Responsive and Gameful SAGE Assessment

Final Report

COMS6901 – Projects in Computer Science, Fall 2017

Johan Sulaiman/js5063

Contents

Motivation

Increasingly in the last 50 years, video games have been wildly successful in demanding big chunks of timeshare and mindshare from the daily lives of school-age children. Many teachers and parents regard games' influence as a serious threat to students' academic success, as students regularly choose to spend hours playing games and discussing them with friends instead of devoting their time and energy towards their study. Even so, there are some among educators and researchers who take the view that phenomena that attract voluntary investments of time, energy, and attention among children are worth evaluating for their redeeming qualities towards fostering learning. One researcher for example, views game-playing in a more positive light, as a problem-solving activity approached with a playful attitude (Schell, 2008).

Some students also discover other powerful motivation to playing games beyond just for recreational purposes, as they simultaneously approach game playing as experimentations of different problem-solving ideas and strategies while remaining in the safety of the game's controlled and simulated environments. Students through simulated role-playing in games are empowered to assume roles and professions they do not normally perform yet in real-life, immersing and familiarizing themselves early with the rich context and varying facets of the job. In effect, this type of training through epistemic games (Rupp et al, 2010) can serve as excellent preparation for entering the professional workforce in the future. From this perspective, gaming can positively foster creativity, deep familiarity, and appropriate risk-taking essential in building the students' knowledge and skills, while limiting potentially permanent and considerable loss of financial, relational, or even physical capital if similar exposure occurred in real-life.

At the same time, proponents of this view acknowledge that due to a lack of convincing empirical evidence and curriculum-specific learning outcomes, there currently is not enough conclusive proof that games are an educationally effective solution, at least at their current form (Kazimoglu et al, 2010). As such, there is a need for researches that explore, quantify, develop, and formalize the kinds of games, gaming interactions, and gaming behaviors that are conducive towards obtaining academic and professional knowledge. As part of Columbia Programming Systems Lab's Social Addictive Gameful Engineering (SAGE) project, this report aims to contribute to this effort by developing a highly responsive and Gameful students' proficiency assessment system that aims to maximize Computational Thinking learning and shorten the pathway towards competency and mastery.

Related Work

Computational Thinking

One area of learning that is especially strategic to prepare students for the jobs of the 21st century is Computational Thinking. Popularized by Dr. Jeannette Wing in 2006, Computational Thinking recognizes that the jobs of the future will increasingly rely on the aid of programmable machines to meet increasingly higher standard of acceptable productivity in all industries and workplaces (Wing, 2006). Workers who seek to maximize their output through machine-and-computer-based enablers are required to master multi-layered concepts and mental competencies such as abstraction, decomposition, recursion, iteration, modeling, caching, and many others. These myriad areas of learning are complex concepts that are necessarily abstract, in order for them to be applicable to virtually every industry that can leverage machine computations to augment human workers. This translates to a steep and extensive learning curve in mastering Computational Thinking, requiring persistence, creativity, and mental resiliency from students because pay off for the time and energy invested only appears to be more worthwhile closer to the end of the curve as theoretical and practical proficiency of the major concepts finally come together to realize real-world productivity gains.

This abstract and complex nature of the domain presents at least two major challenges. First, the daunting learning curve makes it difficult to get students interested over a long period of time to undertake Computational Thinking learning in both formal and informal settings. Second, once the hurdle of getting students engaged is somehow overcome, there is a shortage of teachers who have been adequately trained to be proficient in Computational Thinking, and are adept in teaching the subject to students. The SAGE project seeks to answer these challenges by 1) seeking to increase and maintain students interests through attractive presentation of Computational Thinking in a highly visual and game-like package, 2) providing intuitive curriculum design and assessment tools to support teachers, and 3) through an automated intelligent tutoring system and activation of a rich and supportive social hub of fellow learners, substituting some of the roles traditionally performed by teachers in providing guidance, hints, evaluation, and progress feedback of students' performance along pre-defined learning path. Understanding Computational Thinking also provides researchers solid grounding on the different competencies that can be used to design objective assessments and measurements.

PECT, Dr. Scratch, and Hairball

Progression of Early Computational Thinking (PECT) model was especially designed to understand and assess development of computational thinking in primary grades, which contains metrics called Evidence Variables (Seifer et al, 2013) that could be leveraged in any development of Computational-Thinking-oriented curriculum. Figure 1 shows an example of a set of Evidence Variables organized into mastery categorizations (basic, developing, proficient) and categorized into different CT learning concepts such as looping, conditional, parallelization, and others. Past SAGE researches have also frequently cited Dr. Scratch (Moreno et al, 2015) and the Hairball plug-ins (Boe et al, 2013) to produce CT score from Scratch project analysis. These prior works provide good reference points when building SAGE's responsive and gameful assessment.

	1 -Basic	2 - Developing	3 -Proficient
Looks	Say, think.	Next Costume. Show. Hide.	Switch to costume. Set, change color/size/etc.
Sound	Play sound,note,etc.	Play vs Play until done.	
Motion	Move, goto sprite, point,turn.	Goto x,y. Glide to x,y.	Set,change X, Y.
Variables	Scratch variable (sprite, mouse pointer, answer, etc)	New variable (set,change)	New list.
Sequence & Looping	Sequence	Repeat, Forever.	Forever If. Repeat Until.
Boolean Expressions	Sensing operators.	<, =, >.	And, or, not.
Operators	Math	String and random	List
Conditional	If.	If... else....	Nested If/ If... Else...
Coordination	Wait.	Broadcast, When I Receive.	Wait Until.
User Interface Event	Green Flag clicked.	Key press, sprite clicked.	Ask and wait.
Parallelization	2 scripts start on same event.		
Initialize location	Set location properties (x,y,etc.) on green flag.		
Initialize looks	Set looks properties (costume, visibility,etc.) on green flag.		

Figure 1. PECT's Evidence Variables

Scratch

One example of a widely adopted gaming-for-learning application is Scratch. Scratch is a visual programming environment (see figure 1) inspired by the LOGO programming language (Maloney et al., 2010), where children are encouraged to create programming projects using block objects similar to LEGO blocks. Scratch can be used to create games and interactive media in ways that are welcoming and non-threatening to younger age audiences new to programming. Scratch provides the unrestrictive open-ended game design space that students respond well to (Resnick et al, 2009), allowing for self-paced explorations and personalization of virtual blocks that when linked correctly become executable programs that come alive and shareable with Scratch's active multimillion user-base. One noted limitation of scratch stems from its strength: the unbounded freedom to create makes it difficult for novices to organically develop more complex design patterns through understanding of advanced programming concepts (Xie & Zhong, 2016). In any learning endeavor, novice students typically would benefit the most from: 1) guidance from domain experts who are capable and effective in imparting knowledge and 2) cross-pollinating interactions with fellow motivated learners. As the student increases their domain expertise, the most effective blend between the two kinds of interactions would shift from initially more guidance heavy mirroring a student-teacher relationship to more peer-like interactions with other individuals in the same domain. The challenge is how to make these interactions traditionally seen in physical and organized classroom settings translatable in the context of an open-ended and online learning environment. In this report, we focus on building a responsive feedback mechanism that mimics the experience of having an effective guide/teacher sitting alongside the student as she works on Scratch assignments.

