SeekWhence
A Retrospective Analysis Tool for General Game Design

Tiago Machado
New York University
tiago.machado@nyu.edu

Andy Nealen
New York University
nealen@nyu.edu

Julian Togelius
New York University
julian.togelius@nyu.edu

ABSTRACT

This paper describes the design of SeekWhence, a retrospective analysis tool for gameplay session. SeekWhence is a new addition to the Cicero AI-assisted game design tool, which is built on top of the Video Game Description Language (VGDL) and the General Video Game Framework (GVG-AI). With SeekWhence, designers can prototype their games and record gameplay sessions simulated by agents or human players. They can go back and forth on every frame of the recorded session, analyzing it step by step and import it into their current project to edit it. This paper explains the technical details of SeekWhence and gives examples of its usage.

CCS CONCEPTS

•**Computing methodologies** → *Artificial intelligence; Search methodologies;*

KEYWORDS

Gameplay Sessions, Game Analysis and Visualizations, Game Prototyping, Retrospective Analysis

ACM Reference format:
Tiago Machado, Andy Nealen, and Julian Togelius. 2017. SeekWhence
A Retrospective Analysis Tool for General Game Design. In *Proceedings of FDG'17, Hyannis, MA, USA, August 14-17, 2017,* 6 pages.
DOI: 10.1145/3102071.3102090

1 INTRODUCTION

AI-driven design assistance tools have now been developed for many creative tasks. Especially in the cultural and creative industries, the number of tools which providing assistant components to help aspirant and professionals users are available for a wide range of activities. From text writing [17] to digital filmmaking [3], there are plenty of research and commercial tools.

Many such tools have been developed for games as well. Examples include Unity Analytics [24] and Bioware's Skynet [28]. Both called market and research attention by performing telemetry and analysis on data gathered from players across the world. With this information, game developers can work with accuracy to keep and expand their user basis, besides apply their findings to improve the games according to the desires of their target audience. Besides that, these tools also can offer assistance in tasks like game balancing, bug tracking, usability, profiling and log analysis as well [6, 14].

Many of these tasks are related to advanced stages within a game development process. However, we already have been seeing tools designed to offer assistance in early stages. For example, Tanagra [23] and Ropossom [19] are tools whose main focus is the creation and evaluation of game levels. Independent of their final purposes, many of these systems are straightly attached to just one game or a game genre in the best hypothesis.

Cicero is an AI-assistant tool developed to address this lack of generality. It allows designers to prototype 2D games across different genres like action, shooters and puzzles that users can simulate by themselves or by using intelligent agents. Previous evaluations of Cicero revealed that the users were in need of a tool to offer them the possibility of going back and forth in a gameplay sequence and highlight it with the Cicero's visualization system to help them in evaluating their players (human or agents) and find system failures.

SeekWhence was designed to attend these requirements, it is a retrospective analysis tool enriched by a visualization system with the power to replay games played by humans or artificial agents. It does telemetry whenever a game is playing. Every frame of the gameplay session is stored and becomes available to the users in a sequential media file. The users can play it and analyze all the gameplay session frame by frame. They also can activate a visualization system to further help with the analysis task. Finally, they can import any frame of the sequence into their current project to edit it with all the information about the game and its components already available, from the beginning until that point (the frame which was chosen to be imported).

In this paper, we report our experiences regarding the SeekWhence development and the benefits that game developers can have with such a retrospective analysis tool. We discuss our approach, the background which motivated this work and our telemetry process. Finally, use cases and examples which illustrate the uses of the tool are presented as well.

2 BACKGROUND

Despite the significant contribution to the game development process in general and its large adoption by companies and research institutions in the recent years, we can observe that game telemetry and the analytics tools based on it, especially visualization, is still in its infancy [1]. Nevertheless, developers can have a more accurate analysis of the (in-game) behavior of players who play their products [2, 10], they can track a myriad of events (deaths, enemies attacked, items used, etc.) and even know when a player is about to abandon their games [13].

