consistent (Related to age level as well as correlates to programming terminology)	3	• Text is simple and easy to understand (age appropriate) • Loops and conditional statements can be easily translated into if and for loops
Appropriate Symbolism	3	• The ability to connect blocks show a sequential order of code • Loops allow for statements to be executed inside • If.. do statements
Attractive Text/Aesthetics/Visually Consistent	3	• Extremely attractive; feel like you out at sea; music, flowers, tribal print • Moving waves of the ocean • Feel of the wind blowing the boat • Text is clear, concise • Appropriate ocean, Hawaii like music plays in the background • Water makes sounds
Constructive Feedback/	3	• "Oops" on wrong answers (graphic

Help users recognize, diagnose, and recover from errors (problem hints)		image of the characters looking mad at you) • Ex: "Add move forward three times to reach the fish" • Hints on what and how many blocks to use • Hints are only displayed if level completion was wrong
Learn by mistakes/Accessibility to return to previous lessons	3	• You are able to go back • Hint button that will give you a description of what to do in order to complete the level
Visibility/Audibility of System Status	2	• Level completion and current level status is displayed at the top of the screen • No audio display of level completion

Instruction Manual/Documentation (usage hints - block doesn't fit)	3	• Gives a tutorial of the game display
		  • Left is the MOANA game space
		  • Instruction are above the game space
		  • Tool box in the middle: block commands you can use
		  • Yellow space: workspace, build program
		  • Shows the number of blocks you will need in order to solve the puzzle
		  • Drag and Drop attributes; delete
		  • Introduction video for loops
		• Instructions are given in text and audibly during the course of the game
		• However, each level does not give audio instructions

Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	0	• No Ability to share code

Appendix E: Snap! Evaluation

https://snap.berkeley.edu

Feature	Rating	Explanation/Comments
Clear Goal/Purpose of Application	1	There is no clear goal or purpose of the application; there are not clear instructions where to start and how to start.
Creativity	2	There is not creativeness in the design and structure of the language. It is not an attractive language. however, once you get familiar with the language it allows you to colors, features, sounds and other features in the games.
Extrinsic Rewards	0	No rewards
Positive Feedback	0	No feedback at all
Appropriate Terminology and consistant (Related to age level as well as correlates to programming terminology)	5	Both the language/vocabulary match the intended user and correlate to programming languages • Text is simple; short and common words (age appropriate)
Appropriate Symbolism (i.e., container blocks look like containers)	5	The usage of visual representations is diverse and appropriate for the assigned tasks. The symbolism is great in programming concepts • Loops • set variables • operators
Attractive Text/Aesthetics/Visually	3	• The colors and text is attractive but the design in general is not.

Consistent		• The stage (the display area) is boring • Contains no irrelevant information
Constructive Feedback/ Help users recognize, diagnose, and recover from errors (problem hints)	0	No feedback given
Learn by mistakes/Accessibility to return to previous lessons	2	There is an option to go back
Visibility/Audibility of System Status	3	• There is not a visual status bar display • But, there a sound pallet that the user can use at any time to give sound to the game
Instruction Manual/Documentation (usage hints - block doesn't fit)	2	• No hints • There are manuals in the Snap website but not inside the Snap game environment. So If the user needs help, he/she has to go to the website and find the manual.
Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	5	The software allows for easy shareability in a variety of formats and manners. • Arduino • Lego NXT package

Appendix F: Alice Evaluation

Feature	Rating	Explanation/Comments
Clear Goal and Purpose of Application	0	• Lacks a plot or a purpose
Creativity	3	• Numerous amounts of features, attributes • One can say it has too many options because the selection is so expansive
Extrinsic Rewards	0	• No rewards are given
Positive Feedback	0	• No feedback is given
Appropriate Terminology and consistent (Related to age level as well as correlates to programming terminology)	3	• Very technical terminology that easily relates to a programming language
Appropriate Symbolism	3	
Attractive Text/Aesthetics/Visually Consistent	1	• The graphics are very simple • There is a lot of text that give a cluttered feeling
Constructive Feedback/ Help users recognize, diagnose, and recover from errors (problem hints)	0	• No constructive feedback is given
Learn by mistakes/Accessibility to return to previous lessons	0	• There are not any levels
Visibility/Audibility of System Status	0	• No progression of status is given
Instruction Manual/Documentation (usage hints - block doesn't fit)	3	• There is a help menu tab
Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	3	• Can easily save progress and current work

