2014 Brazilian Symposium on Computer Games and Digital Entertainment

A Non-intrusive Approach for 2D Platform Game Design Analysis Based on Provenance Data Extracted from Game Streaming

Lidson B. Jacob, Troy C. Kohwalter,
Esteban W. G. Clua and Daniel de Oliveira
Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói, RJ, Brazil
{ljacob, tkohwalter, esteban, danielcmo} @ic.uff.br

Alex F. V. Machado
Departamento de Computação
Instituto Federal de Educação, Ciência e Tecnologia
Sudeste de Minas Gerias
Rio Pomba, MG, Brazil
alex.machado@ifsudestemg.edu.br

Abstract - **The usage of provenance data drastically increases the potential for game data mining since it is able to record causes, effects and relationships of events and objects during a game session. However, it commonly requires modifications in the game engine in order to collect such provenance data. The modifications in the game engine may be unviable in commercial (and not open source) systems. In this paper, we propose a novel and non-intrusive approach for collecting provenance data in digital games. Our proposal collects provenance data using image processing mechanisms and pre-defined image patterns, thus avoiding accessing and modifying the source code of the game. Using our approach, we are able to generate, analyze and visualize game design features based on the gameplay flow using provenance data. Furthermore, we evaluated our proposal with a well known commercial 2D game, called "*Super Mario World*".**

Keywords – provenance; game analytics; image processing; image analysis

I. INTRODUCTION

The worldwide video game marketplace reached US$ 93 billion in 2013 according to [5]. Only in 2013 a huge number of game titles were released, including new versions of well-established games, such as "Assassin's Creed" and "Call of Duty". With thousands of new game titles released every year, the game industry faces a difficult and important task of attracting and maintaining the interest of players for long periods of time [2]. The popularization of each game title is essential to generate income and to finance the game's production, as well as the possibility of making future games.

Analyzing and understanding every part of a game session is an important information in order to improve game quality [8]. Thus, we claim that an analysis tool allows for an easier study of player behavior and game design. Therefore, such tool may aid the game designer during the game development and improve the overall quality of the game. By analyzing game data captured from previous game sessions and successful games, the game designer can obtain useful information about features or gameplay mechanics for future productions or updates.

This game data is traditionally represented as a *log* file that registers the game flow in details. However, *log* files are not structured and do not allow to perform queries on the data, which reduces the potential of inferences and data mining. To overcome this limitation, we claim that games should gather and represent game data in a provenance repository. Provenance data represents the ancestry of an object [4]. Provenance of an object, such as an NPC in the game session, contains information about the actions involved with this NPC. It provides important documentation that is essential to preserve the data and to interpret and validate results of a game session.

However, to the best of our knowledge, to capture such provenance data within a game session we have to modify the game engine to be able to register information in certain points of the game flow. This approach may be unviable in several cases since most commercial game titles has proprietary code, which means that no modifications in the engine are allowed. Some approaches already provide ways to capture game data in a structured form such as the recently proposed *Provenance in Games* framework [9], but all of them are intrusive in respect to the source code of the game.

The main goal of this paper is to present a non-intrusive framework capable of gathering and storing provenance data of a game session for analysis. However, unlike the *Provenance in Games* framework, the proposed framework does not require access to the source code in order to gather provenance data. The proposed approach is based on histogram and image processing techniques, thus only requiring accessing the real time streaming of the game session. This way, it can be applied in a variety of proprietary games without access to its source code.

After gathering the provenance data through image processing techniques, we need to display the collected data in a way that can be easily understood and analyzed by the end user (or the game designer). Thus, we have chosen to represent the gathered data in the form of a graph, showing all actions taken by a player in the form of vertices and the relationships between them as edges.

We used the game "*Super Mario World*" (SMW) [12] as a case study. This game was chosen because it is a classic game and with great success. Another reason is because SMW is still used as a template for a large number of games titles of the same genre. It is a platform-based game, where the main character has to follow a path picking items and destroying or deviating enemies.

This paper is organized in four sections besides this introduction: Section II introduces important concepts about provenance and histogram used in this paper. Section III shows related work about provenance in games. Section IV shows how the provenance data is acquired, organized, analyzed, stored, and displayed to game designer in our case study. Finally, Section V concludes this paper, listing contributions, limitations and future work.

II. BACKGROUND KNOWLEDGE

The main purpose of this paper consists on colecting provenance data of a game flow, using image processing techniques. Following we present some important concepts for a better understanting of the proposed approach.

A. *Provenance*

As defined by [10] and [11], provenance is traditionally adopted in several areas as arts and digital libraries. The data provenance refers to the historical documentation of an object or documentation process life cycle of digital objects. The historical documentation of an execution application provides a better comprehension by tracking all transformations and changes as well as causal relationshiops between entities. According to [10], computational methods should be transformed with the purpose of generating a qualified provenance so it can be recovered, analyzed and trustworthy, so that we may understand the result of a historic object, answering questions related to how the elements were achieved and why [4].

Detached in [1], the fundamental provenance concepts, such as storing information of time and location. The provenance references the origin of an information which should contain its identification, generator, date or time information, and sequences of processes applied to it.