As games tend to diversify from one to another, most of the tools are created within a specific project, for instance, *Data Cracker* is a tool designed to gather and analyze data from one of the titles in the *Dead Space* franchise [14]. *Cure Runners*, a 2D runner game also had a telemetry and visualization system implemented just for it. In [26], the authors used the game as a case study about how to integrate game analytics tools into the development cycle. Finally, the game *Super Mario Bros 3* was used by [22] to analyze gameplay through input controllers. The same is true for AI-game design assisted tools, where the systems, usually, are also developed within a specific game project. A notable example is Tanagra [23], a tool that provides assistance in designing 2D platform games. Its features include real-time editing and multiple partial levels generations with a guarantee of playability.

Ropossum [19] is an application that shares some levels of similarities with Tanagra. However, it is entirely based on the game *Cut The Rope*. It helps users in creating and evaluating their own levels with features such as real-time feedback level solutions. Similarly, Smith et al. [21], generates solvable levels, however, according to the player progression in the educational puzzle game *Refraction*.

Slightly more general, we have Sentient Sketchbook [9]. Its assistance is also focusing on generating game levels, however across the genres of strategy or roguelike. It provides a real-time suggestion system that shows up level's recommendations at the time that users are creating their own.

Finally, StreamLevels [7] provides the tools to create the shape of 2D platform game levels. However, the main idea is to have a general level editor. The user interacts by drawing strokes on the UI canvas, the system uses this input as a player trace and generates the shapes. The idea is that the users can adapt the shape in their future game project.

As mentioned earlier [11, 12], Cicero is an AI-assisted tool developed with the intention of reducing this lack of generality. It provides a user interface to prototype games as well as a myriad of AI agents to play them. By using its agents, Cicero is capable of gather in-game data.

In order to analyze these data, SeekWhence allows the designers to inspect their games step-by-step, they can interpret the data as a whole sequence of events or any event in the sequence separately. Besides that, they can import any frame of a sequence into their current project, edit, test and play it as if it was designed at that single moment. The development of SeekWhence was inspired by a previous evaluation of the Cicero system. When testing the application, several users suggested that it would be helpful in having a tool to help them in seeing what was happening in the game at every time-step. Almost like a slow-motion retrospective analysis.

Retrospective analysis was previously used by [25] as a player-centric visualization work geared to evaluate types of visualizations in the team-based combat game *World of Tanks* [27]. It was also presented in the serious and open-ended game *RumbleBlocks* [8] where the authors used the technique to solve alignment issues.

Following the terminology defined by [1], whereas the work of Wallner and Kriglstein [25] is a retrospective analysis tool designed with the purpose of training players and the work of [8] is designed to align game design with educational outcomes our work is a retrospective analysis tool designed to help developers in tasks such as debugging and balancing. Therefore, its purpose, target audience, its usage across different game genres, the use of agents to simulate the games and the visualizations to enrich the navigation are the main contributions of SeekWhence.

3 PREVIOUS WORK ON CICERO

Cicero is an acronym for Computationally Intelligent Collaborative EnviROnment for game and level design. It is built on top of the GVGAI Framework and allows designers to prototype games by using the VGDL language. As SeekWhence uses many resources of Cicero, we will do an explanation of it and its features to better explain later how SeekWhence was developed.

3.1 VGDL and GVGAI

The Video Game Description Language is a language which allows the development of 2D games like some classical ones available for the Atari 2600 and the Nintendo Entertainment System [18]. A game definition consists of two text files, one to describe the game rules and another one to design the game level (See Figure 1); a game can have any number of levels. The game description file consists of four different sections: SpriteSet, LevelMapping, InteractionSet and TerminationSet.

