In Search of Learning: Facilitating Data Analysis in Educational Games

Erik Harpstead, Brad A. Myers, and Vincent Aleven
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{eharpste,bam,aleven}@cs.cmu.edu

ABSTRACT
The field of Educational Games has seen many calls for added rigor. One avenue for improving the rigor of the field is developing more generalizable methods for measuring student learning within games. Throughout the process of development, what is relevant to measure and assess may change as a game evolves into a finished product. The field needs an approach for game developers and researchers to be able to prototype and experiment with different measures that can stand up to rigorous scrutiny, as well as provide insight into possible new directions for development. We demonstrate a toolkit and analysis tools that capture and analyze students' performance within open educational games. The system records relevant events during play, which can be used for analysis of player learning by designers. The tools support replaying student sessions within the original game's environment, which allows researchers and developers to explore possible explanations for student behavior. Using this system, we were able to facilitate a number of analyses of student learning in an open educational game developed by a team of our collaborators as well as gain greater insight into student learning with the game and where to focus as we iterate.

Author Keywords
Educational Games, Analysis of Learning, Toolkits, Logging.

ACM Classification Keywords
H.5.2 User Interfaces: Evaluation/methodology; K.3.1 Computer Uses in Education: Computer-assisted instruction (CAI); K.8.0 Personal Computing: Games.

INTRODUCTION
Video games have become a compelling medium for instruction. Games possess many potential benefits for education including their ability to motivate students as well as provide them with authentic environments to practice new skills with minimal personal risk. This has led education researchers to begin to establish a field of educational games studies [9,10]. Despite this glowing potential of games for education, many researchers demand that more rigor be applied to the methods of proving educational effectiveness within games [10,21]. One of the challenges in meeting this demand for rigor is a lack of generalizable methods for measuring learning within educational games. While many methods have been proposed [8,14,18,19], it remains difficult for game developers to know which method of measurement to use and how to integrate particular method into their games during the process of iterative development. Additionally, because games often incorporate novel student activities for which there are no well-established existing measurement methods, the measures often need to be developed along with the game, also in an iterative fashion. These challenges are compounded if a researcher or developer wants to apply multiple methods of measuring learning to their game, as each method may require a slightly different perspective of the player's behavior. To address these challenges, we have developed a toolkit for capturing and analyzing student behavior within open educational games that can support the application of multiple methods of measurement and can also be used to address questions beyond whether learning is observed from the game. This approach was developed as part of a larger collaborative effort between game designers and researchers to develop a collection of educational games to teach basic physics concepts to young children, the first of which is RumbleBlocks [5] (see Figure 1).

Figure 1. RumbleBlocks is a game where players must construct a tower high enough to reach the alien while also covering all of the blue energy balls.

CHI 2013, April 27–May 2, 2013, Paris, France.

The contributions of this paper are:

- A novel approach to recording and analyzing students' performance in educational games, which uses introspection of the properties of re-created game states to facilitate analyses of player learning in the game. This approach can be used both as a formative analysis during iterative development to improve the game, and after the game is deployed to evaluate the students' learning.
- A toolkit, implemented in C# and Unity (unity3d.com/), which supports this approach, and has been used to record and analyze logs for 174 children playing the RumbleBlocks game. This toolkit can be used to answer many different questions posed by researchers and educators about the children's learning and performance.
- A collection of measures for the RumbleBlocks game derived from re-created game states which other researchers have demonstrated successfully measure learning of physics principles in 5-8 year olds.

RumbleBlocks

RumbleBlocks (Figure 1) is an educational game designed to teach basic structural stability and balance concepts to children in kindergarten through grade 3 (5-8 years old). Its main focus is on three basic principles of stability: objects with wider bases are more stable, objects that are symmetrical are more stable, and objects with lower centers of mass are more stable. These principles are derived from goals outlined in the National Research Council's Framework for New Science Educational Standards [7] and other science education curricula for the target age group.

The game follows a sci-fi narrative where the player is helping a group of aliens who are stranded across four planets when their mother ship is damaged. Each level consists of an alien stuck on a cliff with their deactivated space ship lying on the ground. The player must use blocks to construct a tower that is tall enough to reach the alien on the cliff. Additionally, they must make sure that their tower covers a series of blue "energy balls" so that the spaceship can have enough power to take off. Once the player is confident about their tower, they can place the spaceship on top, which triggers an earthquake. If, after the earthquake ends, the tower is still standing and the spaceship is still on top of it, then the player progresses to the next level. If the tower or spaceship falls, then the player has to start over from the beginning of the level.

Each set of levels in RumbleBlocks is designed to focus on a different principle of stability. The targeting of different principles is accomplished mainly though level design. The energy balls can be used to both scaffold and limit the students' solution to a level, forcing them to prioritize one principle over another. However, even with this scaffold design, there are an unknown number of possible valid solutions to any given level because the earthquake mechanic relies on the dynamics of a real-time physics engine to evaluate the student's structure. That is, even though the level designer may intend for a particular tower design to be the solution, other designs may also work (see Figure 2).

RumbleBlocks is built within the Unity game engine. Unity is a popular game engine and development environment that is designed to allow game developers to rapidly prototype and iterate on game ideas by providing a suite of tools that automate many common game development tasks. One of these tools is the built-in physics engine, NVidia's PhysX engine, used to simulate realistic rigid body physics. This allows developers to create an accurate physical environment with much less effort. RumbleBlocks takes advantage of this feature to simulate the interactions of the blocks and the spaceship within the game. This means that the spaceship and blocks will respond to gravity and collisions as if they were real objects.

