2024 IEEE Conference on Games (CoG) | 979-8-3503-5067-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/CoG60054.2024.10645573

Open Game Data: Defining a Pipeline and Standards for Educational Data Mining and Learning Analytics with Video Game Data

David J. Gagnon
Field Day Lab
University of Wisconsin - Madison
Madison, USA
https://orcid.org/0000-0001-5126-0476

Luke Swanson
Field Day Lab
University of Wisconsin - Madison
Madison, USA
https://orcid.org/0000-0002-8455-7441

Erik Harpstead
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, USA
https://orcid.org/0000-0003-3019-3627

Abstract—**In this paper we describe the need for a framework to support collaborative educational research with game data, then demonstrate a promising solution. We review existing efforts, explore a collection of use cases and requirements, then propose a new data architecture with related data standards. The approach provides modularity to the various stages of game data generation and analysis, exposing intermediate transformations and work products. Foregrounding flexibility, each stage of the pipeline generates datasets for use in other tools and workflows. A series of interconnected standards allow for the development of reusable analysis and visualization tools across games, while remaining responsive to the diversity of potential game designs. Finally, we demonstrate the feasibility of the approach through an existing implementation that uses this architecture to process and analyze data from a wide range of games developed by multiple institutions, at scale, supporting a variety of research projects.**

Index Terms—**games, educational data mining, learning analytics**

I. Introduction

The digital nature of educational video games affords low cost distribution to arbitrarily large and diverse audiences. This effect is highlighted with web-based games which can be deployed nearly instantly and are popular in schools, in no small part in connection to the historic investment into chromebook devices and high speed internet in schools following the COVID-19 pandemic. A 2021 survey of school leaders reported that 90 percent of students in grades 6-12 and 84 percent of grade 1-5 in the United States have a personal chromebook device [1]. Studies over the last few years demonstrate that game adoption has grown mainstream, with users of games mimicking the general population across dimensions such as gender, age and race [2].

Despite the ease of distribution, game interaction data can be quite rich. As digital systems, players are interacting with simulated environments where every attribute is stored as some part of the systems state. Every interaction made by a player is similarly well-defined. These state changes and interactions are both high frequency, but also fine-grained. When carefully considered through interpretive lenses such as Evidence Centered Design [3], game interaction data becomes a method for understanding player beliefs, attitudes and knowledge. Following, games offer an superb tool for educational research, supporting research in fields such as cognitive science, learning science and learning engineering. Breakthroughs in Education Data Mining and Learning Analytics [4] provide the mechanisms to conduct these forms of research with very large audiences by automating the analysis methods. Researchers have used these approaches to detect computational thinking strategies [5], predict quitting [6] and wheel spinning [7], and assess content knowledge [8].

Unfortunately, there is very little shared research infrastructure or even widely adopted conventions that facilitate research with educational games. Some researchers may have access to the variety of resources required to develop and instrument games, recruit audiences, develop data capture and systems and then analyze the data, but for many individuals or smaller teams, the scope of these efforts is a significant barrier. Practically this results in game research either being conducted by a small number of institutions that have all these capacities in-house, or much smaller studies that do not benefit from the affordances of large datasets from large audiences.

This paper is a response to the promise of using large-scale educational game data in concert with educational data mining and learning analytic approaches, and offers an initial step in developing a shared research infrastructure. We begin by describing the need for a new data pipeline for conducting educational game data research that enables open collaboration between developers and researchers, flexibility in game designs and research agendas, and modularity to support the reuse of tools and methods across games. Next, we provide a proposed architecture for such a pipeline and standards for how new games can be added and analysis developed. Finally, we demonstrate an initial implementation of this pipeline and

This work was supported by the National Science Foundation under grants 2243668, 2142103, 2116046, 1907384 and The Learning Agency TOOLS competition

standards in a production research infrastructure being used by various institutions across 20 diverse learning games and millions of players yearly.

II. Existing Approaches to Data in Games Research

Commercial tools have been developed to provide analytics capacities to game developers who do not wish to develop their own infrastructure. The popular Unity game engine provides an integrated analytics platform, Unity analytics, that provides game developers with key metrics and visualizations of their games performance and player experience, such as the daily active users and average engagement time. The Game Analytics product provides similar analysis and works for any game engine. Both products continue to grow and expand, recently adding capacities such as A/B testing for developers to conduct experiments. Both products are primarily focused on providing analytics that will help monetize games, retain users, and improve design. Neither attempt to create a pipeline for data processing, or create or share reusable analysis.