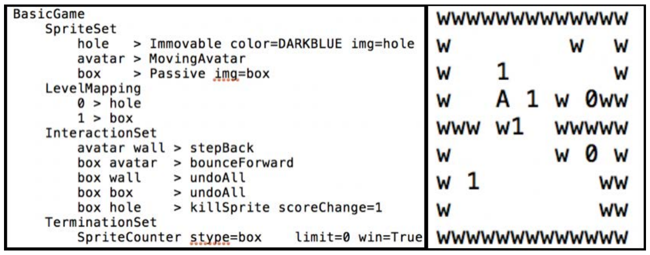
SpriteSet Defines the behaviors of actors (avatar, enemies and immovable objects). A sprite in VGDL is not just an image, it carries information about how the image should behave according to the game rules.

LevelMapping Creates a symbol for each sprite in the SpriteSet. The symbols are used to design the game levels.

InteractionSet Defines what happens with the sprites when they colliding each other.

TerminationSet Defines the conditions which lead the game to an end (Win or Lose).

The General Video Game AI [15] framework is designed to allow AI developers to create general agents to play VGDL games without any previous knowledge about them. Every year the GVG-AI competition pushes developer skills in creating new agents for the framework. The agents can be based on several techniques and algorithms such as tree search and evolutionary ones. The framework is entirely developed in Java and uses only Java native libraries.



```
BasicGame
    SpriteSet
        hole   > Immovable color=DARKBLUE img=hole
        avatar > MovingAvatar
        box    > Passive img=box
    LevelMapping
        0 > hole
        1 > box
    InteractionSet
        avatar wall > stepBack
        box avatar  > bounceForward
        box wall    > undoAll
        box box     > undoAll
        box hole    > killSprite scoreChange=1
    TerminationSet
        SpriteCounter stype=box    limit=0 win=True
```

```
wwwwwwwwwwwww
w        w  w
w   1       w
w   A 1 w 0ww
www w1  wwwww
w       w 0 w
w 1        ww
w          ww
wwwwwwwwwwwww
```

Figure 1: Sokoban game written in VGDL. Game rules (left) and game level (right).

3.2 Cicero Features

Cicero has a user interface in which the users can edit game rules and levels without any previous knowledge about VGDL. For every action of the user, underneath, the system generates lines of VGDL code which fills the description and level files explained in the previous section. When the users are done with the game definitions they can choose one of the available agents to play their games. At this point, the system combines the description and level files to generate a game to be played by the agent previously selected by the user. Besides the tasks of editing and playing VGDL games, Cicero offers three other features: a stats tool, a mechanics recommender and a visualization system.

3.2.1 Stats Tool. The stats tool tracks every interaction in a game. An interaction in VGDL is part of the core mechanics of each game. When running a prototype, the user can have access to a real-time report that shows a diagnosis about what interactions are in use, from the most to the least accessed or if there are rules not accessed at all. This way, the user can take the information of this diagnostic to design test cases and decide if the interaction continues relevant.

3.2.2 Mechanics Recommender. The Mechanics Recommender is inspired by the situationalist school of creativity, as defined by Ben Scheneiderman [20]. It works whenever a user presses a button to get recommendations from the system. The system compares the game in development with the ones available in the VGDL library. Then it defines a rank of similarities and suggests sprites and interactions (the core elements which represents game mechanics in VGDL) [11].

3.2.3 Visualization System. The visualization system, as the name indicates, shows the behaviors of the game elements (agent, enemies, power-ups, portals, etc.). In real-time, the user can see what each element do during a gameplay session.

3.2.4 Evaluation of Cicero. In previous work, a user study with 10 participants was performed. Overall, Cicero's features were well accepted. Most of the users agreed with their usefulness and that they would like to see more prototype tools like the ones available during the study. However, they also mentioned that they would like to see some kind of sequential analysis. While they could see the data in real-time, they could not stop the game to analyze a particular event. They said that the stats and visualizations reports are interesting, but see when and where the events happen could be even more useful. These results were the main motivation behind the development of SeekWhence.

4 RETROSPECTIVE ANALYSIS AND VISUALIZATION IN CICERO

This section details the implementation of SeekWhence, with explanations about the GVGAI and Cicero's UI extensions.