For provenance related to computational tasks, there are two variations: prospective and retrospective. In the prospective provenance, the steps followed or processes used to generate a product are captured, allowing the registration of an specification of a computational tasks. The data recorded refers to the required steps to reach a specific result. The retrospective provenance captures the steps that were executed, in addition to information about the circumstance that generated a specific data product. Retrospective provenance also have a *log* that details the execution of computational tasks [4].

First explored provenance in game production and detach the importance of a game designer to visualize what scripts and objects were modified, and how these objects change over time [13]. [13] concluded that data provenance is even more important if the script runtime has an unusual execution model and detach that game-aware runtimes are more difficult to implement than language features. Language features can often be implemented piecemeal, as programming patterns are identified and new language features can be added without adversely affecting the old. Runtimes elements, once architected, can be very interdependent and difficult to change.

B. *Provenance In Games*

In order to adopt provenance for the context of games, it is necessary to map each type of vertices of the provenance graph into elements that can be represented in games. The PROV model uses three types of vertex: *Entities*, *Activities*, and *Agents*. In order to use these vertex types, it is first necessary to define their counterparts in the game context.

In the context of provenance, *entities* are defined as physical or digital objects. Trivially, in our approach they are mapped into objects present in the game, such as weapons and potions. In provenance, an *agent* corresponds to a person, an organization, or anything with responsibilities. In the game context, agents are mapped into characters present in the game, such as non-playable characters (NPCs), monsters, and players. It can also be used to map event controllers, plot triggers, or the game's artificial intelligence overseer that manages the plot. Thus, *agents* represent beings capable of making decisions, while *entities* represent inanimate objects. Lastly, *activities* are defined as actions taken by agents or interactions with other agents or entities. In the game context, *activities* are defined as actions or events executed throughout the game, such as attacking, dodging, and jumping.

With all three types of vertex mapped into the game context, it is also necessary to map their causal relations to create the provenance graph. The PROV model defines some causal relations that can be used similarly to their original context. However, it also provides rules to extend these relationships or to create new ones. For instance, it is possible to create relationships to express the damage done to a character or relationships that affect specific core mechanics from the game, like attack rolls, healing, and interactions with NPCs or objects. Also, the PROV model deals well with the aspect of time, which can be heavily explored in games, especially on games focused on storytelling.

Each NPC in the game should explicitly model its behavior in order to generate and control its actions, providing an array of behavior possibilities. With this explicit model, a behavior controller can register information about the action when it is executed. The main reason of using provenance is to produce a graph containing details that can be tracked to determine why something occurred the way it did.

The information collected during the game is used for the generation of the *game flux log*, which in turn is used for generating the provenance graph. In other words, the information collected throughout the game session is the information displayed by the provenance graph for analysis. Thus, all relevant data should be registered, preferentially at fine grain. The way of measuring relevance varies from game to game, but ideally it is any information that can be used to aid the analysis process.

C. *Histogram*

Since the proposed approach is based on the analysis of game streaming, histogram analysis is a key issue. According to [3], the histogram of an image is a collection of numbers that indicate the percentage of pixels in the image with a determined tonality color. These values are easy to understand and analyze through a bar graph that provides the number of pixels for each tonality corresponding in the image. Through visualization and analysis of an image histogram, we can identify various characteristics, such as the indication of its quality, the contrast level, and its medium shine (if the image is predominantly light or dark).

Each element of this collection is computed as:

$$p_r(r_k) = \frac{n_k}{n} \tag{1}$$

Where:

$0 \leq r_k \leq 1$

$k = 0, 1, \ldots, L-1$, L is the number tonalities of the digitized image;

n = total number pixels in the image;

$p_r(r_k)$ = probability k-th of tonality;

n_k = number of pixels with tonalities that corresponds to k.

III. RELATED WORK

Introduced in [6] a system that captures information from a game of infinite run genre. The captured information is represented as a graphic, so that it becomes quick and easy to understand the behaviors occurred during the game session. It also informs to the game designer characteristics and behaviors of the gameplay as well as the player's behavior during the game.

Some important features that can be analyzed graphically are ilustrated in Figure 1, including distance achieved by the player, the distribution of coins and special items gathered by the player. By analyzing this graphic, it is clear that the player achieved low distances in the initial rounds of the game session. This behavior may be related to being the first contact the player has with the game. Thus the result suggests that the game need a better help or tutorial before starting the game.

Another possible analysis is the distribution of coins and special items. By analysing the graphic, the game designer can verify if they were collected as intended. An important feature can be observed by analyzing the relationships between the data. For example, in the section of the graph where the capture of special items are growing and the distance is constant (round 5 to 12), and immediately after when special items are constant and the distance is growing (round 15 to 20), is possible to infer that the capture of elements disturbed the player's progress. This may be because the items were in difficult local to be captured, inducing the player to get lost. However, the system requires accessing and modifying the source code of the game. The represented information is limited, in our proposal shows more information to the end user.

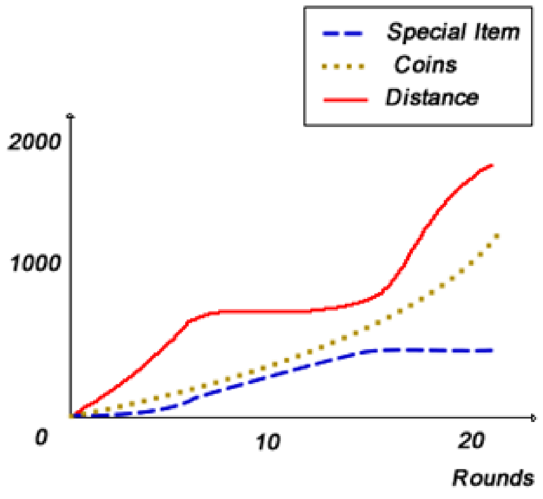


Figure 1: Combined Analysis. [Jacob *et al.* 2013].