Turning to conventions for interoperable game data, Charleer et al. [9] proposed the xVGM specification that is presented as a simplification of the well known xAPI specification designed for describing player actions within competitive multiplayer games like League of Legends. While the specification is inspired by xAPI it only makes use of the high level event structure and eschews the use of interoperable URIs for verb descriptions, instead relying on the fact that each game would define its own verb set.

The Serious Game Profile for xAPI (xAPI-SG) [10] is another approach that has been taken in the broader context of Learning Analytics in games. This specification provides additional games specific semantics to make it easier to express game actions within the broader language of xAPI. This includes activities for levels and missions, as well as actions for progression and completion of challenges. While this specification does adhere to established standards, the authors note challenges related to its verbosity and potential to face bandwidth issues in the context of highly dynamic games. Despite these challenges, analytics aggregation and reporting systems have been built on top of the xAPI-SG profile. For example, the T-Mon system [11] provides tools for automatically aggregating and calculating progression and time metrics.

III. Intended Audiences and Use Cases

This paper is not purely theoretical, but a blueprint for a data architecture based on previous experiments into game data. While there are certainly unexplored approaches to come, we have already seen a number of promising uses of game data analysis for different stakeholders and use cases that this data pipeline should support.

We focus on four high level stakeholders for the outputs of game data analysis: designers, researchers, educators and players. Designers need ways of understanding players experiences in order to improve the game. Researchers require ways to discover patterns, and conduct experiments that will further our understanding about how players think, learn and feel. Similarly, educators need tools that help them understand how learners are thinking and progressing so they form interventions. Finally, players need ways to understand their own thinking and affect so they can make changes to their own playing and learning strategies, and their games need ways of adapting to individual players.

Across these stakeholders and previous research, we focus on six use cases for a data pipleine:

Learning Engineering with Games. Learning engineering is an emerging field that uses evidence to inform educational design. For example, researchers developed and disseminated different versions of a games script [12] and various conditions of difficulty and support [13] to large audiences to determine which versions produced desirable outcomes and inform design theory.

Player Experience Visualization. Design researchers are in constant development of new ways to visualize play experiences. Some of the promising directions include simple heat mapping [15] to more complex analysis to consider pathways for progression [14].

Player and Player Experience Classification. Player typologies are useful tools for user-centric design, as well as uncovering unexpected patterns of play [15].

Replay of Player Experience. Replay is a process of using game data to recreate a representation of the original play experience. Among other examples, text replay logs have been used to qualitatively label segments of gameplay to identify players who are struggling [16], and video replays have been used to identify specific computational thinking approaches [5].

Realtime detectors to support qualitative research. Similar to replay, machine learning analysis can be used to support qualitative researchers, but in real time, while the gameplay is taking place to direct the focus of costly research [17].

Realtime analysis for summarizing player experiences. Researchers have developed web based [18], and augmented reality [19] dashboards to synthesize player data and communicate useful insights to educators to augment their perceptions of how learners are performing. These models could also be applied to adapt gameplay in response to player performance.

IV. Design Principles for the Data Pipeline and Standard

In response to the opportunities for using game data, the previous work and the use cases this project intends to support, we develop a collection of goals and properties of the required data pipeline and standards.

Principle 1: Data Follows Design. The data structures and vocabulary needs to be extremely flexible, in the same way that game design is flexible. While conventions exist and will emerge, their adoption remains optional. Similarly, adopting this pipeline and standard should require the minimal amount of effort as possible of the game studio and pose no risk to game performance.

Principle 2: Capture then Analyze. Enough data should be captured so that unforeseen future analysis can be conducted by researchers outside the game developers or initial research team. Optimally, enough data should be present to completely reconstruct the game experience from telemetry events.

Principle 3: Support a Diversity of Workflows. In contrast to creating large, integrated infrastructure with many features, provide users with the easiest to use interfaces so they can use the tools of their choosing. For example, provide outputs in simple formats that are widely compatible, such as .TSV files so that even novice researchers are able to work with the datasets in the tools that are most comfortable to them. Similarly, the pipeline should support the development of many reusable tools that are loosely coupled to any particular game or research agenda through a series of emergent and optional standards, minimal requirements for leveraging a particular tool or analysis, that each developer can choose to adopt.

Principle 4: Transparency and Openness. The pipeline should expose both the logic as well as the intermediate work outputs and many phases for outside review and secondary analysis.