4.1 GVGAI Extensions

GVGAI has two classes which are fundamental to this implementation: the **StateObservation** and **AbstractPlayer**. The former is responsible for keeping information about the game at every state, the latter is the class that every agent has to extend in order to execute their functions when playing. Every agent overrides a method called **act** from **AbstractPlayer**. This method receives a **StateObservation** as a parameter. It analyzes the current game state and as a result it takes one of the actions available to move to another state. The agent has one game tick to take a decision and every decision leads to a new state configuration.

This means that at every game tick, the sprite level matrix, contained into the **StateObservation** is stored. This matrix contains the position of every sprite in the level and the sprite unique ID, which we can use to access sprite information at that particular game tick.

Also, at every game tick, the position of every sprite is stored in order to do a summation and come out with the normalized alpha value for the color track at that position. It is storing the visualization information of each sprite for every game tick. This computation considers the range that goes from the the first to the current game tick. In other words, it is pre-caching the necessary information in order to provide to the user step-by-step visualization.

Then, when a gameplay session is ended by an agent winning or being defeated (or by a decision of the developer) every information tracked on the fly is then stored in different files.

4.2 SeekWhence File Generation

For each new game tick, a SeekWhence file is generating by combining the collections of level map matrices and the collections of stored color tracks (see Figure 2) of sprites and events. More specifically there are four files to know:

Map Matrices This file stores the position of every sprite in the level in the form of a 2D matrix. It is straight related to the **LevelMapping** section of the VGDL game description (See ssection 3.1) and each index of the matrix contains the necessary information to rebuild the level at the specified game tick.

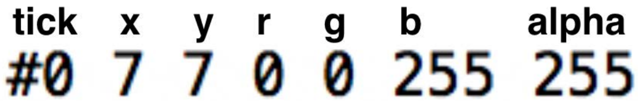
Avatar This file stores the color tracks of the positions on the level visited by the avatar (human player or AI agent).

Enemies Analogue to the last one. However focused on enemies[1].

Events Stores the color tracks of any given event. An event in a VGDL game is every interaction among two sprites described in the **InteractionSet** (See section 3.1). We have kill events, clone events, hit events, dropping resource events and so on.

The first step to assembly the SeekWhence file is to read the Map Matrices file and for every tick, we recreate, in a secondary file, the level as it was.

So, when having the level sequences all together and indexed by game ticks, we can access every frame from a stored game session step-by-step. Furthermore, based on the same tick we can print the visualization color tracks for game elements and events.



tick	x	y	r	g	b	alpha
#0	7	7	0	0	255	255

Figure 2: Every color track stores the game tick when it was registered in a gameplay session, its *x* and *y* position on the matrix and the its RGBA values.

4.3 Retrospective analysis - Cicero UI extension

In order to allow the users to analyze the sequence of events, we extended the Cicero's UI. We implemented a component similar to a video player (Figure 3). The user can interact with the component by clicking on the arrow buttons (left and right) to go back and forth in the sequence frame by frame. A slider bar also does this service allowing the user to skip quickly several frames. The component also has an export button, which allows the user to export any particular frame to the Cicero's main interface. An important highlight of SeekWhence is that it is not just a tool for *Replay Theater* in which the users can watch a replay of a game session and have a control over what they are watching. More formally, *Replay Theater* in general, works like a video where players use to see their own and their opponents progress. It is also in use by developers to catch bugs. However, in essence, it restricts viewers to watch a non-interactive animation [1]. SeekWhence enhances the *Replay Theater* concept by allowing a designer to actually use any given frame of any gameplay video sequence, edit it, and changing whatever game element they want, like the agents for example.

By editing, we mean the process of describing the level map. By importing a frame from SeekWhence, the user gets all the information of the level (and its elements) at that state of the game. By changing it, the user will change the sequence of game states, from the one being edited to last until the game finishes. Obviously, it can reach many of the former gameplay session states, but it is not guaranteed since it depends on the kind of changes the user did and the algorithm (or player) decisions when playing again.

[1]As other game elements are analogue, we decided to skip their explanations for the sake of space.