Introduced in [9] a framework that captures provenance data during a game session. This data is exported as a game flow log that can be used to generate a provenance graph. The proposed framework has the intention of capturing data while the user plays the game in order to generate a provenance graph that can be shown at the end of the session. This graph is able to help the player to understand the reasons during the game that induced the final result, allowing a feedback to the player about the decisions and actions executed during the game in order to aid in the learning process and identify his mistakes for future sessions. However, the framework requires modifications in the game engine, thus being intrusive.

As a case of study, the authors used a serious game called *Software Development Manager* (SDM). In this game, the user have to manage a software development environment using the concepts of software engineering to better develop the software following the requirements stablished by the company's client. The player has to manage a group of employees by deciding strategies of software development in order to meet the established prerequisites and restrictions. At the end of the development, the player receives a payment in accordance with the quality analysis.

Following this work, [8] proposed an approach for the representation of the provenance data gathered during a game session in the form of a provenance graph. The proposed graph has the intention of representing the actions and decisions made by the player while running the game. This graph allows for the developers and designers to identify possible problems in gameplay by analyzing the provenance graph from the game session.

The graph shown in Figure 2 is the visualization tool named *Prov Viewer* created to generate the provenance graph. The ilustrated graph is just an example that represents a small flow of captured data of a game. The graph notations used follows the exisiting provenance models, with vertices representing the activities, entities and agents according to the legend. The edges represents the relationship between the vertices and can be positively or negatively in accordance with the color. The edge's thickness represents the intensity of the relationship, thick edge represents bigger infuences, dotted edges are neutral relationships and have little importance besides associating vertices neighbors or owners.

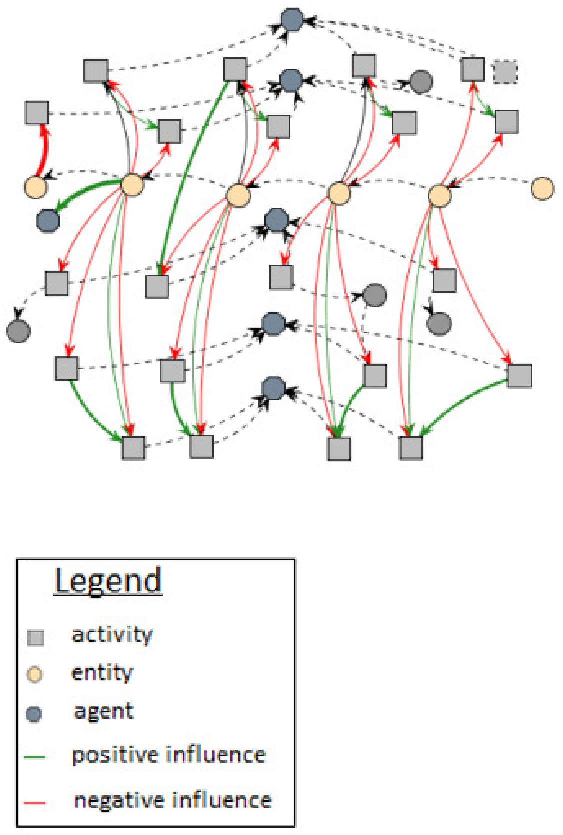


Figure 2: Example of a generated provenance graph. [Kohwalter et al. 2013].

IV. A NON-INTRUSIVE FRAMEWORK FOR EXTRACTING PROVENANCE DATA FROM GAMES

In this paper we propose a novel strategy for extracting and storing provenance data from game session based on image processing techniques. The proposed approach is based in a workflow that is showed in Figure 3. The solution starts with a player interacting with a game in a game session. The entire streaming is captured and the rendered images are stored to be further analyzed. It is important to highlight that this rendered image extraction is not a simple and fast process to be performed. In order to analyze the captired images, we need to have template images that will be identified and recognized in the extracted rendered images. For example, in the case of SMW game, template images for each character (*e.g.* Mario, Koopa, Toad, Yoshi, *etc.*) have to be provided in order to perform image recognization. We also store metadata associated to the template images such as names of the characters and influences in the game.

This way, with the template images and the extractect rendered images we can indeitify the elements in the entire game session by comparing the histograms of each one of the extractetd images with the historgram of template images to verificy if the characters are parte of a specific session. If a character is identified and an action associated to this character is also identified (*e.g.* Mario touches an enemy) we have to store this information in our provenance repository. This provenance repository is further queried by *Prov Viewer* to generate a provenance graph. This way, during the player interaction, his input data are also stored and will be connected with image features, generating a graph with the dynamic elements. The game designer can interpret what happened during the game session by analyzing the provenance graph and can plan future developments or correct bugs in the current version of the game, for example.