V. Open Game Data Pipeline v0.1

The proposed pipeline (see Fig. 1) contains several components that operate primarily in a linear fashion and are each designed to take the results of the prior component as input, starting with the games themselves. The first stage of the pipeline is a data **Logger**. This component captures the time series telemetry events directly from the games, commits them to long term storage, and makes them available for use as downloadable files and via an API. The next stage in the pipeline is the **Detector** component. Detectors are simple algorithms or models that inspect the time series, raw data in order to infer events that are not logged from the games themselves e.g detecting a idleness, or frustration. Detectors insert calculated events within the time series in-between the raw events from the game.

From this stage forward, each of the components are operating as some level of aggregation, transforming the sequences of events into features that describe the different attributes of play experience. In each case, a row in the output is the level of aggregation (player or population) and columns define individual features. The first step in this process is to calculate the features that describe a specific **Player**. These featured are then further aggregated into **Populations**. Populations are used for any segmentation of players, such as all players or player of a specific game version.

In the first break from a purely linear progression, **Higher Order Features** consider Players in context of entire Populations. The simplest example of this would be a player feature that considers that single players session count in comparison of the number of sessions of all the players in the population. In thus case, the result of the analysis would be additional features added back to Players after the Population features had been calculated.

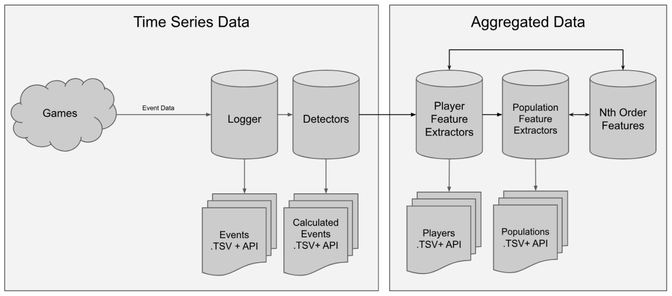

Fig. 1. The proposed data pipeline.

VI. Data Standards v0.1

We now propose initial specifications for the standards identified in the previous section. As we have discussed above, our goal is not to create a new set of detailed, rigorous standards that require significant implementation effort. Instead, our intent is to create simple, flexible conventions that balance the advantages of shared data formats against the disadvantages of high-maintenance implementations of high-detail specifications. The standards we propose include interchange formats for time-series event and aggregate feature data, both for tabular dataset distribution (i.e. .tsv files) and for web APIs. We also propose a series of smaller ”ontology” standards that can be implemented by individual games to ensure compatibility with external tools and reusable analysis methods.

A. Event Data Standard

In table I, we document the set of properties every individual event must include, and indicate whether ”null” is a valid value for each. Our standard for events includes elements within five categories, namely identifiers, sequencing, versioning, user metadata, game state, and event data. As described in the pipeline segment on event data, events may be generated directly from the game, or inferred from the content of game-logged events. We call the latter “calculated events.” The structure of a calculated event does not differ from the standard game-generated event; we distinguish between log events and calculated events with the event source element. Event detectors that generate calculated events should fill in all elements implemented by the game whose data the detector operates on. In general, the best practice is to copy identifier, versioning, and sequencing items from whatever game event triggered the detector. There is one exception, namely for instance and session sequence indices. Simply copying these would result in redundant indexing. Instead, a system implementing event detectors should re-index subsequent events whenever a calculated event is inserted into the event stream.

We now describe each category of event properties and, as necessary, specific rules for each property:

1) Identifiers: are used to indicate the application, game instance, user, and user-session from which a given event is logged. The instance ID is optional, supporting multiplayer games where instances of the game system exist independently

TABLE I
STANDARD PROPERTIES FOR EVENT DATA

Property	Data Type	Allow NULL	Description	Example
game_id	String, 32	NO	A string identifying which game from which the event came	MY_GAME
instance_id	String, 32	YES	For multiplayer games, an ID for the instance of the game that generated the event.	48272075
player_id	String, 32	YES	A custom, per-player ID that may be re-used across game sessions	BlueWhale
session_id	String, 32	NO	Unique identifier for the gameplay session	24020195392739482
timestamp	Datetime	NO	A UTC timestamp for when the event was logged	2024-02-22 00:00:11
offset	timedelta	NO	The local offset of the event time from GMT	+6:00:00
instance_sequence_index	Int	YES	Counter of events in the game instance, for multiplayer games that use instancing, starting from 0. A row with index = i is the (i+1)-th event on the game instance	78
session_sequence_index	Int	NO	Counter of events in the session,starting from 0. A row with index = i is the (i+1)-th event of the session	13
event_name	String, 32	NO	The type of event logged	start_game
event_source	String,32	NO	GAME if the event was logged directly from the game, or the name of the event detector module that generated the event.	GAME
source_hash	String, 32	YES	A hash string indicating the precise revision or version of the source that generated the event	a63eb9
event_data	JSON	NO	Data specific to an event type, encoded as a JSON-formatted string	`{ "continue":TRUE, "language":"ENG" }`
game_state	JSON	YES	Metadata about the state of the game when the event occurred	`{ "score" : 72, "level" : 5 }`
player_metadata	JSON	YES	Metadata specific to a player ID	`{ "join_date" : 1/1/2022 }`
game_version	String, 32	NO	The version of the game from which the event came. Convention is a semantic versioning-style string: `[Major].[Minor].[Patch]-[Branch]`	1.2.1-no_narrative
game_log_version	Int	NO	The version of the logging code for the game from which the event came. Format is a single, strictly-increasing number.	7

