# CRESST REPORT 814

# A PRIMER ON DATA LOGGING TO SUPPORT EXTRACTION OF MEANINGFUL INFORMATION FROM EDUCATIONAL GAMES: AN EXAMPLE FROM SAVE PATCH

MARCH, 2012

*Gregory K.W.K. Chung*
*Deirdre S. Kerr*

National Center for Research
on Evaluation, Standards, & Student Testing
UCLA | Graduate School of Education & Information Studies

# A Primer on Data Logging to Support Extraction of Meaningful Information from Educational Games: An Example from Save Patch

CRESST Report 814

Gregory K. W. K. Chung and Deirdre S. Kerr
CRESST/University of California, Los Angeles

March 2012

National Center for Research on Evaluation,
Standards, and Student Testing (CRESST)
Center for the Study of Evaluation (CSE)
Graduate School of Education & Information Studies
University of California, Los Angeles
300 Charles E. Young Drive North
GSE&IS Bldg., Box 951522
Los Angeles, CA 90095-1522
(310) 206-1532

The work reported herein was supported under the Educational Research and Development Centers Program, PR/Award Number R305C080015.



To cite from this report, please use the following as your APA reference: Chung, G.K.W.K. & Kerr, D. (2012). *A primer on data logging to support extraction of meaningful information from educational games: An example from Save Patch.* (CRESST Report 814). Los Angeles, CA: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

## TABLE OF CONTENTS

# A PRIMER ON DATA LOGGING TO SUPPORT EXTRACTION OF MEANINGFUL INFORMATION FROM EDUCATIONAL GAMES: AN EXAMPLE FROM SAVE PATCH

Gregory K. W. K. Chung and Deirdre S. Kerr
CRESST/University of California, Los Angeles

## Abstract

In this primer we briefly describe our perspective and experience in using data logging to support measurement of student learning in a game testbed (*Save Patch*) we developed for research purposes. The goal of data logging is to support the derivation of cognitively meaningful measures and affectively meaningful measures from a combination of player behaviors, game events, and game states. Key best practices we have developed are to record data that reflects behavior rather than inferences about the behavior, specify the behavior to log ahead of time, log in-game behaviors that map directly to targeted knowledge, skills, and attitudes, encode sufficient information so that the data elements are unambiguous at the desired grain size, capture context to allowing linking of the data element to an individual's specific game experience (e.g., school, teacher, period, stage, level, game event, game state), and use a structured and delimited record format. The capturing of behavioral events that presumably is a manifestation of cognitive and affective processes allows for the investigation of numerous research questions that connect game play to students' background, strategy use, knowledge, and cognitive processes.

## Introduction

This primer is intended to describe CRESST's perspective and experience with data logging issues related to the games developed by the Center for Advanced Technology in Schools (CATS). This primer describes the method of data logging that has been used successfully in CATS-developed games. The examples we use are drawn from a CATS-developed game, *Save Patch*. *Save Patch* was developed to teach fraction concepts to middle school students. A description of *Save Patch* is given in Appendix A.

We use the term *data logging* to connote the systematic specification, capture, and logging of events that occur in a game (i.e., player-initiated or game-initiated events) or game states to a permanent external store using a predefined record format. We do not mean the logging of unstructured output or the ad hoc capture and storage of events that are based on arbitrary criteria or convenience.

## General Approach

Our perspective on data logging flows from the behavioral observation tradition (e.g., Bakeman & Gottman, 1997). Two features underlie behavioral observations: (a) systematic observations—that is, the set of behavioral acts of interest are defined prior to the observation, and (b) reliable coding of the observed behavior.[1]

**Goal.** The goal of data logging is to support the derivation of cognitively meaningful measures and affectively meaningful measures from a combination of player behaviors and game states. By *meaningful* we mean that the measures should: (a) help researchers and designers interpret why players are performing the way they are, and (b) exhibit a systematic (e.g., statistical) relationship with complementary measures; differentiate between players with different degrees of content knowledge, different degrees of game experience, or different backgrounds (e.g., language skills); and differentiate between players who receive different instructional treatments or different game designs. Our major assumption is that player behavior—what players do at a specific point in the game—is a manifestation of their ongoing cognitive and affective processes (e.g., knowledge, judgment, decision making, problem solving, self-regulation, self-efficacy, beliefs, and attitudes).

