Entertainment Computing 52 (2025) 100778

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom

Using provenance and replay for qualitative analysis of gameplay sessions

Leonardo Thurler *, Sidney Melo, Leonardo Murta, Troy Kohwalter, Esteban Clua

Institute of Computing, Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brazil

ARTICLE INFO

Keywords:
Game telemetry
Game provenance
Game data analysis
Provenance graph data visualization
Spatio-temporal data visualization
Qualitative analysis

ABSTRACT

There is an increasing interest to use game telemetry for analyzing gameplay sessions, with numerous techniques created to help game developers analyze different game aspects like game balancing and behavioral analysis. Among different gameplay session analysis techniques, the collection of provenance data has stood out due to a crucial advantage of this approach: the possibility to identify cause–effect relationships between game events. In previous work, we presented our conceptual framework called Prov-Replay, which provides a replay synchronized with an interactive provenance graph visualization. We validate Prov-Replay by creating PinGU Replay, a tool that implements our conceptual framework and applied it in a commercial game. Due to the promising results, this paper extends our previous work by presenting a detailed overview about Prov-Replay implementation, introducing a new feature that provides an analysis dashboard, and applying our experiment methodology to a new commercial game. We also enhance the concept of analytics related to provenance through the replay pipeline. Finally, we made PinGU Replay available as open-source software. Our new experiment results reinforce that, when using our conceptual framework fundamentals, it is possible to improve the efficiency and effectiveness of qualitative analysis process.

1. Introduction

Lately, there has been a notable surge in enthusiasm surrounding game data utilization, often called game telemetry. This heightened interest spans across both academic and industry circles [1–3]. This is attributed to the diversity of use that this area has, which includes but is not limited to game balancing [4–6], detection of failures during game design [7,8], and player profiling [9–11]. Kleinman et al. [12] observed that data analytics remains one of the most efficient ways to collect large-scale generalized insights regarding what players do in-game.

Kleinman et al. [13] conclude that there is a notable movement to use data visualization with the objective of making the analysis of granular telemetry more transparent and accessible to be analyzed by humans. This movement led to the creation of several game data visualization techniques such as Charts and Diagrams [14,15], Heatmaps [16,17], Movement visualizations [18,19], Self-organizing maps [20,21], and Node-link representations [22,23]. The authors still observed a significant growth of **spatio-temporal visualization** technique compared with other game data visualization approaches. They associate this growth with the ease with which this technique has to present low-level, granular, and context-sensitive information [12,24,25]. The authors highlight that such visualizations play a crucial role in assisting human analysts in deriving meaningful insights from context-sensitive information, providing positive reactions from users [26–30]. They also mentioned two works [31,32] provides a interactive spatio-temporal visualization technique, the main difference is that this interactive approach allow the user to manipulate the elements that are being presented in an interactive way, further helping in the analysis process when compared with non-interactive approach.

Using provenance data in games was first proposed by Kohwalter et al. [33] with the Provenance in Games (PinG) conceptual framework. Kohwalter et al. [33] introduced a conceptual framework called Provenance in Games (PinG) to leverage game telemetry through usage of **provenance data in game** context. Their main objective is to collect provenance data that, when analyzed, can provide feedback to the developers, making it possible to understand the cause–effect relations of players' decisions. They implemented a provenance gathering framework for the Unity[1] game engine called Provenance in Games for Unity (PinGU) [34], providing an easy way to collect provenance data. In another work, they extended the use of PinGU to collect provenance data from different games while showing examples of how to do some common analysis using provenance data [35]. Subsequently, numerous works have expanded upon and applied these frameworks for diverse purposes [11,36,37].

* Corresponding author.
E-mail addresses: lp.thurler@gmail.com (L. Thurler), sidneymelo@id.uff.br (S. Melo), leomurta@ic.uff.br (L. Murta), troy@ic.uff.br (T. Kohwalter), esteban@ic.uff.br (E. Clua).

[1] https://unity.com

https://doi.org/10.1016/j.entcom.2024.100778
Received 17 April 2024; Accepted 15 June 2024
Available online 24 June 2024

Nevertheless, one of the major challenges found in studies involving the analysis of game provenance graphs is comprehending the contextual information associated with the graph. This understanding is crucial for facilitating the interpretation of the cause–effect relationships presented within the graphs. Motivated by this issue, in [38] we proposed a new approach to analyzing provenance graphs on games through Prov-Replay, a conceptual framework that combines a provenance tracker module and provenance graph viewer controller integrated with a replay module for providing a interactive spatio-temporal visualization from game session data, which can facilitate user analysis, visualizing both the provenance graph and game state in game space, allowing understanding of the entire context of a desired moment. To evaluate Prov-Replay, we implemented the Provenance in Games for Unity with Replay (PinGU Replay), a tool for the Unity game engine that incorporates the elements of our conceptual framework. Furthermore, we applied PinGU Replay in Smoke Squadron, a split screen local multiplayer arcade flight battle three-dimensional (3D) under-development commercial game, and we collected real game session data and performed an experiment with its developers using our proposed solution to analyze game sessions.

Due to the promising results, this paper extends our previous work by providing more detail about Prov-Replay implementation through PinGU Replay, improving PinGU Replay with the inclusion of a new feature that provides an efficient way for users to consult all content contained in the session's provenance data and that is relevant to update their visualization and filters rule through an analysis dashboard. This new visualization concept groups all functionality and information in just one place. We validated our experiment methodology into Survivor Heroes, an online multiplayer roguelike survival adventure two-dimensional (2D) under-development commercial game. Furthermore, we provide additional comparisons with related work and a more detailed discussion about our experiment methodology, case study results, and conclusions, indicating limitations and supplementary future works. We made PinGU Replay available as open-source software with a sample project.

Our new experiment results reinforce that, when using our conceptual framework fundamentals, it is possible to improve the efficiency and effectiveness of qualitative analysis processes by providing effective ways to capture and access important information, enabling a more informed qualitative analysis. Finally, with the additional information of this work together with the availability of PinGU Replay, we provide an efficient system to other researchers and game developers to use and expand our work.

This paper is organized as follows: Section 2 presents some related work in game data analysis with visualization techniques. Section 3 provides background information about the usage of provenance in games and issues to consider when making a replay system. Section 4 presents details about Prov-Replay implementation through PinGU Replay and provides information about PinGU Replay availability as an open-source software with a sample project. Section 5 presents our case studies' methodology, implementations, and results. Finally, Section 6 presents our conclusions with considerations about limitations and future work.

2. Related work

Liu et al. [23] proposed a method for visually analyzing play traces called Playtracer. It illustrates how groups of players move through the game space, aiding the designer by showing common pathways and alternatives that players use to succeed or fail in their tasks and identifying pitfalls and anomalies in the scene. Nonetheless, Playtracer does not consider temporal information or preserve the order of the states visited by players. Besides that, incorporating Playtracer in the game design is challenging because it requires designers to define a state distance metric and identify relevant states.

Wallner and Kriglstein [25,39] described and formalized the Play-Graph visualization system. This system captures and illustrates the sequence of states and the actions that caused the player's state changes throughout the game. The game state describes a certain configuration of the game or an entity, while actions consist of player interactions within the game, such as shooting, jumping, or using an object. Due to the nature of how the data is structured in Play-Graph, the understanding of player behavior is guided by the player's progression in the game (*e.g.*, killed a boss) and not by how he/she interacted with the world (*e.g.*, player changes direction to try to collect an item). From the available documentation, only the state changes caused by players' actions can be identified in the graph. Thus, the graph does not present influences and causal relationships in the player's action.

Kuan et al. [31] presented a study with an analysis process using an interactive, spatio-temporal visualization system for analyzing battle information from StarCraft 2.[2] Their analysis involved utilizing their custom-developed tool in conjunction with an official replay system provided by the game. This work is limited only for real-time strategy (RTS) game genre and presents information that depends on an external tool and does not display the data inside the game space. Moreover, this study has a significant distinction when compared with our work. They focus on giving a tool for players to analyze matches to improve their performance, and our work focuses on giving a tool for game developers to understand the elements that influence players' decisions and improve the game based on this information.

Afonso et al. [32] presented a study that used animated maps for spatio-temporal data analysis applied to Multiplayer Online Battle Arena (MOBA) game genre using League of Legends (LoL)[3] as a case study. In this study, they designed a prototype tool named VisuaLeague. This tool uses animated maps to allow users to visualize player trajectories and events during a LoL match. This study focused on understanding if animated maps are an effective visualization technique to help inexperienced users (*i.e.*, players inexperienced with data visualization) extract relevant information to solve tasks related to players' performance in the game. Similar to [31], this study is limited only to the MOBA game genre. Besides that, the presented information depends on an external tool presented by the authors, and the study focus in give a tool for players analyze matches to improve them performance.

Ahmad et al. [24] proposed Interactive Behavior Analytics (IBA) a methodological approach for study player behavior using two visualization tools: a spatio-temporal visualization tool for game events called StratMapper and a graph-based visualization tool to investigate player behavior in a fine-grained level named Glyph. Both visualization tools are combined to provide an iterative game session visualization with rich contextual information. However, to analyze the information considering spatio-temporal information, users need to follow a complex workflow that consists of labeling the game using StratMapper and exporting data to be visualized in Glyph. Kleinman et al. [12] expanded this work, adding a sequence analysis at the beginning of the process. This addition facilitated the development of preliminary high-level behavioral labels, but the rest of the process remained. Although promising, this is not a simple approach to use and involves getting used to several tools. In addition, the tools are limited to a 2D visualization of the game space, making some information difficult to visualize and understand adequately, such as the position of agents, entities, and activities of a 3D game.