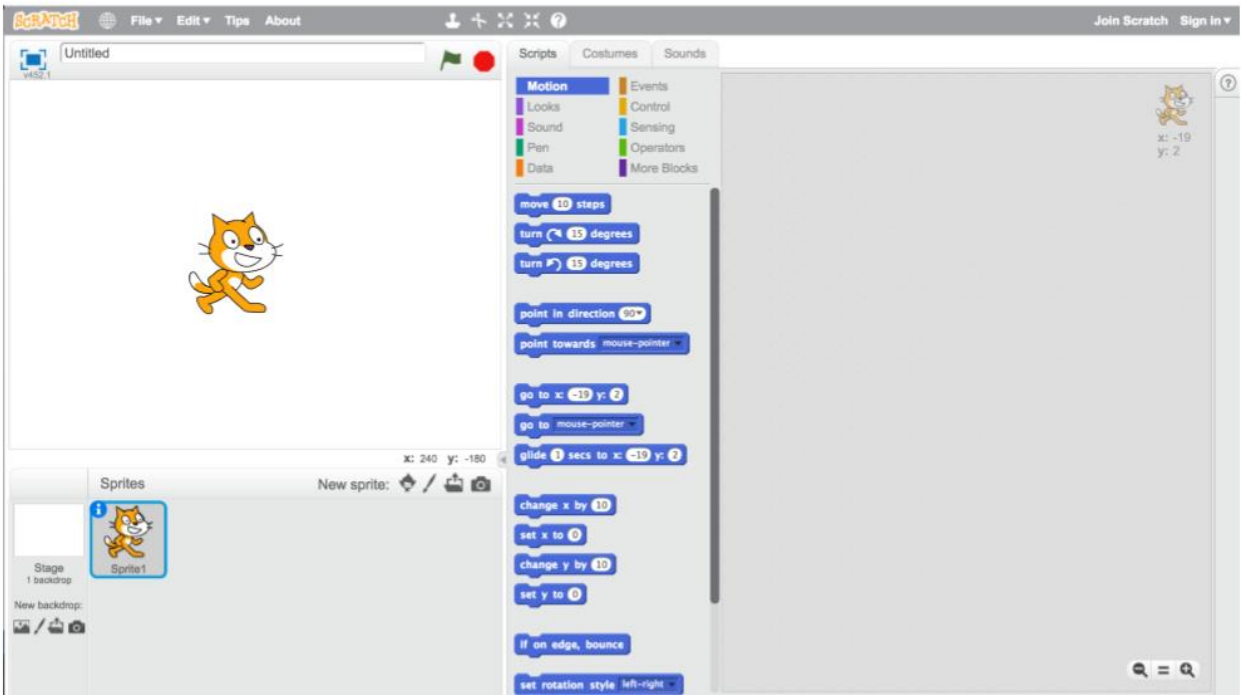


Figure 2. Scratch Interface

Past SAGE Research and Modules

Columbia University's Social Addictive Gameful Engineering (SAGE) project (Bender, J., 2014 & 2015) aims to leverage games' addictiveness and their allure for voluntary temporal and energy investments to infuse Computational Thinking within grade 6-8 curricula. In doing so, the construction-oriented Scratch programming language is determined as an ideal medium (Bender, 2015). Latest developments in data mining, data analytics, artificial intelligence, and machine learning have shown promising applications that could make it easier to not only replicate effective teaching strategies of real-world classroom, but also to revolutionize and innovate formal educational processes that have stayed mostly the same throughout the past decades. One of SAGE's aims is to provide integrated solutions leveraging findings in these technological knowledge areas to increase the effectiveness of Scratch. Figure 3 shows the different SAGE research projects that this current project builds upon, especially the work on formative SAGE assessments (Pava, 2016), gamification foundation (Zhang, Chien, & Tsai, 2016), front-end dashboard (Xie & Zhong, 2016), and integrated learning platform (Khandelwal & Mohanty, 2017). The project also aims to setup an interactive, personalized user experience medium that can deliver automated machine-generated intelligent hints to aid SAGE Scratch users (Anand & Sawyer, 2017). The previously built modules that have been produced by past SAGE researches as part of the research effort serve as excellent foundation for the current work (see Appendix B).

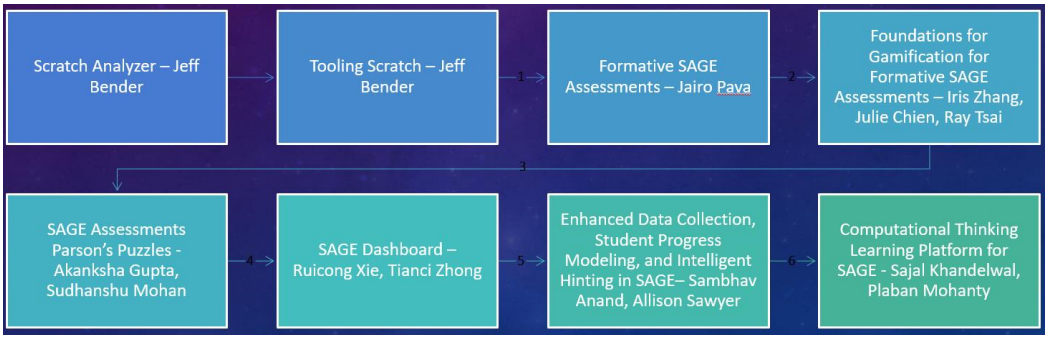


Figure 3. Past SAGE Research

Constructionism

Because wide-ranging constructionism is an essential part of Scratch, it is important to realize the unique characteristics of constructionistic games. Supplying a sufficient degree of freedom that encourages exploration for example, is an important prerequisite for a gameful implementation of Scratch-based curricula (Weintrop et al, 2012). It is also important to recognize and anticipate constructionist games' resistance to grading, ranking, and classifying children as bad/underachievers, because inherent in the idea of constructionism is that all students can engage in deep learning if the environment, tools, and facilitation are well-designed (Berland et al, 2014). On the data gathering spectrum, constructionist games do provide a wealth of unstructured data conducive for application of data mining techniques, much more so compared to traditional games (Berland, Baker, Blikstein, 2014). A non-intrusive and integrated assessment system will go a long way in cultivating the creativity enabled by Constructionism but still providing the external guiderails and support the maximize engagement, enjoyment, and stickiness.

Evidence-Centered Design

Despite all these potential gains from games that foster learning, proponents of game-based learning acknowledge more empirical evidence and curriculum-specific learning outcomes are needed to conclusively prove that games are an educationally effective solution (Kazimoglu et al, 2010). Devising an effective assessment strategy in measuring student's learning progress while using construction-oriented games thus is an area of importance. Frameworks such as evidence-centered design (ECD) are attractive (Mishlevy et al, 2016, Figure 4), because they aim to structure learners' performance data that are continuously collected from the game, utilizing machine-based reasoning techniques to make inferences about the learner's competencies resulting in a learner model (Rupp et al, 2010). It is argued that stealth assessment and ECD can assess not only content-specific learning but also general abilities, dispositions and beliefs, thus being more adequate to assess the so-called 21st century competences such as problem-solving skills, persistence, and creativity (Carvalho, 2017). This is important research that meshes well with the work done in SAGE Visual Assessment Editor, SAGE Affinity Space, and SAGE Intelligent Hinting System. The gameful approach of the SAGE user experience in the front-end also serves functions that are similar to the Presentation Model in the ECD framework.

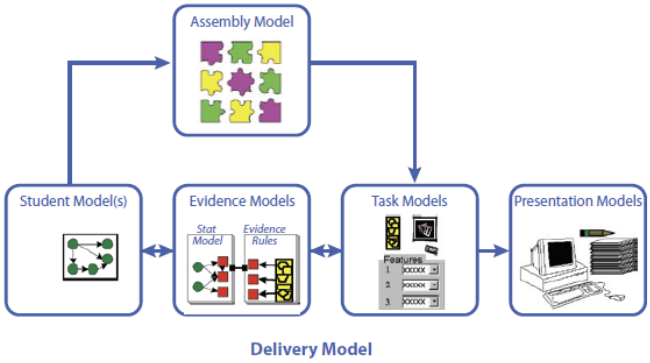


Figure 4. Schematic Representation of the Models in ECD Framework

ADAGE

ADAGE (Halverson, Owen, 2014) is another example of an open-source data collection and analysis framework that interprets real-time click-stream (telemetry) data from games and other digital learning environments into formative evidence of learning. These data can be used to identify patterns in play within and across players (using data mining and learning analytic techniques) as well as statistical methods for testing hypotheses that compare play to content models (Groff et al, 2015). Gamified elements such as units of learning, critical achievements, and boss level listed under the assessment mechanics in the Adage framework (Figure 5) are useful concepts when thinking about a gameful assessment system.