**Specification of what to log.** The critical aspect of data logging is the specification of the behaviors, events, and states to log. If the focus is on learning, teaching, and assessment—then behaviors[2] related to each of those foci should be logged. The challenge is in mapping specific in-game behavior to unobservable cognitive and affective processes such that the ambiguity of the datum is minimized. A specific behavioral act can be a manifestation of numerous underlying processes. Judicious structuring of the interaction and the capture of contextual information surrounding the interaction can help eliminate alternative explanations underlying the behavior.

As an example, if the research question asks about whether students know that two fractions with unlike denominators cannot be added together, then the game should allow and log students' attempt to add unlike denominators, rather than disallow the behavior entirely. Additionally, it is important to know the context in which the attempted addition occurs. An

[1] In the case of computer interactions, establishing high reliability of coding is still an issue but different in nature from establishing high human rater reliability. The sources of error are different from human rater errors. Computer-based data logging errors are largely due to poorly specified requirements, and to a far lesser extent poor software design and programming errors.

[2] The use of the terms player behaviors, events, and states is used loosely. In general, the use of ~~"b~~behavior" connotes all three dimensions.

attempted addition of 1/4 to 1/2 when the answer is 3/4 has a different explanation than an attempted addition of 1/4 to 1/2 when the answer is 2/4.

**Target behaviors that reflect the use of cognitive demands.** By cognitive demands we mean the set of intellectual skills required of learners to succeed in the game. Examples of broad categories of cognitive demands include adaptive problem solving, situation awareness, decision making, self-regulation, teamwork, conceptual and procedural learning of content, and application and transfer of learning. In a game (or any other task where inferences about cognition are made based on observations), it is important to conduct a cognitive task analysis that provides insight about the mental operations students invoke during the course of carrying out a task.

> Because cognitive processes cannot be observed directly, inferences about the use (or non-use) of a particular cognitive process and the appropriate use (or inappropriate use) of that process can be based only on what learners do in the game—their in-game behaviors and the associated game state.

Ideally, the design of the user interface and game mechanics will allow only those learners who have knowledge of *X* to execute game mechanic *x*. To the extent that is possible, game mechanic *x* becomes a potential measure of *X*.

**Data logging record structure.** Our approach to data logging is to record data at the finest usable grain size. By *finest usable grain size*, we mean a data element that has a clear definition associated with it. For example, a data element that refers to "click" is often unusable whereas a datum that qualifies the click (e.g., "clicked on the reset button) is usable. For example, in *Save Patch* logging an attempted addition is not at the finest usable grain size because some attempted additions have the same denominator and some do not. In this case, the finest usable grain size would be logging an attempted addition with different denominators.

In general, each record should contain sufficient information to describe the context in which the event occurred and in sufficient detail to link the data to a specific school, teacher, period, player, game level, and game state. One way to think about this is to suppose the data were recorded on index cards (e.g., a sorted deck of 150,000 cards composing the game experience of 130 students across 5 teachers, 4 periods, and 4 different versions of the game) and the card deck was dropped: What information would need to be recorded on each index card so that the original card deck could be reconstructed perfectly? As an example, our data log record related to an attempted addition with different denominators would also include fields documenting the student that made the addition, the level in which the addition was

made, the time at which it occurred, the fraction value being added, the fraction value it was added to, the location on the grid where the addition occurred, and the values placed on all other grid locations at the time.

The format we have used is a *flat file* representation of the data as this format is simple, easy to explain, easily understood, and portable, and is required when logging data to a local text file. We have also used more efficient representations, but only when reliable connections to a database server can be guaranteed. Regardless of the particular format of the data store, the eventual destination of the data itself is a statistical analysis tool, where often a flat file representation is the easiest format to use for the greatest number of users.

Our approach has emphasized ease of use by end-users of the data (e.g., the data analyst, researchers) and not computational efficiency. This trade-off is intentional and assumes that multiple data analysts and researchers will touch the data over its lifespan; thus making the data log as simple and usable as possible is a high priority. An example of this trade-off is including a field called data_description that includes the following text for any record logging an attempted addition: "~~a~~dded ropes incorrectly: at `[position]` tried to add `[value added]` to value `[existing value]`" which contains no additional information not already logged, but allows the end-user to interpret the rest of the data in the record without referring to a handbook or manual.