Kohwalter et al. [40,41] presented the ProvViewer,[4] a visualization tool for provenance graphs generated using the PROV-N notation [42]. This tool displays data in a 2D visualization and provides several options such as filters, merging, highlighting, and collapsing information

[2] https://starcraft2.com
[3] https://www.leagueoflegends.com
[4] http://gems-uff.github.io/prov-viewer

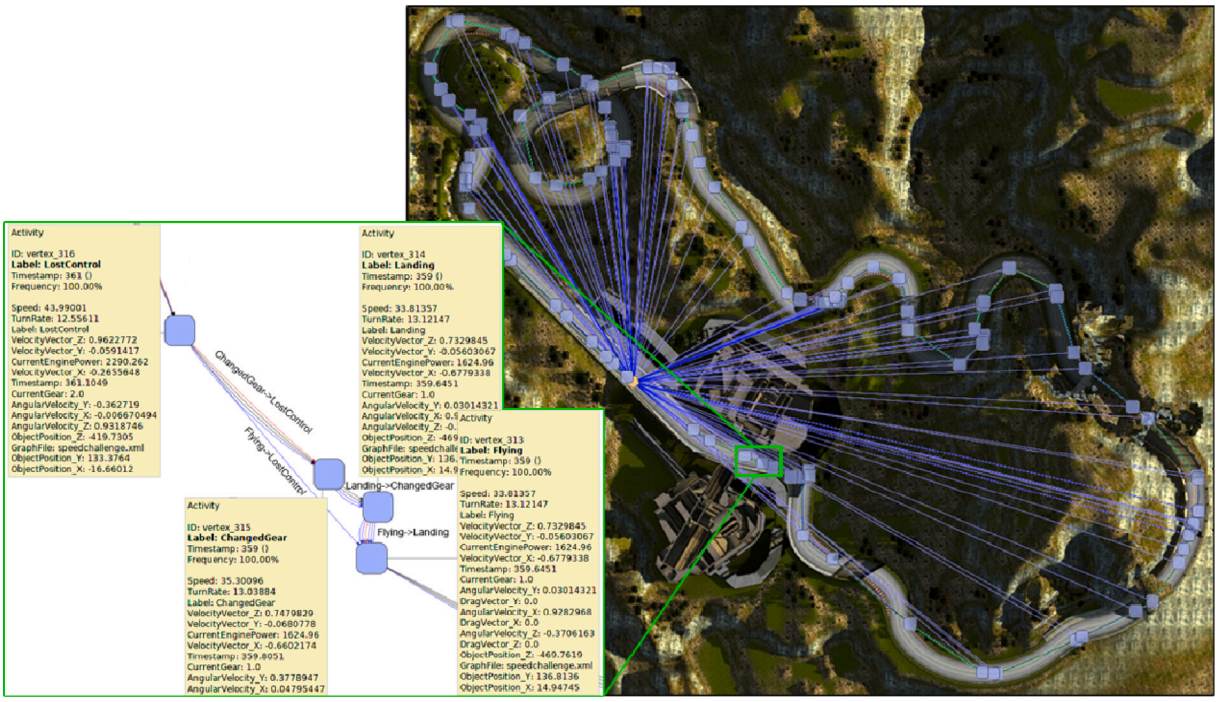

Fig. 1. Racing game provenance graph example with a zoom of a particular moment of provenance graph showing some of the collected activities and their attributes. Source: [11].

that facilitates the visualization and analysis. Although ProvViewer provides visualization and facilitates data analysis, it has some limitations. Similar to StratMapper, this tool presents all data collected from a 3D game space in a 2D visualization. Another limitation is the impossibility of verifying the state of game elements that were not captured in provenance data at a specific moment in the graph, making it difficult to analyze and understand the entire context of the game.

Thus, compared with the cited papers, our work makes a contribution by presenting a framework that is integrated with the game engine, presenting information inside the game space without the need of any external tools, and that can be applied to different game styles [35,37]. In addition, provenance data allows to capture and display data about the relationship between graph vertex, which, combined with the replay display, allows users to analyze the game context in detail in order to understand the cause–effect relations of players' decisions.

3. Background

In art and digital libraries context, provenance refers to the documented history of an art object or the documentation of processes in a digital object's life cycle [43]. Moreau et al. [44] assumed that the provenance of objects is represented by an annotated causality graph, which is a directed acyclic graph enriched with annotation capturing further information pertaining to execution.

Using provenance data in games was first proposed by Kohwalter et al. [33] with the Provenance in Games (PinG) conceptual framework. This framework is based on the PROV model [43] and is used to map provenance concepts to the context of games. PinG relates each type of provenance graph data to elements found in games. In the provenance context, there are three vertex types: Entities, Agents, and Activities. PinG maps these vertexes as follows: Entities represent objects without autonomous behavior, like items, weapons, and static obstacles in the environment; Agents represent objects capable of making decisions or have some responsibilities in the game, like players, enemies, and non-playable characters (NPCs); Activities are actions or events that occur throughout the game like attacking, jumping, collect items and others; The activities could be related with other activity, entity or agent, like "Activity A was triggered by Activity B", these relationships are mapped as edges between vertex elements. Following provenance definitions, PinG provides a solution to associate attributes with each of these types of vertices, allowing you to collect extra information when the vertex was generated, such as the player's name, amount of life of the player or enemy, which items the player had, and many others.

Later, Kohwalter et al. [34] provides a PinG implementation for the Unity game engine, named Provenance in Games for Unity (PinGU). PinGU is a generic implementation that can be easily integrated with any game in the Unity game engine to collect all provenance data (*i.e.*, agents, entities, activities, relationships, and attributes). The collected data throughout a game session is saved into a .xml file. This file contains the information that documents the whole session and can be used to generate and analyze the provenance graph or generate the provenance graph visualization in other tools like ProvViewer [41].

PinGU implements several methods to facilitate the capture of domain-specific provenance information and attach them to each entity in the game, and the amount of data gathered depends on analytic choices from game developers [35]. So, game developers control the granularity of the provenance data. They define what and how provenance data needs to be captured, considering their needs and the game's nature. For example, the game designer can define to capture an action named "PlayerMove" every second and configure this action to include information about the player's position and items. The captured provenance data includes contextual information and causal relationships, enriching the telemetry data that a simple log could not capture in detail. Fig. 1 presents the diversity of information that can be captured in provenance graph through a racing game's provenance graph, in this figure is possible to observe four distinct activities (*i.e.*, LostControl, Landing, ChangedGear, Flying), their components (*i.e.*, Speed, TurnRate, CurrentEnginePower), and influences between activities (*i.e.*, ChangedGear influences LostControl).

There are still numerous works that use these frameworks for diverse purposes [11,36,37]. Melo et al. [36] developed a framework called PINGUMIL[5] with the objective of detecting long-range cause-and-effect relationships by leveraging methods of representation learning

[5] https://github.com/sidneyaraujomelo/PingUMiL

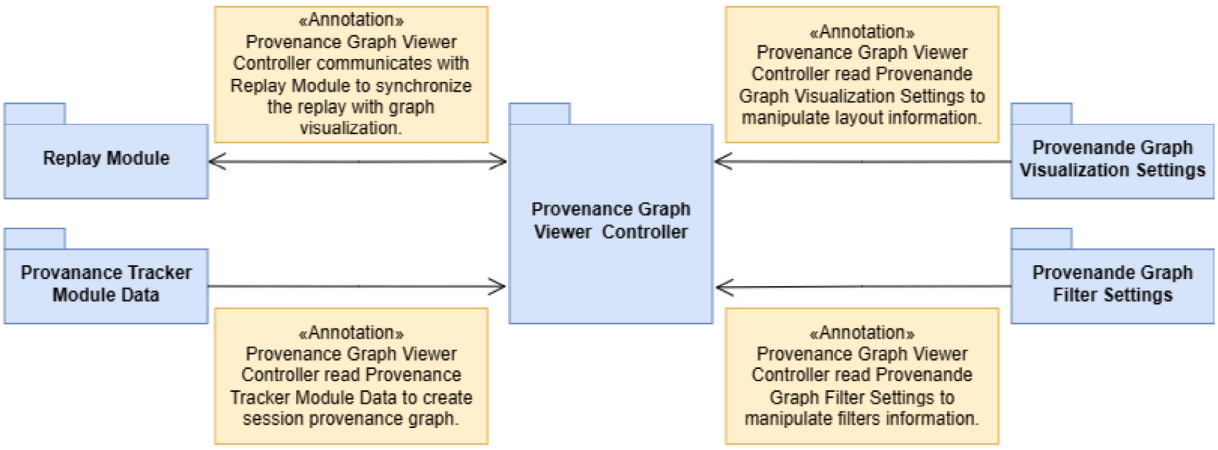

Fig. 2. Prov-Replay high-level architecture.

in game provenance graphs. In [11], the authors use the provenance data to analyze player behavior profiles with machine learning. Kohwalter et al. [37] proposed a probabilistic approach for game analytics named Provenance for generating stochastic models (Provchastic). This approach merges the tracked provenance data from multiple game sessions into a stochastic graph that is used to understand and predict game events.