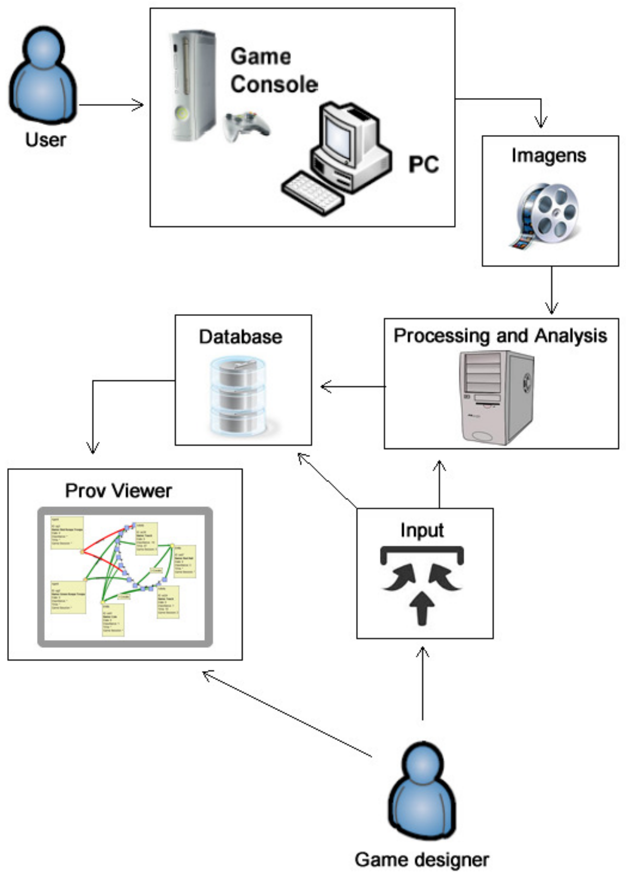


Figure 3: Organization Modules

A. Data Capture

The framework starts capturing images of the game session. In the proposed approach, n frames are analyzed per second, with n being a parameter that is defined by the end user. Ideally, n should have the smallest possible value without incurring in extremely large scale processing that demands too much time, thus being unfeasible to be used. Since processing images require a high computational cost, it may require a long time to process even for medium amounts of images. However, the n number cannot be too low, otherwise information could be lost in the process. For example, if in one second the character performed σ actions, with σ being bigger then n, we then have σ - n actions that will not be appropriately captured. We can state that in Figure 4 where the character *Mario* can perform various actions in one second. By analyzing of Figure 5, we found that n = 3 is a good balance between execution time optimization and reliability.

After the image acquisition, the framework identifies the agents and entities involved. The agents in a game are the main character, such as *"Mario"* and other objects that have some behavior in the game. For example, the *"Yoshi", "Red Koopa"*, and *"Green Koopa"* are classified as agents by framework. The entities are defined as elements that the main character can capture or interact, such as the *"Red Mushroom", "Green Mushroom",* and *"coins"* in SMW example.

Agents and entities are detected by comparing their color histogram. In this paper, we use the RGB color pattern. This is achieved by analyzing the histogram of each agent and entity with a section of image that we call layer. This search is performed by checking if the histogram of the template image is contained in a specified

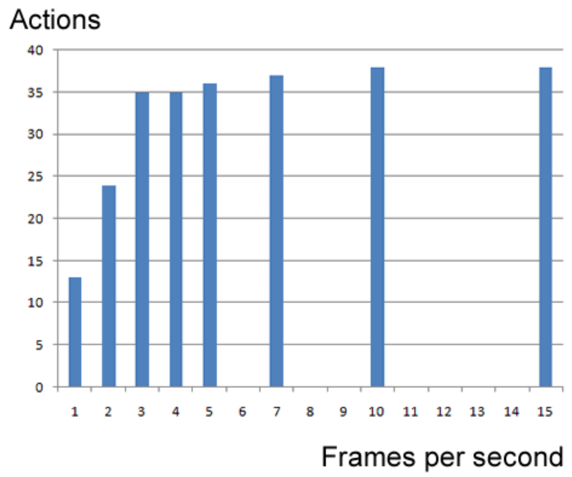


Figure 5: Relation between actions and frames per second showing the loss of information to value n in one game session.

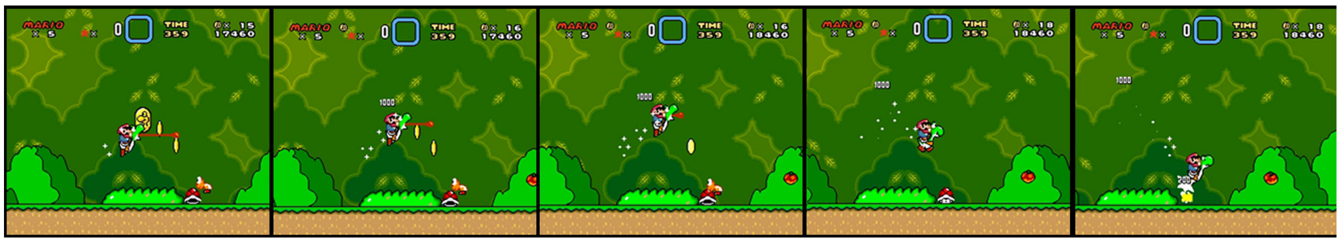


Figure 4: Sequence of pictures that represents each second during a game session.

part of the game frame. The process of searching consists on scanning the template image through the complete frame window. If a template image is not in the layer, then it is considered as not being an object and proceeds to the analysis of the next layer.