Table 1

Sample Log File Record Format Used in CATS Game *Save Patch*

| Field | Data type | Description |
|---|---|---|
| Sn | long integer | Increments from 1 to *n*. Use to uniquely identify each record in the data and to sort the records in the order they were recorded. |
| timestamp | formatted time of day | The time the data was captured in the following format: mm/dd/yy hh:mm:ss.mmm |
| game_time | long integer | The time in seconds since the game was loaded. |
| user_id | integer | The login ID of the current player. |
| Stage | integer | The current stage of the game. Set to 1 if there is only one stage in the game. |
| Level | integer | The current level of the game. Set to level name if not included in scripts. |
| data_code | integer | The numeric code that uniquely describes this type of data. There should be a 1:1 correspondence between a data code and the type of data logged (e.g., data_description). |

| Field | Data type | Description |
|---|---|---|
| data_description | string | A general description of the data being logged by the corresponding data_code. |
| data01 | string | data_code specific value. |
| data02 | string | data_code specific value. |
| data03 | string | data_code specific value. |
| data04 | string | data_code specific value. |
| data05 | string | Spare |
| data06 | string | Spare |
| data07 | string | Spare |
| game_state | string | A list of the values currently placed on the grid. List the position of the sign, space, and the value on the sign. Separate signs with "]". Do not use "," or tab (\t) to delimit values. |

*Note*. Tab (\t) to delimit each field of the record. Newline (\n) to delimit each record.

## General Data Element Properties[3]

In general, when we log an event, we attempt to ensure that the data element has the following properties: (a) is a description of behavior (and not an inference about why the behavior occurs), (b) is unambiguous (i.e., the data point refers to a single event and not a collection of events—the difference between "clicked on button 1" vs. "clicked on a button"), and (c) contains sufficient context information to allow linking of the data element to a specific student at a specific point in the game.

**Descriptive.** Our general approach is to record a description of the event and not an interpretation of the event. For example, suppose in a fractions game the game mechanic supports adding two objects where each object represents a fraction. Adding two things incorrectly can be represented descriptively as "incorrect addition" or inferentially as "player does not understand how to add fractions."

The issues with logging inferences are: (a) Unless validity evidence has been gathered on the specific interpretation, the interpretation may not be accurate. (b) An interpretation layered over the actual event may create restrictions on subsequent data analyses. For example, statements about *what* the player did in the game (which may be useful for usability analyses) may not be possible if the data element reflects understanding. Data logged as "does not understand adding fractions" says little about the actual game play itself. (c) The inference may subsume multiple events, in which case the subsumed events are unavailable

[3] Not all elements will be described in the text.

for analyses. This aggregation may lead to uninterpretability of inference data (i.e., an action logged as "student understands adding fractions" immediately followed by "student does not understand adding fractions").

The trade-off is that the volume of data is much greater for descriptive data compared to inference data. However, for exploratory analyses, particularly during the early stages of game development when little or no empirical data exist, our perspective is that the data should be logged at the descriptive level and not the inference level.

**Unambiguous.** For maximum flexibility (particularly for statistical analyses), the data element should be unambiguous. By unambiguous we mean a 1:1 correspondence between the data element and an event. For example, suppose there are 10 buttons and we are interested in recording button click events. The data should be recorded in such a way to uniquely identify which of the 10 buttons was clicked on, as well as support easy aggregation across the 10 buttons. The first capability allows us to examine a particular behavioral act, and the latter case allows us to examine a class of behavioral acts. If only the latter capability exists, then there is a loss of information and potentially important behavioral acts may be masked by the aggregation. In our log files we separate these into the data_code field, which allows us to examine a class of behavioral acts, and the data01 through data07 fields, which add additional details that allow us to examine specific behavioral acts.

**Contextualized.** The idea of contextualizing is to encode as much relevant information about the conditions under which the data were generated as feasible. The purpose for gathering context information is to rule out alternative explanations for the observed data and in general, to help researchers understand why an event occurred in the game.

Contextual information consists of two classes of information. First, information about the student—background information such as schooling (e.g., school, period, teacher, grade), domain-specific information (e.g., prior knowledge on the topic of the game, game experience), demographic information (e.g., age, sex), and other information that may influence performance and learning in the game (e.g., motivational information). The second class of information is related to the game experience itself. Contextual information during the game can be the values of various game state variables, type of feedback, or any other information that may qualify the data.

In our log files we capture information about the student in the *user_id* field, which can be linked to a separate dataset containing additional information about the student. We capture information about the game experience in the *game_state* field, which records information about the state of the problem, as well as various individual *data_code* entries

that record specific game information or game events (e.g., when feedback is given, the version of the game that is being played).