RumbleBlocks falls into a class of games that Spring and Pellegrino referred to as *open games* [19]. Open games are characterized by having a set of overarching rules that gov-

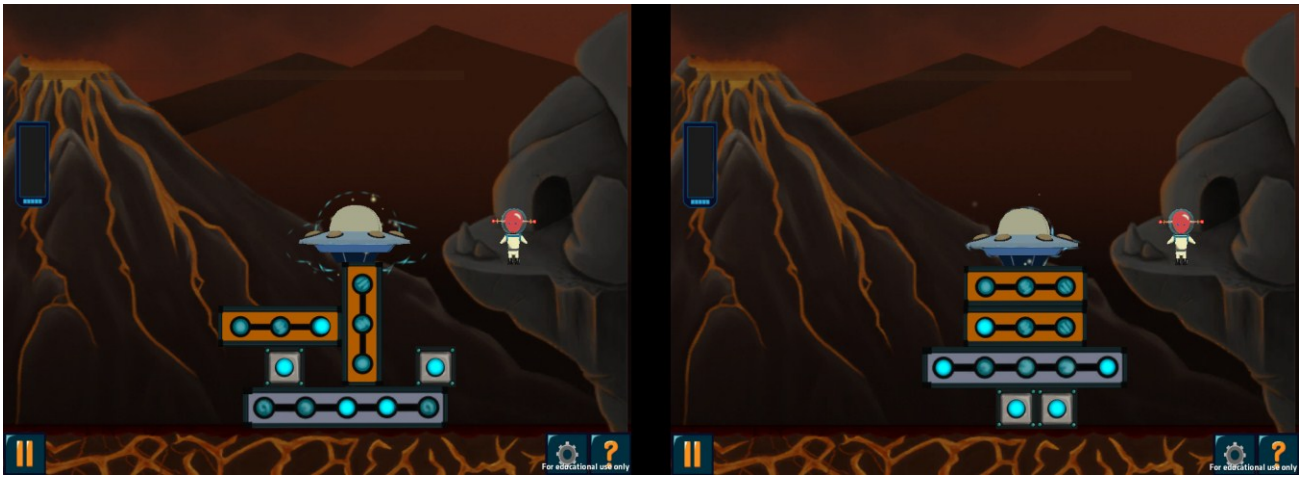

Figure 2. Two example solutions to a level in RumbleBlocks. Both solutions were successful, but knowing only that, it is hard to tell which principles each student understands.

ern the interactions of the game's elements, as well as a non-restrictive structure that allows players to explore multiple paths to a solution. These elements make open games attractive from an educational perspective because they allow students to gain lots of practice in an authentic setting that carries few substantive consequences for failure, all tenants of Gee's principles of learning in games [9]. This authentic setting, however, can make it difficult for designers to know whether or not a student's behavior demonstrates understanding of a specific piece of target knowledge. For an illustration of this issue in RumbleBlocks, refer to Figure 2. In this example, the level is designed to target the principle of structures with a low center of mass are more stable. Both student answers were successful after the earthquake and passed the level, but it is not readily clear how well either student has grasped the target principle. The tower on the right represents the answer envisioned by the level designer; however, the tower on the left actually has a lower center of mass.

Measuring Learning in Games

Educational games are ultimately judged by their capacity to show transferable learning outside of the game environment. Given how important evidence of learning is, it benefits educational game developers and researchers to have methods for measuring student learning from gameplay. The only way to truly claim that a game results in student learning is to conduct controlled lab or classroom experiments where students' improvement in understanding can be measured external tests. These kinds of evaluations, at least in their most rigorous form, are best saved for the end of the development cycle due to their cost in time and labor. The truly ideal situation is to develop measures for learning that can be evaluated within the game. While these measures also require grounding through controlled studies that measure out-of-game transfer of learning, it can be helpful to develop and iterate on them along with the game. Once validated, in-game measures allow the game to serve as an assessment in itself, opening up avenues for large scale data collection such as those used by Stamper *et al.* and Anderson *et al.* [2,20].

A number of methods for measuring within-game learning from student play exist in the literature. One simple method is to measure how many levels a player passed, or, relatedly, recording the last level the student completed. The benefit of this approach is that it is very straightforward and relatively easy to implement. Delacruz, Chung, and Baker used a measure of last level beaten when they found validity evidence for a game to be used as a standalone assessment of student math ability [8]. A single level success metric was also used by Andersen *et al.* [2] in a study of the effects of different tutorial styles on gameplay learning, i.e. how well a player was learning to play a game. This approach works well when the mechanics of gameplay are well-aligned to the target content being taught, and in the case of Andersen *et al.* they are the same thing. However, it cannot always be assumed *a priori* that gameplay and target knowledge are well aligned. An example of poor alignment came up during the early stages of developing RumbleBlocks. During playtests it was found that some students' towers were falling against the alien's cliff, which prevented them from truly falling over, and allowed players to succeed at levels they maybe should not have. This had the effect of providing players with inconsistent feedback on their constructions. If a player who uses a poor strategy passes a level that they should not have, they will never be required to reflect on the faults in their design, and thus will miss out on an important learning opportunity. To address this problem, the game designers moved the cliff farther from the construction area in an attempt to bring the game's feedback into better alignment with its educational goals.