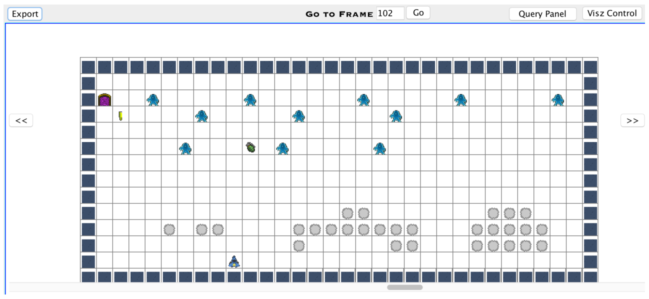


Figure 3: The SeekWhence UI contains buttons to allow a user to go back and forth in the stored gameplay session. It also contains a slide bar, a search for a frame specified by its tick number, button to turn the visualization on and the button to export the frame to the Cicero's main UI.

5 USE CASES

This section presents some use cases and examples where SeekWhence can be used: agent and level evaluation. We also present our findings based on an informal test conducted with three volunteers.

5.1 Agent evaluation

Designers can use SeekWhence to help they in analyzing their agent's implementation. In the normal way, to specify in detail the agent behavior, the user needs to use the available Java debugging tools included in the most popular IDEs among developers. However, these tools do not offer an easy way to navigate through all the data the users needs to have a clear and easy comprehension about what their agents are doing. There are no graphics or other visual clues that could help. Besides that, it is a responsibility of the users to write some parts of the code in order to access specific information. SeekWhence, in this case, can present to the users a visual interface to go back and forth through a recovered gameplay session. They will not only know where is the agent position, they will see where it is at any given game tick. More than that, they will also see where are all the other elements in the level and how they may (or may not) influence agents decisions. This is much more convenient than checking printed lines in a console showing positions and actions of game elements. Particularly, if the users are trying to do a hybrid agent, SeekWhence offers the option of changing the game and level configurations whenever they want. So they can start a sequence with a *AStar* agent, change it to a *Monte Carlo Tree Search* and finally use one of the champion agents of the GVGAI competition. Then, by running the recorded sequence, they analyze the agents individually and come out with their conclusions about how to improve their hybridization. This kind of advantage, import the stored frames and editing the level, makes SeekWhence more than just a *Replay Theater* application as mentioned before in the subsection 4.3. In the example illustrated in this section, the designer switches among a human player and *adrienctx* (a previous GVGAI champion).

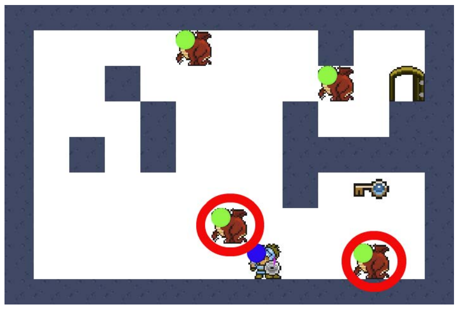


Figure 4: Human player playing a cave level. This is the last frame before he got killed by the enemies highlighted by the red circles.

In the Figure 4 we see a human player who is constantly killed by random enemies. The enemies are fast and they make hard his mission of getting the key. The designer got the last frame of this session played by the human (the frame before his avatar get killed by one of the enemies) and replaced him by the *adrienctx* agent. It was able to kill the monsters and get the key. However, it wastes too much time walking in circles and attacking empty spaces. At this point, the designer gives the control back to the human player to play again and complete the level.

5.2 Level evaluation

One use of SeekWhence is level editing and evaluation. Although there are many methods and heuristics involved [4], level evaluations are done by players following formal or informal procedures [5] in which they answer questions and provide feedback about what they think. Sometimes both the designer and the player needs to use their memories in order to remember all the events that happened during the session, what can be pretty inaccurate. So by providing a way of accessing these events, SeekWhence can help users and players to better express themselves and increase the accuracy of their communication [16]. Besides that, if the players have the same opinion about a similar event, it is difficult for the designer to access the exact point and rebuild the level to edit it.