**Example.** In a game on fractions that CRESST developed (*Save Patch*), one event that is logged is an incorrect addition attempt. In addition to the standard information that is logged with each data element (serial number, timestamp, game time, user ID, stage, and level; see Table 1), event-specific data are logged as shown in Table 2. The entry for an incorrect addition attempt is assigned data code 3011 and specific context information is captured by the three data columns—where in the game board the error occurred (data column 1), the value being added incorrectly (data column 2), and the existing value on the game board (data column 3). By assigning the data element to a class (incorrect addition) and recording specific context associated with the event (position, incorrect addend, existing addend), the incorrect additions by learners can be summarized (e.g., by counting all data elements with data code 3011 and analyzing by game level, by individuals, or by classes) or subjected to more precise analyses (e.g., examining the types of denominators used in the incorrect addition by level).

Table 2

Sample Data Log Entry For Data Code 3011 (Incorrect Addition) in the CATS Game *Save Patch*.

| Data code | Data description | Data 1 | Data 2 | Data 3 |
|---|---|---|---|---|
| 3011 | added ropes incorrectly: at `[position]` tried to add `[value added]` to value `[existing value]` | `Position` | `value added` | `existing value` |

Table 3

Snippet from the Raw Log File in the CATS Game *Save Patch*

| S/N | Time-stamp | Game time | User ID | Stage | Level | Data Code | Data Description | Data 1[a] | Data 2 | Data 3[b] | Game state[a] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 448 | 6/28/2011 9:51:57 AM | 1068.74 | 151fish | 1 | 10 | 3000 | made an individual selection: selected rope of value [rope value] | 1o1 | | | 3o2_0o2 0o1\|2o2_0o2 0o1\|0o2_0o2 0o1 |
| 449 | 6/28/2011 9:51:59 AM | 1070.351 | 151fish | 1 | 10 | 3010 | added ropes to a sign: at [position] added [value added] to yield [resulting value] | 0o2_0o2 | 1o1 | 1o1 | 3o2_0o2 0o1\|2o2_0o2 0o1\|0o2_0o2 1o1 |
| 450 | 6/28/2011 9:52:02 AM | 1073.413 | 151fish | 1 | 10 | 2000 | toggled fraction on rope: from [old value] to [new value] | 1o1 | 2o2 | | 3o2_0o2 0o1\|2o2_0o2 0o1\|0o2_0o2 1o1 |
| 451 | 6/28/2011 9:52:03 AM | 1074.053 | 151fish | 1 | 10 | 3000 | made an individual selection: selected rope of value [rope value] | 1o2 | | | 3o2_0o2 0o1\|2o2_0o2 0o1\|0o2_0o2 1o1 |
| 452 | 6/28/2011 9:52:05 AM | 1075.823 | 151fish | 1 | 10 | 3010 | added ropes to a sign: at [position] added [value added] to yield [resulting value] | 2o2_0o2 | 1o2 | 1o2 | 3o2_0o2 0o1\|2o2_0o2 1o2\|0o2_0o2 1o1 |
| 453 | 6/28/2011 9:52:08 AM | 1077.714 | 151fish | 1 | 10 | 3000 | made an individual selection: selected rope of value [rope value] | 1o2 | | | 3o2_0o2 0o1\|2o2_0o2 1o2\|0o2_0o2 1o1 |
| 454 | 6/28/2011 9:52:08 AM | 1078.154 | 151fish | 1 | 10 | 3011 | added ropes incorrectly: at [position] tried to add [value added] to value [existing value] | 0o2_0o2 | 1o2 | 1o1 | 3o2_0o2 0o1\|2o2_0o2 1o2\|0o2_0o2 1o1 |

| S/N | Time-stamp | Game time | User ID | Stage | Level | Data Code | Data Description | Data 1[a] | Data 2 | Data 3[b] | Game state[a] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 455 | 6/28/2011 9:52:09 AM | 1079.166 | 151fish | 1 | 10 | 3000 | made an individual selection: selected rope of value [rope value] | 1o2 | | | 3o2_0o2 0o1\|2o2_0o2 1o2\|0o2_0o2 1o1 |
| 456 | 6/28/2011 9:52:11 AM | 1080.709 | 151fish | 1 | 10 | 3010 | added ropes to a sign: at [position] added [value added] to yield [resulting value] | 3o2_0o2 | 1o2 | 1o2 | 3o2_0o2 1o2\|2o2_0o2 1o2\|0o2_0o2 1o1 |
| 457 | 6/28/2011 9:52:14 AM | 1082.56 | 151fish | 1 | 10 | 3020 | submitted answer: clicked Go | | | | 3o2_0o2 1o2\|2o2_0o2 1o2\|0o2_0o2 1o1 |

*Note*. [a]A list of the values currently placed on the grid. List the position of the sign, space, the value on the sign. Separate signs with "|". Do not use "," or (\t) to delimit values. All numeric values are written as numerator, o (for over), denominator (i.e. 1o1). All positions are written as horizontal location, underscore, vertical location (i.e. 1o1_1o2). For example: 1o1_0o1v1o2|1o1_1o1v0o0
[b]Data columns 4 to 7 are omitted because they are empty.