A further limitation of using success per level as an indicator of learning is that it is very coarse-grained. This approach might work well when the learning task within each level is relatively straightforward, and does not involve a wide range of knowledge components, so that it is straightforward to attribute failure on a level to specific missing or incorrect piece of knowledge. However, games do not always have that characteristic. In fact, one reason why games are attractive for purposes of learning and assessment is that they can present learners with tasks that are more complex than those typically found in classrooms or paper-and-pencil tests.

A number of more complex methods also exist in the realm of measuring learning in games. One such method is the use of Evidence Centered Design (ECD) [16]. ECD relies on three major components: a competency model, an evidence model, and a task model. The competency model describes the target knowledge for the system. The evidence model analyzes a student's interactions with a system based on a researcher-defined rubric for scoring. The evidence model also expresses a relationship between the competency model and the observed scores. Finally, the task model provides a framework for describing the tasks a learner performs within the system. One of the benefits of ECD is that it does not require direct instrumentation into a game; however, this comes at the expense of having to analyze video recordings of player performance. Shute and colleagues have applied the ECD process successfully to existing commercial games [18].

Another complex method, drawn from work on intelligent tutoring systems, is Bayesian Knowledge Tracing (BKT) [6]. BKT tries to infer a student's knowledge state for a set of knowledge components based on their performance. It does this by assigning every action in a system to different knowledge components that can be correctly or incorrectly performed. It then takes the sequence of correct and incorrect student actions and runs them through a Bayesian Network to determine the probability that the student understands each knowledge component based on the observed pattern. Manske and Conati have used similar Bayesian

network analysis to measure student learning within an educational mathematics game [14].

A further method from the intelligent tutoring literature is the use of empirical learning curves to plot the performance of students [15]. This method assigns student actions as correct or incorrect in terms of a particular piece of target knowledge, similar to BKT. It then plots the error rate of each knowledge component on each successive opportunity to apply the knowledge component. If a student is learning the content (i.e., the targeted knowledge component) being measured then the learning curve will follow the shape of a power curve, following the "power law of practice" [17]. Formulating student performance this way allows for comparison between systems based on the slopes of their respective curves. Baker and colleagues have applied learning curve analysis as a way of measuring the increase in fluency of players' ability to judge divisors of numbers in an educational math game [3].

Each of these methods requires a different formulation of data, a different perspective of student performance, and a different grain size of examination. When looking at raw performance within a game, such as the rate at which the student passes levels, or how many levels they manage to beat, the analysis is concerned with the results of an entire level, whereas BKT requires information about individual student knowledge components, and ECD and learning curve analysis can work at multiple levels. Data for one method is not generally compatible with another without translation, forcing designers to choose a particular direction. Our approach is designed to enable all of these kinds of analyses to be done on the log data.

Questions beyond Learning

While evidence of learning is the primary goal when looking at educational games, it is not the only question that can be asked about a game. Many other questions can arise over the course of development. Here is a brief list of questions that were raised by researchers during the development of RumbleBlocks:

- Are the goals of the game actually rewarding the kinds of behaviors we want to be encouraging? Does success in the game (e.g., a tower that withstands an earthquake) align well with educational objectives?
- Can we measure the "difficulty" of levels? Does the difficulty of the levels progress in an appropriate way?
- How many different solutions are there to each level? Are there solutions to levels that we did not intend?
- What measures correlate with the "fun" of levels?

Answering these questions can affect the interpretation of learning measures as well as provide implications for redesigning gameplay mechanics. Furthermore, we have found that the researchers and educators on the team continually come up with new questions they would like to answer about the game. When selecting methods to measure learning, it would be helpful if they could also address some of these kinds of questions, and could facilitate answering new questions on the existing dataset from previous players.

OUR APPROACH

To this end, we have developed our approach as a way of exploring multiple questions from the same set of player data. The system we developed had to be capable of answering learning-related questions as well as facilitating exploration of other possible questions. Our approach has two major components in it: a toolkit for logging player actions within the game, and an architecture for a Replay Analysis Engine that rebuilds and annotates a player's session. Currently the system is implemented within the Unity game engine, but it is not conceptually restricted to this environment.[1]

Play log analysis is a common practice of game user research with many pre-existing examples in the field [1,12]. Our approach differs from prior approaches by examining a student's play session as a live instantiation of the game state within a running game engine. This allows analysis to consider a much deeper picture of a student's performance, which can be analyzed from multiple perspectives, and provides access to more contextual properties of game objects than a designer may have thought to measure initially. One of the hallmarks of our design is the ability to adapt new analysis onto existing data, freeing education researchers from having to rerun large classroom studies while experimenting with new measurement or analysis techniques.

Logging System

The logging system uses an adaptation of the PSLC DataShop XML format, which is a commonly used format for describing student interactions with intelligent tutoring systems [13]. The primary motivation behind the design of the logs in our system is to be descriptive and to defer assessment to the analysis stage, whereas the original DataShop format records both the action the student took and the assessment of the tutor system as a single transaction.