Appendix D: Scratch, Jr. Evaluation

Feature	Rating	Explanation/Comments
Clear Goal and Purpose of Application	4	• Goal: Students to create interactive stories and games
Creativity	5	• Students can snap together graphical programming blocks to make characters move, jump, dance, and sing. • Students can can modify characters in the paint editor, add their own voices and sounds, even insert photos of themselves and then use the programming blocks to make their characters come to life.
Extrinsic Rewards	5	
Positive Feedback	0	
Appropriate Terminology and consistent (Related to age level as well as correlates to programming terminology)	5	
Appropriate Symbolism	4	
Attractive Text/Aesthetics/Visually Consistent		• Interface and programming language are developmentally appropriate for younger students. • Designed features to match young students cognitive, personal, social, and emotional development.
Constructive Feedback/ Help users recognize, diagnose, and recover from errors (problem hints)	4	
Learn by mistakes/Accessibility to return to previous lessons	0	
Visibility/Audibility of System Status	4	
Instruction Manual/Documentation (usage hints - block doesn't fit)	5	• Has a section tab called "Learn", which is very informative with graphics and arrows to show what the Scratch Jr. interface is able to do. AKA infographics ?

Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	0	

Appendix H: App Inventor Evaluation

Feature	Rating	Explanation/Comments
Clear Goal and Purpose of Application	5	• Allows even an inexperienced student the ability to create a basic, fully functional app within an hour or less.
Creativity	5	
Extrinsic Rewards	5	• The reward of having a simple functioning app
Positive Feedback	0	
Appropriate Terminology and consistent (Related to age level as well as correlates to programming terminology)	5	
Appropriate Symbolism	4	
Attractive Text/Aesthetics/Visually Consistent	3	
Constructive Feedback/ Help users recognize, diagnose, and recover from errors (problem hints)	0	
Learn by mistakes/Accessibility to return to previous lessons	0	
Visibility/Audibility of System Status	4	
Instruction Manual/Documentation (usage hints - block doesn't fit)	0	
Shareability (transference of created task to other public forms i.e. can be sent through email, posted online)	5	

Appendix I: Blocks required for story.

- Getting Ready for School
  - Block: "My character is" -drop down option of names of different characters
    - On execution, the outline of the character is yellow as if it is selected. Then, the scene will change to kitchen
  - Block: "For breakfast, I would like to eat" - drop down menu of breakfast options "apple", "yogurt", "cereal"
    - On execution, that food item is brought to the table for the user to eat
  - Block: "Move forward"
  - Block: "Turn Right"
  - Block: "Turn Left"
  - Block: "Sit at kitchen table"
    - On execution, the student sits down at the table and food is in front of them with a glass of juice
  - Block: "Eat"
    - Sounds of someone eating food; Food item disappears and a mess appears (plate, napkin, glass of juice empty)
  - Block: "Clean Dishes"
    - On executions, Sounds of watering running; Clean dishes appear on the counter
  - Block: "Pick up book bag"
    - User picks up book bag
  - Block: "Go to bus"
    - scene switches to outside where the bus is waiting
  - Block: "Get on bus"
- Entering the Classroom
  - Block: "Put book bag in cubby"
    - On execution, book bag is in the shelf
  - Block: "Sit down"
  - Block: "Say"-dropdown option "Hello, do you want to be friends?", "What is your name?"
    - "Hello, do you want to be friends?" Option: "Yea, sure. I would love to."
    - "What is your name?" Option: "My name is Sophia. Nice to meet you."
- Lunchtime
  - Block: "Say"-dropdown option "Hello, do you want to be friends?", "What is your name?"
    - "Hello, do you want to be friends?" Option: "Yea, sure. I would love to."
    - "What is your name?" Option: "My name is Sophia. Nice to meet you."
  - Block: "Sit next to your friend"
  - Block: "Say"- "What did you get for lunch?" or "Today has been fun"
    - "What did you get for lunch?" Option: "I got chicken tenders. They are so good."
    - "Today has been fun" Option: "It has been great. I love school."
- Recess
  - Block: "I want to play" -dropdown menu with the options "soccer" or "on the jungle gym"
    - "soccer" option: the scene changes to the soccer field and the goal

