Kwiri - What, When, Where and Who: Everything you ever wanted to know about your game but didn't know how to ask

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

Daniel Gopstein
New York University
dgopstein@nyu.edu

Andy Nealen
University of Southern California
anealen@cinema.usc.edu

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

Abstract

We designed Kwiri, a query system, as a support tool for an AI Game Design Assistant. With our tool we allow users to query for game events in terms of what (was the event), when (did it happen), where (did it happen), and who (was involved). With such a tool, we explore the possibilities of applying a query system to provide game-general AI-based design assistance. Throughout this paper, we will discuss the motivation, the design of Kwiri, use cases, and a preliminary qualitative study. Our first results shows that Kwiri has the potential to help designers in game debugging tasks, and has been served as infrastructure to build another system which relies on querying for game events.

AI-assisted design systems promise to help humans perform design tasks faster, better and/or more creatively. Within the field of game design there have been several prototype systems that showcase how artificial intelligence can be used in a specific role to assist a human game designer (Smith, Whitehead, and Mateas 2010; Liapis, Yannakakis, and Togelius 2013; Shaker, Shaker, and Togelius 2013; Butler et al. 2013; Smith et al. 2012). For example, AI can be used to evaluate game content that the human designs, creating new content to human specifications, and/or suggesting changes to content that is being created.

Most existing prototypes focus on a single type of AI-assisted design for content (typically levels) in a single game. In contrast, *Cicero* is a tool that aims to be a *general* AI-assisted game design tool, that is, providing multiple types of AI-based design assistance not just for a single game, but for any game that can be represented in the language used by the tool (Machado, Nealen, and Togelius 2017a). It includes features such as a recommendation engine for game mechanics, AI-based automatic playtesting, and editable game replays.

On top of Cicero, we designed Kwiri. A query system motivated by the idea that a designer (or tester) of a game will often want to figure out when and where something happens, and this might not be evident when either playing a game or watching a replay. For example, imagine that a particular NPC (Non-Player Character) occasionally dies even when one of the player's bullets do not hit it. To find out what's going on, a designer would need to rewatch endless replays attentively. However, what if you could simply ask the game when and where an NPC died? Kwiri makes use of the fact that the games that Cicero work have formally defined game mechanics, and provides the ability to interrogate replays for particular combinations of events.

Kwiri contribution relies on its generality. It is a single query system, whose interface allows designers to ask questions about any genre of game they can design with the platform. It also makes use of gameplay simulations and does not depend on human users to generate in-data game to be analyzed.

Background

This section discusses the primary references and inspirations related to the development of our system.

AI-assisted game design tools

Tanagra (Smith, Whitehead, and Mateas 2010) is a tool that assists humans in the task of designing levels for 2D platform games. The system works in real time. It creates many different possibilities for a level and provides a guarantee that they are playable. Therefore, there is no need to playtest the level to verify inconsistencies.

Similar to Tanagra, Ropossum (Shaker, Shaker, and Togelius 2013), also generates and solves levels for the popular physics puzzle game Cut The Rope. The user is assisted in the tasks of level design and evaluation. The tool is optimized to allow real-time feedback from a given state, after users inputs. It generates the next possible actions of the player until it finds a solution, if available.

Sentient Sketchbook (Liapis, Yannakakis, and Togelius 2013) is also a tool to assist the creation of game levels, but offers more generality than the two previous tools in this section since it provides assistance for strategy and roguelike games. The system shows level suggestions in real-time. It allows users to interact by editing their levels while generating recommendations based on choices made previously by users.

Also in the field of level generation, we have the work of Smith et al. (Smith et al. 2012) and Butler et al. (Butler et al. 2013). They present independent implementations of three diverse level design automation tools in the game *Refraction*. They use Procedural Content Generation (PCG) techniques and Answer Set Programming (ASP) to explore the intended design space and offer levels with playability guarantee.

All of these works presented significant results in the realm of AI-assisted game design tools. However, they are attached to a single game or a game genre in the best case. They have a lack of generality because their techniques need to be reimplemented every time someone starts a new game project. Their main focus is not querying for game events, but the way they provide assistance based on AI, even limited to one game, is an inspiration for our system's design.

Game Visualization systems

Game visualization is a topic which is gaining more attention every day (El-Nasr, Drachen, and Canossa 2013). Major game companies like Unity and Bioware have released their solutions with specific features to work with visualization analysis, Unity Analytics (Uni 2017) and Skynet (Zoeller 2010), respectively. In the academic field, many projects have been developed using different visualization techniques.