Logs of student behavior are captured at the level of a basic action, defined as the smallest unit of meaningful action that a player can exert on the game world. These actions are meant to be contextualized to the game world, i.e. picking up or dropping an object, rather than the raw input of the player, i.e. mouse down at position (x, y). The reason for recording the smallest possible actions is to allow for analysis at various grain sizes by capturing data at the finest grain size. Additionally, it is easy for the game implementer

[1] The library for logging actions can be found at https://github.com/eharpste/Unity-Logger and the implementation of the replay system for RumbleBlocks is available upon request from the authors.

to see where logging calls need to be inserted into the game code because the basic actions make up the base mechanics of the game.

In addition to basic actions, logs also carry contextual information, such as the beginnings and ends of sessions and levels and whether levels were beaten or not. This information can also carry the initial conditions of the level, which might be relevant to analysis. Capturing this information facilitates coarse grain analyses dealing with whole level performance. This information can also be used to focus fine grain analysis on particular levels.

A log of a single action is described using the Selection-Action-Input paradigm borrowed from DataShop [13]. In a log, the Selection is defined as the entity the player is acting on, in most cases the name of the entity, or some other unique identifier. The Action is what the player is doing to the entity, a text description of the type of action. The Input is a parameter to the action being performed. The meaning of the Input field is contextual to the type of action, and is not always meaningful, such as in the case of button presses which use "-1" for Input as a standard convention. In the case of RumbleBlocks, the Inputs of block-related actions correspond to the position, rotation, and velocity of the block being acted on, the origin position when picking up blocks and the resultant position when releasing them.

There is also capacity to log actions that are not performed directly by the player through the same mechanism, which we refer to as engine actions. In RumbleBlocks, engine actions are used to record when the system activates the earthquake mechanism, and when the blue energy balls are covered and uncovered.

The novel aspect of our logging system is that each action is also paired with a description of the state of the game just before the action took place; this appears in the logs as a separate entry that is paired with the action by a unique transaction ID. This allows analyses to consider each action within the context in which it took place, and know the initial conditions of an action that may take time to broadly affect the game environment. The logging system also uses the state just before the action to account for actions that might remove an entity. The contents of a state are defined by the game developer and should contain only objects that are deemed relevant to ensure accurate replay. There is also capacity for simpler actions to not have accompanying states, such as pressing a "Level Start" GUI Button. In RumbleBlocks, the state describes the position, rotation, and velocity of the spaceship and all of the blocks currently on the field. The position of the cliff and blue energy balls are not needed in the state descriptions because they are immutable and instead are logged as initial conditions.

It should be noted that the actions being captured are not necessarily the set of actions that are deemed relevant by a method of analysis. ECD, for example, may consider a set of actions irrelevant to measuring a particular piece of learning. The recorded actions are the set of actions that are relevant to gameplay. This stance is in keeping with the motivation to make logs descriptive of what is happening in the game. Using purely descriptive logs allows researchers to revisit old data in light of new findings or new analysis methods without having to run additional expensive field or classroom studies.

The logging library was written in C# and is designed to be straightforward to use. For each player action and special event such as level start and end, the implementer adds a call to a function that takes as parameters the Selection, Action, and Input of the action. (For convenience, there are also simpler versions of this call that leverage the standard properties that all Unity game objects have.) In order to log the state, the implementer keeps up-to-date a global list of all relevant game entities, and the logger automatically logs the relevant properties. For convenience, the implementer can alternatively attach a script to these objects using Unity's interactive builder at design-time rather than adding them to the list programmatically.

The highly descriptive nature of the logs can make them somewhat large. The entire RumbleBlocks dataset we discuss here, contains logs from two sessions of play (approx. 20 minutes per session) by 174 children, and is roughly 850MB in size. The system is primarily designed to log to a webserver; however, when running classroom studies, we have found that the volume of logs can become intractable for some school networks and so a locally logged version is also available.

Replay and Analysis Engine

A novel aspect of our system is that the analyses of student performance are evaluated using an active replay of the student's game session in the running engine rather than on the raw logs themselves. This replay process happens within the second main component of the approach, which we call the Replay Analysis Engine (RAE). This tool is also implemented in C# for the Unity engine. The log replay system reads in a student log and reconstructs the student's play session action-by-action. For each action, the RAE first constructs the state in which the action took place and then enacts the student action to let the Unity game engine calculate the consequences of that action. For example, if the action is to let go of a block, then the RAE will use the game engine to calculate where the block ends up after it finishes falling.

The RAE makes use of Unity's "prefab" architecture. A *prefab* is a Unity construct that serves as a master copy of a complex game object to allow for easy instantiation and duplication. Using this system allows the analysis designer to ensure that the game objects on which they run their analysis behave exactly the same as the ones the player experienced. This, in turn, ensures that any analysis is a valid reproduction of student performance in the game.

The RAE has a few different options for determining when to stop the simulation and run the analyses. By default, the logged state is re-instantiated and simulation is stopped immediately. Another option (and the primary one used in RumbleBlocks analysis) is that once the state is re-instantiated, the simulation is run until all the objects in the interface have stopped moving. Once this happens, the simulation is stopped and the analyses are run. There is also an option to run the simulation for a defined amount of game time, which can also be accelerated without loss of accuracy.