Wagner [45] defines a replay system as an in-game system that lets the player record a gameplay sequence and replay it, just like recording a video with a phone and playing it again. Despite there being several ways to implement a replay system, the two most popular approaches to making a replay system are state-based and input-based. Wagner [45], Montville [46], and Engel [47] show some issues to consider when making a replay system with these approaches, as discussed in the following.

The state-based strategy captures a sequence of states from desired game elements inside a frame interval. A state represents a snapshot of the properties of an entity in a specific moment; for example, in frame 1, the state of the player character can have the position value (X:0, Y:1, Z:0), and in frame 20 (X:5, Y:6, Z:7). To replay, the replay system read the frames in order and apply the saved values to respectively game elements.

The input-based strategy consists of capturing an initial state from game elements and a sequence of inputs (*i.e.*, keypresses, touch gestures, joystick buttons) that can update their state values. To replay, the replay system sets the entire state of game elements at the beginning and then processes the frames by applying the recorded inputs for that frame.

Both strategies have advantages and disadvantages. The state-based strategy is more versatile than the input-based because it can reproduce faithfully in non-deterministic and deterministic games, while the input-based only reproduces a faithful replay in deterministic games. On the other hand, the input-based strategy needs less memory space than the state-based since it requires saving only the initial state and input information to reproduce replay, while the state-based needs to save the state of several frames. So, the choice of which approach should be used depends on the characteristics of the game in which it will be applied.

4. PinGU Replay

In [38], we proposed a new conceptual framework named Prov-Replay to perform interactive analysis through gameplay sessions using spatio-temporal visualization techniques synchronized with replay. Our solution provides a visual representation of the provenance graph inside the game level space, allowing users to configure rules to interactively manipulate the graph by making it possible to omit or highlight desired elements during the analysis process. The displayed graph is integrated with the replay data, allowing for advancing and rewinding the replay time to view the exact moment of a graph event and display graph data based on the current state of the replay.

To validate Prov-Replay, we created a tool for the Unity game engine named PinGU Replay that incorporates the elements of our conceptual framework. Fig. 2 presents the communication between Replay Module, Provenance Tracker Module, Provenance Graph Viewer Controller, Provenance Graph Visualization Settings and Provenance Graph Filter Setting, that are key elements of Prov-Replay architecture. In this section, we detail how PinGU Replay implements each element of Prov-Replay architecture and provide information about the availability of PinGU Replay as an open-source software and sample project.

4.1. Replay module

The Replay module is responsible for recording and reproducing the replay of game sessions. This module must provide functions allowing the user to play, pause, stop, and forward the replay to a specific game session time. This module must reproduce a faithful gameplay session replay. This is necessary to allow users to see the states of any objects that might impact the gameplay experience and player decisions, thus understanding the whole context information.

Since Unity is not a deterministic game engine, we created a state-based replay module to reproduce a faithful replay. In Fig. 3, we present an overview of the architectural design of our replay module, with the core elements that our replay module uses to identify which game object and data must be saved. The ReplayController plays a pivotal role in handling the various tasks associated with recording and reproducing replays. It takes charge of processing replay readings, configuring the necessary settings, and creating saved objects to ensure an accurate and faithful reproduction of the recorded events. IReplayComponent is the interface that a component must implement to save information into replay data. ReplayComponentBase is an abstract component that implements IReplayComponent and needs to be inherited by the component that saves any information into the replay. The ReplayTransformComponent inherited ReplayComponentBase and defines how to save game objects' transform information (*i.e.*, position, rotation, and scale), it is possible to create other components that inherit from ReplayComponentBase to store and load other information relevant to game replay (*i.e.*, animations, domain-specific information, and others). ReplayObject is the class that identifies a game object as an object that must be saved and contains a list of ReplayComponentBase. ReplaySettings defines general information about how replay must be saved (*i.e.*, the frequency of replay information must be saved and the path to spawn the correct replay object when reproducing replay).

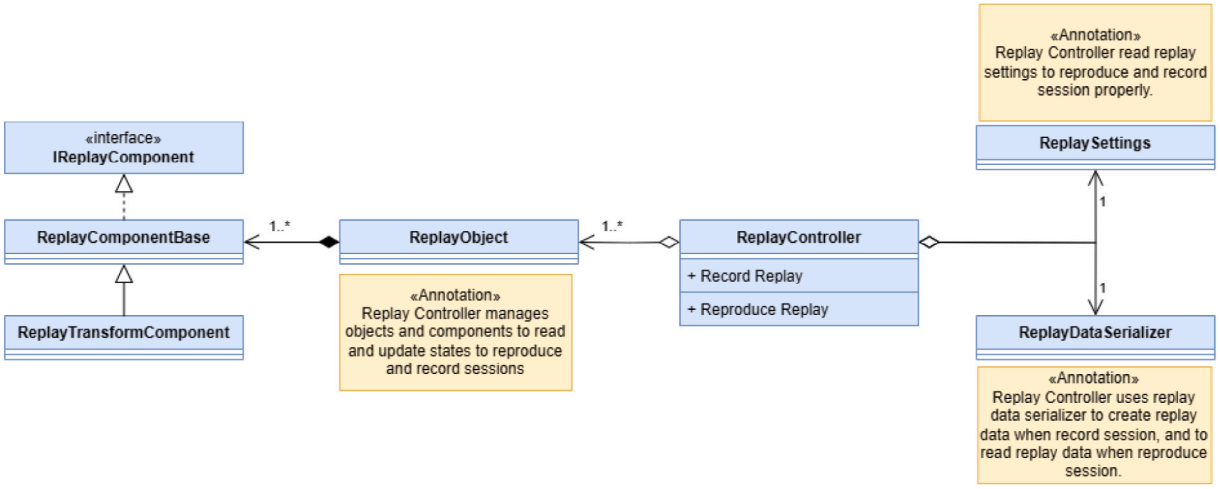

Fig. 3. Replay Module architecture overview.

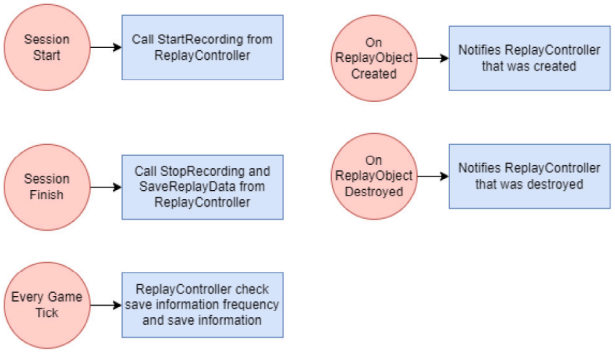

Fig. 4. Replay module important events.

Lastly, ReplayDataSerializer is responsible for defining how the replay data will be stored.

Fig. 4 shows important events that the replay module uses to save replay information. Red circles indicate events and blue rectangles describe, in high-level, what occurs after the event activation. When the session starts, we need to call StartRecording from ReplayController to the replay module to start saving information. While a session is executing, the ReplayController use ReplaySettings to check save information frequency to save information if needed. The ReplayObjects notify ReplayController when they are created and destroyed, so ReplayController can store and remove these objects from an internal list to save information. When the session finishes, it is necessary to call StopRecording and SaveReplayData from ReplayController to replay model stop saving information and store saved information in a file that will be read to reproduce the replay later.

To reproduce replay, ReplayController uses ReplayDataSerializer to load stored data and ReplaySettings to spawn, destroy, and set up ReplayObjects state information according to game session time.

4.1.1. Integrating replay module into an existing game

The configuration for the save session consists of four stages. The *first stage* is to configure the module to save replay data, which consists of identifying and configuring ReplayObject and ReplayTransform components on game objects that represent agents and entities. These components allow the system to identify each object that should be reproduced on replay and what information should be stored. The *second stage* consists of creating a ReplaySettings scriptable. This file contains configurations to generate replay files, locate replay objects prefabs, and replay save data frequency. The *third stage* consists of creating a game object in the scene to act as a controller for the replay system. This game object will have two attached components, ReplayController and ReplayDataSerializer, and will reference the ReplaySettings. The *last stage* consists of updating your code to use ReplayController methods to start and stop records and save replay data into a file.

To reproduce a saved session, it is necessary to duplicate the original game scene and remove all objects related to game logic and controls, leaving only the environment and the ReplayController object in the scene. Afterward, it is necessary to configure ReplayDataAsset in ReplayController with the saved file of the session to be reproduced. Finally, one must add the ReplayCanvas object to the game scene. This object implements a user interface that allows all replay functionalities (*i.e.*, play, pause, and stop replay).

4.2. Provenance tracker module

The Provenance Tracker module is responsible for gathering provenance information about game sessions and storing it into a file that can be used to generate a provenance graph of the game session. PinGU Replay uses PinGU as a provenance tracker module, so all provenance data of activities, agents, entities, and their relations are collected by PinGU and stored in a .xml file that the provenance graph viewer controller processes to generate provenance graph visualization. Kohwalter et al. [34] described the details of PinGU implementation and how to integrate this framework into a game to gather and store provenance information.