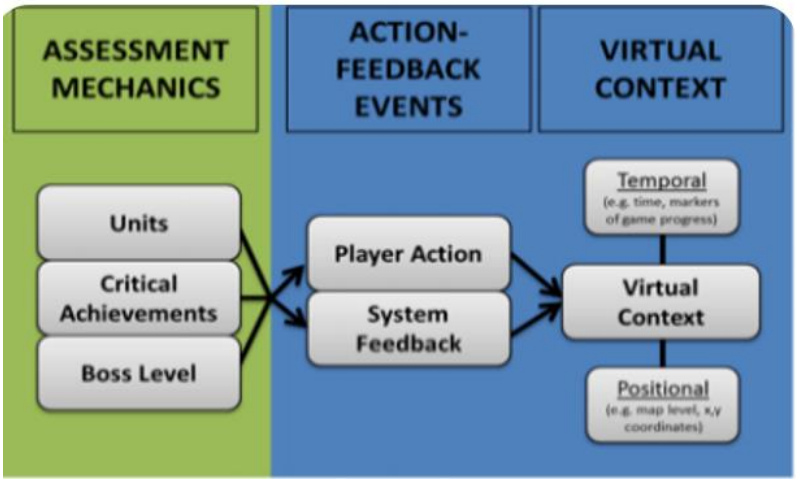


Figure 5. Adage Framework

Khan Academy

The web-based learning management platform Khan Academy (www.khanacademy.org) provides some inspiration as it serves as an excellent example of a tight and unified integrations of different front-end elements and the associated back-end functionalities in production mode. Its system of points, badges, progress tracker, and accomplishments progressions implements a functioning Gamified Accomplishments tied with persistent progress visualization and badge system that students can build upon session after session, reinforcing the importance of learning as a multi-layered and

scaffolding endeavor.

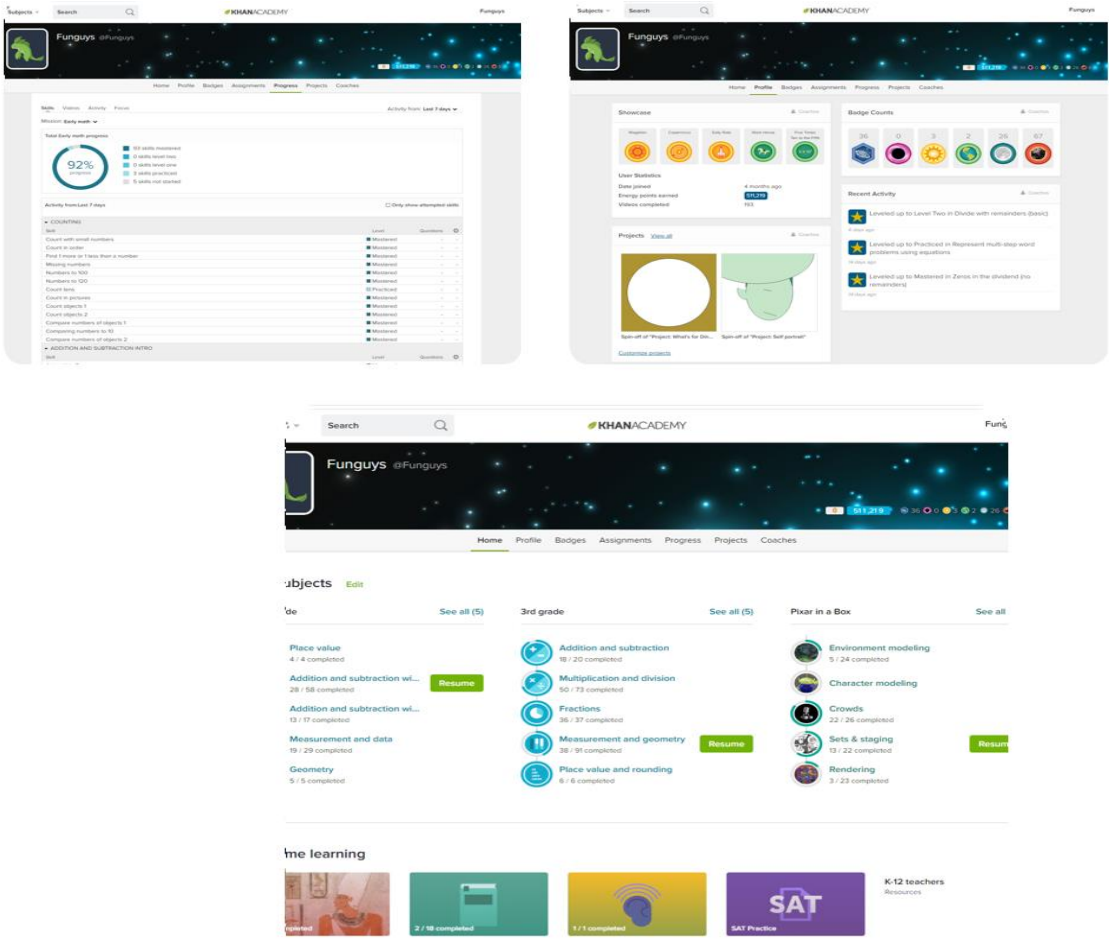


Figure 6. Khan Academy Dashboards

Project Planning

Approach

This project aims to increase Social Addictive Gamefulness during instillation of computational thinking, specifically through a personalized and individualized assessment system that provides real-time response to the students' activities and interactions with Scratch. Moreover, there is added awareness for a need to provide up-to-date, unifying structure on the conceptual and architectural level with a Gameful orientation. It was identified by past researchers that building a one-stop platform for SAGE that integrates all SAGE components under one web platform may lead to better engagement (Khandelwal & Mohanty, 2017), which is tied to an important future goal to evaluate how to use SAGE in experimentations, both in classroom settings (Anand & Sawyer, 2017) and within the Scratch community at large (Khandelwal & Mohanty, 2017). An analysis on the evolution of SAGE modules shows that there is a high degree of inter-connectedness and inter-dependencies between modules that necessitates the SAGE platform to be developed with a holistic, integrated-product paradigm in mind modules (SAGE Wiki, 2017, see Appendix B). Conducting experiments and collecting usage and experimentation data are

of special importance for the SAGE Machine Learning module, which utilized algorithms that self improves as the volume of usage data increases.

With this view in mind, the project is broken down into three stages:

Figure 7. Project Stages

Feature Porting

The first step towards enhancing SAGE's Gameful Assessment System involves the migration and refactoring of all assessment related features that exist in the various SAGE modules (especially the Go-based Assessment Server, the Visual Assessment Editor, the Node.js version of Assessment Server, and SAGE Affinity Space). A major part of this migration is the porting of assessment server functionalities from the prior Go language-based version to the current Node.js based version of the server. Successful completion of the port will re-enable a closed continuous feedback loop from the front-end, to the assessment server, and back to the front-end, similar to the flow in the ECD framework from the Student Model, to the Assembly Model, to the Task Model, and the Evidence Model (see figure 4 in the Related Work section). Part of the work in this first step involves a joint-effort with the concurrent work done by the Gameful Affinity Space team.

Platform Documentation and Terminology Review

Around the same time, a review will be conducted to evaluate, and when necessary refactor, the naming and structures of the various assessment objects that exist in these modules and repositories, with the objective to achieve cohesiveness and a one-platform feel throughout SAGE assessment components, that will also be compatible and comprehensible in the context of the various related research and findings from the academic literature that has influenced SAGE over the years. This information will be stored in the SAGE Wiki online knowledge repository for easy reference by current and future SAGE researchers.

Personalized Gameful Assessment

After this is accomplished, what comes next is the build of gameful features on top of the integrated platform. Of special interest is the build of effective presentation of instructional information and student motivators through a highly interactive, gameful, even addicting user experience. There will be modifications to the Assessment space in the student facing area for this purpose, where a personalized avatar will be shown, with a high degree of interactivity with the students as the students create using the Scratch user interface. This involves further utilization of the point systems throughout the play experience to relay to the students the various feedback from the game: persistent progress indicator, milestone reaching notifications, and the associated rewards and celebratory messages. In

addition to real-time communications at relevant points of the gameplay, these feedback occurrences summarized and displayed in a pleasing, meaningful, and gameful manner.

Milestones

The stages of the project were executed following the milestones below.