The RAE supports two different ways to analyze the logs. The first way is through an API that allows access to the states created by student actions. The analysis implementer can write C# code which leverages this library to generate metrics. This is useful when the analyzer needs to access specific properties of the objects. For example, in RumbleBlocks we were interested in the properties of structures students built, such as their centers of mass. We used our API to examine the instantiated version of blocks to find which ones were part of a connected tower, which then allowed us to average their centers of mass to create a center of mass for the structure as a whole. Without the ability to access the collider information or rigid body properties of game objects in the live engine we would not have been able to do this.

Implementing an analysis using the RAE API resembles creating a new level for the game. The analysis developer first creates a skeleton level containing any entities that are common to all levels of the game and an entity provided by the library that runs the actual replay. The developer must also include any prefabs of objects that may appear in a state message. The RAE provides a set of routines to iterate through the actions in a student log, and to inspect the elements of a current action state. It also provides routines for accessing the success and failure of each level in the log so the analysis designer can easily have access to whatever properties of the performance they want to analyze.

The second way of using the RAE is by running student logs through predefined analyses. This process generates spreadsheets of data that can be used for statistical analysis. We have used the API provided by the RAE to develop a set of common measures to filter logs based on their properties, such as outputting only specific levels or users. Access to these common filtering operations is provided through onscreen menus while the RAE is running to control which data are recorded in the spreadsheets.

The benefit of the RAE approach is the ability to create a set of potential measures and analyses and iterate on them separately from the development of the game itself. Since the analysis system contains all common level elements, it is agnostic to level design and to some changes in mechanical design. One slight limitation is that it requires the game developers to maintain old copies of prefabs within the game files so that future analysis of old data will remain consistent. There are situations where one might want to break this rule, to explore if the responses to previous players' actions may have had different results if a particular rule was changed.

USE IN ANALYSIS FOR RUMBLEBLOCKS

We used our toolkit with RumbleBlocks as part of a formative analysis of the game's design, so we and other members of our team of collaborators were able to perform a wide range of analyses. The data we discuss here were gathered as part of a classroom study in two Pittsburgh area public schools, with subjects taken from kindergarten to third grade classrooms (roughly 5 to 8 year-olds). This study was meant to both check the progress of the game design and attempt to ground, for the first time, measures of learning within the game against external (i.e., out-of-game) assessments of the educational objectives of the game.

For the purposes of this study, two sets of levels were selected to be used as in-game pre- and post-tests counterbalanced across students. These levels were chosen out of the collection of regular building levels but altered to have their blue energy balls removed, in addition to removing the ability to allow students to retry on a failed attempt. These special levels were placed after a short collection of tutorial levels, which explained the mechanics of the game. The intention of this design was to allow us to get a sense of the kinds of structures students would build before they had a large amount of experience with the game, and after they had a chance to practice building. In addition to these in-game evaluations, players also took out-of-game paper and pencil tests, developed by our colleagues, before and after playing the game. These tests contained items relating to stability and construction, based on our three principles of base width, low center of mass, and symmetry. We only touch on a subset of these learning analyses here, since the full results will appear elsewhere.

Learning Analysis

Our team was interested in exploring many questions, such as those listed above about RumbleBlocks but chief among them was whether or not we could say there was evidence of within-game learning related to the game's educational objectives (i.e., understanding and application of the three principles of stability). We measured a slight, yet significant, increase in performance from pretest to posttest on the out-of-game measures ($p < .05$); however, we were also interested in whether learning could be measured in the game, using data provided by our system. In looking at the difference in raw performance, i.e., the pass rate of the in-game pre-test and post-test levels, there is a small, yet significant, improvement in the passing rate ($p < .001$). This improvement however could potentially be attributable to simply learning the game mechanics (although that explanation would not account for the increase in performance measured on the out-of-game pre-test and post-test).

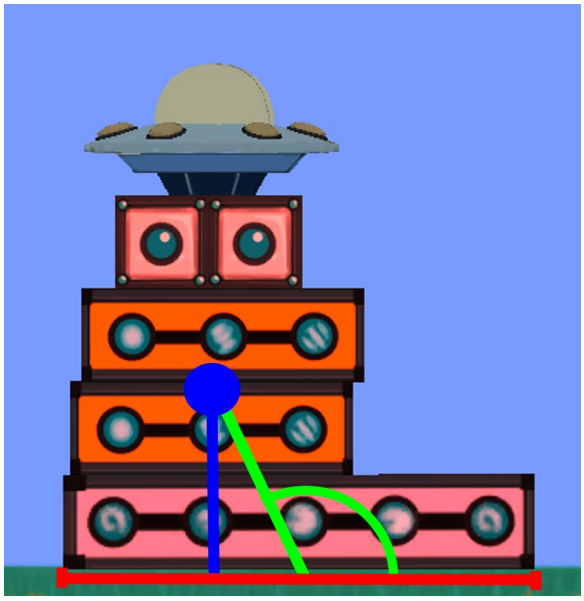

Figure 3. Illustrations of measurements used in RumbleBlocks evaluation. The length of the red line is the base width, the length of the blue line is the height of the center of mass and the size of the green angle measures symmetry.