SeekWhence allows the designers to have these kinds of access. They can go to the mentioned events, import the frames and all the information attached to, inspect them as many times they want to and perform the changes they find necessary.

We did small changes in a VGDL clone of the classical game *Space Invaders*. We decreased the enemies cooldown, what does the game pace increases, and some barriers cannot be destroyed by enemies' bombs. To evaluate the gameplay session, the designers needs to use SeekWhence, otherwise, by not see it step-by-step, they can lose important details due to the speed of the game.

In the figure 5 we have the last frame of the VGDL *Space Invaders* clone 5a. The red dots shows areas where the enemies were killed. Green dots shows the enemies‘ moves and the blue ones shows the player's moves.

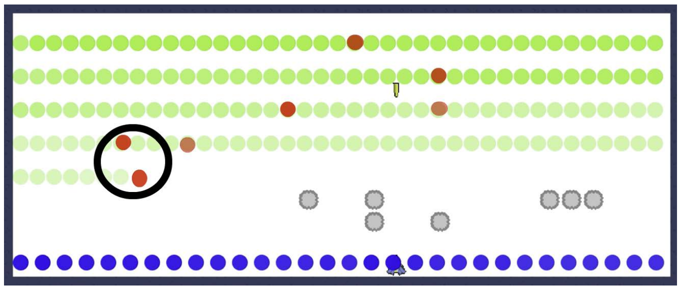
By running SeekWhence and moving the gameplay session backward the users can see that a barrier is killing enemies 5c. This fact is highlighted by the visualization system. In the figure 5a, the black left circle shows two killing spots, one of them is exactly the position of a barrier. Also, they can notice another flaw rule in the figure 5b. While in the left black circle they can see an enemy getting a shot from the player in the right black circle the enemies' bombs are not destroying the barriers.

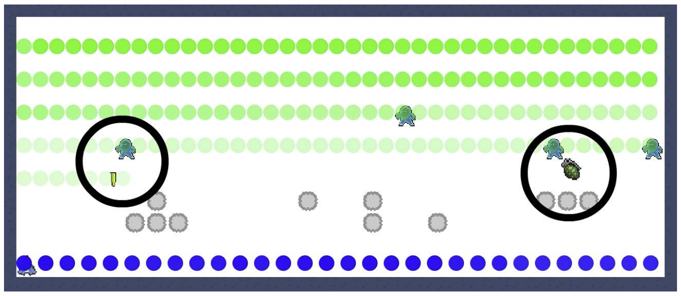
5.3 Informal User Study

We conducted an informal study in order to evaluate users‘ reactions related to SeekWhence. In this study, we were interested in seeing if our users would be able to find bugs in a gameplay session. Three users helped us with the test, all of them are Ph.D. students doing research in areas related to game design and artificial intelligence.

The setup for the study included making some modifications to a VGDL game called *Firestorms*. It is a puzzle game where the player must find the exit in a labyrinth crowded with portals, which cast fireballs in different directions.

We introduced two chaser enemies (so called because they chase the player) and two random enemies, which just move randomly


(a)


(b)

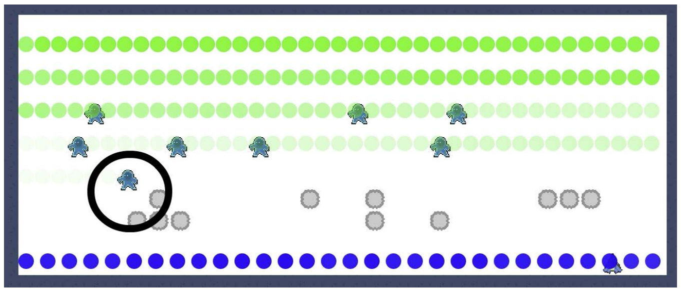

(c)

Figure 5: From (a) to (c) we can see frames of a Space Invaders gameplay session. The black circles highlights areas where the users found flaws in the game rules.

around the level and shoot. Then, we created some bugs, on purpose, like enemies going through walls or portals which donfit cast fireballs.