4.3. Provenance graph viewer controller

The Provenance Graph Viewer Controller is responsible for creating and updating provenance graph nodes and connecting with the replay module to synchronize the provenance graph with the replay. This controller must allow users to manipulate the graph interactively, enabling users to define what and how information must be displayed. It also manages other elements involved in this process to enable the manipulation and visualization of the provenance graph.

To develop the Provenance Graph Viewer Controller, we followed our Prov-Replay architecture presented in Fig. 2. So, the Provenance Graph Viewer Controller communicates with the Replay Module and uses the provenance data stored in the .xml file generated by PinGU with information of Provenance Graph Visualization Settings and Provenance Graph Layout Settings to display provenance graph information to the user.

Fig. 5 shows **Provenance Graph Visualization Settings**, these settings allow the user to configure and create presets with information for the graph layout, which provides functionalities to highlight

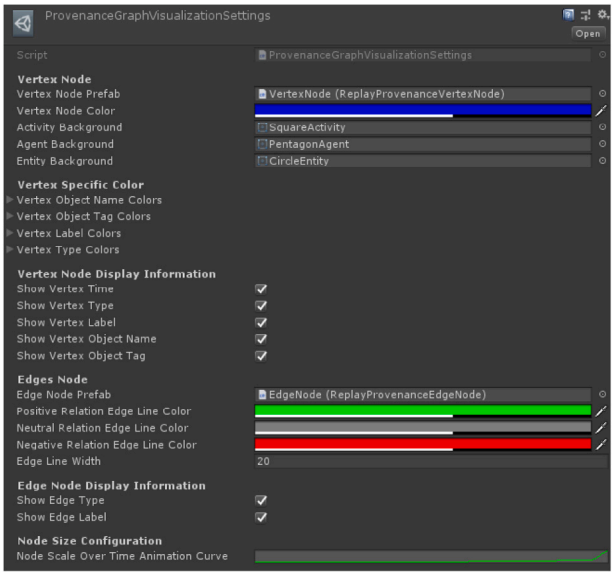

Fig. 5. Provenance Graph Visualization Settings.

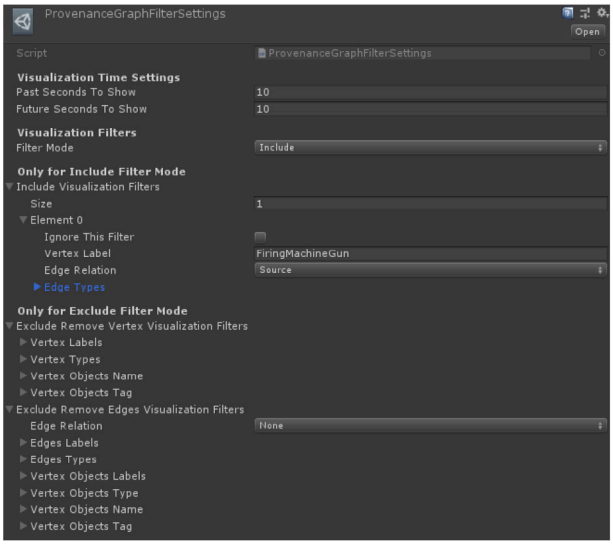

Fig. 6. Provenance Graph Filter Settings.

desired elements using rules to colorize and resize provenance graph elements. The visualization strategy is based on different shapes and colors for vertices. Vertices' shapes are used to map semantic concepts from the provenance: activities (rectangle), entities (circles), and agents (pentagons). Colors are used to highlight vertices and show edges' relationship intensity. The user can configure rules to change vertex colors based on vertex name, tag, label, and type, which allows them to highlight fine-grain information for specific analyses (*i.e.*, modify the color of a vertex with "Player 1" or "Player 2" name). The edge value, if any, defines the color (*i.e.*, green to positive values, red to negative, and gray when the value is zero or empty) and is more common on influences (*i.e.*, relations with wasInfluencedBy type). Also, edge lines are thicker when they are close to the source vertex and thinner when they are close to the target vertex. Furthermore, the user can configure what information will be shown on vertices and edges and configure the vertex size curve to increase or decrease closed nodes to replay at the current time.

Fig. 6 shows **Provenance Graph Filter Settings**, these settings allows the user to configure and create presets with information related to filters in the displayed graph, which provides functionalities that show or hide specific provenance graph elements using rules and time configurations. The time setting allows you to hide information that is very far from the current moment of the replay, so if it is configured to only see information that is 10 s before and after the current moment, any event beyond that time frame will not be displayed. The user must define the filter mode, we created two filtering modes: Exclude and Include. The *Exclude filter* mode uses a top-down visualization strategy, where the provenance graph excludes only information that matches user-configured rules and presents everything else. The user can configure much information to exclude, like vertex label, type, object name, object tag, edges relations (*i.e.*, Source and Target), labels, type, and all vertex information linked by a given edge relation. Furthermore, users can use the exclude filter mode without any rule to show all provenance graph information. The *Include filter* mode uses a bottom-up visualization strategy, where the provenance graph shows only information that matches user-configured rules. The user can configure vertex labels to show and define whether they display relations that involve desired vertices. Moreover, users can configure the exclude rules to be applied only on elements with configured vertex labels (*i.e.*, Configure to show vertex with the label "CollectItem" and exclude tags "Player02"). Both modes require the user to configure a visualization time.

Depending on the provenance information, creating or updating many nodes at once is necessary. This operation can be heavy to process and cause lag in replay processing. So, to improve performance, we create the ProvenanceNodesPoolManager component. This component implements an object pooling[6] to create and consume the objects that represent graph node information in replay processing. Fig. 7 presents provenance nodes object pooling elements. In addition, it shows the communication between ProvenanceGraphViewerController and ProvenanceNodesPoolManager. IProvenanceNodePoolComponent is an interface implemented by ProvenanceNodesPoolComponent to define a game object as a provenance node that ProvenanceNodesPoolManager will use. ReplayProvenanceVertexNode and ReplayProvenanceEdgeNode are two components that represent a graph node without relationships and with relationships, respectively. These components implement the IProvenanceNodePoolCleanUp interface to define how to reset information when the object will be reused.

4.3.1. *Configure to show graph with session replay*

To visualize the provenance graph, we must create a game object in the scene to act as a controller for the provenance graph viewer system. We need to attach the ProvenanceGraphViewerController component to this game object, and this component is responsible for setting the provenance graph input graph data (*i.e.*, the .xml file generated by PinGU), provenance graph visualization settings and provenance graph filter settings.

4.4. *Analysis dashboard*

In this work, we also include a new feature that provides an efficient way for users to consult all content within the session's provenance data, which is relevant to updating their visualization and filter rule through an analysis dashboard. The primary purpose of this dashboard is to enhance the user's analysis experience by offering improved usability across various aspects of the Prov-Replay framework. Since it is essential that this tool groups all the information necessary for the user to carry out the analysis process, different features are designed:

- Settings for showing and hiding layout and filter settings and updates of this information.

[6] https://learn.unity.com/tutorial/introduction-to-object-pooling

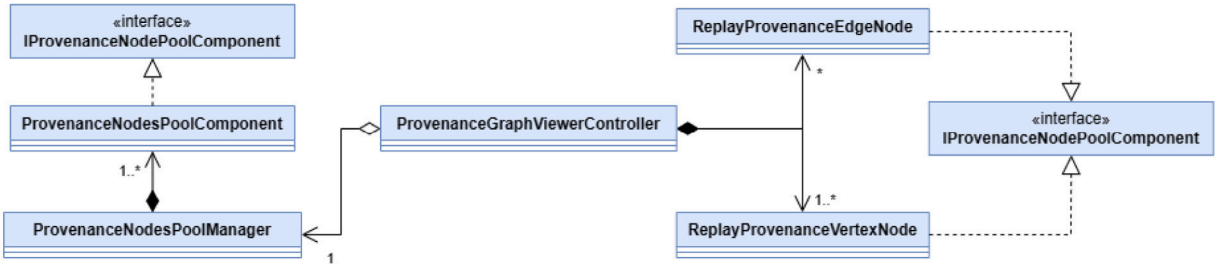

Fig. 7. Provenance nodes object pooling elements.

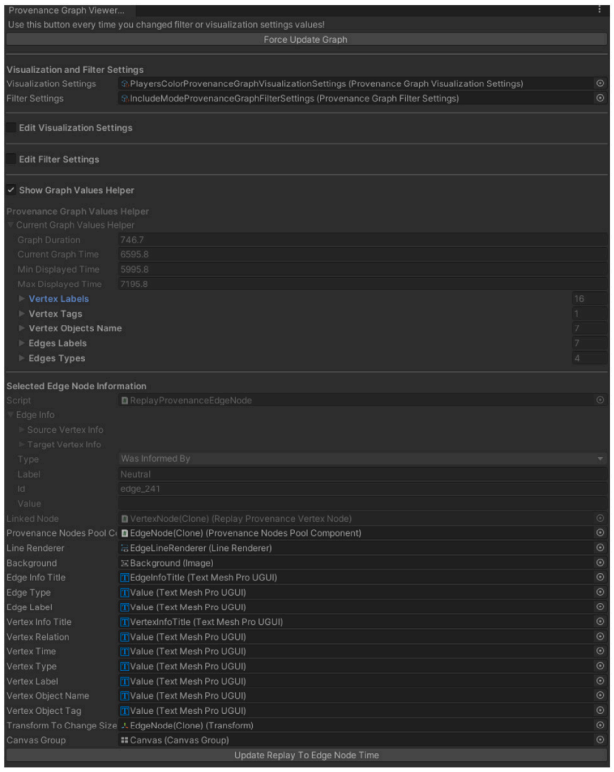

Fig. 8. Provenance Graph Viewer Inspector window.