To explore this issue further, we wanted to see if, within the game, there were performance gains in terms of the specific principles. We are interested in this measurement as a way of getting a clearer picture of *exactly what students were learning* and not just that they were getting better at passing levels. This would mean that from pre to post, students would build towers that showed a better awareness that (1) a structure with a wide base is more stable, (2) a structure with a lower center of mass is more stable, and (3) a structure that is symmetrical is more stable. To find out whether they did, we instrumented the replay system to calculate a variety of metrics based on the final state of each level that a student played. These metrics were: the width of the tower's base, the position of the tower's center of mass relative to the ground, and a measure of symmetry defined as the angle formed by a ray from the center of the base to the center of mass and the ground (illustrated in Figure 3). These measures were then compared to values calculated across all other players for that same level in order to create a relative score for each player. In the case of base width, this score is relative to the maximum observed width for that level; for center of mass position, the score is relative to the lowest observed position for that level; and for symmetry a score is calculated relative to 90 degrees which would represent perfect symmetry. When comparing the average relativized pre and post scores, we saw a significant improvement, with a one-tailed t-test, for the wide base and symmetry principles (see Table 1), meaning that at the end of playing the game, students were beginning to design towers that had wider bases and more symmetrical layouts. However, we did not see any significant difference in terms of center of mass position, meaning that students did not seem to attempt to lower the center of mass of their structures. This result would suggest that the game is currently not successfully teaching the center of mass principle. A recommendation to the game designers from this finding would be to explore ways of making the center of mass of a tower more salient to the player so that they might be able to account for it when creating their solution to a level.

Measure	Mean (SE)		T-Statistic
	Pretest	Posttest	
Width of Base	0.60 (.01)	0.64 (.01)	-2.767**
Symmetry Angle	5.98 (.34)	5.20 (.27)	1.9783*
Center of Mass	1.61 (.02)	1.63 (.02)	-0.663

Table 1. Pre and posttest means and significance values for metrics. Note for the width of base measure, a higher value is more desirable, whereas for the symmetry angle and center of mass measures, a lower value is more desirable.
*** p < .05, ** p < .01**

Gameplay Alignment Analysis

Another interesting question that we explored was how well the game was aligned to its educational objectives. Many researchers have pointed to the importance of alignment of game goals to instructional goals [15,18,22]. The basic question is: does the game properly incentivize the behavioral patterns that we want to foster? Additionally, in the case of RumbleBlocks, we wanted to know if levels that were designed to target a particular piece of knowledge (i.e., one of the three principles of stability) actually succeeded in doing so. Answering this question during a formative evaluation can have implications for level design and even on the design of a game's core mechanics if there are alignment problems. We instrumented a new set of metrics in addition to the ones used in the learning analysis described above, including what we call the "stability metric." From the standpoint of realistic earthquake physics, the stability metric is the amount of energy that would be required to overturn a tower about its weakest foot, and so a high value in this metric would represent a stronger design.

In using these metrics for analyzing alignment, we wanted to explore how well metrics that should indicate a well-constructed tower actually corresponded to students passing a given level. To do this, we created 3 groups by collecting together the levels that target each of the three principles, and then we performed logistic regressions using each of the metrics of the players' towers to predict success on a level. The results of the regression analyses can be found in Table 2. When looking at the level-relevant metrics for base width, center of mass height, symmetry, and the stability metric, we found a significant predictive relationship between them and success on level, which is what would be expected, however the direction of the relationship for the center of mass metric was in the wrong direction. This

Group	Coefficient	Estimate	Standard Error	Z-Statistic
Levels targeting the Wide Base Principle	Width of Base	0.26227	0.05667	4.628***
	Center of Mass	-0.13203	0.122101	-1.091
	Symmetry Angle	-0.03086	0.01041	-2.965**
	Energy Required	-0.28414	0.12115	-2.345*
Levels targeting the Low Center of Mass Principle	Width of Base	0.036602	0.026918	1.360
	Center of Mass	0.783418	0.107301	7.301***
	Symmetry Angle	-0.003038	0.005309	-.0572
	Energy Required	-0.061895	0.114265	-0.542
Levels targeting the Symmetry Principle	Width of Base	0.066311	0.028741	2.230*
	Center of Mass	0.167317	0.073404	2.279*
	Symmetry Angle	0.022834	0.008717	2.620**
	Energy Required	0.389674	0.107243	3.634***

Table 2. The results of logistic regression predicting level success based on metrics within groups of levels targeting a specific principle. Note for the width of base measure, a higher value is more desirable, whereas for the symmetry angle and center of mass measures, a lower value is more desirable.
*** p < .05, ** p < .01, *** p < .001**

would mean that, counter to what the target principles suggest, students that build with higher relative centers of mass are more likely to succeed on the levels that target the center of mass principle. One possible explanation for this comes from the level designs for the center of mass levels. In an attempt to have the center of mass levels isolate the center of mass principle from the highly related wide base principle, the level designers created a few levels where the intended solution had a very narrow base. This has the effect of actually raising the center of mass for successful structures and possibly introducing confounds. The results of this alignment analysis as well as the learning analysis suggest that the design of these levels should be revisited.

Solution Space Analysis

An ongoing initiative in our research group is to attempt to answer the question of how many unique solutions students have created for each level. While every level was designed around a particular envisioned solution, there is no requirement that players use it. Understanding the solutions that students actually created can lead to many further analyses, such as finding solutions we did not intend that might bypass the learning goals of the game, or exploring relationships between particular solution patterns and better performance on transfer measures, or analyses of how challenge, fun, and levels having multiple solutions are related.