One of the games of the Dead Space franchise uses the tool *Data Cracker* to collect, analyze and summarize data about player performance in a visual way (Medler and others 2009).

The game *Cure Runners* has a visualization system used to track the player and assist designers on level balance tasks. This work is a case study in the integration of an analytics tool into a commercial game (Wallner et al. 2014).

GPlay (Canossa, Nguyen, and El-Nasr 2016) presents visualization and event queries on a spatial-temporal interface. The UI allows users to select the game elements whose behaviors they want to track.

As well as the AI-assisted tools, most of these works are directly related to a single game. The visualization package of Unity Analytics is a more general approach, but it does not have agents to play the games and generate gameplay simulations.

Kwiri takes influence from these systems and applies visualization as a way to enhance the answers to the designer's question (What, When, Where, and Who).

Declarative game engines and query systems for games

Declarative game engines go beyond the common idea of having databases only as a data storage method. The work of White et al (White et al. 2008) is an example. It develops the concept of *state-effect* patterns, a technique that let game designers develop parts of their games declaratively.

Deutch et al. (Deutch et al. 2012) developed a framework, based on SQL, to perform data sourcing in games. It extends SQL commands to allow recursive rule invocations and probabilistic operations. It has a runtime monitor that watches the game execution and give notifications to the user of properties violations. These properties formally specify expected behaviors of the game.

A more traditional use of databases can be seen on (Srisuphab et al. 2012). Here the authors store gameplay sessions of Go matches in a database. The goal is to use the stored matches to train novice human players through a GUI (Graphical User Interface).

Finally, Varvaressos et al. (Varvaressos et al. 2014) detail the process of implementing a bug tracker in six different games. The infrastructure is concentrated on the game main "loop". The authors implemented specific code that captures events of the games. The data is stored in an XML file. The process of finding bugs is based on properties about the expected behavior of the game being expressed in a formal language. During runtime, a monitor observes the game and notifies the user when some property is violated.

GVGAI and VGDL

The General Video Game AI framework (GVGAI) was designed for serving as a testbed for general video game playing research (Perez et al. 2015). A competition, based on the framework, runs annually and allow competitors to submit their agents. These are then judged based on how well they can play a set of unseen games. The Video Game Description Language (Schaul 2013; Ebner et al. 2013) (VGDL) is the language used to describe games in this framework; the language is compact and human-readable. Despite its simplicity, it is capable of expressing a large range of 2D games, like clones of classical games developed for the Atari 2600 and the Nintendo Entertainment System (NES). The GVGAI competition has now, about 100 VGDL games available and several dozen effective AI agents, with different strengths on a variety set of games (Bontrager et al. 2016).

The use of a simple, and analytically tractable game description language, as well as a collection of AI agents, gives GVGAI important benefits related to game editors and engines such as *Unity*, *Unreal* or *GameMaker*. The AI agents allow us to perform automatic gameplay and game testing, what is still not possible in the cited engines because of the lack of uniformity in how their games are specified. They are, of course, versatile and powerful, but not flexible about the use of general AI agents in the way we need for collecting data and implement a general query system.

Cicero

Cicero is a *general* AI-assisted game design tool based on the GVGAI framework and the VGDL. As stated before, existing tools are mostly limited to a single game and a single mode of design assistance. As the AI-assisted game design tool is still an emerging paradigm (Lucas et al. 2012) it is sometimes difficult to know what features and interactions should be included in an individual tool. Because of this problem, we adopted an Interaction Design approach to developing our tool. In the first iteration, we developed the general features for creating and editing games. Accompanying that, the system also included three additional features: a statistics tool, a visualization system, and a mechanics recommender. After a first evaluation, it was not enough for the users to perform an accurate analysis of the data collected by the agents (humans or AI algorithms) during a gameplay session. These data were pretty quantitative, and the users said that they would want information about when and where some events happened when an agent was playing a game. This motivated us to develop SeekWhence(Machado, Nealen, and Togelius 2017b), a retrospective analysis tool. SeekWhence allows users to replay a stored gameplay session frame-by-frame as if it were a video. Informal evaluations showed us that SeekWhence was well accepted. However, it required time and focused attention from the users because a single frame can contain multiple events. Therefore, if many sprites are in the same space, even by playing a game step-by-step, it is hard to identify which is doing what. To solve this problem, we developed Kwiri, which allows users to make queries about what, where, and when a specific event happened and which sprites were involved with it. The query system is integrated with SeekWhence and the visualization system.