We explained the game to the users and our modifications (except for the bugs), followed by a short demonstration of an agent playing it. Then, we explained how SeekWhence works and asked them to spend some time trying the tool and reporting us if they could find something wrong with the *Firestorms* gameplay session.

At the beginning of the session, the users seemed confused because of the sheer amount of events happening in the game. However, they mastered the tool very quickly and could easily find the bugs. What surprised us is that they could even find bugs that were not listed for us (this was a finding that made us confident about this tool purpose). A portal which is not casting fireballs, enemy's bullets which go over the level top and bottom borders, fireballs which appear suddenly on the screen and enemies which go through walls were the most reported bugs. Overall, the users stated that they think it is a useful feature and the option of export and edit the level frames was very appreciated. One of the users even reported the frames where he found bugs in order to make easier the process of correcting them.

6 LIMITATIONS

SeekWhence is limited to the space of VGDL games. It is based on a discrete game state structure what makes it capture and reproduce in-game data in a very reliable way, however, it is not suitable for continuous spaces. Also, float-point events like game objects with an increasing speed tend to mislead users.

7 FUTURE WORK

We are currently working to addresses some limitations of SeekWhence. For example, if we have more than one sprite competing for the same game space, a single game tile, just one of them will be rendered. This was a limitation identified during our informal tests that confused our testers in the beginning of the system evaluation. We also are working on a query system in order to facilitate the process of knowing what, when, where and who were involved in game events (VGDL interactions). We believe that a system like that, combined with the navigation provided by SeekWhence, together with the visualizations, will enhance the analysis tasks involved in the game development process.

8 CONCLUSION

In this paper, we presented SeekWhence. It is a tool that allows users to do sequential analysis in a recorded gameplay session. SeekWhence is an extension of Cicero, an AI-assisted game design tool built on top of the GVGAI framework. As SeekWhence works along with GVGAI, it can store actions of sessions of a game played by an agent or a human player. The users can analyze a level step-by-step and use their conclusions, at least, in two situations: to build and debugging agents and to edit and evaluate levels. As users can go back and forth in all the sequence of events stored in a session, they can use every frame of that sequence and export it to Cicero. Then they can edit the level as they wish and with all the information until that point already available. We did informal tests and the first reactions were positive. We are also using our findings to improve the experience with SeekWhence and develop a query system to enhance the analysis tasks.

ACKNOWLEDGMENTS

Tiago Machado is supported by the Conselho Nacional de Desenvolvimento Cientfico e Tecnolgico (CNPQ), under the Science without Borders scholarship 202859/2015-0

REFERENCES

[1] Brian Bowman, Niklas Elmqvist, and TJ Jankun-Kelly. 2012. Toward visualization for games: Theory, design space, and patterns. *IEEE transactions on visualization and computer graphics* 18, 11 (2012), 1956–1968.

[2] Paul Coulton, Will Bamford, Keith Cheverst, and Omer Rashid. 2008. 3D Space-time visualization of player behaviour in pervasive location-based games. *International Journal of Computer Games Technology* 2008 (2008), 2.

[3] Nicholas Davis, Alexander Zook, Brian O'Neill, Brandon Headrick, Mark Riedl, Ashton Grosz, and Michael Nitsche. 2013. Creativity Support for Novice Digital Filmmaking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 651–660. DOI: http://dx.doi.org/10.1145/2470654.2470747

[4] Heather Desurvire, Martin Caplan, and Jozsef A Toth. 2004. Using heuristics to evaluate the playability of games. In *CHI'04 extended abstracts on Human factors in computing systems*. ACM, 1509–1512.

[5] Heather Desurvire and Charlotte Wiberg. 2009. Game usability heuristics (PLAY) for evaluating and designing better games: The next iteration. In *International Conference on Online Communities and Social Computing*. Springer, 557–566.

[6] Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa. 2013. *Game Analytics: Maximizing the Value of Player Data.* Springer Publishing Company, Incorporated.