        - Block: “Kick ball into goal” -dropdown option of how many times (1-5) the ball will be kicked into goal
            - On execution, a loop should have the user kick the ball into goal
    - “on the jungle gym” option: the scene changes to the slide part of the jungle gym
- Block: “Slide down the slide” - -dropdown option of how many times (1-5) the users will slide down the slide
    - On execution, a loop should iterate how many times the user is sliding down the slide

- Class Activity
    - Block: “The book I would like to read is” - dropdown options “Dr. Suess”, “My Best Friend”, and “How to be a Good Dog”
        - “Dr. Suess” Option: that book is selected
        - “My Best Friend” Option: that book is selected
        - “How to be a Good Dog” option: that book is selected
    - Block: “Hand book to teacher.”
    - Block: “Say”{ - dropdown options “We should read this.”, “I like this is book”
- Going Home
    - Block: “I want to sit next to” - dropdown options “Cat”, “Bear”, “alone”
        - “Cat” Option: user goes to sit next to Cat
        - “Bear” Option: user goes to sit next to Bear
        - “alone” Option: user goes and sits in a empty seat

Appendix J: Common Core Math for First Grade

In Grade 1, instructional time should focus on four critical areas:

1. Developing understanding of addition, subtraction, and strategies for addition and subtraction within 20.

   Students develop strategies for adding and subtracting whole numbers based on their prior work with small numbers.

   - add-to, take-from, put-together, take-apart, and compare situations to develop meaning for the operations of addition and subtraction,
   - Develop strategies to solve arithmetic problems with these operations.
   - Students understand connections between counting and addition and subtraction
   - Adding two is the same as counting on two.
   - Making tens
   - Solve addition and substation problems within 20.

2. Developing understanding of whole number relationships and place value, including grouping in tens and ones

   - Methods to add within 100 and subtract multiples of 10.
   - Compare whole numbers (at least to 100) to develop understanding of and solve problems involving their relative sizes.

     - think of whole numbers between 10 and 100 in terms of tens and ones
     - recognizing the numbers 11 to 19 as composed of a ten and some ones
     - understand the order of the counting numbers and their relative magnitudes.

3. Developing understanding of linear measurement and measuring lengths as iterating length units.

   Students develop an understanding of the meaning and processes of

   - Direct measurement, including underlying concepts such as iterating (the mental activity of building up the length of an object with equal-sized units)
   - Indirect measurement.

4. Reasoning about attributes of, and composing and decomposing geometric shapes.

   Students compose and decompose plane or solid figures

   Example:

- put two triangles together to make a quadrilateral
- Combine shapes and recognize them from different perspectives and orientations
- describe their geometric attributes, and determine how they are alike and different
- develop the background for measurement and for initial understandings of properties such as congruence and symmetry.

Grade 1 overview

operations and algebraic thinking

- represent and solve problems involving addition and subtraction.
- Understand and apply properties of operations and the relationship between addition and subtraction.
- add and subtract within 20.
- Work with addition and subtraction equations.

number and operations in Base ten

- extend the counting sequence.
- Understand place value.
- Use place value understanding and properties of operations to add and subtract

measurement and data

- measure lengths indirectly and by iterating length units.
- tell and write time.
- represent and interpret data.

Geometry

- reason with shapes and their attributes.

Mathematics

Term	Definition
Addition	the operation of combining numbers so as to obtain an equivalent simple quantity
Subtraction	the operation of deducting one number from another
Sum	the whole amount
Difference	The degree or amount which things differ in quantity or measure
Group	To combine
Counting on	To start with a number in a counting sequence and continue
Making ten	To make a set of ten
Combinations	Different ways to solve a problem (example : 2+2= 4, 3+1=4, 4+0=4)
Equal sign	=
True	Exact or accurate
False	Incorrect, wrong
Unknown	Not known; not within the range of one's knowledge
Digits	Number figures: 0-9
Two-digit number	Numbers having two digits; each digit is a different place value
Greater than sign	>
Less than sign	<
Mental math	Math calculated in a student's head without paper and pencil
Unit	Standard for measurement
Centimeter	Unit of the metric system
Inch	A unit of measurement equal to $1/12^{th}$ of a foot
Hours	A measure of time equal to 60 minutes
Half hour	A measure of time equal to 30 minutes
Minute	The unit of time with equals 60 seconds
Digital	Expressed in digits
Clock	An instrument that measures time
Trapezoid	A four sided plan figure with two parallel sides
Half-circle	A portion that is equivalent to ½
Quarter-circle	A portion that is equivalent to $1/4^{th}$
Cube	A regular solid with six equal squares
Rectangular	A parallelogram with all right angles
Prism	A solid figure with triangular ends and rectangular sides
Cone	A solid body that is tapered evenly to a point from a base that is circular
Cylinder	A long round body that is either solid or hollow
Half	Being one of two equal parts
Fourth	Being one of four equal parts
Quarter (fraction)	Being one of four equal parts

