We'll Fix It in Post: What Do Bug Fixes in Video Game Update Notes Tell Us?

Andrew Truelove
University of California, Irvine
Irvine, CA, USA
truelova@uci.edu

Eduardo Santana de Almeida
Federal University of Bahia
Salvador, Brazil
esa@rise.com.br

Iftekhar Ahmed
University of California, Irvine
Irvine, CA, USA
iftekha@uci.edu

***Abstract*—Bugs that persist into releases of video games can have negative impacts on both developers and users, but particular aspects of testing in game development can lead to difficulties in effectively catching these missed bugs. It has become common practice for developers to apply updates to games in order to fix missed bugs. These updates are often accompanied by notes that describe the changes to the game included in the update. However, some bugs reappear even after an update attempts to fix them. In this paper, we develop a taxonomy for bug types in games that is based on prior work. We examine 12,122 bug fixes from 723 updates for 30 popular games on the Steam platform.We label the bug fixes included in these updates to identify the frequency of these different bug types, the rate at which bug types recur over multiple updates, and which bug types are treated as more severe. Additionally, we survey game developers regarding their experience with different bug types and what aspects of game development they most strongly associate with bug appearance. We find that *Information* bugs appear the most frequently in updates, while *Crash* bugs recur the most frequently and are often treated as more severe than other bug types. Finally, we find that challenges in testing, code quality, and bug reproduction have a close association with bug persistence. These findings should help developers identify which aspects of game development could benefit from greater attention in order to prevent bugs. Researchers can use our results in devising tools and methods to better identify and address certain bug types.**

I. Introduction

Similar to any regular software, the presence of bugs in video games can cause considerable problems for game developers, including lost sales, damaged public image, and even lawsuits [15], [19]. These bugs can also impact the users–and their ability to enjoy the full range of game features–in a variety of ways. Certain bugs can cause players to lose hours of progress [13], while others can create misunderstandings between players that might result in receiving "abuse from angry teammates" [8]. Despite the negative consequences a bug can have on developers and users alike, it is common for games to release with bugs that are fixed through subsequent updates. In 2014, for example, over a third of big-budget games "released on Xbox One, Wii U and PS4" received an update within 24 hours of the game's initial release [22].

One of the reasons bugs may not get detected before releasing is because of the difficulty in comprehensively testing all aspects of a video game [21], [28]. Developers have difficulty writing comprehensive tests, because games can have a significantly large number of possible user interactions compared to other types of software [28]. Players can act unpredictably when playing games, and automated testing tools struggle to replicate the full range of interactions that human players might attempt when playing a game [28]. As a result, many games release with undiscovered bugs that only reveal themselves once customers begin playing the game [22].

One way developers address previously undetected bugs post-release is by applying updates to their games that fix these bugs [18]. These updates are usually accompanied by update notes, a textual description of the changes in the accompanying update [30]. These update notes are often published as news items on the game's website (e.g., [9]) or on an official online game forum (e.g., [32]). While some updates might include content changes or additions, other updates might only attempt to fix a small number of severe bugs [18]. Some of these smaller updates are referred to as "hotfixes"; these updates are typically meant to remedy more pressing issues or bugs that need immediate attention [30].

Additionally, some bugs might reappear or recur in multiple updates. There exist cases in which developers have attempted to fix a bug in one update, only for that bug to reappear in the game anyway, necessitating another attempted fix in a later update [16]. For example, in the game *Warframe*, the notes for an update posted in August of 2019 included a bug fix statement that said, *"Fixed inability to complete the Mastery Rank 12 test if you fall off the starting platform"* [12]. In September of 2019, the notes for a later update also included an identical line, which implies that the August update did not successfully fix the bug [11].

Previous research has focused on creating a taxonomy for video game bugs [17]. While this taxonomy provides a solid basis for categorizing bugs in games, a cursory examination of bug fixes described in game update notes indicated that this taxonomy did not cover all the bugs addressed in these updates. Since the taxonomy developed by Lewis et al. was developed primarily by observing videos of game play and reading articles related to bug complaints from users [17], it fails to identify the bugs that are more subtle and less readily noticeable, such as bugs related to background variables that are not presented to the player.