## Deriving Measures from Log Data

In general, three types of measures can be derived from game play: (a) game performance, (b) in-game learning, and (c) in-game strategies. Each type of measure has certain uses and the measure used in an analysis depends on the question being asked.

### Overall Game Performance

Measures of game performance reflect a player's achievement in the game. For example, in *Save Patch*, the last level attained was the primary game performance measure because it was the most direct measure of achievement. Had we designed the game differently, there might be other measures of overall game performance (e.g., total score, number of achievements). When designing measures of overall game performance, the important questions are: (a) What behaviors or game states reflect overall achievement in the game? (b) Can overall achievement in the game be partitioned into components related to learning, game skills, and engagement and motivation?

### In-game Performance

Measures of in-game performance are game dependent and are derived from an analysis of the cognitive demands required of the game. In *Save Patch*, the in-game performance measures reflect the math knowledge that presumably underlies overt behavior. A measure of poor in-game performance is the number of unsuccessful events reflecting the adding fractions operation such as resets and deaths. When designing in-game measures, the important questions are: (a) What in-game behaviors reflect cognitive demand *X*? (b) What behaviors might reflect productive and unproductive use of cognitive demand *X*? (c) What behaviors might reflect common errors in the domain? (d) What if any transformations need to be applied to the raw data to adjust for game design (e.g., normalization procedures to adjust for how far the learner progressed in the game)?

### In-game Strategies

Compared to in-game performance measures, measures of in-game strategies can be derived from aggregated performance, performance classifications, or other means of describing a player's game play over time. The goal of measuring strategies is to be able to summarize how a player's game play unfolded over the course of the game level (or other unit of time). Thus, data are gathered over time and subjected to various types of analyses that take order of player events into account (e.g., Markov chain analyses, time series analyses, lag sequential analyses) or sets of co-occurring player events (e.g., cluster analyses, neural network analyses). When designing measures of in-game strategies, the important

questions are: (a) What sets or sequences of in-game behaviors reflect cognitive demand *X*? (b) What sets or sequences of in-game behaviors might reflect productive and unproductive use of cognitive demand *X*? (c) What sets or sequences of in-game behaviors might reflect common errors in the domain? (d) What if any transformations need to be applied to the raw data to adjust for game design (e.g., normalization procedures to adjust for how far the learner progressed in the game)?

For our CATS game, we have used cluster analyses to identify sets of co-occurring events that reflect the ideal solution (presumably reflecting adequate fractions knowledge), errors that are consistent with common fraction misconceptions, and game strategies that are not mathematical in nature.

## Analytical Approaches to Developing Measures from Log Data

In general, procedures for developing measures can be done directly from the game design itself (referred in this document as *a priori*), from mining of the data (referred in this document as *post hoc*), or a combination of both.

### Measures Defined a Priori (Based on Game Mechanics)

Measures based on game mechanics should reflect, as directly as possible, the targeted knowledge and skills. The more directly a game mechanic supports a cognitive operation, the more likely that measure will be sensitive to differences in knowledge. For example, in *Save Patch*, one targeted outcome of the game is the idea that only quantities with the same unit can be added together. In fractions, this concept is reflected by addition of fractions with the same denominator. A central game mechanic is adding together objects (e.g., pieces of rope) that represent fractional pieces of a whole unit. The act of adding two pieces is recorded as either a successful addition or an unsuccessful addition. Contextual information such as the value of the numerator and denominator is recorded as well, and if the addition was unsuccessful, where in the solution path the error occurred. The single data element provides information on the nature of the error, when the error occurred, and where the error occurred. The measure can be used singly by aggregating across data code 3011 (i.e., the number of addition errors) or as part of data mining procedure whereby the data element is far more specific and unique (e.g., error with numerator *x*, denominator *y*, location *z*). Similarly, the requirement for students to press a "go" button when they think they have the solution to the level is a proxy for learners' judgment of solution adequacy.

The key point is that designing game mechanics to require use of particular knowledge will result in a measure that will be sensitive to the presence or absence of that knowledge.

Similarly, a game mechanic designed to invoke the desired type of cognitive processing will result in a measure that will be sensitive to productive (or unproductive) processing.