As the agents move during the game, the color histogram changes as well. One way to treat this is by comparing the histogram of each frame with the animation of the agents. However, this process costs too much processing time. For example, to analyze an image with 800 × 600 pixels that has a window of 50 × 50, then 37,500 histograms should be analyzed. If we consider an animation being composed of 9 frames, then the amount of histogram to be analyzed for this agent would be 337,500. Remember that this value is calculated for a single image. Nevertheless, we need to analyze a video with hundreds or even thousands of images.

One way we use to optimize this process is to create a setting to define whether or not a layer contains or not a template image. Let us consider ε as the margin of error for each index tonality of RGB and a value δ as margin of error for total percentage. If n_k of the layer is larger than n_k - ε of the template image, we consider that the layer contains k of the template image. Moreover, if $\sum_0^k n_k$ of the layer that contains k of the template image is larger than the $\sum_0^k n_k$ - δ, then we also consider that the layer contains the template image.

Another optimization for this process was sliding the layer with more than one pixel of offset. When a window does not contain a P_{min} value (specified minimum percentage) of the desired object, then the layer does not increments by a number of pixels, but skips an offset corresponding to its complete size. For this paper, we assume that the objects do not occupy the same place. When is confirmed a presence of an object in the window, then the entire layer is skipped.

In the vast majority of games, the main character is usually located in the center of the screen. Thus, we are analyzing only the red central region as presented in Figure 7 to further reduce the processing time. For the framework execution, we painted the border of the layer which contained the histogram of the template image so we can analyze what was correctly being captured. Figure 6 is an example of colors used to represent some objects.

Figure 8 (A) shows an example where the matching correctly occurs. However, objects with similar histogram may lead to incorrect matching, as can be observed with the pipes and coins in Figure 8 (B), caused by the similarity of colors, resulting in a false positive. We can identify this false positive, after an analysis of the data that will be detailed in the next section by comparing what was captured with what should have been captured. We can see in TABLE 1 the sample objects captured with their

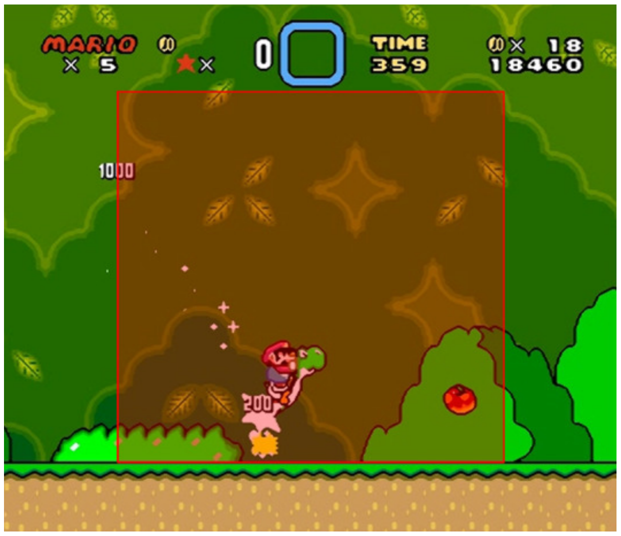


Figure 7: Central area where search happens.

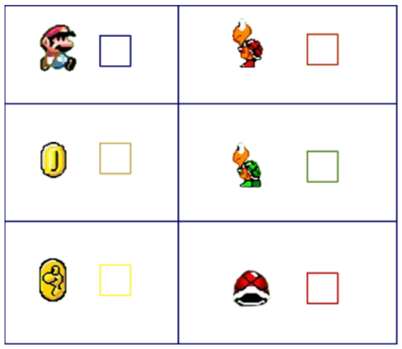


Figure 6: Referential colors of objects.

respective errors, which corresponds approximately 90.2% of accuracy.

B. Activity Identification

After having the position of each agent or entity in the screen, it is possible to infer which actions were taken by the player. We consider the captured entities when the player is closer than a specified distance from another entity, with the exception of some entities that enable clamp, such as turtle shells in case of SMW.

If the player's position is higher than agent, then it is considered that he/she is stepping in the agent. If the height of the player is approximately equal to the object, then it is considered that he/she caught the agent. Furthermore, if the height of the player is smaller than the object, it is considered that he touched in the agent. The possible actions that the main character can realize in relation to some other agents and entities are illustrated in TABLE 2.