Design of Kwiri

As stated before, the query system is developed on top of Cicero, and it integrates SeekWhence and the visualization system. In this section, we will highlight some of the implementation details about these two systems in order to facilitate further discussion about how the query system works.

How SeekWhence and the visualization system work

SeekWhence stores every frame of a gameplay session played by an agent. In order to store the frames, we capture every game state, at every game tick. The game state contains all the information of the set of game elements available, such as their positions in the level for example. The indexation by a game tick is what makes SeekWhence runs like a video player.

Every game element can be assigned to a specific color by the designer. It activates the visualization system [Figure 1], which captures all the positions of the elements in the level and apply a heatmap to show which areas were explored more.

Kwiri Implementation

Our query system is implemented on top of everything we previously discussed. It adds a database (***MySQL***). The database stores events and performs searches related to them. An event in VGDL happens whenever two sprites collide, it is up to the designer to define what kind of event will be fired at the moment of the collision. It can be a *killSprite*, a *cloneSprite*, and more than 20 other ones. In other words, the events are the rules of the games. The use of a database makes the implementation of the storage and the searching process easier and straightforward. The queries responsible for searching answers for the questions What, Where, When and Who are provided to the users as a GUI so that they can insert the query's parameters [Figure 2].

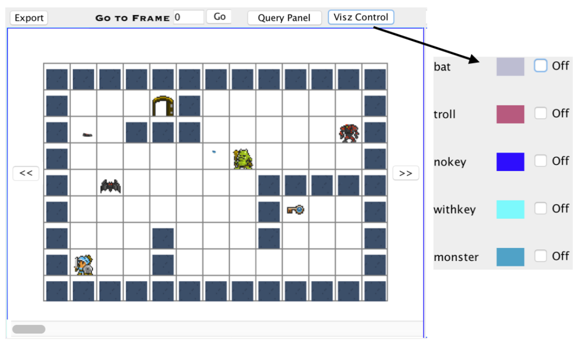
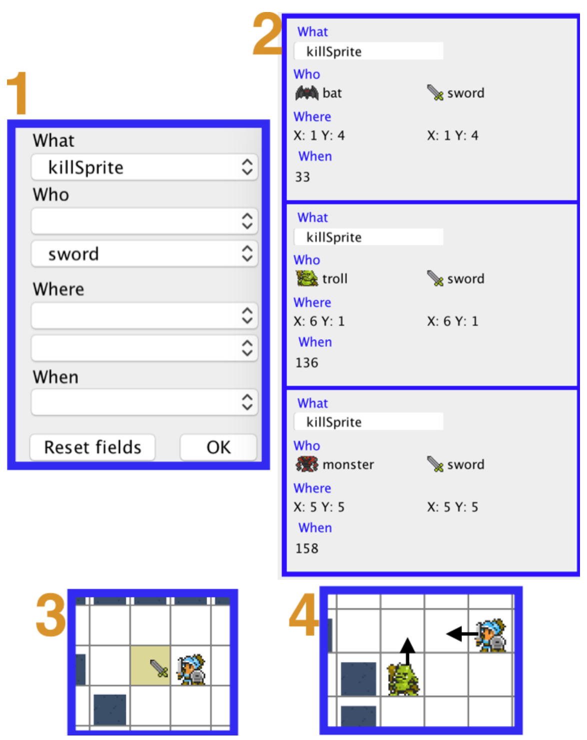


Figure 1: SeekWhence panel and its visualization control



What
killSprite
Who

sword
Where

When

Reset fields
OK

What
killSprite
Who
bat sword
Where
X: 1 Y: 4 X: 1 Y: 4
When
33

What
killSprite
Who
troll sword
Where
X: 6 Y: 1 X: 6 Y: 1
When
136

What
killSprite
Who
monster sword
Where
X: 5 Y: 5 X: 5 Y: 5
When
158

Figure 2: (1) The user query for "kill" events involving the use of the avatar's sword. (2) the system presents the query results. (3) After the user clicks on the middle panel, the system changes its focus to SeekWhence and shows the exact frame *when* the event happened. It also highlights the position *Where* the event happened. (4) By moving one frame backwards, the user can see the sprites positions after the event.