To address the shortcomings of the existing taxonomy and to attain a deeper understanding of the types of bugs in games, in this paper, we expand the taxonomy of bug types. We posit

that improving the taxonomy will improve the chances of identifying the types of bugs missed in testing and will also improve the development pipeline in order to better identify these bugs. This will allow us to gain a deeper understanding of the association between different bug types to different game development techniques and processes.

Using this expanded taxonomy, we collect update notes for 30 popular games on the Steam platform [3], [4] and categorize the bug fixes present in these updates. We analyze the frequency at which the different bug types appear in the update notes and investigate which types of bugs recur more often over multiple updates. Additionally, we investigate which types of bugs most frequently appear in urgent updates or hotfixes, as the bugs that appear in these updates are more likely to have a severe negative impact on users [30]. Finally, we survey game developers on their experience with these different types of bugs as well as what challenges and techniques are involved in fixing these bugs.

While past research has investigated the content and timing of game updates [18], to the best of our knowledge, our work is the first to examine the frequency in which certain types of bugs appear and recur in game updates and the first to investigate the reasons why certain types of bugs appear more frequently than others. Specifically, this paper addresses the following research questions:

RQ1: What are the most frequently fixed types of bugs through game updates?

RQ2: What are the most frequently recurring types of bugs?

RQ3: Are all types of bugs equally severe in negatively impacting the game experience?

II. Related Work

Prior research on video games have looked into creating a taxonomy of bugs, identifying design patterns and smells. Lewis et al. created a taxonomy for video game bugs after surveying online videos, articles and online communities dedicated to video game issues [17]. This taxonomy serves at the basis for the taxonomy used in our study. As part of a broader investigation into the relationship between programming languages and software quality, Ray et al. developed a taxonomy of bugs based on the cause and impact of the bug [25]. This taxonomy was used by Pascarella et al. when classifying faults in open source games [23]. However, we found that this taxonomy was not suitable for our purposes. For one, it is not specific to games, which limits their utility for aiding game developers with the issues that are more particular to their environment. Secondly, this taxonomy is based on non-functional characteristics including Performance, Concurrency, Algorithm, Memory, Programming, and Security, and was developed by analyzing bug fix commit messages from open source projects [25]. This type of information is not available in release notes for popular non-open source games [25].

Ampatzoglou et al. performed a case study on a collection of open-source Java games to find possible correlations between the application rate of design patterns, the defect frequency and the debugging efficiency [7]. They found that certain patterns such as the Adapter pattern and Observer pattern exhibited some correlations with defects and debugging, but could not identify causal relationships [7]. Borrelli et al. identified seven types of bad smells in game projects that were created with the Unity game engine and presented UnityLinter, a static analysis tool designed to help Unity developers detect these smells [10].

Some researchers have focused on identifying differences between game development and development of other software. Murphy-Hill et al. conducted surveys and interviews with game developers [21]. Some of the findings were related to difficulties with testing; some game developers expressed difficulty in designing tests that could explore the state space in games, while others indicated that there was a lack of automated, low-level testing [21]. Santos et al. examined the particularities of software testing in game development, and, through the use of case studies and surveys, found a number of difficulties that game developers face while testing their software, which, in addition to challenges in test automation, included rapidly changing requirements and impracticality in testing interactive components [28].

There exists prior research on software release notes. Abebe et al. investigated the types of content that appear in release notes for updates to general software. They found six types of information that are generally in release notes, including a description of the issues addressed in the release [6].

Lin et al. performed an empirical analysis of urgent updates for games on the Steam platform–including which types of games released urgent updates, the timing of these updates, and the reasons for releasing urgent updates [18]. With respect to the reasons for releasing urgent updates, the authors created a taxonomy for these primary reasons [18]. However, this taxonomy was not tailored specifically to bugs and included reasons that were not strictly related to bugs [18]. When matching their taxonomy to the taxonomy of Lewis et al. some of the Lewis et al. categories fell under multiple categories proposed by Lin et al., while one of the Lin et al. categories was matched to seven of the 11 Lewis et al. categories [17], [18]. In this paper, our taxonomy is focused on bug fixes rather than the general content of updates. Additionally, Lin et al. did not look into why certain types of bugs appear in updates, what kind of bugs recur over multiple updates and why, and how severe these different bug types are in comparison to each other. Furthermore, they did not survey game developers about their experience with game updates. These are all major components of our work.