**Measures Defined Post Hoc (Discovery of Patterns)**

Measures based on the discovery of interesting patterns are more tenuous in that once a pattern is identified, the pattern needs to be interpreted in light of the task and the student's presumed knowledge of the domain. As is true of *a priori* measures, the discovered patterns of student behavior must reflect the targeted knowledge and skills for those patterns to be sensitive to differences in knowledge. In the CATS work, we have used cluster analyses of game events to identify sets of events that co-occur. The cluster of events is interpreted given the particular level design (resources, complexity) and the math knowledge targeted in the level.

Patterns of student behavior can be identified from the log files using data mining techniques such as cluster analysis (Merceron & Yacef, 2004; Romero & Ventura, 2007). Cluster analysis groups individual actions into patterns of behavior by determining which actions co-occurred (Berkhin, 2006; James & McCulloch, 1990; Romero, Gonzalez, Ventura, del Jesus, & Herrera, 2009). Two individual actions are considered to belong to the same pattern of behavior (cluster) if they are both made by the same students. Two individual actions are considered to belong to different patterns of behavior (clusters) if the two actions are made by two different groups of students.

For example, in *Save Patch*, cluster analysis was able to identify groups of actions that reflected different patterns of student behavior within a level (Kerr, Chung, & Iseli, 2011). These groupings were then interpreted, given the level design and targeted math knowledge, as indicators of different strategies students were using to solve game levels. For instance, some students appeared to attempt to solve levels using correct mathematical techniques, others appeared to believe specific mathematical misconceptions, and still others appeared to attempt to solve levels by using game strategies rather than mathematical techniques.

The design of the game mechanic to require use of particular knowledge and the design of the game levels to represent different levels of knowledge will result in a *post hoc* measure that will be sensitive to specific levels of knowledge. Similarly, the design of the game mechanic to invoke desired types of cognitive processing and the design of the game levels to represent different levels of cognitive processing will result in a *post hoc* measure that will be sensitive to specific levels of each type of cognitive processing present in the game. Once a high level of confidence exists in the discovered patterns, then software can be developed for automated detection of the patterns.

## Usage

In this section we briefly describe how the log data are used. In general, we use log data to support (a) game forensics—what players were doing at a particular point in the game - and (b) pedagogical research.

### Forensics

We refer to game forensics as the process of attempting to understand what players were doing at a particular point in the game. Log data are particularly useful when recorded at a resolution that corresponds directly to user interface events. We have engaged in game forensics to resolve problems with data collection and to explore anomalies in game play. For example, given unusual behavior, we have examined specific levels for design features that make it difficult for students to understand the task demands. We also found misleading level design or feedback, and identified features of a level that contribute to difficulty.

### Research

The primary use of log data is to support the development of the measures. Once the measures are operationalized, values are assigned to each person based on their individual performance on all of the measures. These "scores" are then fused with other data and become part of a set of analyses to answer research questions such as:

- **Which game design is most effective for whom?** To answer this question, game-based measures are compared across different game designs or different populations.
- **For those students who learn from the game experience, what did they do in the game?** To answer this question, relationships between pretest-posttest gains and in-game measures can be examined.
- **To what extent do players with different backgrounds do things differently in the game?** To answer this question, relationships between player background and game play measures can be examined.
- **To what extent does degree of knowledge, game experience, ELL status, or other background characteristics relate to performance in the game?** To answer this question, relationships between pretest and game play measures can be examined.
- **To what extent does game play predict performance on the outcome measure?** To answer this question, relationships between game play measures and the posttest can be examined.
- **To what extent does opportunity-to-fail predict performance on the outcome measure?** To answer this question, relationships between exposure to game play designed around common errors and the posttest can be examined.

## Summary

In this primer we briefly described our perspective and experience in using data logging to support measurement of student learning in a game. Our general approach is to derive cognitively meaningful measures and affectively meaningful measures from a combination of player behaviors, game events, and game states. A variety of analytical and practical issues arise, particularly the need to specify the behavior to log ahead of time, logging in-game behaviors that map directly to targeted knowledge, skills, and attitudes, the use of a structured record format, and capturing of context to allowing linking of the data element to an individual's specific game experience (e.g., school, teacher, period, stage, level, event, game state).

Our general approach is to record data that reflects behavior rather than inferences about the behavior. Some of our best practices include encoding sufficient information in the data element so that the data elements are unambiguous at the desired grain size, ensure there is a link in the data to an individual and specific game state, and use of a structured and delimited record format.