of the individual players. For non-multiplayer games, this element may be left null. Similarly, user ID is optional, used by games that implement some form of player identification for repeat plays. Other games may leave this item null and use only the session ID.

2) Sequencing: The sequencing elements include a UTC-formatted timestamp and timezone offset, as well as an instance sequence index and session sequence index that indicate a strict ordering of events within a given game instance and session, respectively. These indices help avoid any potential issues with timestamp precision on older storage systems that may lead to ambiguous event sequencing. Note, while we specify the timestamp and offset as datetime and timedelta data types, respectively, appropriately formatted strings may be used for systems that do not implement equivalent data types.

3) User Metadata: This category consists of a single, JSON-formatted object tracking information about an individual user. Games that do not implement a user identification scheme may leave this element null. The specific members of the JSON-formatted object are left to the implementer, but all events from games sharing a common user identity system should also share a common set of elements in this object.

4) Game State: This consists of a single, JSON-formatted object to track the state of a given game at the instant an event occurred. As with user metadata, the specific members of the JSON-formatted object are left to the implementer. However, all events within a single game should use the same set of elements. Game state is optional, and may be left null; the recommended best practice is to include elements of game state that are generally useful in contextualizing player behaviors in later analysis, such as current level or player total score.

5) Event Data: This consists of a name of the specific event type, an event source, and a JSON-formatted object for the specific details of the event that occurred. Again, the specific elements of the JSON object are left to the implementer. Unlike the other JSON objects, the event data object will contain a different set of elements for each unique type of event. Where game state records some contextual state data at the moment the event occurred, the event data JSON object records information about changes in state as a result of the event. For example, an event for completing a puzzle would record the score immediately before the puzzle was completed in the game state, and the new score as a result of finishing the puzzle in the event data.

6) Versioning: These elements are used to independently track the game experience, event logging set, and code revision that collectively led to the generation of an event. The app version refers to the version of the game experience itself. This should be specified with a semantic versioning scheme [20]; we relax the usage of the pre-release version to allow any branch name. For example, a game may have an official release that logs the app version as 1.0.0-main, as well as a temporary release with an edited script for a research study, whose version is 1.0.0-newscript. This allows experimental versions of a game to be used without needing to fit into the sequential progression of the games development.

The "log version," on the other hand, is a single, strictly-increasing integer value that reflects the version of the game's event specification. Note that because logging code exists only to record the occurrences within a game, it is assumed that there are no branching versions, only an increasing set of possible events. The log version should be increased any time new changes to the logging code are deployed, and *must* be increased when the internal structure of any events data elements are changed.

Finally, the "source version" is a precise identifier of the version of the event source that generated the event. Generally, this should be a revision hash from a version control system. The source hash must be unique for the given source. Then the combination of game_id and source_hash gives a unique identifier for the source of a game-generated event, and event_source + source_hash provides a unique identifier for detector-generated events.

B. Events File Format

For distribution of full datasets, a tab-delimited file should be used. Character-delimited files are unparalleled in their compatibility with existing data tools. Nearly every package for data analysis is capable of reading comma-delimited files, and typically this capability spreads to other character delimiters. Our standard uses tab-delimitation in order to allow for the usage of comma-delimited JSON data elements. This is a conscious compromise between flexibility across games, consistency between games, and compatibility with analysis tools. Each element of the event data structure, then, is a column of the tabular TSV file. Any complex data types, such as timestamp and JSON elements, should be rendered into an appropriately formatted string. Event files containing only the original game-logged events should be named with a suffix of `_raw-events`, and event files containing both game and generated events should be given a suffix of `_all-events`. Additionally, every TSV dataset file should be accompanied by a README.md and a META.json file.