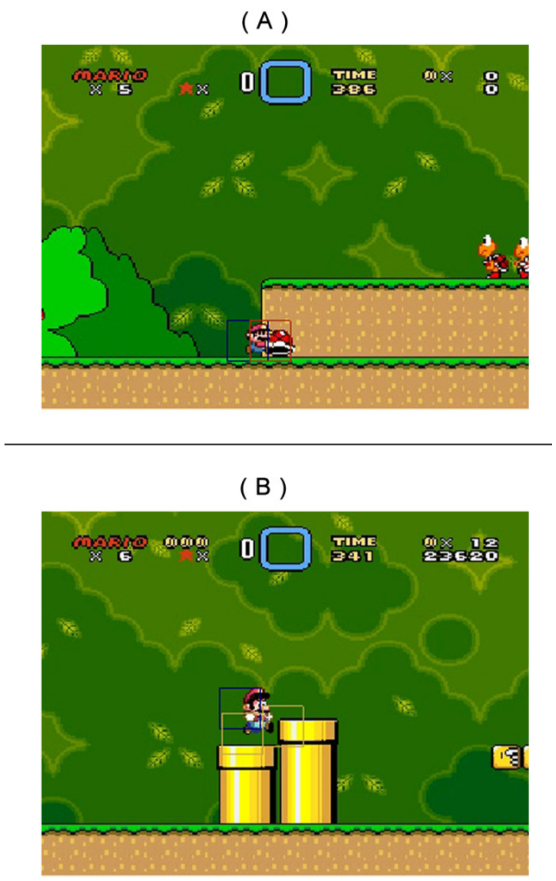


Figure 8: Object capturing process.

TABLE 1: SAMPLE OBJECTS CAPTURED COMPARED WITH WHAT SHOULD BE CAPTURED.

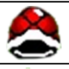
	Expected	Captured	Error
	420	432	12
	16	6	10
	4	4	0
	4	2	2
	16	8	8
	8	8	0
	50	68	18
	6	4	2
Total	**524**	**532**	**52**

TABLE 2: POSSIBLE ACTIONS RELATED TO SOME AGENTS AND ENTITIES.

Catch	Touch	Step
	Touch	

C. Storing Provenance Data

After capturing and analyzing the elements, the data is organized and stored in the provenance repository that follows the provenance schema represented in the class diagram showed in Figure 9. The "*agent*" table contains the information from the game agents for further analysis. Example of information includes the name, attributes, goals, and location of each agent. For example, the *"Red Koopa"* has the hull as an attribute and she walks on the ground.

The "*entity*" table is filled by the game designer and contains the name, type, importance, location and attributes of each entity in the game. The attribute type is used to identify as possible help or hinder the main character. The importance field serves to represent how much this entity mean to the game designer, considering his analysis. Finally, the location represents where the entity is found in the game and can be used for playability analysis.

The table "*activitytype*" is also filled by the game designer and contains the possible actions types during a game, such as catch, touch, or step in another agent or entity. The table "*gamesession*" is filled after the processing and analysis of the data, saving information about time and phase of the game. The table "*activity*", stores the agent or entity involved in the action with the player, as well as the type of action. This table is also stored for each game session. However, a game session may have various activities. In other words, table

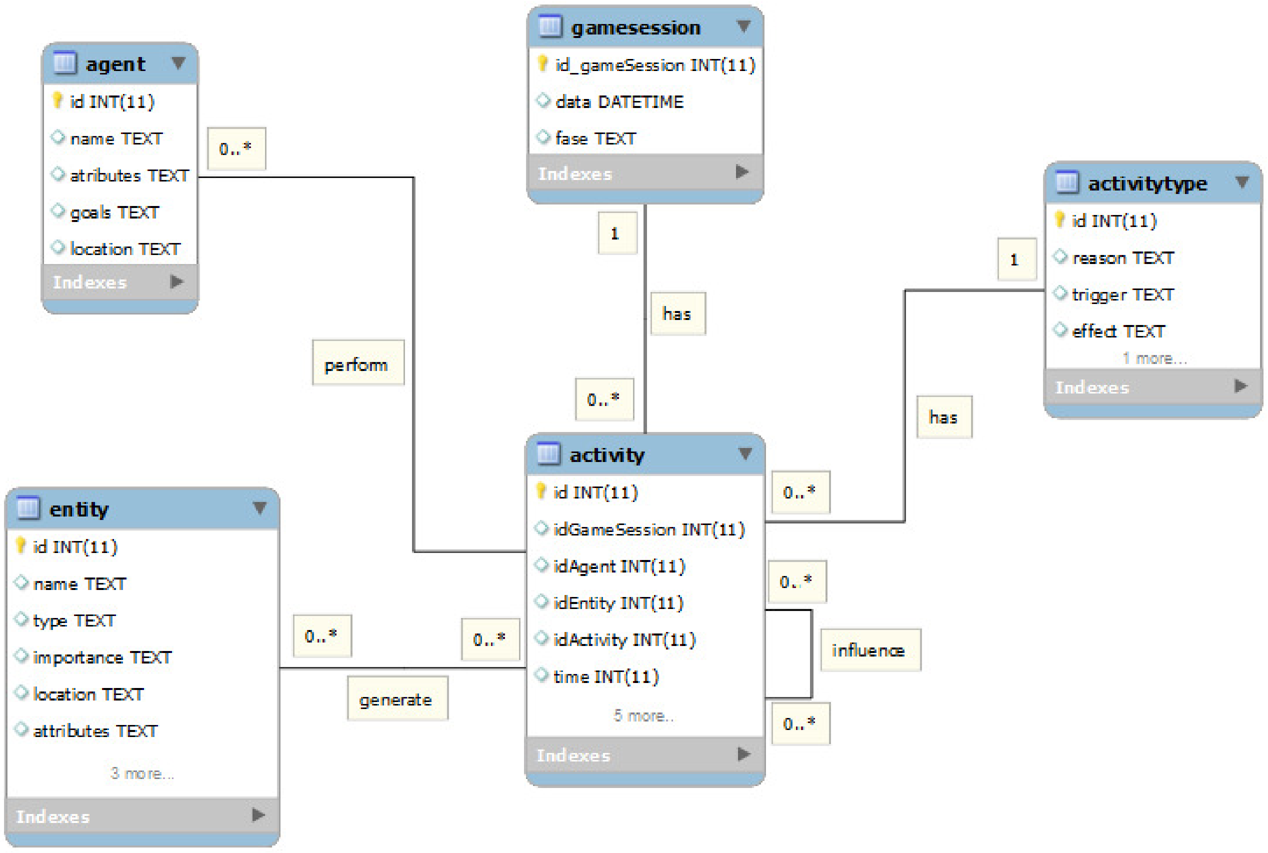


Figure 9: Class Diagram representing the provenance schema of the provenance repository.