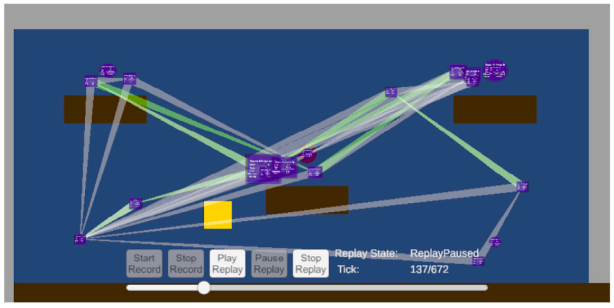

Fig. 9. Running sample project replay with ReplayCanvas object showing provenance information.

- Display information about the current graph, like possible values for each piece of information that can be configured in layout, filter rules, and session duration.
- Display selected vertex or edge information
- Update replay time to the selected vertex or edge.
- Refresh provenance graph visualization

The Fig. 8 presents our custom window with all the mentioned features. This window has "Edit Visualization Settings" and "Edit Filter Settings" flags that allow the user to update visualization and/or filter rules that were shown in Fig. 5 and Fig. 6 respectively. Selecting the "Show Graph Values Helper" flag, the user can observe different information about the current game session, such as graph duration, all vertex labels, tags, object names, edges labels, and types captured in the game session. Furthermore, when the user selects a node, this window presents all provenance information of the selected node and has a button to update replay time to the exact moment that the node event occurs in the game session.

4.5. Available as open-source software

In order to allow other researchers and game developers to use our tool, we make it available at GitHub.[7] We provide the tool's source code, two videos, and a sample project on the GitHub page.

About the videos, the first[8] shows how to use PinGU Replay features, and the second[9] shows an analysis sample that illustrates how game developers can use this tool to analyze a game session.

Fig. 9 shows the sample project, and it is a 2D platform game where the player needs to collect five coins to finish. This project is configured to save player and coins information to reproduce replay and store provenance information about the agent (*i.e.*, player), entities (*i.e.*, collectible), activities (*i.e.*, player position, collect and spawn collectibles), influences (*i.e.*, player position influences the next position) and attributes (*i.e.*, the collectible entity has the "CollectableName" attribute).

For the sake of conciseness, we list below the overall features that a node from a provenance graph of this sample project might contain:

- **Common features**: ObjectPosition_X, ObjectPosition_Y, ObjectPosition_Z, ObjectName, ObjectTag, ObjectID
- **Collectable features**: CollectableName

However, each node type draws a different subset of the above-mentioned features.

Table 1 presents an overview of sample project node feature information. Additionally, PlayerPosition activity is collected every 1.5 s and has an influence with the next PlayerPosition activity with value 1 and configured to be consumed after generating one edge, so the player's previous position has a positive relation with the player's next position, which is illustrated by green edges in Fig. 9.

5. Case studies

In our previous work, we performed a case study to evaluate ProvReplay. This case study encompassed various stages, including defining research hypotheses (RH), integrating PinGU and PinGU Replay into

[7] https://github.com/gems-uff/ping
[8] https://www.youtube.com/watch?v=MYl8XwFbTVM
[9] https://www.youtube.com/watch?v=NXfzvyxo3w8

Table 1
Sample project provenance node and feature information.

Node label	Node type	Node feature set
Player	Agent	Common
PlayerPosition	Activity	Common
SpawnCollectable	Entity	Common, Collectable
CollectCollectable	Activity	Common, Collectable

Smoke Squadron, collecting game sessions with actual players, conducting experiments with game developers analyzing game sessions using PinGU Replay, and evaluating the developers' responses through a questionnaire and interview to validate our research hypotheses.

In this paper, we defined the methodology behind our case study experiment where game developers analyze game sessions using PinGU Replay, presented a more detailed analysis of the Smoke Squadron case study, integrated PinGU and PinGU Replay into Survivor Heroes, collected real game session data and applied our game developers qualitative analysis methodology into Survivor Heroes game.

5.1. Research hypotheses

In order to assess the efficiency and effectiveness of Prov-Replay, we formulated two research hypotheses (RH).

- **RH 1** - The visualization of the provenance graph with the replay provides more information about the context than only using the replay, facilitating the understanding of different elements involved in players' decision-making and allowing more in-depth qualitative analysis;
- **RH 2** - Visualizing the provenance graph with the replay allows the user to quickly identify points with relevant events in the game session, making it easier to analyze points of interest in the game session.

5.2. Game developers qualitative analysis methodology

We conducted analysis sessions with six developers, three from Smoke Squadron and three from Survivor Heroes. Each developer analyzed sessions of their own game. Ops Game Studio and Double Dash Studios are small companies. Hence, they have a limited number of developers for each project, so this number of participants was almost the entire team involved in the game's production.

Each analysis session lasted 2 h and was divided into seven steps. We introduce participants to the experiment roadmap and objectives in the *first step*. In the *second step*, the participants had 15 min to analyze the first game session using only the replay module without any information about the provenance graph. In the *third step*, we introduce participants to provenance concepts and show how to use the interactive provenance graph module of PinGU Replay. This step had 20 min of duration. In the *fourth step*, the participants had 15 min to analyze the first game session again, but using replay with interactive provenance graph visualization; the objective of this step is for participants to confirm previously analyzed information and to acclimate themselves to how to use the visualization tool. In the *fifth step*, the participants had 30 min to analyze the second game session using all PinGU replay features. In the *sixth step*, the participants answered questions about their experience in analyzing the game using only the replay and using the replay with graph visualization, as well as the positive and negative aspects and how we could improve PinGU Replay. In the *final step*, we conducted a small interview to collect any additional feedback and find out if they believed the tool could help them improve future projects they are developing. 2nd, 4th, and 5th steps were recorded so that we could observe how they used PinGU Replay features.

Furthermore, at the end of 2nd, 4th, 5th, and 6th steps, the developers answered a form with some questions about the analyzed game sessions and their experience using PinGU Replay. Below are listed questions that all developers answered in 6th step. These questions were about PinGU Replay. The other step questions were specific to each game and are described in the following sessions.

Fig. 10. Screenshots of Smoke Squadron game.

1. Did you feel that visualizing the graph in conjunction with replay improved (or hampered) the analysis process compared to using only replay? Why?
2. What did you like about PinGU Replay?
3. What did you dislike about PinGU Replay?
4. What can we improve in PinGU Replay?
5. Do you want to comment on anything else?

5.3. Case study I - Smoke Squadron

Smoke Squadron[10] is a split screen local multiplayer arcade flight battle game under development by the indie game studio Ops Game Studio,[11] which provided access to game source code to capture provenance data and realize experiments.

The game match consists of players controlling a small remote airplane. Players must battle each other using machine guns, missiles, and a solid smoke trail that kills at touch. The match ends when one of the players loses all his life. Fig. 10 presents a screenshot of the game.

5.3.1. PinGU and PinGU Replay integration

Previously, Melo et al. [48] integrated the PinGU framework into Smoke Squadron to collect provenance data. The study details all the provenance information that PinGU collects in this game. We list below the most relevant attributes that the Smoke Squadron provenance graph might contain:

- **Type attribute**: Provenance node type (*i.e.*, Agent, Activity, or Entity)
- **Date attribute**: Time the node happened since the game started.
- **Object Tag attribute**: determines the game element that instantiates that node. The possible values are Player01, Player02, Smoke, Rocket, SmokeItem, SmokeItemHalf, and SmokeSpawner.

[10] https://youtu.be/g9h720URmlw
[11] https://www.opsgamestudio.com

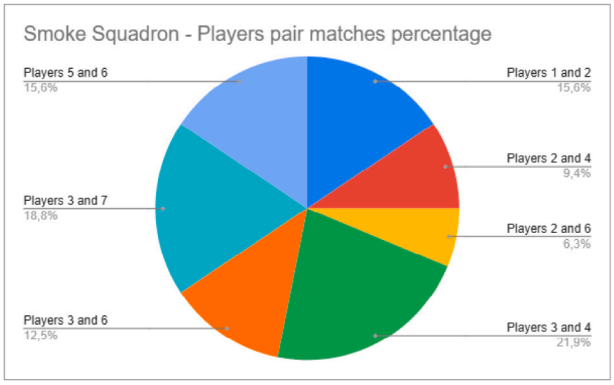

Fig. 11. Smoke Squadron played matches' percentage of players pair.

- **Label attribute**: determines a label for the node. The possible values depend on the Type and Object Tag values. For example, the label value of a node with Object Tag Missile and Type Entity is the type of the missile. Conversely, the label value of a node with the same ObjectTag but with Type Activity is the action performed by that missile.
- **Movement related attribute**: position, direction, and rotation in axis X, Y, and Z, and Speed.
- **Weapon related attributes**: Weapons and Smoke's counters, cooldowns, and explosion timers.