By starting with a clear idea of the knowledge and skills to measure, the data logging design becomes simple and focused. The capturing of behavioral events that presumably is a manifestation of cognitive and affective processes allows for the investigation of numerous research questions that connect game play to students' background, strategy use, knowledge, and cognitive processes.

## References


Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis* (2nd ed.). Cambridge, UK: Cambridge University Press.

Berkhin, R. (2006). A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, & M. Teboulle (Eds.), *Grouping multidimensional data* (pp. 25-72). New York, NY: Springer.

James, F., & McCulloch, C. (1990). Multivariate analysis in ecology and systematic: Panacea or Pandora's box? *Annual Review of Ecology and Systematics*, *21*, 129-166.

Kerr, D., Chung, G. K. W. K., & Iseli, M. R. (2011). *The feasibility of using cluster analysis to examine log data from educational video games* (CRESST Report 791). Los Angeles: University of California, National Center for Research on Evaluation, Standards, and Student Testing (CRESST).

Linn, R. L., & Gronlund, N. (2000). *Measurement and assessment in teaching*. Upper Saddle River, NJ: Prentice-Hall.

Merceron, A., & Yacef, K. (2004). Mining student data captured from a web-based tutoring tool: Initial exploration and results. *Journal of Interactive Learning Research, 15*, 319-346.

Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert systems with applications*, *35*, 135-146.

Romero, C., Gonzalez, P., Ventura, S., del Jesus, M. J., & Herrera, F. (2009). Evolutionary algorithms for subgroup discovery in e-learning: A practical application using Moodle data. *Expert systems with applications, 39,* 1632-1644.

## Appendix A:
## Brief Summary of Save Patch


*Figure 1*. Screen shot of *Save Patch.*

The game *Save Patch* was designed to teach the concept of a unit in rational numbers. We first developed general specifications around two key ideas about rational numbers. The first idea is that all rational numbers (integers and fractions) are defined relative to a single, unit quantity (e.g., a unit of count, measure, area, volume). The second idea is that rational numbers can be summed only if the unit quantities are identical (e.g., 1/4 + 3/4 is permissible but 1/2 + 3/4 is not because the unit or size of the fractions is unequal). These two ideas formed the basis of what we expected to measure from students' game play. Table A1 an example of the specifications for the idea of unit. Additional specifications included the meaning of addition, the meaning of the denominator in a fraction, the meaning of the numerator in a fraction, and the idea that any rational number can be written as a fraction.

Table A1

Sample Excerpt of the Specification for the Idea of *Unit* in Rational Numbers

| Item | Objective |
|---|---|
| 1.0.0 | Does the student understand the meaning of one unit in the context of rational numbers? One unit can be descriptive (e.g., apple, car, rocket) or quantitative (interval on a number line, a kilometer, a square foot, etc.) and may be stated implicitly or explicitly. |
| 1.1.0 | The size of a rational number is relative to how one Whole Unit is defined. |
| 1.2.0 | In mathematics, one unit is understood to be one of some quantity (intervals, areas, volumes, etc.). |
| 1.3.0 | In our number system, the unit can be represented as one whole interval on a number line. |
| 1.3.1 | Positive integers are represented by successive whole intervals on the positive side of zero. |
| 1.3.2 | The interval between each integer is constant once it is established. |
| 1.3.3 | Positive non-integers are represented by fractional parts of the interval between whole numbers. |
| 1.3.4 | All rational numbers can be represented as additions of integers or fractions. |

The specifications in Table A1 guided the development of the game mechanics (the main game play operations). A key property of the game, if it was to have assessment utility, is that game play required the cognitive demands outlined in Table A1 (i.e., the two central ideas of unit and addition of like-sized unit quantities). The game design reflected these key ideas in two ways. First, the basic task presented students with essentially a number line, where whole units were demarked with vertical posts and each whole unit could be further divided into fractional pieces (demarked by smaller posts). The game scenario was to help the character, Patch, move from his or her initial position to the goal position to free the trapped cat (the cage in the screen shot in Figure 1). Patch could only move by following a path that was specified by ropes, and the distance Patch traveled was determined by the size of the rope segment. Players specify the distance and direction that Patch travels at each sign post by adding rope segments to the sign.