Example usage

We believe that Kwiri can be used to explore solutions for common and novel game problems. In the following subsections we show some examples.

Quantitative User Study Kwiri was also used in a quantitative user study (Machado et al. 2018). The goal of the work was showing that humans with AI assistance can be more accurate in-game bug detection than humans without assistance. In one of the tasks, agents collected data and the users of group A had to use Kwiri as a way to filter events and figure out what was causing the failures. For the same task, group B users just had SeekWhence available. The possibility of filter the events made the users approximately 32% better than the ones without it.

Automatic Game Tutorials The work of Green et al (Green et al. 2018). introduces a fully automatic method for generating video game tutorials. The AtDELFI system (AuTomatically DEsigning Legible, Full Instructions for games) was designed to research procedural generation of tutorials that teach players how to play video games. In the paper, the authors present models of game rules and mechanics using a graph system as well as a tutorial generation method. The concept was demonstrated by testing it on games within the General Video Game Artificial Intelligence (GVG-AI) framework. AtDELFI uses Kwiri as a way to search for the critical events that make a player win and lose a game. The graph generated by AtDELFI starts the query engine that captures the events and returns a sequence of frames, which are used to generate videos of the tutorials.

Preliminary User study

We employed a qualitative method in order to understand, from our users, the benefits of Kwiri, as well as to solicit suggestions for future additions and improvements.

Study Design

We created one inconsistency in the rules for each of three different VGDL games: Zelda, Aliens, and FireStorms. Their official rules are listed below, as well their inconsistencies.

Zelda - An action game (a clone of the cave levels of *Zelda: A Link to The Past*)

1. Player
   - Cannot go through walls;
   - Can kill enemies with a sword;
   - Changes its sprites when it gets the key;
   - To win the level needs to get the key and access the gate.
2. Enemies
   - Cannot go through walls;
   - Can occupy the same sprite of another enemy, a key and a gate;
   - Cannot kill other enemies;
   - Kill Player when colliding with it.

The inconsistency in this game is that some enemies can kill other enemies.

Aliens - A Space Invaders clone

1. Player
   - Kill enemies and barriers by shooting towards them;
   - Cannot go through walls;
   - To win the level needs to kill all the enemies.
2. Enemies
   - Cannot go through walls;
   - Kill Player by shooting towards it or when colliding with it.
3. Barriers
   - Are destroyed when hit by enemy bombs, by player bullets or when colliding with enemies.
4. Bombs/Bullets
   - Just kills enemies/ Just kills the player;
   - Are destroyed by walls.

The inconsistency here is that some barriers are not destroyed by bombs.

FireStorms - A puzzle game

1. Player
   - Cannot go through walls;
   - Cannot kill enemies;
   - Is killed when colliding with enemies or fire balls;
   - Wins the game when it reaches the closed gate.
2. Enemies
   - Cannot go through walls;
   - Can occupy the same sprite of another enemy, a purple portal, the closed gate and the fireballs;
   - Cannot kill other enemies;
   - Kill Player when colliding with it.

The inconsistency is that one of the enemies can walk through walls.

User Tasks

The user tasks consisted of finding the inconsistency in each one of the three games. To perform the tasks they were allowed to use Kwiri, and obviously, combine it with SeekWhence and the visualization system.

Participants

The study had nine participants, all of them male, eight of them were enrolled in a university program (7 Ph.D. students and one undergraduate student), and one was a digital media professional. Just two of them did not study/work with games/gaming. The other ones had mixed experience between industry and academic fields, varying from two to fifteen years. The most cited engines and frameworks by them were Unity, Phaser, and GameMaker. They were recruited through the department e-mailling list.

Procedure

The first step of the study was to ask the participants to fill out a form about their demographic data and game development experience. After that, we asked if they would be comfortable with their voices being recorded. Then we explained how SeekWhence and Kwiri work. The explanation took less than five minutes. In the end, we asked the users if they wanted to ask any question or skip it to use the tools for a quick warm up.

The second step started by introducing the users to what would be their tasks during the experiment. We informed them that we would give them three different tasks and that they should use the set of features to find a design inconsistency in each one of them. For each task, first, we handed the users a sheet of paper with all the rules of the game in the evaluation. After they read the rules, we ran the game with the agent *"adrienctx"*. We choose this agent because it is a former winner of the GVGAI competition. It is able to play the games well, and most of the times can beat the levels it is playing. Then, all the data were available to the user to actually start to work with them in order to try to solve the task.