To integrate PinGU Replay in Smoke Squadron, we configured the replay object and replay transform component on the following game elements: players' airplanes and pick-up items, Missiles, and Smokes. These configurations collected all information about position and rotation from game objects. Furthermore, we need to create a replay smoke component to allow the collection of domain-specific information since this information is very important for developers to understand the entire context of the game. Ultimately, it was necessary to make these settings in 15 game objects.

5.3.2. Playtest data collect and session select

We conducted a playtest with seven male players, aged between 22 and 34 years old, with intermediate to advanced game skills. The playtest session lasted 4 h and generated 32 game sessions.

During the playtest, the players only needed to play the game normally and try to win the adversary. We arranged for players to face each other during some matches, allowing them to get used to the game with the aim of having matches where players were learning how to play and others where they already knew how to play. Fig. 11 presents the played matches percentage of players pair.

Lately, we selected two specific sessions that we considered to contain distinct data to be analyzed by game developers. The first selected session had 114 s of duration. This session was the second match between players 5 and 6. These players had some difficulties getting accustomed to the game, so both players were still learning how to play. We considered this session as a sample match between beginner players. The resulting provenance graph of this session has 3422 edges and 2465 vertices. The second session is 343 s long. This session was the third match between players 1 and 2. Unlike previous players, these players quickly got accustomed to the game, so both players were already familiar with it. We considered this session as a sample match between intermediate players. The resulting provenance graph of this session has 12749 edges and 8912 vertices.

5.3.3. Specific analysis information

The primary roles of Smoke Squadron developers who participated in the test are 3D Artist, Programmer, and Game Designer/Producer. Nevertheless, in a small company, it is common for developers to be aware and help in other production areas, so all three have knowledge of game design and usually analyze players' behavior during playtest sessions. However, none of them have prior experience using provenance or other tools to assist in this analysis. This distribution of primary roles also allowed us to analyze the tool with three different and complementary user profiles. Below are listed questions they answered during steps 2, 4, and 5 of the analysis process.

1. What were the causes of player deaths?
2. Which moments did the players die?
3. What weapons did the players use throughout this session?
4. Were there times when a MachineGun weapon hit a player?
5. In your opinion, which weapon was most effective in this session? Why?
6. Which items did each player collect?
7. Describe any other information or observation you consider relevant.

5.3.4. Results

> **Finding 1:** Using a provenance graph combined with a replay to analyze game sessions allowed Smoke Squadron developers to become more confident in their analyses and correct some observations when using only the replay.

When using only the replay to analyze game sessions, all game developers remained in doubt about the causes of player deaths, whether players collected items, and whether the MachineGun weapon hit players. Due to this confusion, they answered some of these questions incorrectly, either because of a lack of clarity or not being able to observe the moment of the replay in which it happened. When they used provenance graph data and visualization with replay information, they could confirm their assumptions by filtering out all moments in which these events happened during the game session and checking the provenance relation. This allowed them to fix some of their responses and update their conclusions. Consequently, they agreed that the graph played a crucial role in enhancing their understanding of the context and confirming their initial assumptions.

> **Finding 2:** The possibility of filtering desired events in the provenance graph and synchronizing the replay with the moment of this event allowed Smoke Squadron developers to more efficiently navigate the replay during analysis.

When game developers analyzed the second session in the *fifth step*, two out of three developers started their analysis using the graph to visualize the desired information and answer the questions directly. In contrast, the other developer preferred to watch the entire replay to understand how the session went. However, as soon as the game developer started answering the questions, they also used the graph to see the desired moments and analyze them again. Furthermore, their answers confirmed that they all believe that the graph made it easier to locate the desired events and provided a faster way to find the moments that should be analyzed.

> **Finding 3:** Smoke Squadron developers believe this kind of tool would help them improve their games.

Developers mentioned that the tool is intuitive and practical, although it requires practice to use it more efficiently. The developers were optimistic, believing this kind of tool would help them improve

Fig. 12. Screenshots of Survivor Heroes game.

their games and apply them to different game productions. The developers use layout settings that change players' node color, confirming the layout module's usefulness in helping the analysis. Also, two of three developers mentioned that they believe the possibility of creating presets will be greatly useful in the long term.

Furthermore, the developers expressed specific areas for improvement, with a primary focus on the process of provenance collection. This happens because a specialist, instead of the developers, instrumented the provenance collection. In an ideal scenario, the developers should instrument this collection, making the analysis process even easier. However, another important point to note is that the developers were not aware of provenance in games, and even if they did not know and did not participate in the collection instrumentation, they managed to benefit from the tool even without having previous experience. Other mentioned improvements were related to tool usability and performance.

5.4. Case study II - Survivor Heroes

Survivor Heroes[12] is an online multiplayer roguelike survival adventure game under development by the indie game studio Double Dash Studios,[13] which provided access to game source code to capture provenance data and realize experiments.

The game match consists of a team of 3 players fighting against huge hordes of enemies. Players need to control the hero to defeat enemies, level up, and choose skills to overcome upcoming hordes of enemies. The match ends when all players die or when they survive for 8 min and defeat the boss. Fig. 12 presents screenshots of the game.

5.4.1. PinGU and PinGU Replay integration

Like most games, Survivor Heroes presents several game objects with different feature sets. In Survivor Heroes' provenance graphs, we focused on collecting information about players and Items. For the sake of conciseness, we list below the overall features that a node from a Survivor Heroes provenance graph might contain:

- **Common features**: ObjectPosition_X, ObjectPosition_Y, ObjectPosition_Z, ObjectName, ObjectTag, ObjectID
- **Player features**: IsBot, DamageUpgrades, HealthUpgrades, DefUpgrades, PotUpgrades
- **Player healthy features**: HpPercent
- **Player group features**: Players
- **Player name features**: OldName, NewName
- **Player abilities features**: Attacks, Passives
- **Item features**: CollectableName

[12] https://youtu.be/E7c0pTs2Zz8
[13] https://www.doubledashstudios.com

Table 2
Survivor Heroes' provenance node and feature information.

Node label	Node type	Node feature set
Player1	Agent	Common, Player, Player name
Player2	Agent	Common, Player, Player name
Player3	Agent	Common, Player, Player name
PlayerPosition	Activity	Common
PlayerHpDecrease	Activity	Common, Player healthy
TwoPlayersSeparated	Activity	Common, Player group
TwoPlayersTogether	Activity	Common, Player group
ThreePlayersSeparated	Activity	Common
ThreePlayersTogether	Activity	Common
PlayerDie	Activity	Common
PlayerResurrect	Activity	Common
PlayerResurrect	Activity	Common
SpawnItem	Entity	Common, Item
CollectItem	Activity	Common, Item
UpgradeAbilities	Activity	Common, Player abilities
UpdateObjectName	Activity	Common, Player name

Table 3
Survivor Heroes' activities node influences.

Source	Target	Expire	Influence
PlayerPosition	PlayerPosition	–	0
PlayerHpDecrease	TwoPlayersSeparated	10	−1
PlayerHpDecrease	ThreePlayersSeparated	10	−1
TwoPlayersTogether	TwoPlayersSeparated	10	−1
TwoPlayersTogether	ThreePlayersTogether	10	1
ThreePlayersTogether	ThreePlayersSeparated	10	−1
PlayerDie	TwoPlayersTogether	15	−1
PlayerDie	ThreePlayersTogether	15	−1
CollectItem	TwoPlayersSeparated	10	−1
CollectItem	ThreePlayersTogether	10	−1
UpgradeAbilities	UpgradeAbilities	–	0

However, each node type draws a different subset of the above-mentioned features. Table 2 presents an overview of Survivor Heroes' node feature information.

Another important piece of information about provenance data is the influences. This information is responsible for generating the edges between provenance vertices. In our implementation for Survivor Heroes, all influences are configured to be consumed after generating one edge. However, some influences expire if they do not generate a link after a configured expire seconds value (*i.e.*, PlayerHpDecrease generates an influence that expires after 10 s if TwoPlayersSeparated activity does not happen). Furthermore, each influence has a configured influence value (*i.e.*, TwoPlayersTogether activity generates a positive influence to ThreePlayersTogether and a negative influence to TwoPlayersSeparated). Table 3 presents an overview of Survivor Heroes' activities node influences information.

To integrate PinGU Replay in Survivor Heroes, we configured the replay object and replay transform component on the following game elements: players' heroes, enemies, buildings, weapons, and abilities. These configurations collected all information about position, rotation, and scale from game objects. Furthermore, we need to create replay match time, replay player name, and replay player HP components to allow the collection of domain-specific information since this information is important for developers to understand the entire context of the game. In the end, it was necessary to make these settings in 87 game objects.

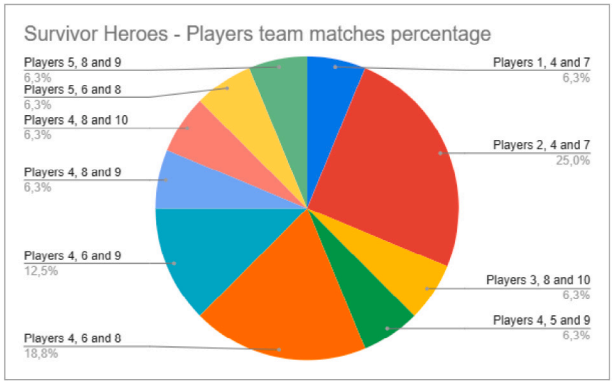

Fig. 13. Survivor Heroes played matches' percentage of players team.