The README.md file is intended as human-readable documentation of the dataset contained in the TSV. It should include license information, a suggested dataset citation, and some information about the dataset itself, as well as documentation of the game state data and event types for the game whose data is stored in the file. We recommend a Creative Commons CCO 1.0 Universal license, (see https://creativecommons.org/publicdomain/zero/1.0/). Dataset citations can be generated through the citation file format tool [21] The dataset metadata should include the version of the event data standard to which the data conforms, the name and URL of the game whose data is contained in the file, the name and URL of the games developer. Finally, the exact format of the game state and event type information is left to the user, but must include a name, data type, and description of each element of the game state object, and a name, description, and listing of elements in the event data column for each event type. These elements, in turn, must be given a name, data type, and description.

The META.json is machine-readable documentation that a compatible tool can read to determine its compatibility with the dataset. Table II describes the elements that must be included in the META file.

TABLE II
ELEMENTS OF A META.JSON FILE

Key	Description
dataset_id	An identifier for the dataset.
dataset_type	"Event", "Player", or "Population", indicating whether the dataset contains event, player feature, or population feature data.
game_id	The name of the game whose data is included in the dataset.
instance_count	The number of unique instance IDs in the dataset.
player_count	The number of unique player IDs in the dataset.
session_count	The number of unique session IDs in the dataset.
start_date	The date of the earliest event or session in the dataset.
end_date	The date of the latest event or session in the dataset.
event_collections	A list of event collection standards implemented by the game whose data is included in the dataset, if the dataset contains event data.

C. Feature Data Standard

Like our Event Data Standard, the Feature standard contains elements in several categories. In this case, the categories are identifiers, feature data, and versioning.

1) Identifiers: These are largely the same as in the case of event data. Game, instance, player, and session identifiers carry the same meaning as the corresponding identifiers in event data. The feature standard introduces a "unit" ID as well, which indicates a unit of gameplay for which the feature was calculated. For example, a unit ID of "lvl01" would indicate the feature was calculated for level 01 of the given game.

In addition, we allow any of the identifiers to hold a special value of "$\star$", which indicates the feature was calculated by aggregating across all identifiers of that type. Thus, a feature with session ID and unit ID set to "$\star$" would indicate the feature value was calculated for the identified player, across all gameplay sessions and units of play.

TABLE III
STANDARD PROPERTIES FOR FEATURE DATA

Property	Data Type	Allow NULL	Description	Example
game_id	String, 32	NO	A string identifying which game from which the event came.	MY_GAME
population_id	String, 32	YES	For higher-order features calculated at a population level of granularity, and ID for the group of players or sessions making up the population. By default, this is just "dataset," i.e. the population made up of all data in the given set.	version_8
instance_id	String, 32	YES	For multiplayer games, an ID for the instance of the game that generated the event, or * if aggregated across instances.	48272075
player_id	String, 32	YES	A custom, per-player ID that may be re-used across game sessions, or * if aggregated across players.	BlueWhale
session_id	String, 32	NO	Unique identifier for the gameplay session, or * if aggregated across sessions.	24020195392739482
unit_id	String, 32	YES	An identifier of an in-game unit of interaction, such as a level or a questionnaire; * if aggregated across gameplay units.	lvl1
feature_name	String, 32	NO	The name of the feature.	AverageScore
feature_value	Int | Float | String | JSON	NO	The value of the feature, across all data in the dataset for the specified instance, user, session, and unit IDs. Feature values may be a single number, a string, or a complex JSON object.	100
feature_source	String, 32	NO	The name of the feature extraction module that generated the feature	ScoringModule
source_hash	String, 32	NO	A precise identifier of the version of the feature extractor module that generated the feature value. Generally, this should be a revision hash from a version control system. The source hash must be unique for the given feature extractor.	a63eb9
feature_dependencies	List [Tuple [str, str]]	YES	A compound JSON-style object, which contains a list of pairs. Each pair is the name of an event detector or feature extractor whose outputs are accepted as input by the given feature, and the source_hash of that module.	

2) Feature Data: This category includes the name and value of the feature, as well as information on the "source" of the feature. The feature_source is equivalent to event_source, with the only exception being that "GAME" is no longer a valid source. We recommend but do not require the feature_name match the feature_source; this allows the flexibility for a single source module to calculate multiple features.

3) Versioning: Because features may use outputs from event detectors as well as (in the case of 2nd- and higher-order features) feature extractors in order to calculate the feature value, versioning is a somewhat property to track. For feature data, we include a source_hash, which is equivalent to the source_hash for detector-generated event data. We also require a feature_dependencies property, which is a JSON-formatted list of pairs. Each pair indicates a detector or feature module that generates input for the given feature, and the source_hash of the module. In this way, the full chain of data-generating modules leading to a given feature value can be traced.