The third and last step of the procedure consisted of a conversation where the users could express their opinions about the tested features and compare it with others they used before.

We checked in our users after five minutes of working on a task. We decided on this number after three pilot tests executed with a preliminary version of the system. However, this time is not a measure of success. It was only used to verify if the user is feeling tired and/or uninterested in continue after five minutes of effort. We gave them the option to keep trying or move to another task.

Source of Data

We used three sources of data in order to collect the users' activities: direct observation, audio recordings of test sessions, and a design questionnaire.

Direct observation As participants interacted with three different games and were allowed to speak about their findings, problems, and suggestions related to the tools, they had their voices recorded, and written notes were made of their overall patterns of use and verbal comments. Attention was paid to participants' interaction between the different systems (SeekWhence, Kwiri, and visualization), how hard they had to work to find the inconsistency, what design features attracted their attention, and whether, at any stage during the study, they seemed to lose interest in the activity.

Audio recordings of test sessions Every user had their voice recorded. This helped us to validate our written notes since we used them as tags to pay attention when listening to their recordings. Also, we used this source to clarify some actions they performed which were not initially clear to us.

Design questionnaire Our design questionnaire is based on semi-structured interview and required factual (e.g., "When I was querying for what a sprite was doing."), perceptual (e.g., "I think that the query results annoyed me with too much information.") and comparative (e.g., "It would be a plus to have these features in the tools I have used before"). We started the questionnaire by asking general questions in order to let the user feel comfortable like "Which are your thoughts about these features?". Then we moved to specific ones, many of them influenced by our notes, like: "In the second task, could you explain what you were trying to do when you asked if you could type your queries?". Finally, we asked if the users had more suggestions besides the ones they suggested during the tasks.

Data analysis

Our data analysis is based on text transcriptions of the design questionnaire discussed in the previous section. We used a classical procedure, Qualitative Content Analysis since it is considered suitable to analysis text materials, which varies from media products to interview data (Bauer and Gaskell 2000). Normally it creates key common points identified among different users interviews in a technique called by codification theory, which helps researches in quick organizing and managing qualitative data. To facilitate the codification process, we used the trial version of the software Atlas.ti (GmbH 2017).

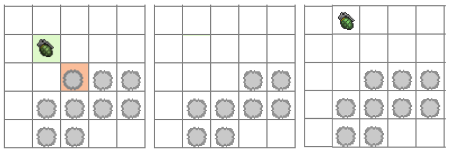


Figure 3: (Left) The users found a bug we were not aware of by querying for kill events involving a bomb and a barrier. (Center and Right) The users navigating forward and backward to confirm the inconsistency: the bomb was destroying a barrier which wasn't on its line of fire. Later on during the interviews, some users stated that the green and red patterns (Left figure), used to express the sprite who does the action and the one who suffers, should also be indicated on the query panel.

Results

As explained in the previous section, we did a qualitative study to know from our users which are the significant gains of our tools and which are the points of improvements in a next iteration of our design process. We will present the data first with a general discussion of our findings by observing the users. Then we will present our conclusions based on the users' point of view.

Task results

Most of the users were able to complete all three of the tasks with success. Just two gave up, one during the *Zelda* task, and another during the *Aliens* task.

They started by exploring their options, right after reading the game rules given to them. Some started by using SeekWhence and tried to figure out the problem just by replaying the game frame-by-frame. Then they switched to the query system to filter based on their suspicions. Other users did the opposite, starting by querying and then switched to SeekWhence. This exploratory step was skipped by two users who decided to do a warm-up during the explanation of how the system works.

One fact that grabbed our attention was that five of our users could identify one inconsistency that we were not aware of. In the game *Alien* they could observe that a bomb was destroying a barrier in a position it was not designed to. Figure 3.

Praise

In general, the users agreed that Kwiri was a valuable resource for finding game design inconsistencies. Some of them pointed to personal experiences with situations in

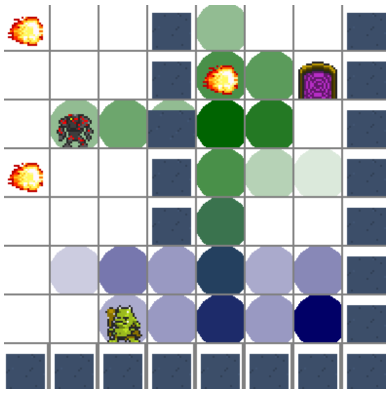


Figure 4: User solving the inconsistency on the third task. By navigating on the gameplay session with SeekWhence and using the visualizations he could see the enemy walking through walls.

which they needed tools like the one presented here, but instead, they ended up hand-rolling their own.