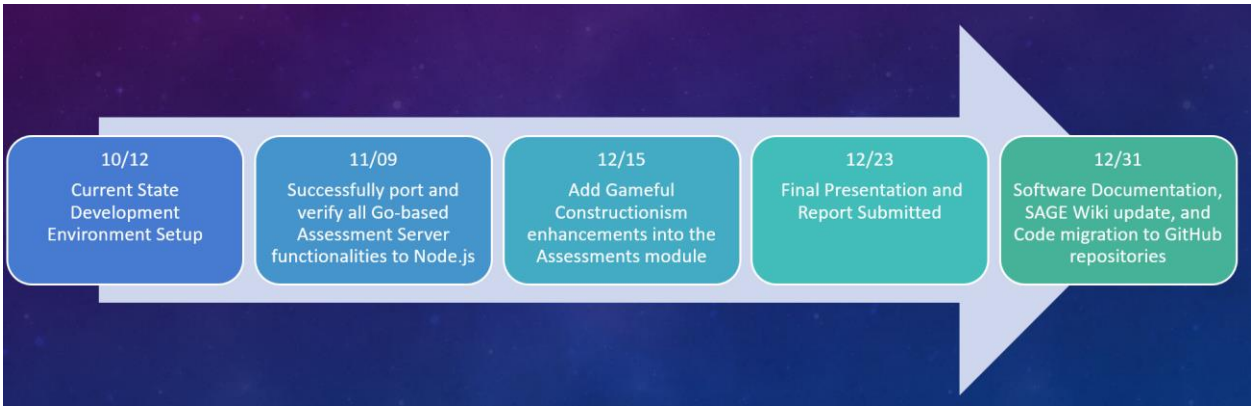


Figure 8. Project Milestones

Implementation

Architecture

Server and Database Components

SAGE's Assessment architecture consists of an Angular-based front-end served by a Node.js server that presents the SAGE Affinity Space to students, teachers, and researchers. The Front-end server periodically sends POST requests containing game information to the Node.js Assessment server, and also receives assessment results via a GET request which then get passed onto the student Affinity Space. Both the assessment server and the front-end server store MongoDB documents onto an mLab repository on the cloud.

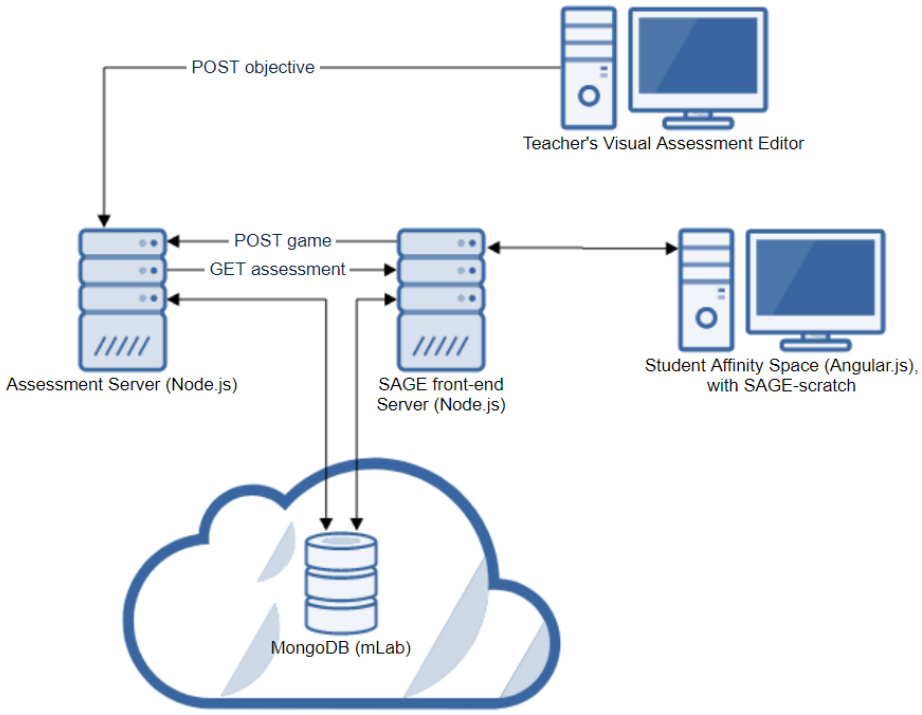


Figure 9. SAGE Architecture

Features

Porting the Assessment Feature from Go version to Node.js Version

This section describes the porting of models, services, and controllers from the Go version of the Assessment Server into the Node.js version. A necessary component and a logical starting point for the Assessment feature is a set of assessment statements for a specific game, now called a game "objective", that is created in the Visual Assessment Editor (Figure 10). New services and controllers are also created in the Assessment Server to handle the new and updated POST and GET requests as described below.

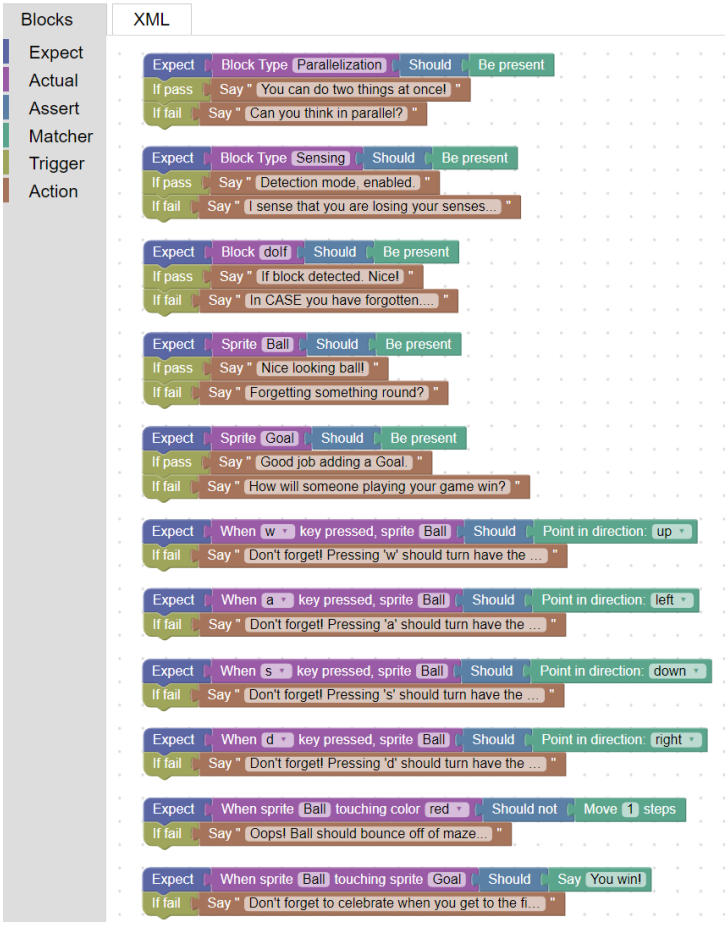
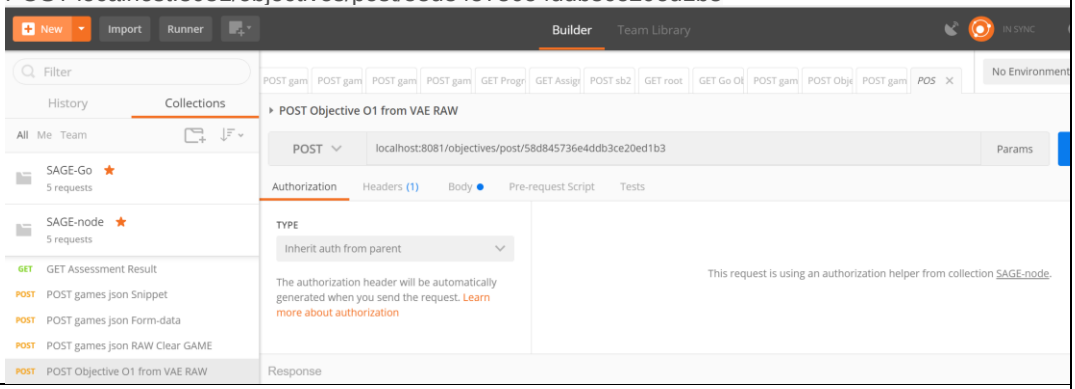


Figure 10. An example of a game objective in the Visual Assessment Editor

Objective

When a teacher completes an objective set for a game in the Visual Assessment Editor, the Editor then sends the information via a POST request to the Assessment Server.

Route	POST /objective/post/{objectiveID}
Example	POST localhost:8081/objectives/post/58d845736e4ddb3ce20ed1b3 
Response Type	200, on Success

Content-Type	application/x-www-form-urlencoded; charset=UTF-8

This information is handled by the Assessment server's "objective" controller/service mechanism, and then stored in an "objective" model, capturing the information in both the raw XML form (in the "objectiveXML" element) and a structured JSON form (as part of the "testcases" element as an array), as shown in Figure 11.