D. Feature File Format

As with events files, we define a standard reshaping of feature data into a form suitable for file-based dataset distribution. Again, files should be distributed in a tab-delimited format, reserving comma-delimitation feature properties that may use JSON formatting for their values. In this case, however, we do not recommend a direct translation of elements from the data standard into TSV columns. Instead, feature data should be provided in one of two kinds of file. Specifically, these are population and player files. A player file includes features *within* player, session, and game units, which collectively describe individual gameplay experiences. Population files include features *across* players and sessions. These files essentially describe the game itself, based on the collective outcomes of many players. We discuss each in detail below:

1) Player Feature Files: These files include data only for individual players and sessions. That is, features which do not have "$\star$" as the value of both player_id and session_id. The first three columns of a feature file should be player_id, session_id, and unit_id, leaving aside the population and instance IDs. Then each row of the file will contain data for a unique combination of those IDs. The remaining columns of the file should be the unique feature_names across the entire set of feature data, with the corresponding feature values in those columns. This format is illustrated in table IV.

In this example, we see various levels of analysis for the *BlueWhale* user and one record for *GreenGiant* to illustrate that this file is for multiple players. In the topmost record, we output a * for both Session ID and Unit ID, indicating that these analysis are utilizing all sessions and all units of game play to derive the player features. In the next two records, only session contains a * and the Unit ID has a value. This indicates that the analysis will use all the sessions for the player to make calculations, but the only features that are reasonable to report are the individual unit features. In the forth record from the top, we see that the *BlueWhale* player also has a session specified, but still has a * for the Unit ID. This indicates that the analysis will be considering all the possible game units, but only for one session for the player. Therefore, only the session features are populated. Finally, we see a record where Player ID, Session ID and Unit ID all have values. This describes the most narrow use of the common aggregation levels, describing the features of a specific unit of game play within a specific session of a specific player.

TABLE IV
STRUCTURE OF A PLAYER FILE

Player ID	Session ID	Unit ID	Player Features	Session Features	Unit Features
BlueWhale	*	*	values	null	null
BlueWhale	*	lvl1	null	null	values
BlueWhale	*	lvl2	null	null	values
BlueWhale	123456	*	null	values	null
BlueWhale	123456	lvl1	null	null	values
Green Giant	213456	lvl1	null	null	values

TABLE V
STRUCTURE OF A POPULATION FILE

Population ID	Instance ID	Unit ID	Population Features	Instance Features...	Unit Features
dataset	null	*	values	null	null
dataset	null	lvl1	null	null	values
dataset	null	lvl2	null	null	values

2) Population Feature Files: Population files contain features that represent aggregations of play experiences, rather than aggregations over time series data. That is, they always contain data only from higher-order features. Thus, instead of session and player ID columns, a population file uses the population_id and instance_id of its features, in addition to the unit_id. Then each row represents data for a unique combination of population group, game instance (in the case of multiplayer games), and gameplay unit. Like player files, the remaining columns are feature names, whose values correspond the feature_value matching the combination of feature name, population ID, instance ID, and unit ID.

As with player files, this format is illustrated in table V. That example assumes a non-multiplayer game, leaving the instance ID null. In all other respects, the function is the same as a player feature file; only the meaning of the data differs.

3) Readme and Meta Files: As with event datasets, any player or population feature dataset must include a README.md and META.json file, again providing human-readable and machine-readable documentation of the data. The readme file is principally the same, including license information, a recommended citation, and names and URLs of the game and game developer. Instead of a version of the game logging set and descriptions of individual events, however, the readme should contain a list of all dataset features, including the name, type, source, source_hash, dependencies, and description of the feature. Thus, the properties of a feature that are not included directly in the dataset (which unlikely to be useful for analysis) are instead provided to analysts via the readme. The meta file, on the other hand, is nearly identical to the meta for an event dataset. However, it should not include an event_collections key, and instead should include "units". This element should be a list of types of gameplay units, each paired with a count. For example, a game with 20 levels and 4 questionnaires would list its units as: `[('lvl', 20), ('qst', 4)]` or similar.

E. Ontologies

Finally, we introduce standard *ontologies*, collections of common sets of events and/or features seen across many games that enable the use of analysis methods or external tools. Detectors, feature extractors and external tools may specify specific game events and features that, if implemented, will enable use of their tool. Every ontology is optional for any given game or data pipeline; it is up to implementers to identify any tools or standard features that are desired for use with the games data, and to implement the corresponding event collections. While formal definitions are dynamic and outside of the scope of this document, examples include:

1) sessions-0.1: - Provides player and population features to describe active play duration, platforms, and new vs. continued play.