One of the users explained that he had to write his system in order to evaluate how an AI was behaving. *"I liked these features. Some time ago, I had some issues with an AI that I was developing for playing a game. The tool I was using didn't have this feature about navigating and use visualizations. Then I had to write my own system to do exactly what we were doing here! Because here, I really can see what the agent is doing."*

Another user, similarly, does not have the appropriate tools to inspect his game projects. *"It is a really cool feature, pretty deal! I would like to use it on my work now, especially this frame navigating tool. But I'm having to implement everything from scratch."*

A third user revealed that his work consisted of play his game to figure what was causing the problems. *"I developed a game once, but I did not have how to record it and play it again as I did here. So every time that I found a bug, the only way to debug it was by playing it again. If I had a tool like that, at least my debug process would be way easier"*. Figure 4

Issues

Some users said they had problems in understanding the queries, for example, one of them stated that at some point it pops up too much information making the process not so attractive to follow. However, another user stated the opposite. In his speech, information volume does not seem to be a problem. The way that the UI leads to them is what concerns him. *"I really liked these tool, but I would like to see more information, however with less clicks."* Still, about the query system, one user said that he was not sure about the roles of a sprite in an event. *"I would like to know who is killing who in a 'killSprite' event. The query helped me to confirm my suspicion that an enemy was killing another, but the panel in the Who area should say who is the one doing the action."*. Overall, as pointed in this section, the main issues were related to the query system. Design them as a filter tool does not seem enough. At least for the tasks evaluated, despite the majority of the users were able to use it correctly, it should provide details about the sprite roles in the game and be more explicit about the event as a user stated: *"I wasn't sure about what this event - transfrom to - mean. I was doing assumptions based on what I have seen before on other tools like GameMaker. Fortunately, it is similar, but I would like to have this information before."*

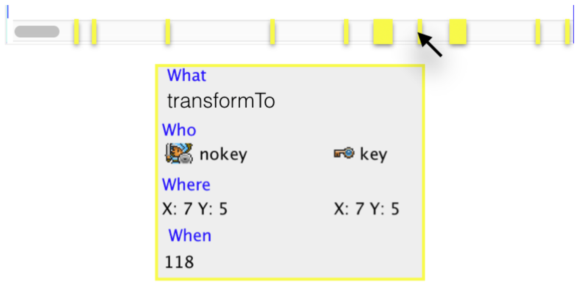


Figure 5: A quick prototype of a user suggestion. Yellow ticks on the timeline tells in which frames we have event(s) happening. By hovering the mouse over them we can see a preview of the event(s).

Suggestions

We got many suggestions from our users that we will add in future iterations of the system. One user said that he would like to see small ticks on the timeline bar. The ticks would be used to let a user know that one or more event is happening on that part of the gameplay session. To assure what he was suggesting we used the Youtube (Youtube 2017) yellow advertise ticks as a design metaphor. They are used to explain to a user when an add will pop up. He promptly confirmed that it was exactly what he had in mind. Figure 5.

Another user suggested us to present the query options as a tree visualization where he could filter based on options available in each branch of the tree. It is an interesting suggestion, and we think it can reduce user efforts and present a better way to lead them to what they are trying to find.

Finally, another user said that the same color patterns used to represent the agents that does (green) and suffers (red) the actions should be available on the panel used to make the queries. He said that in general the queries help when one has to identify who is involved in an event. However, it is also important to know who started the cause and who is getting the consequences. It was also stated as an issue by another user and is something straightforward to fix in the query UI panel.

Conclusion and Future Work

We have presented a system for querying game events in space and time. This system is a new addition to Cicero, a general mixed-initiative game design assistance tool. We evaluated our tool in the context of finding design inconsistencies/bugs. The users were able to solve their given tasks promptly and provided us with valuable suggestions; notably, they found bugs we were not even aware of, speaking to the usefulness of the system. To our surprise, the users were significantly attracted to the replay analysis system (SeekWhence) than to the query one (Kwiri) to solve their tasks. It contradicts a previous quantitative study that shows users having much better results, in similar tasks, by using a query system than a replay analysis.