"*activity*", stored all the information collected from the images analysis. This enables the reconstruction of sequence of actions that occurred during the game execution.

D. Data Representation

We used the tool *Prov Viewer* from [7] and made some changes to become possible to represent our collected data, allowing it to be easily compreensible for the game designers during their analysis process. The main change was related to the vertices structures, where we added importance, time and gamesession elements, which are required for our context.

This *input.xml* file is created by querying the provenancedatabase. The vertex tags represent each agent with their proper information. We created in the same manner a tag for each activity and entity. In the same *input.xml* file we added an edge tag with *Neutral* type in order to associate all actions with the next action. The edge tags contain the element *sourceid,* it is the current action and the *targetid* containing the next action. We then generated a graph with edges organizing actions in chronological order. Assuming that all actions are made by the main character, we generate a tag for every action with *targetid* containing the entity or agent which relates to the main character. We can see this in sections of *input.xml* file as follows.

```
<vertex>
  <id>ac38</id>
  <type>Activity</type>
  <label>Touch</label>
  <date>0</date>
  <importance>1</importance>
  <time>14</time>
  <gamesession>4</gamesession>
  <details/>
</vertex>

<edge>
  <id>e53</id>
  <type>Neutral</type>
  <label></label>
  <value>1</value>
```

```
 <time>66</time>
 <gamesession>4</gamesession>
 <sourceid>ac53</sourceid>
 <targetid>ac54</targetid>
</edge>
```

The Figure 10 illustrates the graph generated by *Prov Viewer* with our modifications. Purple squares represents activities performed by the main character. The agents are represented by an orange pentagon. Entities are represented by a yellow circle. The edges represents the relationship between two vertices. The green edge represents a positive action bringing benefits to the character with its proper associated importance and the red edge is a bad action that brings prejudice to the character.

It can be observed that the edges have different thickness according to its value. The beige frames are the detailed information of each vertex or edge. This frame is a tooltip that is shown when we pass the mouse cursor over the vertices or edges. We made this motage in Figure 10 showing various frames at the same time to be represented in a single figure.

With this type of graph and the information that brings to the game designer, we can see all actions performed by the main character in chronological order. We can identify possible problems in the game session. For example, the repetition of actions. It happens probably, because the