2) achievements-0.1: - Provides player and population features to describe achievement sequences and popularity. Enables the use of the progression web tool.

3) challenges-0.1: - Provides player and population features to describe how players accept, fail and complete different challenges. Used to understand both linear and non-linear sequences and popularity. Enables the use of the progression web tool.

4) replay-vr-deterministic-0.1: - Enables the use of the VR replay tool which constructs videos of gameplay sessions from log data.

VII. CURRENT OPEN GAME DATA IMPLEMENTATION

We now discuss an implementation of the proposed pipeline and several of these standards in a series of packages collectively referred to as Open Game Data. Source for all of these components is available at https://github.com/opengamedata. First, we provide packages for logging event data from Unity- and JavaScript-based game applications in accordance with the format described in the event data standard. The event output from these packages is directed to an instance of [opengamedata-logger], which collects events into a short-term database. We have implemented logging in 21 games to date that use this system, collectively contributing 2-10 million new events per day to a growing public repository of gameplay data. We use nightly automations via the GitHub Actions service to move event data to long-term storage in a BigQuery database. The feature processing portion of the pipeline is implemented in [opengamedata-processor]. We have created event detectors and feature extractors for 12 of the 21 games, and feature processing uses a monthly automation. At the beginning of each month, we process all the previous months data for each game, releasing all event and feature data to the website [opengamedata-website] instantiated at https://opengamedata.fielddaylab.wisc.edu.

However, some future work remains. Our [opengamedata-processor] implementation is not yet compliant with all proposed standards. Specifically, feature data currently uses an older format that does not align with the feature data format outlined above. Additionally, we do not yet support the system feature modules to be paired with event collection standards, and our implementation of event detectors is not fully compliant with the rules for sequence indexing.

VIII. Conclusion

While promising, the use of large scale game data for educational research is a complex endeavour, requiring significant existing infrastructure. Through a review of previous efforts and defining a collection of specific use cases, we propose a modular data architecture and related data standards that provide a number of interfaces with a shared infrastructure, from integrating new games to developing new analysis and visualization tools. We then demonstrated the feasibility of the approach through an initial infrastructure that contains a portfolio of games developed by a number of studios and collects large amounts of data used by researchers. While much ongoing effort is required to continue developing these approaches and standards, this paper creates a foundation to invite participation from a variety of educational game designers and researchers.

References

[1] A. Klein, "During COVID-19, Schools Have Made a Mad Dash to 1-to-1 Computing. What Happens Next?" *Education Week*, Apr. 2021. [Online]. Available: https://www.edweek.org/technology/during-covid-19-schools-have-made-a-mad-dash-to-1-to-1-computing-what-happens-next/2021/04

[2] B. Engelsttter and M. R. Ward, "Video games become more mainstream," *Entertainment Computing*, vol. 42, p. 100494, May 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1875952122000180

[3] R. J. Mislevy, S. Corrigan, A. Oranje, K. DiCerbo, M. I. Bauer, A. von Davier, and M. John, "Psychometrics and game-based assessment," in *Technology and testing: Improving educational and psychological measurement*. Routledge New York, 2016, pp. 23–48.

[4] R. S. Baker and P. S. Inventado, "Educational Data Mining and Learning Analytics," in *Learning Analytics*, J. A. Larusson and B. White, Eds. New York, NY: Springer New York, 2014, pp. 61–75. [Online]. Available: http://link.springer.com/10.1007/978-1-4614-3305-7_4

[5] E. Rowe, M. V. Almeda, J. Asbell-Clarke, R. Scruggs, R. Baker, E. Bardar, and S. Gasca, "Assessing implicit computational thinking in Zoombinis puzzle gameplay," *Computers in Human Behavior*, vol. 120, p. 106707, Jul. 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0747563221000297

[6] S. Karumbaiah, R. S. Baker, and V. Shute, "Predicting Quitting in Students Playing a Learning Game," *researchgate.net*, 2018. [Online]. Available: https://www.researchgate.net/profile/Ryan_Baker14/publication/325263595_Predicting_Quitting_in_Students_Playing_a_Learning_Game/links/5b0273c20f7e9be94bd9c947/Predicting-Quitting-in-Students-Playing-a-Learning-Game.pdf

[7] V. E. Owen, M.-H. Roy, K. P. Thai, V. Burnett, D. Jacobs, and R. S. Baker, "Detecting Wheel-spinning and Productive Persistence in Educational Games," in *Proceedings of the 12th International Conference on Educational Data Mining*. International Educational Data Mining Society, 2019, pp. 378–383.