Appendix K: Common Core Vocabulary for First Grade

Core vocabulary describes a small set of basic words in any language that are used frequently and across contexts [1]. Core words tend to be pronouns, verbs, and demonstratives because they represent words that generally do not change [2]. Words like "big," "little," "give," "eat," "go," and "you" are examples of core vocabulary terms used every day in many situations. Research shows that 80% of what we say is communicated with only the 200 most basic words in our language.

The vocabulary words for first graders will enhance their ability to read, comprehend, communicate, and learn.[1] According to the article "Picturing Language: A 'How-To' Workshop" by Brunce Baker in 2012, this is the word list known at the end of first grade

List 1	List 2	List 3
a	an	after again an
and	are	any
away big blue	at	as
can come	ate	ask
down find	be black brown but	by could every fly from give going
for funny	did	had has her him his how last know
go	do	let
help	eat four get good	live may of
here	have he	old once open over put round some
in	into like new	stop take thank them then think walk
is	no now on	were when
it	our	
jump little look	out please pretty ran	
make me	ride	
my	say	
not one play	she	
red	so soon that there	
run said	they this	
see	too want was well	
the	went what white who	
three to	win with	
two		
up		
we where		
yellow you		

Furthermore, Hyde Park (NY) Central School District Search provides the following *vocabulary list* on the district's web site for first grade.

annoy ignore Prefer

attention
calm
comfortable
consequences
curious
curve
decide
directions
discover
disappointed
embarrassed
enormous
exhausted
explore
fair
fascinating
feast
focus
frustrated
gigantic
grumpy
huge
instead
investigate
invite
important
jealous
leader
list
listen
lovely
measuring
miserable
mumble
negative
nervous
nibbled
note
notice
observing
opposite
ordinary
positive
precious
Problem
protect
proud
question
reminds
repeat
report
rhyme
respect
searching
special
spotless
squirm
stomped
suddenly
suggestion
surprise
uncomfortable
warning
wonder
worried

English Language Arts Standards » Reading: Foundational Skills » Grade 1

Print Concepts:

- Demonstrate understanding of the organization and basic features of print.
- Recognize the distinguishing features of a sentence (e.g., first word, capitalization, ending punctuation).

Phonological Awareness:

- Demonstrate understanding of spoken words, syllables, and sounds (phonemes).
- Distinguish long from short vowel sounds in spoken single-syllable words.
- Orally produce single-syllable words by blending sounds (phonemes), including consonant blends.