Figure 11. An example of an objective model as a MongoDB document

Game

Separately, as soon as a student starts a game in the Student Affinity Space, periodic POST requests from the Affinity Space start occurring every second, sending over the game information to the Assessment server.

Route	POST /games/student/{studentID}/game/{gameID}/objective/{objectiveID}
Example	POST: localhost:8081/games/student/sm4241/game/123/objective/58d845736e4ddb3ce20ed1b3
Response Type	200, on Success
Content-Type	application/x-www-form-urlencoded; charset=UTF-8

This information is stored by the Assessment server in a “game” model, after being handled by the “game” controller/service, as can be seen from a sample game document snippet in Figure 12.

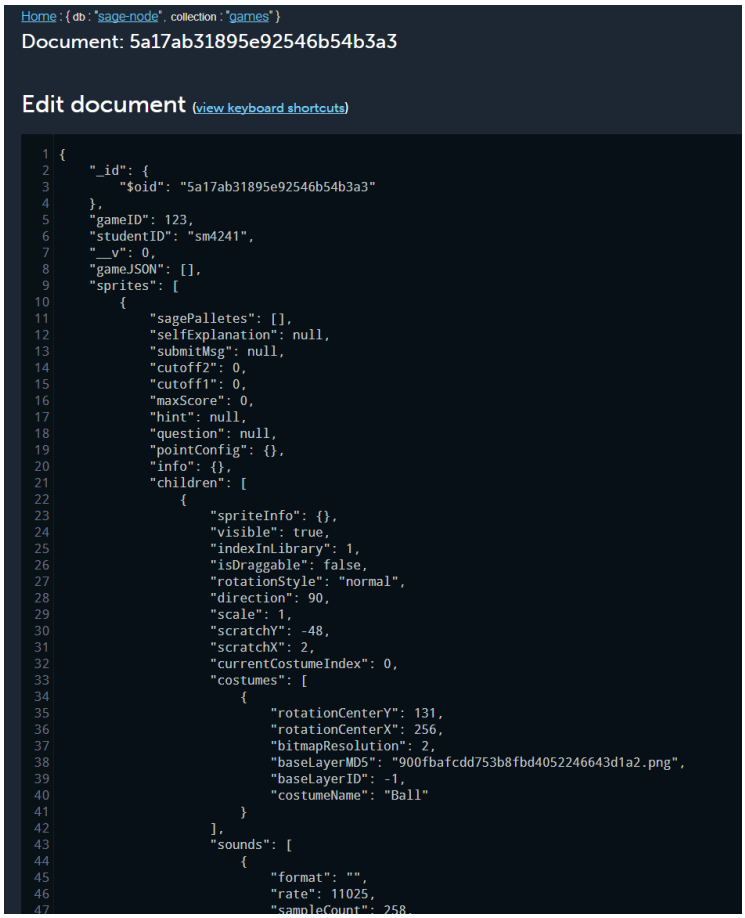


Figure 12. An example of a game model as a MongoDB document

Assess and Result

Once a game session is active in the Student Affinity Space, there is a periodic GET request sent to get an up-to-date assessment results from the Assessment server. The assessment server uses the “assess” controller/service mechanism to produce a result Object (Figure 13) as a response to this GET request. This result is the displayed in a gameful way in the Assessment area.

Route	GET /assess/game/{gameID}/objective/{objectiveID}
Example	GET: localhost:8081/assess/game/123/objective/58d845736e4ddb3ce20ed1b3
Response Type	200, on Success
Content-Type	N/A

Figure 13. A snippet from a sample result model in MongoDB

For further clarity, the Entity Relationship Diagram in Figure 14 shows some additional specifications of the three main Assessment models (game, objective, and result) and how they interrelate with each other.

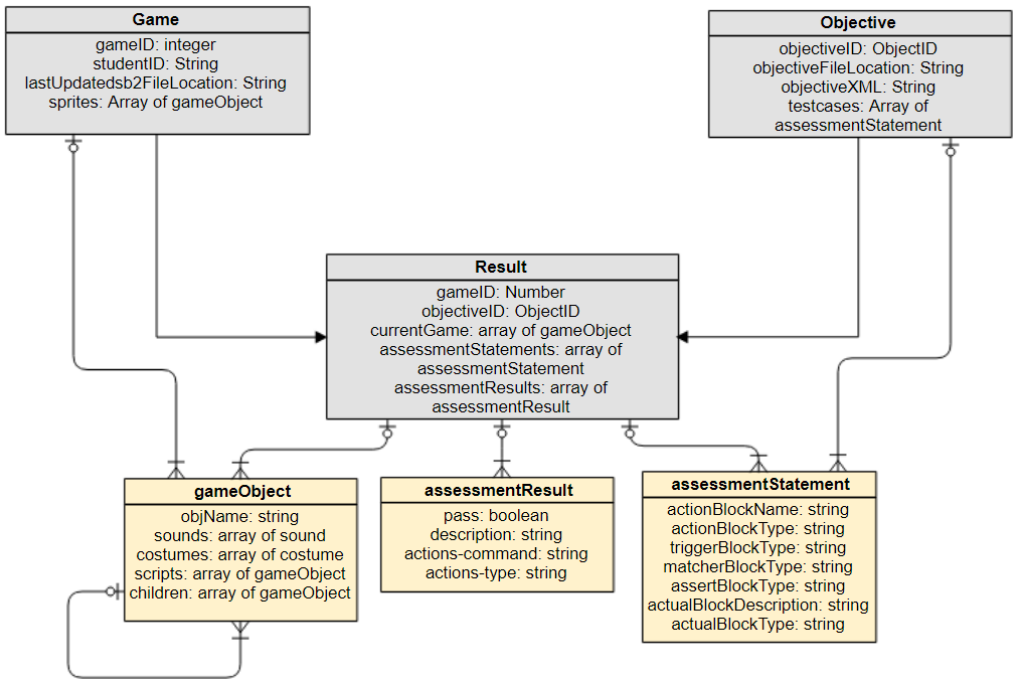


Figure 14. Assessment Entity Relationship Diagram

Platform Documentation and Terminology Review

An online wiki area for SAGE has been setup (see Appendix A) to serve as an organized reference source and that contains the following sections:

- **Getting Started**
  This section provides an up-to-date overview of the SAGE project, including a sub-section showing the evolution of SAGE modules and functionalities. This section also contains a step-by-step walkthrough to install, setup, and configure the various SAGE modules.
- **Application Features**
  This section describes the individual features of each SAGE models.
- **Workstreams**
  This section describes, organizes, and documents the work and research related to the different SAGE modules.
- **Theoretical Foundation**
  This section contains references, summary, and notes pertaining to a collection of academic papers and research related to the SAGE project.

One major goal of the wiki is to present the SAGE project and its different modules as one cohesive body of work. A deliberate effort is made to show how the different modules interact with each other, which will naturally highlight areas where there might be inconsistency in terminologies and code nomenclatures across modules. The decisions to refactor references of what now are known as “objective”, “game”, “assess”, and “result” for example, were made after finding some incongruency in the SAGE Assessment and the SAGE Affinity Space front-end modules. Past and future researchers are

encouraged to use the wiki as a handy reference for all things SAGE, and are also invited as active collaborators to maintain and enrich the wiki as SAGE continues to grow and evolve.

Gamified Assessment Presentation

Looking at the previous version of Student Affinity Space's Assessment user interface (see Figure x below), there seemed to be an opportunity to make the display of assessment results in a simpler and more gameful manner, partly by reducing text density and by re-arranging the presentation layout.