Successful game play required students to determine the size of the whole unit for a given grid and also the size of any fractional pieces. The second component, additive operations only allowed on like-sized units, was carried out via the game scenario of adding rope segments to the sign post so Patch would travel the appropriate distance. The distance traveled was a function of how many rope segments were added to a sign post. The size of the rope corresponded to a whole unit (1/1) or a fractional unit (e.g., 1/2), and when adding ropes to the sign post, only same-sized rope segments were allowed. This adding operation corresponded to adding fractions with common denominators, and also in the solution to the

level. A successful solution resulted in *Save Patch* traveling from sign post to sign post to the goal position, which mathematically was the sum of all sign post values

## Appendix B:
## Data Code Definition for Save Patch

Table B1

Data Code Definitions for CATS Game *Save Patch*

| | | Data columns | | | |
|---|---|---|---|---|---|
| Data code | Data description | 1 | 2 | 3 | 4 |
| **1000 to 1499: Game startup information, such as start time, build, and level notes.** | | | | | |
| 1000 | application loaded from [path] | path | | | |
| 1010 | game startup: [current time] | current time | | | |
| 1020 | game version: [build] | build | | | |
| 1021 | study condition: [condition] | condition | | | |
| 1022 | login id: [login] [verification file] [verification row number] | login | verificat- ion file | verification row number | |
| 1023 | login data: [district] [school] [teacher] [period] | district | school | teacher | period |
| 1030 | game notes: [game notes] | game notes | | | |
| 1100 | level notes: [stage number] – [level number] [level notes] | stage number | level number | level notes | |
| 1110 | level solution: sign [grid location] direction [direction] value [solution value] | grid location | direction | solution value | |
| 1120 | level resources: [denominator values] provided equals [number of ropes in that row of the resource bin] | Denominator value —ŕ Number of ropes in that row of the resource bin | | | |
| 1200 | tutorial notes: tutorial level [stage number] – [level number] about [tutorial content] | stage number | level number | tutorial content | |

| Data code | Data description | Data columns | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 |
| 1300 | feedback notes: [stage number] – [level number] [feedback notes] | stage number | level number | feedback notes | |
| **1500 to 1999: Game end information, such as end time or a saved game relaunch.** | | | | | |
| 1500 | game end: [current time] | current time | | | |
| 1510 | load saved game: loaded into level [current level] | stage number | level number | | |
| **2000 to 2999: In-game manipulation of objects, such as toggling fractions.** | | | | | |
| 2000 | toggled fraction on rope: from [old value] to [new value] | old value | new value | | |
| 2010 | changed sign direction: changed [position] from [original direction] to [new direction] | position | original direction | new direction | |
| 2020 | scrolled resources: scrolled table [direction] to show [resources shown] | direction | resources shown | | |
| **3000 to 3999: In-game decisions, such as adding fractions.** | | | | | |
| 3000 | made an individual selection: selected rope of value [rope value] | rope value | | | |
| 3010 | added ropes to a sign: at [position] added [value added] to yield [resulting value] | position | value added | resulting value | |
| 3011 | added ropes incorrectly: at [position] tried to add [value added] to value [existing value] | position | value added | existing value | |
| 3020 | Submitted answer: clicked Go | | | | |
| 3021 | moved: direction [direction] from [position] value [value] | direction | position | value | |
| 3030 | clicked undo: removed [value] from [position] | value | position | | |
| 3040 | closed feedback: closed [feedback filename] | feedback filename | | | |

| Data code | Data description | Data columns 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **4000 to 4999: Success states such as deaths, resets, feedback, or success.** | | | | | |
| 4000 | player death: died after moving `[value]` from position `[position]` | `value` | `position` | | |
| 4010 | reset level: reset | | | | |
| 4020 | feedback given to student: received `[description of feedback]` showing `[text of feedback]` | `description of feedback` | `text of feedback` | | |
| **5000 to 5999: Game navigation, such as advanced to next level.** | | | | | |
| 5000 | advanced to the next level: `[stage number]` `[level number]` | `stage number` | `level number` | | |
| **6000 to 6999: Help menu system.** | | | | | |
| 6000 | opened help menu: `[help file name]` | `help file name` | | | |
| 6010 | closed help menu: `[help file name]` | `help file name` | | | |
| **7000 to 7999: In-game assessment system.** | | | | | |
| 7000 | question loaded: `[question file name]` | `question file name` | | | |
| 7010 | answered question: answered question `[question file name]` with `[answer selected]` | `question file name` | `answer selected` | | |
| 7020 | question closed: `[question file name]` | `question file name` | | | |
| **8000 to 8999: Reserved for testing and survey system.** | | | | | |
| **9000 to 9999: Reserved for Bayes net system.** | | | | | |

*Note*. All numeric values are written as numerator, o (for over), denominator (i.e. 1o1). All positions are written as horizontal location, underscore, vertical location (i.e. 1o1_1o2). Variables are indicated by a font change and are enclosed by square brackets, i.e. `[current time]`.