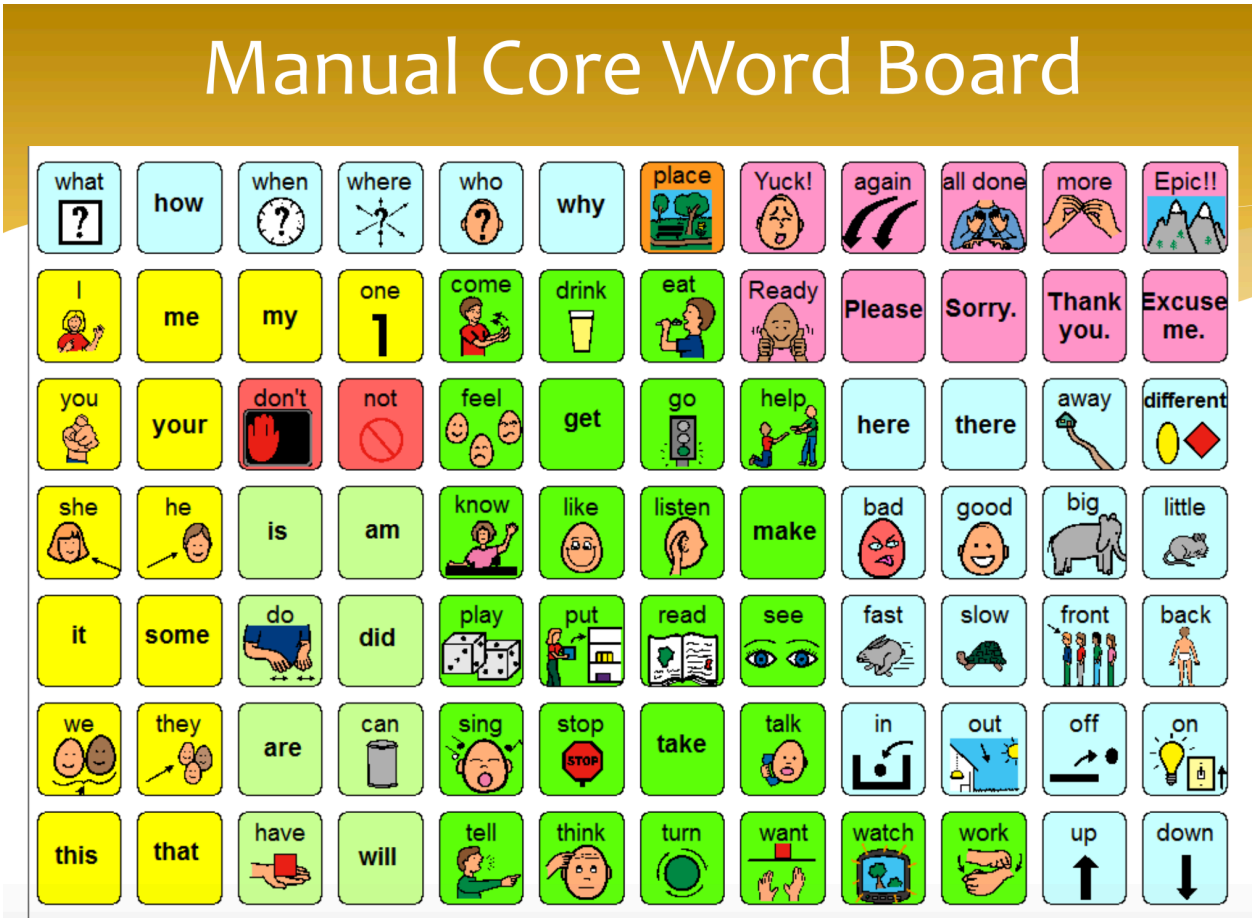
- Isolate and pronounce initial, medial vowel, and final sounds (phonemes) in spoken single-syllable words.
- Segment spoken single-syllable words into their complete sequence of individual sounds (phonemes).

Phonics and Word Recognition:

- Know and apply grade-level phonics and word analysis skills in decoding words.
- Know the spelling-sound correspondences for common consonant digraphs.
- Decode regularly spelled one-syllable words.
- Know final -e and common vowel team conventions for representing long vowel sounds.
- Use knowledge that every syllable must have a vowel sound to determine the number of syllables in a printed word.
- Decode two-syllable words following basic patterns by breaking the words into syllables.
- Read words with inflectional endings.
- Recognize and read grade-appropriate irregularly spelled words.

Fluency:

- Read with sufficient accuracy and fluency to support comprehension.
- Read grade-level text with purpose and understanding.
- Read grade-level text orally with accuracy, appropriate rate, and expression on successive readings.
- Use context to confirm or self-correct word recognition and understanding, rereading as necessary.



Common Core Vocabulary – 1 st Grade

Language Arts

Retell	To tell the story again
Characters	A person in a story, novel, or play
Settings	The background (as time and place) of the action of a story or performance
Prose	Writing that does not have the repeating rhythm used in poetry
Clarify	To make or become easier to understand
Heading	Something (as a title or an address) at the top or beginning (as of a letter or chapter)
Table of Contents	A list of parts or sections of a book and the pages on which they start
Glossary	A list of the hard or unusual words found in a book