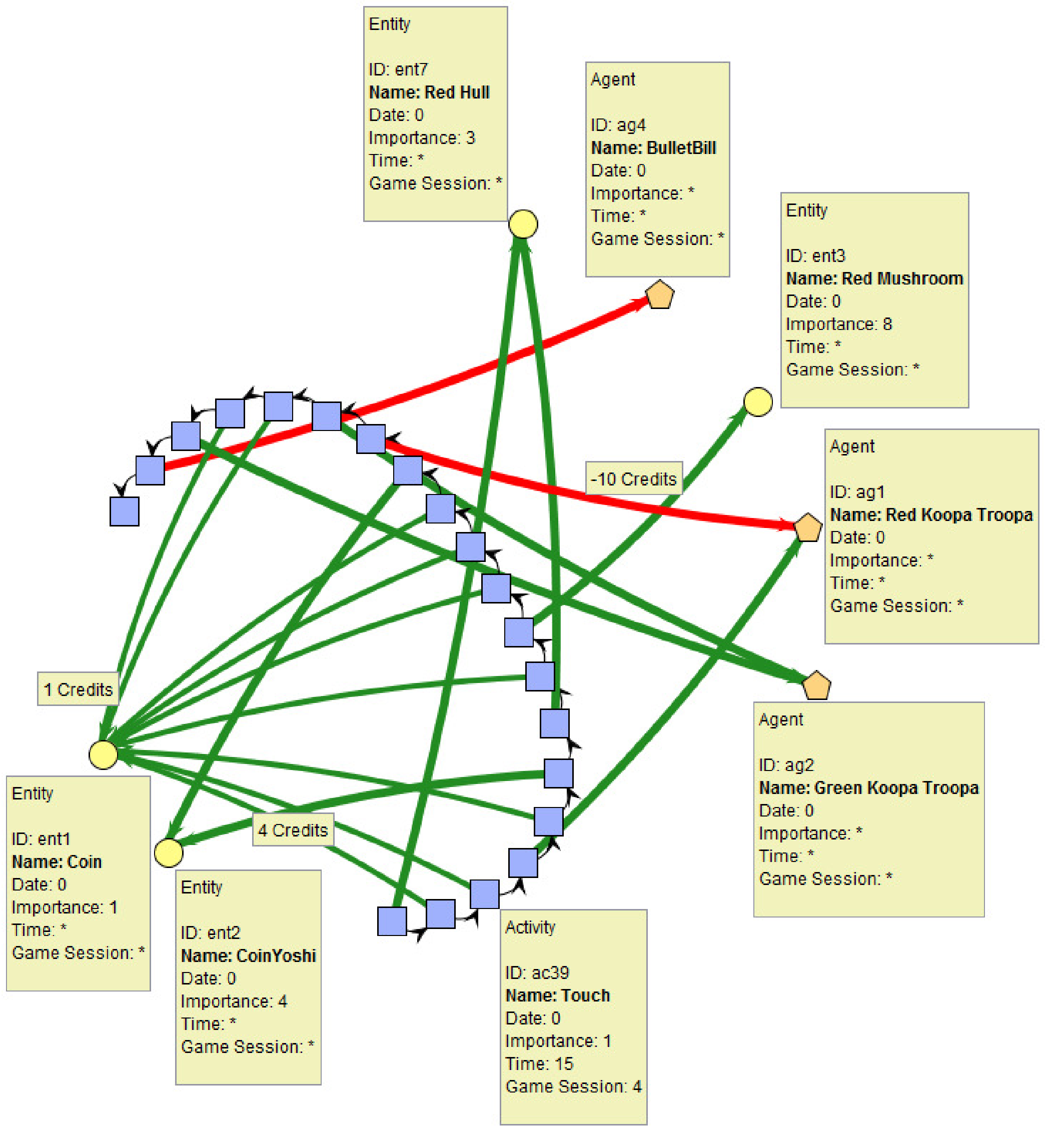


Figure 10: Representation of the provenance graph using the modified Prov Viewer.

associated action is very easy or very difficult to be performed. In Figure 10 we can observe a large number of association of *activitytype Touch* with the *entity Coin* because it is an easy and ordinary task and very common in the game session. Similarly, this hard tasks in the game session would be identified as well, since the user would repeat the same actions several times. However, the associate will be negative and have a red color.

V. CONCLUSION

This paper presents a novel provenance based approach to help the game designer by allowing an analysis of their games. The most important feature of this work is the possibility of analyze gameplay of 2D platform games without depending upon the game's source code. In other words, our approach is non-intrusive and can be applied in various existing games and does not requires the source code in order to generate the game log. By gathering the provenance data using simple image processing techniques and later displaying it in the form of provenance graph, the game designer can better understand the game flow.

We plan to enhance some of the existing features as future work, especially when treating the false positives generated by objects with similar histogram. In order to solve this, it is possible to include other image processing techniques, such as image segmentation. In order to accelerate the image analysis, we may implement all the image analysis at the GPU level.

Although we focused this work for 2D platform games, it is possible to increase the technique for other 2D styles of games. This is a trivial work, being only necessary to formalize the corresponded activities and behaviors. Nonetheless, this paper presents interesting results allowing the visualization and possible actions interpretations in the game flow through data provenance collected from image sequence. This technique provides a basis for future work beyond its use in different type of games and can be studied in other interactive applications outside the game environment.

ACKNOWLEDGEMENTS

We would like to thank CNPq, CAPES, FAPERJ, and FAPEMIG for partially support this work.

REFERENCES

[1] Cruz, S. M. S., Campos, M. L. M., Mattoso, M. 2009. Towards a Taxonomy of Provenance in Scientific Workflow Management Systems. *Los Angeles, CA, Congress on Services - I*, pp.259-266.

[2] Drachen, A., 2012. *Game Analytics*. Available at: http://blog.gameanalytics.com/blog/announcing-game-analytics-maximizing-the-value-of-player-dat.html [Accessed April 22, 2013].

[3] Filho, O. M., & Neto, H. V., 1999. Processamento Digital de Imagens. *Brasport*. pp. 55.

[4] Freire, J., Koop, D., Santos, E., & Silva, C. T., 2008. Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering* , pp. 11-21.

[5] Gartner, Inc., 2013. Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013. Available at: http://www.gartner.com/newsroom/id/2614915 [Accessed July 22, 2014].

[6] Jacob, L. B., *et al.*, 2013. A game design analytic system based on data provenance. *Entertainment Computing – ICEC,* pp.114-119 .

[7] Kohwalter, T. C., 2013. Provenance in Games. Dissertação (Mestrado em Computação) Niterói, RJ, Universidade Federal Fluminense.

[8] Kohwalter, T. C., Clua, E. W., and Murta, L. G., 2013. Game Flux Analysis with Provenance. *Advances in Computer Entertainment* , pp. 320-331.

[9] Kohwalter, T. C., Clua, E. W., and Murta, L. G., 2012. Provenance in Games. *XI SBGames* . pp. 162-171

[10] Moreau, L.*et al.*, 2007a. The open provenance model(v1.00). *Technical report, University of Southampton* .

[11] Moreau, L., *et al.*, 2007b. The Provenance of electronic data. *Communications of the ACM* 51.4 , pp. 52-58.

[12] Nintendo 2014. Super Mario World. Available at: http://www.nintendo.com/games/detail/OnTm1QccFa_Ht39i-dKiI-f8WRu2Cje [Accessed July 22, 2014].

[13] White, W., *et al.* 2009. Better scripts, better games. *Communications of the ACM*, v. 52(3), pp. 42-47.