[7] Lucas N Ferreira. 2015. StreamLevels: Using Visualization to Generate Platform Levels. (2015).

[8] Erik Harpstead, Christopher J MacLellan, Vincent Aleven, and Brad A Myers. 2015. Replay analysis in open-ended educational games. In *Serious games analytics*. Springer, 381–399.

[9] Antonios Liapis, Georgios N Yannakakis, and Julian Togelius. 2013. Sentient Sketchbook: Computer-aided game level authoring.. In *FDG*. 213–220.

[10] Yun-En Liu, Erik Andersen, Richard Snider, Seth Cooper, and Zoran Popović. 2011. Feature-based projections for effective playtrace analysis. In *Proceedings of the 6th international conference on foundations of digital games*. ACM, 69–76.

[11] Tiago Machado, Ivan Bravi, Zhu Wang, Andy Nealen, and Julian Togelius. 2016. Shopping for Game Mechanics. (2016).

[12] Tiago Machado, Andy Nealen, and Julian Togelius. 2017. Computationally Intelligent Collaborative EnviROnment for game and level design. (2017).

[13] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N Yannakakis. 2010. Predicting player behavior in tomb raider: Underworld. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. IEEE, 178–185.

[14] Ben Medler and others. 2009. Generations of game analytics, achievements and high scores. *Eludamos. Journal for Computer Game Culture* 3, 2 (2009), 177–194.

[15] Diego Perez, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon Lucas, Adrien Couëtoux, Jeyull Lee, Chong-U Lim, and Tommy Thompson. 2015. The 2014 general video game playing competition. (2015).

[16] Eric D Ragan and John R Goodall. 2014. Evaluation methodology for comparing memory and communication of analytic processes in visual analytics. In *Proceedings of the Fifth Workshop on Beyond Time and Errors: Novel Evaluation Methods for Visualization*. ACM, 27–34.

[17] Shubhangi Salinkar and Anirudha Joshi. 2015. Jodo: A Tool for Foreigners to Build and Speak Hindi Sentences. In *Proceedings of the 7th International Conference on HCI, IndiaHCI 2015 (IndiaHCI'15)*. ACM, New York, NY, USA, 137–144. DOI: http://dx.doi.org/10.1145/2835966.2836285

[18] Tom Schaul. 2013. A video game description language for model-based or interactive learning. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. IEEE, 1–8.

[19] Noor Shaker, Mohammad Shaker, and Julian Togelius. 2013. Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels.. In *AIIDE*.

[20] Ben Shneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50, 12 (2007), 20–32.

[21] Adam M Smith, Erik Andersen, Michael Mateas, and Zoran Popović. 2012. A case study of expressively constrainable level design automation tools for a puzzle game. In *Proceedings of the International Conference on the Foundations of Digital Games*. ACM, 156–163.

[22] Brian A. Smith and Shree K. Nayar. 2016. Mining Controller Inputs to Understand Gameplay. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 157–168. DOI: http://dx.doi.org/10.1145/2984511.2984543

[23] Gillian Smith, Jim Whitehead, and Michael Mateas. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*. ACM, 209–216.

[24] Unity. 2017 (accessed February 3, 2017). *Unity*. https://unity3d.com

[25] Guenter Wallner and Simone Kriglstein. 2016. 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 (CHI PLAY '16)*. ACM, New York, NY, USA, 22–32. DOI: http://dx.doi.org/10.1145/2967934.2968093

[26] Günter Wallner, Simone Kriglstein, Florian Gnadlinger, Michael Heiml, and Jochen Kranzer. 2014. Game User Telemetry in Practice: A Case Study. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology (ACE '14)*. ACM, New York, NY, USA, Article 45, 4 pages. DOI: http://dx.doi.org/10.1145/2663806.2663859

[27] Wargaming.net. 2017 (accessed February 3, 2017). *World of Tanks*. http://worldoftanks.com/

[28] Georg Zoeller. 2010. Development telemetry in video games projects. In *Game developers conference*.