Icons	A symbol or pictorial symbol on a computer screen
Phonemes	Speech sound
Consonant blends	Two or three consonant letters whose sounds blend together when forming a word (example: gl, bl, str)
Digraphs	A group of two successive letters who phonetic value is a single sound (example: sh, th, wh, ch, ph)
Phonics	A method of teaching beginners to read and pronounce words by learning the phonetic value of letters, letter groups, and especially syllables
Preposition	A word placed in front of a noun or pronoun to show a connection with or to something or someone (example during, beyond, toward)
Conjunction	A word used to join or connect other words, phrases, sentences, or clauses (example: but, and, or, so, because)
Determiners	A word that makes a specific denotation of a noun phrase (example: this, that, these, those)
Personal pronoun	a pronoun designating the person speaking (example: I, me. we, us or the person or thing spoken about ie. he, she, it
Possessive pronoun	One of several pronouns designating possession (example: they, him, her, them)
Indefinite pronoun	A pronoun that does not specify the identity of its object (example: any, some)
Verb	Words that show action (example: run, sit)
Narrative	List of events in a story
Informative/Explanatory text	Reading that explains something
Temporal words	Words that show time (example: soon, tomorrow)
Common nouns	Names a person, place, or thing (example: desk, car, boy)
Proper nouns	Names a specific person, place, or thing (example: Walmart, Susan, Jose)
Possessive nouns	Nouns that show ownership (example: Katherine's, sister's)
Singular nouns	On single noun (example: cat, dog)
Plural nouns	Names more than one noun (example: cats, dogs)

Sources

[1]Baker, B., Hill, K., & Devylder, R. (2000). *Core Vocabulary is the same across environments*. Paper presented at a meeting of the Technology and Persons with Disabilities Conference at California State University, Northridge.

[2]Stubbs, M. (1986) Language development, lexical competence and nuclear vocabulary. In Durkin, K. (Ed.) *Language Development in the School Years*. London: Croom Helm.

Appendix L: First Day of School Storyline

- Getting Ready for School
- Prompt: "Welcome, today is the first day of school. To begin choose a character"
    - Block: "My character is" -drop down option of names of different characters
        - On execution, the outline of the character is yellow as if it is selected
        - Then, the scene will change to kitchen
- Prompt: "Let's get ready for school!"
    - Scene: Student is standing the kitchen; the student will see a kitchen from birds eye view. The kitchen will have a door, a book bag on the floor, a sink, a counter and breakfast options displayed on the counter
        - Task 1: Student will choose Breakfast (Prompt: "Let's eat breakfast")
            - Block: "For breakfast, I would like to eat" - drop down menu of breakfast options
                - apple
                - yogurt
                - cereal
                - On execution, that food item is brought to the table for the user to eat
            - Prompt: "Go to the table to eat your breakfast"
                - Block: "Move forward"
                - Block: "Turn Right"
                - Block: "Turn Left"
                - Block: "Sit at kitchen table"
                    - On execution, the student sits down at the table and food is in front of them with a glass of juice
                - Block: "Eat"
                    - Sounds of someone eating food
                    - Food item disappears and a mess appears (plate, napkin, glass of juice empty)
        - Task 2: Student will put dishes in the sink (Prompt: "let's clean the dishes")
            - User walks to sink (Maybe a banana peel on the floor)
                - Block: "Move Forward"
                - Block: "Turn Left"
                - Block: "Turn Right"
            - Block: "Clean Dishes"
                - On executions, Sounds of watering running
                - Clean dishes appear on the counter
        - Task 3: Student will grab book bag (Prompt: "the bus is here; it's time to go")

        - Bookbag sitting on the floor in the corner
            - Block: “Move Forward”
            - Block: “Turn Left”
            - Block: “Turn Right”
            - Block: “Pick up bookbag”
                - User picks up bookbag
            - Block: “Go to bus”
                - scene switches to outside where the bus is waiting
            - Block: “Get on bus”
- Entering the classroom
- The first level is over and the user gets a congratulatory prompt.
- Scene: The next scene is in the classroom. Student will see a classroom from bird eyes view. The classroom will include a shelf to a book bag (will basically look like a brown box) and seats facing (will basically look like blue squares) a board (basically look like a gray line about ½ in width). One student (animal) will already be seated in one of the seats, and the student’s seat will be next to that person. The teacher is off to the side.
- Prompt: “ Please put your book bag away”
    - **Task 1:** Putting Book bag away
        - Block: “Move Forward”
        - Block: “Turn Left”
        - Block: “Turn Right”
        - Block: “Put bookbag in cubby”
            - Book bag is in the shelf
    - **Task 2:** Sitting Down (Prompt: “Now it’s time to go to your seat; let’s sit by the Cat, he looks nice.”)
        - Prompt: How many empty seats are there?
        - Block: “Move Forward”
        - Block: “Turn Left”
        - Block: “Turn Right”
        - Block: “Sit down”
    - **Task 3:** Say hello to their friend
        - Block: “Say”-dropdown option “Hello, do you want to be friends?”, “What is your name?”
            - “Hello, do you want to be friends?” Option: “Yea, sure. I would love to.”
            - “What is your name?” Option: “My name is Sophia. Nice to meet you.”
- Level 2 is completed.
- Lunchtime
- Prompt: “I hope you’re hungry! It’s lunchtime.”
- Scene: Student will see a bird eye’s view of a cafeteria. This will include a tray area and places to sit(maybe like 3 circular tables with 4 chairs each)
    - **Task 1:** Select a tray to choose what to eat.
        - Move to tray area