5.4.2. Playtest data collect and session select

We conducted a playtest with ten players, nine male and one female, aged between 19 and 33 years old, with intermediate to advanced game skills. The playtest session lasted 3 h and generated 16 game sessions.

During the playtest, the players only needed to play normally and cooperate to win the match. Similar to the Smoke Squadron playtest, we arranged for players to play with the same team during some matches, allowing them to get used to the game with the aim of having matches where players were learning how to play and others where they already knew how to play. Fig. 13 presents the played matches percentage of players team.

Finally, we selected two specific sessions that we considered to contain distinct data to be analyzed by game developers. The first selected session was 323 s long. This session was the team's first match with players 2, 4, and 7, so the players were still learning how to play, and one did not have any experience with this kind of game. We considered this session as a sample match between beginner players. The resulting provenance graph has 1046 edges and 402 vertices. The second session was 746 s long. This session was the team's third match with players 4, 6, and 8. At this point, all the players had already played with other teams and with each other, so all players were already more experienced in the game, and it was the first time they had won the match. We considered this session as a sample match between intermediate players. The resulting provenance graph has 2277 edges and 837 vertices.

5.4.3. Specific analysis information

The primary roles of Survivor Heroes developers are Programmer/Producer, Game Designer, and Technical Artist. Just like Ops Game Studio, the Double Dash Studios is a small studio, so all three game developers have knowledge of game design and usually analyze players' behavior during playtest sessions. None of them have prior experience using provenance or other tools to assist in this analysis. Below are listed questions they answered during steps 2, 4, and 5 of the analysis process.

1. What were the moments when players died?
2. What skills did players choose? And what are the levels of these skills?
3. In your opinion, what were the most effective skills in this session? Why?
4. In your opinion, why did the players fail? (only for game session 1)
5. In your opinion, why did the players win? (only for game session 2)
6. Comment what you think of the skills and experience of each player.
7. Describe any other types of information you were able to analyze and how you arrived at it.

5.4.4. Results

> **Finding 4:** In accordance with Finding 1, Survivor Heroes developers become more confident in their analyses when using a provenance graph combined with replay.

When using only the replay to analyze game sessions, all game developers remained in doubt about the skills players choose, players upgrade levels, and if they were letting some death information pass, as the players were not always on the same screen and even being able to alternate among the players it was complex to ensure that they were seeing everything. Due to this confusion, they were unsure about some answers, either because of a lack of clarity or an inability to observe the moment of the replay in which it happened. Similar to Smoke Squadron developers, when they used provenance graph data and visualization with replay information, they could confirm their assumptions. This allowed them to get safe with their responses and conclusions. Consequently, they agreed that the graph was crucial in enhancing their understanding of the context and confirming their initial assumptions.

> **Finding 5:** In agreement with Finding 2, Survivor Heroes developers filtered desired events in the provenance graph and synchronized replay with these events to navigate in replay during analysis.

When game developers analyzed the second session in the *fifth step*, as soon as game developers started answering the questions, they all used the graph to see the desired moments and updated the replay to this specific moment to analyze. Furthermore, similar to Smoke Squadron developers, their answers confirmed that they all believe that the graph made it easier to locate the desired events and provided a faster way to find the moments that should be analyzed, and it was very useful to facilitate and accelerate the analysis process.

> **Finding 6:** Survivor Heroes developers were optimistic about using this tool in other projects.

Developers mentioned that they managed to see applications of this tool in projects that are being developed, that they would like this tool to be applied to other game engines, and that replay could be done as a spectator mode rather than a classic replay as it would give more information to the analysis.

Furthermore, the developers expressed specific areas for improvement, primarily focusing on the tool's usability. All developers mentioned that the analysis dashboard helped to identify information and filter desired information in the game session. They also mentioned that using the graph in conjunction with the replay improved the analysis process. They also were optimistic, believing this kind of analytics helped them improve their games and apply them to different game productions. However, they also mentioned that the tool needs improvements in its usability to be accessible to less experienced developers and help further in the analysis. Another important point to note is that just like the Smoke Squadron developers, Survivor Heroes developers were not aware of provenance in games, and even not knowing and not participating in the collection instrumentation, they managed to benefit from the tool even without having previous experience.

6. Conclusion

In this paper, we extended our previous work by providing more detail about Prov-Replay implementation through PinGU Replay, applied our experiment methodology to a new under-development commercial game, collected real game sessions, did an experiment where

game developers analyzed specific game sessions using PinGU Replay and validated our findings through results collected from two under-development commercial games of different genres.

Later, we analyzed the results of six game developers' experiments to check our research hypotheses. Regarding **RH 1**, we claim that our approach provides a better understanding of game sessions since all developers reported that this tool provides access to significant information to understand better what was happening in the game and arrive at sounder answers and conclusions. About **RH 2**, our findings confirm that the utilization of provenance graph data and visualization facilitates the rapid identification of key points that contain significant events since all developers reported that this tool provides an easy way to find and access the exact desired event moment. In contrast to seeking this information by watching replay repeatedly, they can visit desired events without watching and making notes about the entire session replay. After this analysis, our results reinforce that Prov-Replay could improve both the efficiency and effectiveness of the qualitative analysis process since it provides a means for a better understanding of the whole game context information and easy identification of relevant events. As a result, this significantly aids users' analysis process. Moreover, we contribute within the field by making PinGU Replay available on GitHub as open-source software with a sample project, providing an example of how to configure PinGU to collect provenance data and PinGU Replay to capture session data and reproduce replay synced with provenance graph visualization. We made two videos available showing PinGU Replay features and how to make an analysis using their features. We believe these elements provide an efficient way for other researchers and game developers to use and expand our work.

During development, we observed points of improvement that made this framework inefficient in some scenarios. The first is quantitative analysis. Our approach is based on providing detailed information about a game session to perform a more in-depth analysis. Still, it has no resource that facilitates its use in quantitative analysis. By creating visualization presets and filters, it is possible to check whether some behavior is reproduced in different game sessions, but this process is still quite laborious. Another important issue is about performance. The more information the provenance graph presents, the more costly it is to process all the visualization rules and filters and synchronize them with the replay. Therefore, depending on the amount of information and the power of the computer used in the analysis, the process of replay reproduction and visualization of the provenance graph may be slow. We believe these limitations can be resolved in future work, making this tool more complete.

Due to the limitations, we intend to improve visualization performance and apply features to make this tool efficient in quantitative analysis. To improve performance, we want to add features to automatically collapse some graph information that could be irrelevant as it presents little difference concerning previous or subsequent information. And about quantitative analysis, we want to add features capable of reproducing more than one replay and provenance graph simultaneously, allowing the user to show, hide, and compare the information of different sessions or different moments of a session at once. Due to the game developers' feedback, we want to implement this tool in other game engines, making it easier for more developers and researchers to use. We intend to improve the tool's usability so that it is accessible to less experienced game developers and has more options to visualize and navigate between the replay and graph visualization more practically. Lastly, we want to implement all Prov-Replay features to be accessible at game runtime, allowing players to analyze their performance in a game session. This means that this tool could be used in the context of a competitive game, not just trying to improve the game but allowing players and professional teams to analyze their matches and their opponents' matches, seeking to improve their performance and win matches.

CRediT authorship contribution statement

Leonardo Thurler: Conceptualization, Formal analysis, Methodology, Software, Writing – original draft, Writing – review & editing. **Sidney Melo:** Conceptualization, Writing – review & editing. **Leonardo Murta:** Writing – review & editing. **Troy Kohwalter:** Writing – review & editing. **Esteban Clua:** Formal analysis, Methodology, Validation, Writing – review & editing.

Declaration of competing interest

We have no financial or non-financial conflict of interest.

Data availability

We are sharing software code as open-source at github with a sample project, we want share provenance and replay data files in the future, but we do not have permission to share games source code.

Acknowledgments

We would like to thank Double Dash Studios and Ops Game Studio for providing access to games' source codes so that we could capture provenance data and realize experiments. We also want to thank all players and developers who helped us in our case studies by participating in playtests and analysis sessions.