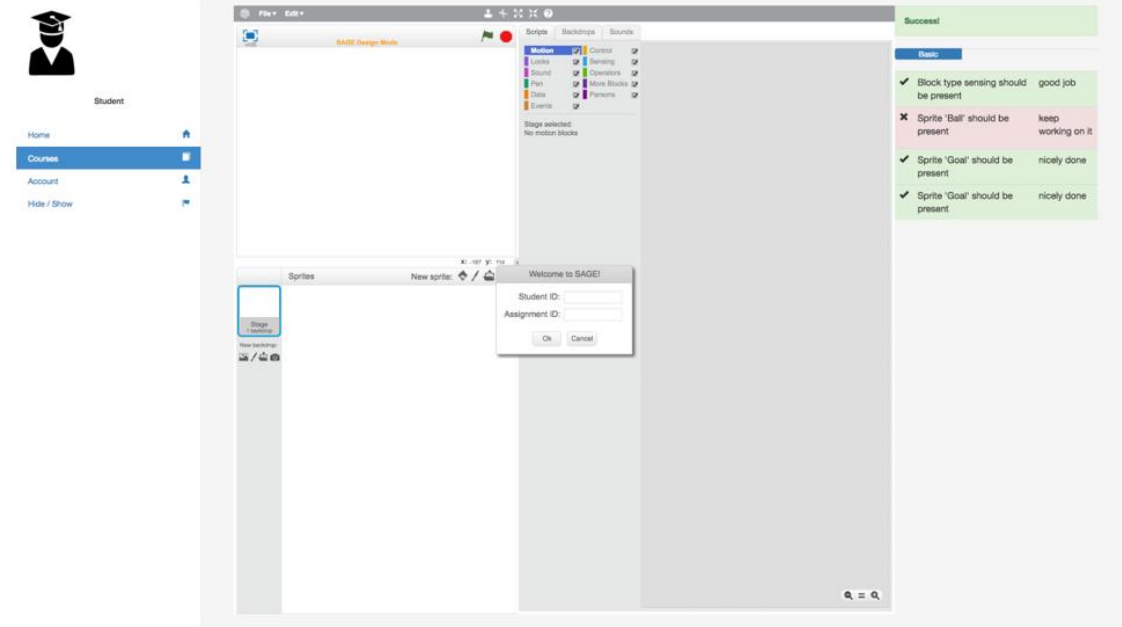


Figure 15. Prior version of Student Affinity Space

In the current updated version, information about assessment results are now grouped under a new Sage Orbs Found section (see Figure 16).

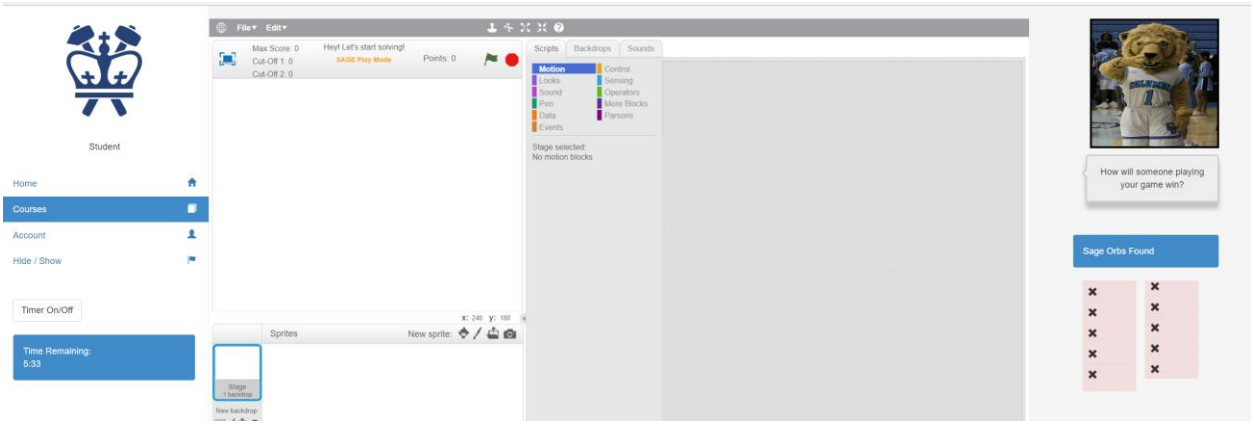


Figure 16. Updated Assessment Space Display

Every single assessment statement that is setup by teachers in the Visual Assessment Editor is now associated and gamefully represented by a new game concept called an Orb, which has two modes:

"Found" (depicted as a checkmark in a green box) or "Missing" (depicted as an X in a red box). A Sage Orb is "Found" when the student has passes an assessment statement. At the beginning of a game, the student is missing all her Sage Orbs, and intuitively is encouraged to collect all the orbs, after which the game will be completed. When a student manages to satisfy the conditions to pass an assessment statement, the periodic GET statement from the front-end server will assign a "pass" status for that particular statement, and graphically shows in the UI that the related Orb has been "Found". As part of the "assess" process of the Assessment server that services this GET request, the "command" variable in the "result" model (see again Figure 13) will be updated with a new sentence that provides motivational or accomplishment feedback for the student, available graphically as words in the chat bubble by a Game Character.

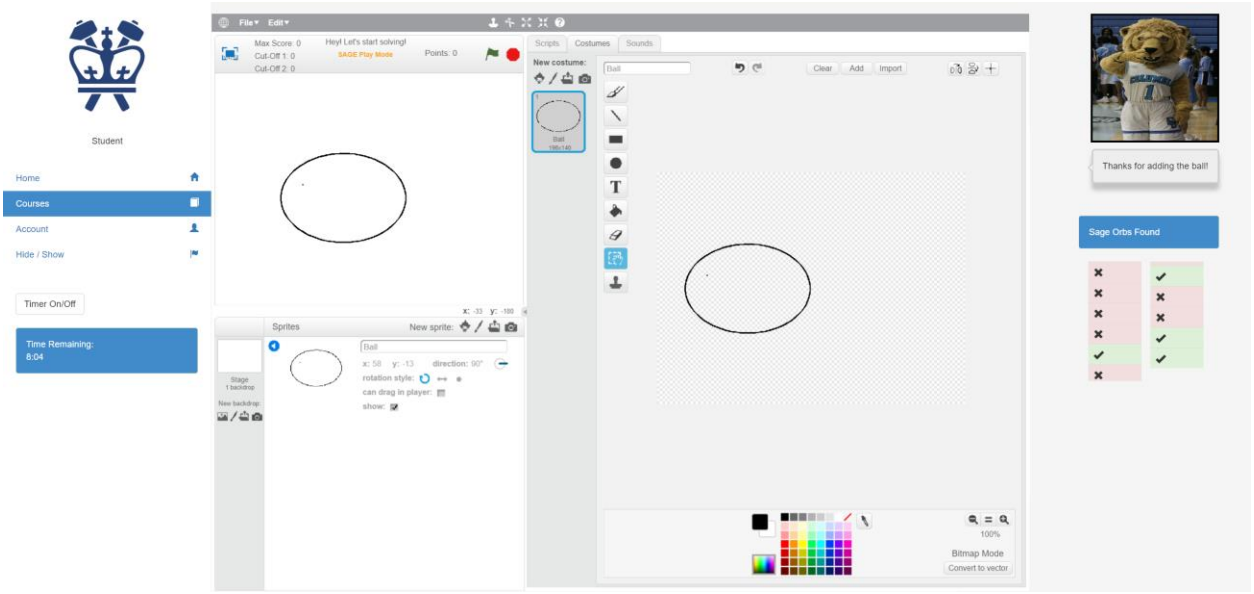


Figure 17. Assessment Space Display with new content in the Chat Bubble, and updated "Sage Orbs Found" section, when the Student successfully created a "Ball" sprite, satisfying an assessment criteria

Interactive Game Character and Dynamic Chat Bubble

The chat bubble then serves as a communication medium to engage students with customizable, dynamic, and periodically updated scripts from the VAL Editor. Future work in SAGE Machine Learning could use the same mechanism and medium to display real-time, dynamic feedback. Moreover, the chat bubble is one part of a greater cohesive game character design, created to bring a sense of game world immersion and simulated interactivity with a persona. The profile picture of this game character is yet another element in this design. In striving to have the program to assume some of the human teacher roles of providing guidance, feedback, and encouragement to the students, having a relatable and interactive character could go a long way in easing the transition and increasing students' acceptance of the role substitution. The presence of a felt-character allows possibilities for future Gameful and Addicting features, such as:

- Character animations, sounds, and message boxes that are interweaved into the gaming experience

- Use the character as a good foil to deliver the result of the back-end machine learning algorithms and Intelligent Tutoring features
- Utilization of storylines that can heighten the sense of knowledge progression through plot development, action, and conflict when integrated with SAGE's concepts of Missions and Quests

Game Timer

A Game Timer is also added to the Student Affinity Space to add another dimension to the gaming experience (Figure 18). This feature can be useful to induce a healthy level of "good" tension that is conducive to higher learning rate, creativity, and productivity. It also presents a new gaming challenge that could add Gamefulness through competitiveness, as students' strive to achieve their personal high scores when measured against a set time limit, and compare their achievements with other students. This information can later be socially displayed in personal and community high-scores in the Affinity Space, using time as a common constraint that establishes a universal benchmark. A time limit can also be useful in classroom settings that have set period lengths, as well as in future SAGE experimentations. It is also a helpful feature to mitigate the inherent weakness of Scratch as a constructionist construct; placing a time restriction to the open-endedness of Scratch can help sharpen students' focus in accomplishing game objectives handed out by their teacher. The additional usage and time-specific data provided by the timer also adds new data points and measurements that have applications for progress tracker, analytics and machine learning algorithm building purposes.