        - Block: “Move Forward”
        - Block: “Turn Left”
        - Block: “Turn Right”
    - Block: “Pick up a tray”
    - Block: “For lunch, I would like to eat” -dropdown option with three options for lunch that are on the counter in from of the user
        - Peanut Butter and Jelly Sandwich
        - Pizza
        - Chicken Tenders
    - Prompt: “Find a seat to eat your lunch”
        - Block: “Move Forward”
        - Block: “Turn Left”
        - Block: “Turn Right”
        - Block: “Sit next to your friend”
        - Block: “Say”- “What did you get for lunch?” or “Today has been fun”
            - “What did you get for lunch?” Option: “I got chicken tenders. They are so good.”
            - “Today has been fun” Option: “It has been great. I love school.”
- **Step 4: Recess**
    - Prompt: “Let’s go outside and play!”
    - Scene: A playground with a soccer area and a jungle gym with a slide
        - Block: “I want to play” -dropdown menu with the options “soccer” or “on the jungle gym”
        - “soccer” option: the scene changes to the soccer field and the goal
            - Block: “Kick ball into goal” -dropdown option of how many times (1-5) the ball will be kicked into goal
                - On execution, a loop should have the user kick the ball into goal
        - “on the jungle gym” option: the scene changes to the slide part of the jungle gym
            - Block: “Slide down the slide” - -dropdown option of how many times (1-5) the users will slide down the slide
                - On execution, a loop should iterate how many times the user is sliding down the slide
- Prompt: “Yay that was fun. Now it is time to go inside.”
- **Step 5: Class Activity**
    - Prompt: “Recess is over, but let’s go back to the classroom it’s Storytime!“
    - Student will see a bird’s eye view of a classroom. The classroom will have a bookshelf, a big rug with 3 other students (animals) sitting and a teacher (animal) sitting on a chair in the front.
    - **Task 1:**

        - Precondition: Student will start at the bookshelf
        - Block: "The book I would like to read is" - dropdown options "Dr. Suess", "My Best Friend", and "How to be a Good Dog"
            - "Dr. Suess" Option: that book is selected
            - "My Best Friend" Option: that book is selected
            - "How to be a Good Dog" option: that book is selected
    - **Task 2:**
        - Student will use a series of move/turn blocks to get to their teacher.
            - Block: "Move Forward"
            - Block: "Turn Left"
            - Block: "Turn Right"
        - Student will select a block to hand the book to the teacher.
            - Block: "Hand book to teacher."
        - Student will select a speaking block to say, "We should read this"
            - Block: "Say" - dropdown options "We should read this.", "I like this is book"
    - **Task 3:**
        - Student will use a series of move/turn blocks to get to their seat on the rug.
            - Block: "Move Forward"
            - Block: "Turn Left"
            - Block: "Turn Right"
            - Math: How many empty seat are there?
        - Students will select the block to sit.
            - Block: "Sit down."
- **Step 6: Going home**
    - Prompt: "We have had a long day. It's time to go home!"
    - Student will see a bird's eye view of the parking lot. The parking lot will include at least 3 buses (remember bird's eye) and a part of the school (should look like a rectangle).
        - Prompt: "Let's get on the bus"
            - Block: "Move Forward"
            - Block: "Turn Left"
            - Block: "Turn Right"
        - Prompt: "Who should we sit next to?"
            - Block: "I want to sit next to" - dropdown options "Cat", "Bear", "alone"
                - "Cat" Option: user goes to sit next to Cat
                - "Bear" Option: user goes to sit next to Bear
                - "alone" Option: user goes and sits in a empty seat
        - Prompt: "You have made it home! Congratulations you have completed the game"