References

[1] A. Drachen, M.S. El-Nasr, A. Canossa, Game Analytics: Maximizing the Value of Player Data, Springer, 2013.
[2] E. Andersen, Y.-E. Liu, R. Snider, R. Szeto, Z. Popović, Placing a value on aesthetics in online casual games, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2011, pp. 1275–1278.
[3] A. Calvo-Morata, D.C. Rotaru, C. Alonso-Fernández, M. Freire-Morán, I. Martínez-Ortiz, B. Fernández-Manjón, Validation of a cyberbullying serious game using game analytics, IEEE Trans. Learn. Technol. 13 (1) (2020) 186–197.
[4] G. Andrade, G. Ramalho, H. Santana, V. Corruble, Extending reinforcement learning to provide dynamic game balancing, in: Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence, IJCAI, 2005, pp. 7–12.
[5] V. Volz, G. Rudolph, B. Naujoks, Demonstrating the feasibility of automatic game balancing, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, pp. 269–276.
[6] C. Pedersen, J. Togelius, G. Yannakakis, Modeling player experience for content creation, Comput. Intell. AI Games IEEE Trans. 2 (2010) 54–67.
[7] A. Zook, B. Harrison, M.O. Riedl, Monte-Carlo tree search for simulation-based strategy analysis, 2019, arXiv preprint arXiv:1908.01423.
[8] P. Guardini, P. Maninetti, Better game experience through game metrics: A rally videogame case study, in: Game Analytics: Maximizing the Value of Player Data, Springer, 2013, pp. 325–361.
[9] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, G.N. Yannakakis, Predicting player behavior in Tomb raider: Underworld, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, 2010, pp. 178–185.
[10] A. Tychsen, A. Canossa, Defining personas in games using metrics, in: Proceedings of the 2008 Conference on Future Play: Research, Play, Share, 2008, pp. 73–80.
[11] S.A. Melo, T.C. Kohwalter, E. Clua, A. Paes, L. Murta, Player behavior profiling through provenance graphs and representation learning, in: Proceedings of the 15th International Conference on the Foundations of Digital Games, 2020, pp. 1–11.
[12] E. Kleinman, S. Ahmad, Z. Teng, A. Bryant, T.-H.D. Nguyen, C. Harteveld, M.S. El-Nasr, "And then they died": Using action sequences for data driven, context aware gameplay analysis, in: Proceedings of the 15th International Conference on the Foundations of Digital Games, 2020.
[13] E. Kleinman, N. Preetham, Z. Teng, A. Bryant, M. Seif El-Nasr, "What happened here!?" a taxonomy for user interaction with spatio-temporal game data visualization, Proc. ACM Hum.-Comput. Interact. 5 (CHI PLAY) (2021).
[14] B. Medler, M. John, J. Lane, Data cracker: Developing a visual game analytic tool for analyzing online gameplay, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 2365–2374.

[15] J.H. Kim, D.V. Gunn, E. Schuh, B. Phillips, R.J. Pagulayan, D. Wixon, Tracking real-time user experience (TRUE): A comprehensive instrumentation solution for complex systems, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 443–452.
[16] A. Drachen, A. Canossa, Analyzing spatial user behavior in computer games using geographic information systems, in: Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era, MindTrek '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 182–189.
[17] M. Ashton, C. Verbrugge, Measuring cooperative gameplay pacing in world of warcraft, in: Proceedings of the 6th International Conference on Foundations of Digital Games, FDG '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 77–83.
[18] N. Hoobler, G. Humphreys, M. Agrawala, Visualizing competitive behaviors in multi-user virtual environments, in: IEEE Visualization 2004, 2004, pp. 163–170.
[19] J.L. Miller, J. Crowcroft, Avatar movement in world of warcraft battlegrounds, in: 2009 8th Annual Workshop on Network and Systems Support for Games, NetGames, 2009, pp. 1–6.
[20] A. Drachen, A. Canossa, G.N. Yannakakis, Player modeling using self-organization in tomb raider: Underworld, in: 2009 IEEE Symposium on Computational Intelligence and Games, 2009, pp. 1–8.
[21] R. Thawonmas, M. Kurashige, K. Iizuka, M. Kantardzic, Clustering of online game users based on their trails using self-organizing map, in: R. Harper, M. Rauterberg, M. Combetto (Eds.), Entertainment Computing - ICEC 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 366–369.
[22] E. Andersen, Y.-E. Liu, E. Apter, F. Boucher-Genesse, Z. Popović, Gameplay analysis through state projection, in: Proceedings of the Fifth International Conference on the Foundations of Digital Games, FDG '10, Association for Computing Machinery, New York, NY, USA, 2010, pp. 1–8.
[23] Y.-E. Liu, E. Andersen, R. Snider, S. Cooper, Z. Popović, Feature-based projections for effective playtrace analysis, in: Proceedings of the 6th International Conference on Foundations of Digital Games, FDG '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 69–76.
[24] S. Ahmad, A. Bryant, E. Kleinman, Z. Teng, T.-H.D. Nguyen, M.S. El-Nasr, Modeling individual and team behavior through spatio-temporal analysis, in: Proceedings of the Annual Symposium on Computer-Human Interaction in Play, 2019.
[25] G. Wallner, S. Kriglstein, A spatiotemporal visualization approach for the analysis of gameplay data, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 1115–1124.
[26] G. Wallner, S. Kriglstein, Visualizations for retrospective analysis of battles in team-based combat games: A user study, in: Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play, 2016, pp. 22–32.
[27] G. Wallner, S. Kriglstein, Multivariate visualization of game metrics: An evaluation of hexbin maps, in: Proceedings of the Annual Symposium on Computer-Human Interaction in Play, in: CHI PLAY '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 572–584.
[28] W. van den Broek, G. Wallner, R. Bernhaupt, Modata – improving dota 2 experience and spectatorship through tangible gameplay visualization, in: Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts, in: CHI PLAY '19 Extended Abstracts, Association for Computing Machinery, New York, NY, USA, 2019, pp. 723–730.
[29] G. Wallner, N. Halabi, P. Mirza-Babaei, Aggregated visualization of playtesting data, in: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1–12.
[30] N. Halabi, G. Wallner, P. Mirza-Babaei, Assessing the impact of visual design on the interpretation of aggregated playtesting data visualization, in: Proceedings of the Annual Symposium on Computer-Human Interaction in Play, in: CHI PLAY '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 639–650.
[31] Y.-T. Kuan, Y.-S. Wang, J.-H. Chuang, Visualizing real-time strategy games: The example of StarCraft II, in: 2017 IEEE Conference on Visual Analytics Science and Technology, VAST, 2017, pp. 71–80.
[32] A. Afonso, M. Carmo, T. Gonçalves, P. Vieira, VisuaLeague: Player performance analysis using spatial-temporal data, Multimedia Tools Appl. 78 (2019).
[33] T. Kohwalter, E. Clua, L. Murta, Provenance in games, in: Braz. Symp. Games Digit. Entertain. SBGAMES, 2012, pp. 162–171.
[34] T. Costa Kohwalter, L. Gresta Paulino Murta, E. Walter Gonzalez Clua, Capturing game telemetry with provenance, in: 2017 16th Brazilian Symposium on Computer Games and Digital Entertainment, SBGames, 2017, pp. 66–75.
[35] T. Kohwalter, F. Figueira, E. Serdeiro, J.R. Da Silva Junior, L. Murta, E. Clua, Understanding game sessions through provenance, Entertain. Comput. 27 (2018).
[36] S. Melo, A. Paes, E. Clua, T. Kohwalter, L. Murta, Detecting long-range cause-effect relationships in game provenance graphs with graph-based representation learning, Entertain. Comput. 32 (2019) 100318.
[37] T.C. Kohwalter, L.G. Murta, E.W. Clua, Provchastic: Understanding and predicting game events using provenance, in: International Conference on Entertainment Computing, Springer, 2020, pp. 90–103.
[38] L. Thurler, S. Melo, E. Clua, T. Kohwalter, Prov-replay: A qualitative analysis framework for gameplay sessions using provenance and replay, in: P. Ciancarini, A. Di Iorio, H. Hlavacs, F. Poggi (Eds.), Entertainment Computing – ICEC 2023, Springer Nature Singapore, Singapore, 2023, pp. 31–40.
[39] G. Wallner, Play-graph: A methodology and visualization approach for the analysis of gameplay data, in: FDG, 2013, pp. 253–260.
[40] T.C. Kohwalter, E.G.W. Clua, L.G.P. Murta, Game flux analysis with provenance, in: D. Reidsma, H. Katayose, A. Nijholt (Eds.), Advances in Computer Entertainment, Springer International Publishing, Cham, 2013, pp. 320–331.
[41] T. Kohwalter, T. Oliveira, J. Freire, E. Clua, L. Murta, Prov viewer: A graph-based visualization tool for interactive exploration of provenance data, in: Provenance and Annotation of Data and Processes: 6th International Provenance and Annotation Workshop, IPAW 2016, McLean, VA, USA, June 7-8, 2016, Proceedings 6, Springer, 2016, pp. 71–82.
[42] L. Moreau, P. Missier, PROV-n: The provenance notation, 2012, URL https://www.w3.org/TR/prov-n/. (Last Accessed 17 Apr 2024).
[43] Y. Gil, S. Miles, PROV model primer, 2010, URL https://www.w3.org/TR/prov-primer/. (Last Accessed 17 Apr 2024).
[44] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, J.V. den Bussche, The open provenance model core specification (v1.1), Future Gener. Comput. Syst. 27 (6) (2011) 743–756.
[45] C. Wagner, Developing your own replay system, 2004, URL https://www.gamedeveloper.com/programming/developing-your-own-replay-system. (Last Accessed 17 Apr 2024).
[46] A. Montville, Implementing a replay system in unity and how i'd do it differently next time, 2014, URL https://www.gamedeveloper.com/programming/implementing-a-replay-system-in-unity-and-how-i-d-do-it-differently-next-time. (Last Accessed 17 Apr 2024).
[47] T. Engel, Creating a replay system in unity, 2020, URL https://www.kodeco.com/7728186-creating-a-replay-system-in-unity. (Last Accessed 17 Apr 2024).
[48] S.A. Melo, E. Clua, A. Paes, Heterogeneous graph dataset with feature set intersection through game provenance, in: Workshop on Graph Learning Benchmarks, 2021, https://Graph-Learning-Benchmarks.Github.Io/Assets/Papers/Heterogeneous-Graph-Dataset-Game-Provenance.Pdf.