A novel way to approach counting unique solutions is to employ expectation maximization clustering along metrics that describe student solutions with cross validation to determine an optimal number of clusters that describes the set of all solutions to a given level, with each cluster representing a unique solution. We have been exploring the use of the metrics derived from our previous analysis as features to feed into the clustering algorithm. This is somewhat problematic as the raw metrics, like center of mass location and width of base, cannot render true equality between two states in the game as quite different structures could have similarly located centers of mass or bases of the same width. Our initial analysis attempts that used clustering on our existing metrics were promising but not readily satisfying. This led us to brainstorming a larger collection of features to build into the RAE that could be beneficial in clustering student solutions together. Among these new metrics are measures of the towers' extents, such as total height and maximum width, as well as measures to capture the position of the spaceship relative to the student's tower. It is our hope that this initiative will render a tractable set of student solutions out of the large set that we currently have in log data. Once a smaller set has been determined, they could be used diagnostically to make inferences as to how a student might perform on transfer tests based on which designs they used.

Another possible way of attacking the question of how many student answers there are is to use plan recognition. This method would require us to decide on a grammar or set of production rules that we can apply in the RAE to annotate the log. Once a collection of higher-order actions is gathered, students can be grouped by the plans they demonstrate. This is a promising direction for analysis and remains part of our ongoing work.

Further Analyses

Going forward, our team will be exploring other analyses through the data afforded by our approach. One of these is to explore more fine-grained analysis by applying BKT as a way of corroborating our coarse grained measures. This would require that we identify knowledge components present within RumbleBlocks that can be mapped to particular actions. Experimenting with different mappings is a time intensive task, but it is made easier by already having a way of formalizing the actions a student can take within the game and having a dataset to test models with.

Another open question that we will tackle is whether or not the difficulty in the game increases in a reasonable manner. One way of measuring problem difficulty is by looking at the amount of time and number of actions it takes to beat different levels. If certain levels seem to take inordinate amounts of time to beat then this might indicate that the level is too difficult. This method, however, cannot properly factor out learning of game mechanics or physics content without randomizing the level order, which is hard to do in a game with a narrative context such as RumbleBlocks. Another way of exploring difficulty is to look at how many times players have to backtrack when trying to find a successful answer (i.e., build a successful tower). This can be accomplished by defining patterns of actions that are indicative of backtracking, such as placing a block back into the inventory or deconstructing a structure and throwing blocks to the side. Levels with a high degree of backtracking could also indicate levels that are more challenging.

A final question that will be explored arises from our game design partners in asking if there is evidence of the game being fun or not. This might be measured by looking at how long players played and how often they returned to the game. This data is readily available from the logs we currently possess; however, since our formative evaluation was conducted in a classroom setting, this data is confounded by the fact that players were a captive audience. Another way we might approach this question is by looking at the rate at which players experiment with the game's mechanics as a measure of playfulness; the cluster analyses and plan recognition initiatives could help with this by looking at the diversity of solutions within a student's actions.

DISCUSSION

So far our approach has facilitated many different analyses that have benefited our educational game development project. We were able to explore course grain learning analysis on a principle level, and found that RumbleBlocks appears to be teaching players to address the wide base and symmetry principles when building simple structures. We were also able to examine the alignment of gameplay to our stated content goals, and found that the alignment is fine for two of the three targeted principles, but that there are potential issues with the alignment of levels targeting low center of mass, which may help explain the results of our learning analysis. Finally, we have also used our technique to begin to map the solution space of RumbleBlocks, giving us a better understanding of what kinds of student answers we are seeing. Each of these findings can help provide feedback for further refining the game.

Each of the analyses we have explored so far requires a slightly different perspective on the performance of players. The learning analysis asked questions about our players by looking at their final solutions to levels. The alignment analysis was concerned with the design of the game itself and used student solutions as a corpus for analysis. Mapping solution space and measuring fine grained progress required a representation of the steps students took throughout the game. Without the benefit of our replay analysis engine exploring each of these questions would have necessitated a different expensive classroom study, and a different characterization of student performance.

As our team continues to move forward and develop new games we will also continue to evolve the system. Currently the implementation of the toolkit is restricted to the Unity environment and many aspects of the log replay system rely heavily on Unity-specific constructs, such as the prefab architecture. There is no reason that the approach could not be implemented in other environments, however. At its core, the approach requires that the entities in the game can be recreated dynamically without loss of accuracy. Given that dynamically creating large numbers of similar game objects is a common requirement for games, many existing engines contain such capabilities.

However, this approach is certainly not applicable to all game types. Our work so far, and into the foreseeable future, is focused on single-player physics games, which do not have to deal with issues such as the interactions of multiple players or the open three dimensional movements of player avatars, such as those present in multi-user virtual environments like River City or Quest Atlantis [4,11]. Applying our process to these kinds of environments would demand more data aggregation on the logging side to avoid information overload from logging every player's minute movements. An alternative might be to only log when players enter and leave certain areas rather than the individual movement commands. This would end up yielding data similar to the "heat map" analysis already commonly used in game user research [1].

As we begin to publish our games online, we are also interested in exploring the capacity to make them adaptive to learners' needs. One of the benefits of our architecture that readily serves this goal is the fact that the analysis measures are built in the exact same environment as the game itself. This means that once we are satisfied by the reliability of a particular measure, it can be easily moved into the live game to serve as an on-line assessment. When the measure reveals that students are struggling, we could then try to serve them levels adaptively to focus their learning.