III. Methodology

The methodology for our study included two primary components. The first source of information encompassed update notes for popular games. The second source of data were responses from a survey distributed to game developers.

A. Data Collection

For our dataset, we used update notes from the most popular games on the Steam platform [3]. We chose the Steam platform, because Steam is the largest vendor for games on

the PC system [31]. To identify a selection of popular games, we found Steam's list of games that had the highest number of players on June 23, 2020 [4]. We chose the top 30 games from this list, excluding items that were not really games (e.g., [5]). The list of games used for this paper can be found in Table I.

Once we had our collection of games, we manually extracted the update notes. For each game, we collected all update notes in the one-year period leading up to the date of the game's most recent update. For example, the most recent update we collected for the game *Counter Strike: Global Offensive* was released on June 22, 2020, which meant the earliest update we collected for this game was released on July 16, 2019. Another game on the list, *Sid Meier's Civilization V*, was much older, with its most recent update being released on October 30, 2013, which meant the earliest update we collected was on March 13, 2013.

Each game on Steam had its own channel that displayed news and updates about the game, including some update notes. We found this source was incomplete for many games, however. Not all of the game's updates were always posted to the Steam channel. For instance, smaller updates that only contained one or two lines–such as hotfix updates–seemed less likely to appear on the Steam channel. For most games, we used the updates notes from the official game or developer web site (e.g., [9]) or online forums (e.g., [32]), relying on the Steam update channel only if there was on other option.

Once we collected the update notes, we used a script to collect the text from the updates related to bug fixes. The text of each update note was split into smaller segments of text based on the line breaks of the source page formatting (referred to as "lines" going forward). Under this scheme, each paragraph or list item was treated as an individual line. A formative analysis of the update notes revealed certain words or phrases that were frequently present in bug fix lines, such as "fix" and "corrected". We applied a simple filtering scheme that scanned through all our update notes and flagged lines with these terms. Each of these flagged lines was recorded in our data alongside supplemental information that included the game the update containing the line came from, the date of the update, and the ID of the update (for cases in which multiple updates were released in a single day).

B. Types of Bugs

The next step was to label the bug fixes in our data. Initially, our goal was to label the bug fixes based on the 11 bug types identified by Lewis et al. [17]. However, we identified that there were bug-fixes that did not fit in any one of the existing types. Hence, two of the authors authors jointly applied open coding [29]. First, we generated short descriptions of the initial 11 bug types based on the descriptions from Lewis et al. [17]. As we labelled bug fixes from our data, if we found a bug fix that did not fall under a currently available label, we created a new label. Additionally, we would search through previously labelled bug fixes to check if any of these fixes better fit under the new label. This step additionally helped remedy instances in which a bug could conceivably fall under multiple categories, a problem Lewis et al. also encountered [17]. In cases like these, we chose which category best described the core impact of the bug. For example, one bug fix for the game *Black Desert Online* said, *"fixed the issue where running left or right was impossible when transformed into an immortal persimmon knight"*. While part of this bug related to incorrect position of a game object, the inability to perform an action seemed to be the core problem. Therefore, the *Action* label was applied instead of the *Position of Object* label.

Finally, after labelling the full collection of bug fixes, we reviewed and consolidated our categories, combining categories that could be classified as sub-types of another category. During this step, we also discarded lines that did not actually contain bug fixes and lines in which the nature of the bug was unclear and could not be categorized. During the coding process, there were no disagreements between the authors. These bug types are described in Table II. In total, 12,122 lines were labelled under this taxonomy; these labelled lines came from 723 different updates.