[8] J. D. Gobert, M. Sao Pedro, J. Raziuddin, and R. S. Baker, "From Log Files to Assessment Metrics: Measuring Students' Science Inquiry Skills Using Educational Data Mining," *Journal of the Learning Sciences*, vol. 22, no. 4, pp. 521–563, Oct. 2013. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/10508406.2013.837391

[9] S. Charleer, F. Gutirrez, K. Gerling, and K. Verbert, "Towards an Open Standard for Gameplay Metrics," in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. Melbourne VIC Australia: ACM, Oct. 2018, pp. 399–406. [Online]. Available: https://dl.acm.org/doi/10.1145/3270316.3271529

[10] . Serrano-Laguna, I. Martnez-Ortiz, J. Haag, D. Regan, A. Johnson, and B. Fernndez-Manjn, "Applying standards to systematize learning analytics in serious games," *Computer Standards & Interfaces*, vol. 50, pp. 116–123, Feb. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0920548916301040

[11] C. Alonso-Fernandez, A. Calvo-Morata, M. Freire, I. Martinez-Ortiz, and B. F. Manjon, "Data science meets standardized game learning analytics," in *2021 IEEE Global Engineering Education Conference (EDUCON)*. Vienna, Austria: IEEE, Apr. 2021, pp. 1546–1552. [Online]. Available: https://ieeexplore.ieee.org/document/9454134/

[12] S. Slater, R. S. Baker, and D. J. Gagnon, "Changing Students Perceptions of a History Exploration Game Using Different Scripts," in *Proceedings of the 30th International Conference on Computers in Education*, Kuala Lumpur, Malaysia, 2022.

[13] J. C. Stamper, D. Lomas, and D. Ching, "The Rise of the Super Experiment," in *Paper presented at the International Conference on Educational Data Mining (EDM)*, Chania, Greece, 2012, p. 4.

[14] Z. Teng, J. Pfau, and M. S. El-Nasr, "Visualization-based Iterative Segmentation to Augment Video Game Analytics," in *2023 IEEE Conference on Games (CoG)*. Boston, MA, USA: IEEE, Aug. 2023, pp. 1–2. [Online]. Available: https://ieeexplore.ieee.org/document/10333151/

[15] L. Swanson, D. Gagnon, E. Harpstead, J. Mccloskey, J. Scianna, S. Slater, and N. Spavacek, "Leveraging Cluster Analysis to Understand Educational Game Player Styles and Support Design," in *GLS 13.0 Conference Proceedings*. Irving, CA: ETC Press, 2022.

[16] X. Liu, S. Slater, J. M. A. L. Andres, L. Swanson, J. Scianna, D. Gagnon, and R. S. Baker, "Struggling to Detect Struggle in Students Playing a Science Exploration Game," in *Companion Proceedings of the Annual Symposium on Computer-Human Interaction in Play*. Stratford ON Canada: ACM, Oct. 2023, pp. 83–88. [Online]. Available: https://dl.acm.org/doi/10.1145/3573382.3616080

[17] Stephen Hutt, Ryan S. Baker, Jaclyn Ocumpaugh, Anabil Munshi, Shamya Karumbaiah, Stefan Slater, Gautam Biswas, Luc Paquette, Nigel Bosch, and Martin van Velsen, "Quick Red Fox: An App Supporting a New Paradigm in Qualitative Research on AIED for STEM," in *Artificial intelligence in STEM education: the paradigmatic shifts in research, education, and technology*, 1st ed., ser. Chapman & Hall/CRC artificial intelligence and robotics series, F. Ouyan, Ed. Boca Raton, FL: CRC Press, 2023.

[18] J. A. Ruiperez-Valiente, M. J. Gomez, P. A. Martinez, and Y. J. Kim, "Ideating and Developing a Visualization Dashboard to Support Teachers Using Educational Games in the Classroom," *IEEE Access*, vol. 9, pp. 83 467–83 481, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9446995/

[19] K. Holstein, B. M. McLaren, and V. Aleven, "Co-Designing a Real-Time Classroom Orchestration Tool to Support TeacherAI Complementarity," *Journal of Learning Analytics*, vol. 6, no. 2, Jul. 2019. [Online]. Available: https://learning-analytics.info/index.php/JLA/article/view/6336

[20] T. Preston-Werner, *Semantic Versioning*, web, Std., 2013. [Online]. Available: http://semver.org/

[21] S. Druskat, J. H. Spaaks, N. Chue Hong, R. Haines, J. Baker, S. Bliven, E. Willighagen, D. Prez-Surez, and O. Konovalov, *Citation File Format*, Std., Aug. 2021.