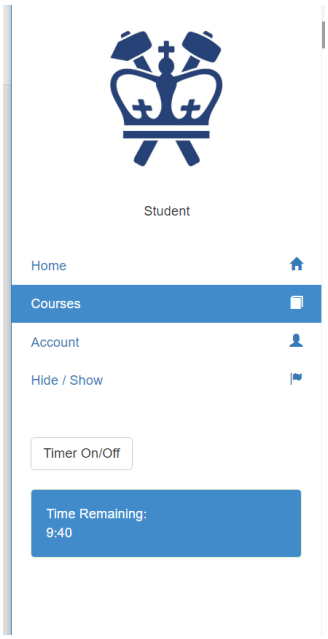


Figure 18. Game Timer

Limitations and Assumptions

At present, the pass-or-fail evaluation logic of the Visual Assessment Editor's assessment statements are fairly rudimentary, consisting of simple keyword searches within the game scripts. The evaluations could benefit greatly from a more robust and comprehensive evaluation logic on how a game should meet the assessment criteria. In addition, the Game timer functionality is currently quite static, as it is currently set at 10 minutes with a permanent placement in the Student profile area. In the future, this functionality could be improved if teachers could modify and customize the Game Timer settings, possibly through the Visual Assessment Editor.

Future Work

The following are some possible trajectories for future research and development:

Branching and Scaffolding of SAGE Learning Paths

There is a lot of potential to devise various learning paths to more formally guide students along their journey in learning computational Thinking. Curriculum designers can create an over-arching story, missions, quests, and games along with the accompanying assessment for every concept outlined in a Computational Thinking curriculum, such as sequences, loops, events, parallelism, conditionals, operators, data, and many others. To allow this, games should be able to be saved and be retrievable online, and teachers should be able to connect and integrate them with particular quests and missions.

Game-End Summary

A Game-end Summary visual can be generated at the end of every game, showing statistics and game play analysis to inform students and provide continuity and motivation to continue interacting with Scratch. This can include both Summary Feedback and a Task Level feedback. If the student completes the game, there will recommendations on what games to play next, further along in the student's learning path. If not all game objectives are completed in the form of collected Sage Orbs, there will still be a progress summary.

Game Boss Character, Game Enhancements, Story-driven Games and Story Enrichment

There is an interesting synergy between the intelligent tutoring module and the assessment module that allows for complex game characters that react dynamically based on the sensed preferences and usage behaviors of the students. This more dynamic approach can also allow for rich and engaging backstories to increase game addictiveness and Gamefulness as well as provide more content for discussion and reactions in the SAGE affinity Space. Existence or absence of optional features such as a time limit, can be creatively integrated into the story arch uniquely design for each game, allowing for numerous plot devices such as character development, tension, conflict, and resolution. When done effectively, these game enhancements have the potential to increase students' immersion, and placing and keeping students' in the zone of proximal flow (Basawapatna et al., 2013).

There is also opportunity to gamify the objectives further, for example by updating the tables based Sage Orbs section with actual sprites of Orbs or Jewels graphics, that can show visual elements

such as an empty orb case when an objective is still yet unfulfilled, and an actual representation of a jewel or an orb once the game has shown evidence of objective completion.

Leverage Usage Data for Insight Generation, Analytics, and Dashboard Reporting

The myriad of data from the time-serialized assessment results, gaming behaviors, timer effects, and observation of students' strategies in achieving full game objectives are useful input points to SAGE's data analytics modules and can be collected into relevant students' progress trackers and learning metrics. As students' continue to use SAGE, the wealth of metric information can yield insightful pattern when analyzed in aggregate, in groups, and by individuals through time-series analysis. Some of these data can be summarized and presented in dashboard form to students, teachers, and researchers. In a similar vein, progress points tracking and summary can then be displayed on the appropriate aggregation levels at appropriate points during students engagements with the Games, Missions, and Questions.

Control Options in the Visual Assessment Editor and Researcher Dashboard

More game controls can be added into the Visual Assessment Editor, such as timer controls, and other teacher-friendly settings such a Constructionism Granularity Slider to control the level of intelligent hinting the teachers intend SAGE to provide to the students. Similar game control features could also be introduced to teachers and researchers to setup A/B Testing capabilities that will aid in experimentation of changes within various SAGE Features. The editor can also be more integrated into the front-end, allowing for a more unified, one-stop place for SAGE usage, which will also contribute towards engagement and stickiness in SAGE Affinity Space.

Uniform Terminology and Code Nomenclature for SAGE as a Whole

Building from the recently completed work on establishing uniform assessment terminology and code nomenclature, future efforts can strive to achieve and maintain the same uniformity and one-product feel throughout the rest of SAGE modules. This endeavor should be seen as a continuous iterative practice as opposed to a one-off task given the highly fluid and dynamic nature of SAGE research and development.

Conclusion

SAGE has grown rapidly in the past several years and has added and evolved distinct modules to realize a fun, engaging, social, comprehensive, and highly effective educational solution to teach complex and abstract Computational Thinking concepts. This exciting and impactful vision is getting closer and closer to fruition, and this project hopes to help SAGE get one step closer by increasing cross-module integration, shared nomenclature, and cohesiveness by building and showing an end-to-end objective-game-assessment process loop that can be leveraged further by future SAGE researchers and developers.

To help future students, learners, and teachers, this project also seeks to improve SAGE's user experience through the introduction of several gameful user interface elements:

- A gamified, streamlined and more visual presentation of assessment progress and results via the Sage Orbs representation
- Creation of a Personalized Interactive Game Character, with dynamic chat bubbles to guidance and motivation for students to stay in the zone of proximal flow, which also serves as delivery mechanism for the back-end results from the machine-learning based intelligent tutoring module
- A Game Timer to achieve heightened sense of good tension, that can also provide new dimensions of usage data for further processing by SAGE's data analytic modules

Preparing the 21st century workforce for jobs that rapidly necessitates ease, comfort and mastery of notoriously daunting and difficult Computational Thinking remains one of the most important and urgent challenges facing educators today. SAGE has tremendous potential to fill this need through its unique game-based solutions, and with excited anticipation I look forward to witness and be a part of SAGE's continual progression in making strategic impact and contributions benefiting many students who are part of the young generations to come, and the ensuing positive change in the society that these students will in turn be influencing.

References

Anand, S., Sawyer, A., 2017. Enhanced Data Collection, Student Progress Modeling, and Intelligent Hinting in SAGE. Columbia Programming Systems Lab. Columbia University, New York, NY

Basawapatna, A. R., Repenning, A., Koh, K. H., & Nickerson, H. (2013, August). The zones of proximal flow: guiding students through a space of computational thinking skills and challenges. In Proceedings of the ninth annual international ACM conference on International computing education research (pp. 67-74). ACM.

Bender, J., 2014. Scratch Analyzer: Transforming Scratch Projects into Inputs Fit for Educational Data Mining and Learning Analytics. Columbia Programming Systems Lab. Columbia University, New York, NY

Bender, J., March, 2015. Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula. Columbia Programming Systems Lab. Columbia University, New York, NY

Berland, M., Baker, R.S. & Blikstein, P., 2014. Educational Data Mining and Learning Analytics: Applications to Constructionist Research. Technology, Knowledge and Learning, Volume 19, Issue 1-2, pp 205-220.