CONCLUSION

Open educational games have proven to be difficult to analyze. We have developed an approach for looking at student performances in open educational games that facilitates multiple kinds of analysis through log reinterpretation, and we implemented this approach in a toolkit and set of analysis tools for Unity in C#. Using a more traditional logging approach, this kind of analysis would have required multiple expensive classroom studies to gather data as new questions and measures were envisioned by our team. Additionally, analysis at the level of individual principles would not have been possible using only success on levels as a metric.

The toolkit is available to others to extend and apply to their own games. It is our hope that others will embrace the methodology for recording players' behavior in games as a way of exploring multiple measures of learning. Having consistent measurement and analysis techniques will help bolster the methodological rigor of the field of educational game design.

ACKNOWLEDGMENTS

We would like to thank development team of RumbleBlocks as well as our collaborators on the ENGAGE Project. This work was supported in part by the DARPA ENGAGE research program under ONR Contract Number N00014-12-C-0284 and by a Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education # R305B090023.

REFERENCES

1. Ambinder, M. Valve's Approach to Playtesting: The Application of Empiricism. *Presented at GDC 2009*, (2009).
2. Andersen, E., Rourke, E.O., Liu, Y., et al. The Impact of Tutorials on Games of Varying Complexity. *Proc. CHI 2012*, (2012), 59–68.
3. Baker, R.S.J. d, Habgood, J.M.P., Ainsworth, S.E., and Corbett, A.T. Modeling the Acquisition of Fluent Skill in Educational Action Games. In *User Modeling 2007*. 2007, 17–26.
4. Barab, S., Thomas, M., Dodge, T., Carteaux, R., and Tuzun, H. Making Learning Fun: Quest Atlantis, a Game without Guns. *Educational Technology Research and Development 53*, 1 (2005), 86–107.
5. Christel, M.G., Stevens, S.M., Maher, B.S., et al. RumbleBlocks: Teaching Science Concepts to Young Children through a Unity Game. *Proc. CGames 2012*, (2012), 162–166.
6. Corbett, A.T. and Anderson, J.R. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modelling and User-Adapted Interaction 4*, 4 (1995), 253–278.
7. Council, N.R. *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. The National Academies Press, 2012.
8. Delacruz, G.C., Chung, G.K.W.K., and Baker, E.L. *Validity Evidence for Games as Assessment Environments (CRESST Report 773)*. Los Angeles, CA, 2010.
9. Gee, J.P. *What video games have to teach us about learning and literacy*. Palgrave Macmillan, New York, 2003.
10. Honey, M.A. and Hilton, M. *Learning Science Through Computer Games and Simulations*. National Academic Press, Washington, D.C., 2011.
11. Ketelhut, D.J. The Impact of Student Self-efficacy on Scientific Inquiry Skills: An Exploratory Investigation in River City, a Multi-user Virtual Environment. *Journal of Science Education and Technology 16*, 1 (2006), 99–111.
12. Kim, J.H., Gunn, D. V, Schuh, E., Phillips, B.C., Pagulayan, R.J., and Wixon, D. Tracking Real-Time User Experience (TRUE): A comprehensive instrumentation solution for complex systems. *Proc. CHI 2008*, (2008), 443–451.
13. Koedinger, K.R., Baker, R.S.J. d, Cunningham, K., Skogsholm, A., Leber, B., and Stamper, J. A Data Repository for the EDM community: The PSLC DataShop. In C. Romero, S. Ventura, M. Pechenizkiy and R.S.J. d. Baker, eds., *Handbook of Educational Data Mining*. 2010, 43–55.
14. Manske, M. and Conati, C. Modelling Learning in an Educational Game. In C.-K. Looi, M. Gord, B. Bredeweg and J. Breuker, eds., *Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*. IOS Press, Amsterdam, Netherlands, 2005, 411–418.
15. Martin, B., Koedinger, K.R., Mitrovic, A., and Mathan, S. On Using Learning Curves to Evaluate ITS. *Proc. AIED 2005*, (2005), 419–426.
16. Mislevy, R.J., Steinberg, L.S., and Almond, R.G. On the Structure of Educational Assessments. *Measurement: Interdisciplinary Research and Perspectives 1*, 1 (2003), 3–62.
17. Newell, A. and Rosenbloom, P.S. Mechanisms of skill acquisition and the law of practice. In J.R. Anderson, ed., *Cognitive skills and their acquisition*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981, 1–56.
18. Shute, V.J. Stealth Assessment in Computer-Based Games to Support Learning. In S. Tobias and J.D. Fletcher, eds., *Computer Games and Instruction*. Information Age Publishers, Charlotte, NC, 2011, 503–524.
19. Spring, F. and Pellegrino, J.W. The Challenge of Assessing Learning in Open Games : HORTUS as a Case Study. *Proc. GLS 2011*, (2011), 200–208.
20. Stamper, J.C., Lomas, D., Ching, D., Ritter, S., Koedinger, K.R., and Steinhart, J. The Rise of the Super Experiment. *Proc. EDM 2012*, (2012).
21. Young, M.F., Slota, S., Cutter, A.B., et al. Our Princess Is in Another Castle: A Review of Trends in Serious Gaming for Education. *Review of Educational Research 82*, 1 (2012), 61–89.