We plan to test the system further, on a large audience of game designers. We also want to design specific tasks for evaluating each one of the features separately and in different kinds of combinations. Besides finding inconsistencies, we will also focus on agent evaluation and game balancing.

Acknowledgment

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

References

Bauer, M. W., and Gaskell, G. 2000. *Qualitative researching with text, image and sound: A practical handbook for social research*. Sage.

Bontrager, P.; Khalifa, A.; Mendes, A.; and Togelius, J. 2016. Matching games and algorithms for general video game playing. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Butler, E.; Smith, A. M.; Liu, Y.-E.; and Popovic, Z. 2013. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, 377–386. ACM.

Canossa, A.; Nguyen, T.-H. D.; and El-Nasr, M. S. 2016. G-player: Exploratory visual analytics for accessible knowledge discovery.

Deutch, D.; Greenshpan, O.; Kostenko, B.; and Milo, T. 2012. Declarative platform for data sourcing games. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, 779–788. New York, NY, USA: ACM.

Ebner, M.; Levine, J.; Lucas, S. M.; Schaul, T.; Thompson, T.; and Togelius, J. 2013. Towards a video game description language.

El-Nasr, M. S.; Drachen, A.; and Canossa, A. 2013. *Game Analytics: Maximizing the Value of Player Data*. Springer Publishing Company, Incorporated.

GmbH, S. S. D. 2017. Atlas.ti.

Green, M. C.; Khalifa, A.; Barros, G. A. B.; Machado, T.; Nealen, A.; and Togelius, J. 2018. Atdelfi: Automatically designing legible, full instructions for games. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, 17:1–17:10. New York, NY, USA: ACM.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *FDG*, 213–220.

Lucas, S. M.; Mateas, M.; Preuss, M.; Spronck, P.; and Togelius, J. 2012. Artificial and Computational Intelligence in Games (Dagstuhl Seminar 12191). *Dagstuhl Reports* 2(5):43–70.

Machado, T.; Gopstein, D.; Nealen, A.; Nov, O.; and Togelius, J. 2018. Ai-assisted game debugging with cicero. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1–8.

Machado, T.; Nealen, A.; and Togelius, J. 2017a. Cicero: Computationally intelligent collaborative environment for game and level design. In *3rd workshop on Computational Creativity and Games (CCGW) at the 8th International Conference on Computational Creativity (ICCC17)*.

Machado, T.; Nealen, A.; and Togelius, J. 2017b. Seekwhence a retrospective analysis tool for general game design. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, FDG '17, 4:1–4:6. New York, NY, USA: ACM.

Medler, B., et al. 2009. Generations of game analytics, achievements and high scores. *Eludamos. Journal for Computer Game Culture* 3(2):177–194.

Perez, D.; Samothrakis, S.; Togelius, J.; Schaul, T.; Lucas, S.; Couëtoux, A.; Lee, J.; Lim, C.-U.; and Thompson, T. 2015. The 2014 general video game playing competition.

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

Shaker, N.; Shaker, M.; and Togelius, J. 2013. Ropossum: An authoring tool for designing, optimizing and solving cut the rope levels. In *AIIDE*.

Smith, A. M.; Andersen, E.; Mateas, M.; and Popović, Z. 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*, 156–163. ACM.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 209–216. ACM.

Srisuphab, A.; Silapachote, P.; Chaivanichanan, T.; Ratanapairojkul, W.; and Porncharoensub, W. 2012. An application for the game of go: Automatic live go recording and searchable go database. In *TENCON 2012-2012 IEEE Region 10 Conference*, 1–6. IEEE.

2017. Unity technologies. unity game engine. `https://unity3d.com`. Accessed: 2017-03-01.

Varvaressos, S.; Lavoie, K.; Massé, A. B.; Gaboury, S.; and Hallé, S. 2014. Automated bug finding in video games: A case study for runtime monitoring. In *Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on*, 143–152. IEEE.

Wallner, G.; Kriglstein, S.; Gnadlinger, F.; Heiml, M.; and Kranzer, J. 2014. Game user telemetry in practice: A case study. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, ACE '14, 45:1–45:4. New York, NY, USA: ACM.

White, W.; Sowell, B.; Gehrke, J.; and Demers, A. 2008. Declarative processing for computer games. In *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games*, 23–30. ACM.

Youtube. 2017. Youtube.

Zoeller, G. 2010. Development telemetry in video games projects. In *Game developers conference*.