B. Boe et al., 2013. Hairball: Lint-inspired static analysis of scratch projects. *Proceeding of the 44th ACM technical symposium on Computer science education*

Carvalho, M. B., 2017. Serious games for learning : a model and a reference architecture for efficient game development. Eindhoven: Technische Universiteit Eindhoven

Groff, J., Clarke-Midura J., Owen, V.E., Rosenheck, L., Beall, M., 2015. Better Learning in Games: A Balanced Design Lens for a New Generation of Learning Games

Kazimoglu, C., Kiernan, M., & Bacon, L., 2010. Learning introductory programming through the use of digital games in higher education. Paper presented at the Game Based Learning Conference.

Khandelwal, S., Mohanty, P., 2017. Computational Thinking Learning Platform for SAGE. Columbia Programming Systems Lab. Columbia University, New York, NY

Kickmeier-Rust, M. D., Albert, D., 2010. "Micro-adaptivity: protecting immersion in didactically adaptive digital educational games," *Journal of Computer Assisted Learning*, vol. 26, no. 2, pp. 95-105

Koh, K. H., Basawapatna, A., Nickerson, H., Repenning, A., 2014. Real Time Assessment of Computational Thinking, IEEE International Symposium on Visual Languages and Human-Centric Computing, Melbourne, Australia

Halverson, R. & Owen, V.E., 2014. Game-Based Assessment: An Integrated Model to Capture Evidence of Learning in Play. International Journal of Learning Technology, 9(2), 111–138

Kickmeier-Rust, M. D., and Albert, D., 2010. "Micro-adaptivity: protecting immersion in didactically adaptive digital educational games," *Journal of Computer Assisted Learning*, vol. 26, no. 2, pp. 95-105

Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E., 2010. "The scratch programming language and environment," ACM Transactions on Computing Education, vol. 4, Oct. 2010

Mislevy, R., Haertel, G., 2006. Implications of Evidence-Centered Design for Educational Testing. *Educational Measurement: Issues and Practice,* volume 25, issue 4, Blackwell Publishing.

Moreno-Leon, J., Robles, G., Roman-Gonzalez, M., 2015. Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking.

Pava, J., 2016. Formative SAGE Assessments. Columbia Programming Systems Lab. Columbia University, New York, NY

Resnick, M., et al, 2009. Scratch: programming for all. ACM 52, 11 (November 2009), 60-67

Rupp, A., Matthew G., Mislevy, R., Shaffer, D., 2010. Evidence-centered Design of Epistemic Games: Measurement Principles for Complex Learning Environments, JTLA, Vol 8, No 4

SAGE Wiki. (2017, December). Columbia Programming Systems Lab. Retrieved December 2017, from SAGE Wiki: https://gudangdaya.atlassian.net/wiki/spaces/SAGE

Schell, J., 2008. The art of game design: A book of lenses. San Francisco, CA: Morgan Kauffman

Seiter, L. and Foreman, B., 2013, August. Modeling the learning progressions of computational thinking of primary grade students. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 59-66).ACM

Xie, R., Zhong, T., 2016. Sage Dashboard: Final Report. Columbia Programming Systems Lab. Columbia University, New York, NY

Weintrop, D., Holbert, N., Wilensky, U., and Horn, M., 2012. "Redefining constructionist video games: marrying constructionism and video game design," in *Proceedings of the Constructionism 2012 Conference*

Wing, J. M., 2006. "Computational thinking," Communications of the ACM, vol. 49, no. 3, p.33

Zhang, I., Chien, Y., Tsai, T., 2016. Foundations for Gamification of Formative SAGE Assessments. Columbia Programming Systems Lab. Columbia University, New York, NY

Appendix A: SAGE Wiki

An online wiki has been setup to store various information and documentation related to the SAGE project. Online access to the wiki or PDF documentation is available for project members.

Appendix B: SAGE Modules

Evolution of SAGE Modules and Functionalities

	2014	2015	Spring 2016	Fall 2016	Spring 2017
Visual Assessment Server and Database https://github.com/cu-sage/sage-node			v1.0: Based on Go. New functionalities: • Manages the storage of assignments, assessments, and assessments results • execute assessments against assignments • evaluate assessment results • Plug-in enabled • Initialize MongoDB data structure/collections	v1.1: • Migrate v1.0 functionality from Go to Node.js • Decommission Go version • Allow saving over and retrieval of existing assignment • Plug-in enhancements through API endpoints expansions and improvements • Updated MongoDB data structure/collections: assignments, quest, assessments, students, teachers classes	
Scratch Analyzer https://github.com/cu-sage/scratch-analyzer	v1.0: • Created Scratch Analyzer, that distills Scratch projects into inputs fit for educational data mining (EDM) and learning analytics (LA) with its three components: Scratch Extractor, Scratch Dispatcher, Scratch Traverser				v1.1: • Add Behavioral Data Processing ability to parse timestamped JSON files to compare successive files and identify any changes made
Visual Assessment Editor https://github.com/cu-sage/sage-editor/tree/master/app/editor			v1.0: Based on Blockly (JavaScript). New functionalities: • Creates assessments that verifies presence of expected student behavior in Scratch Editor • Enable manual uploads to the Assessment Server	v1.1: • Add the ability to edit in-flight assessment • Add capability to assign points for particular actions using conditional • Added Quest - a progressive collection of connected assignments (Note:Instructor area to design quests and assignments is proposed and conceptualized but not implemented)	
Scratch Editor (Formative SAGE Assessment branch) https://github.com/cu-sage/sage-scratch/tree/block-ids		v0.1 New functionality: • SAGE Design and Play modes • Palette restriction system • Block restriction system SAGE learning principles and design tactics identified and documented (Tooling Scratch)	v1.0 New functionality: • Added compatibility with new Visual Assessment Editor module	v1.1: • Periodic auto-upload to the Assessment Server • Enabling/disabling blocks per feedback from assessment server • Assign a point value for each Scratch block • Make available block-based scoring system in Scratch Play Mode • Allow configuration, saving, and loading of Block Point values in Scratch Design mode • Added teachers capability in Scratch Design mode to create Parson programming puzzles • Added student's capability to solve Parson programming puzzles in Scratch Play mode	v1.2: • Automate the collection of additional user behavioral data; these data were then distilled into features that could be leveraged by future machine learning algorithms • Unique ID is now assigned for every individual Scratch Block • Add *latestBlock* variables to keep track of latest block manipulated • Introduces Highlighted Palettes and Shaking Blocks that are automatically activated in by the intelligent hinting system
SAGE Affinity Space (formerly Scratch Dashboard) https://github.com/cu-sage/sage-frontend/tree/master			v1.0. New functionalities: • Release of Assessment Space • Two user types available: Student and Instructor	v1.1: • Added login and register functionality for the Student and Instructor Spaces • Student Space displays Student Account, Student Overview, and Student Tasks • Students can track their courses progress in overview tab • Students can upload personal avatar • Students can track homework assignments scores and class average • Students can assess assignment using hairball mastery to track their progress over time through seven computational thinking attributes (preliminary implementation) • Badges system helps motivate students (requires backend implementation) • Instructor Dashboard displays Account, Overview, and Upload Video	v1.2: • Added a third user type: Researcher, that can activate and de-activate "clustering & hint rule mining" • Make available three machine learning algorithms that generates hints in Scratch Editor: K-medoids clustering, K-means clustering, Generalized Sequential Pattern mining • Instructor Space completed and enhanced with Course Creation, Assignment Creation, and Learning Paths • Added Course recommendations • Integrated Student Dashboard with VAL and Scratch Editor • Hairball implementation enhanced and activated
SAGE Continuous Integration and Delivery Tools			• Utilized Jenkins	• Utilized ESLint, Gulp.js, Mocha, Chai for integration tasks	

SAGE Modules Source Code

	Github Link	Technology and Language
SAGE Assessment Server (legacy)	https://github.com/cu-sage/sage	Go and MongoDB
SAGE Assessment Server (Current Node.js Version)	https://github.com/cu-sage/sage-node	JavaScript, Node.Js and MongoDB (mLab)
SAGE Front-end (Affinity Space)	https://github.com/cu-sage/sage-frontend	JavaScript, Bootstrap, AngularJS, MongoDB-based mLab
SAGE Scratch Editor	https://github.com/cu-sage/sage-editor	JavaScript
SAGE Visual Assessment Editor	https://github.com/cu-sage/sage-editor/tree/master/app/editor	Blockly
Sage Machine Learning	https://github.com/cu-sage/sage-frontend/tree/master/machine_learning	RapidMiner
SAGE Scratch	https://github.com/cu-sage/sage-scratch	ActionScript