C. Bug Recurrence

The next part of our process required us to identify all recurring bug fixes that appeared in multiple updates for a single game. Due to the large number of updates and the large number of bug fixes, manually searching through the data to find all recurring bugs would be prohibitively time consuming. At the same time, we discovered a purely automated approach could also lead to problems. A cursory analysis of repeating bug fixes in the data revealed that some repeating lines were generally written; it was unclear whether or not the fix was applying to the same bug each time. For example, the game *ARK: Survival Evolved* had 5 updates that included the line "fixed a server crash". In this case, it was not apparent whether this line was referring to the same bug each time or different bugs that each caused crashes. Conversely, two updates for the game *Dead by Daylight* both contain the same line: "*fixed an issue that caused a crash when changing the graphic quality settings during a match in the red forest maps*". Here, the apparent cause and effect of the bug are identical; changing certain settings in a certain context causes the game to crash. In this case, both lines are likely referring to the same bug. We were skeptical that an automated tool would be able to differentiate between these two types of recurring bug fixes, so we chose not to use a purely automated approach either.

We ultimately decided on a hybrid approach. We would use an automated approach to limit our analysis to a smaller collection of bug fixes and then manually evaluate these lines to find the true recurring bugs. For this first step, we performed cosine similarity analysis [27] between bug fix lines. For each bug fix line from a game's updates, we compared that line to each line in all subsequent updates for the game. If two lines had a similarity score of at least 90%, they were treated as a potential match and grouped together for manual review. Our small-scale formative analysis showed that a similarity threshold of 90% was sufficient to minimize false positives. A line could have multiple matches if it appeared in more

TABLE I
MOST POPULAR STEAM GAMES ON JUNE 23, 2020

Game Title	Developer	Release Year	Number of Players
Counter-Strike: Global Offensive	Valve, Hidden Path Entertainment	2012	674,145
Dota 2	Valve	2013	383,875
Destiny 2	Bungie	2019	95,419
PLAYERUNKNOWN'S BATTLEGROUNDS	PUBG Corporation	2017	89,878
Path of Exile	Grinding Gear Games	2013	77,919
Grand Theft Auto V	Rockstar North	2015	75,732
Tom Clancy's Rainbow Six Siege	Ubisoft Montreal	2015	69,181
Football Manager 2020	Sports Interactive	2019	65,397
Team Fortress 2	Valve	2007	62,675
ARK: Survival Evolved	Studio Wildcard, Instinct Games, Efecto Studios	2017	60,830
Dead by Daylight	Behaviour Interactive Inc.	2016	60,558
Rocket League	Psyonix LLC	2015	57,055
Terraria	Re-Logic	2011	54,374
Rust	Facepunch Studios	2018	53,173
Sid Meier's Civilization VI	Firaxis Games	2016	37,609
Warframe	Digital Extremes	2013	32,011
Total War: WARHAMMER II	CREATIVE ASSEMBLY,	2017	27,326
PAYDAY 2	OVERKILL - a Starbreeze Studio.	2013	26,251
Europa Universalis IV	Paradox Development Studio	2013	23,782
The Elder Scrolls V: Skyrim Special Edition	Bethesda Game Studios	2016	23,639
The Elder Scrolls Online	Zenimax Online Studios	2014	22,657
Hearts of Iron IV	Paradox Development Studio	2016	21,694
SMITE	Titan Forge Games	2015	21,329
Unturned	Smartly Dressed Games	2017	20,987
Sid Meier's Civilization V	Firaxis Games	2010	20,790
Black Desert Online	Pearl Abyss	2017	18,931
Brawlhalla	Blue Mammoth Games	2017	18,841
War Thunder	Gaijin Entertainment	2013	18,460
Euro Truck Simulator 2	SCS Software	2012	17,912
FINAL FANTASY XIV Online	Square Enix	2014	17,854

than two updates; all instances of these bugs were grouped together. Once we collected all potential matches, we manually inspected them and removed all false positives as well as those matches in which it was unclear whether the matching lines were really referring to the same bug. After filtering the potential matches, we were able to collect information on the types of bugs that recurred over multiple updates.

D. Bug Severity

To identify which bugs were treated as severe, we used metrics employed by Lin et al. to identify urgent updates [18]. Under these metrics, updates were considered urgent if they were: identified as hotfixes within the notes for the update, 0-day updates (same day as the last update), or deployed sooner than normal, based on the game's regular update schedule [18].

In addition to these metrics, we also looked at the length of the update notes. Based on our own observations of the data, it appeared as if updates addressing urgent bugs typically had shorter update notes than regularly scheduled updates. This is likely because urgent updates are only intended to fix a small number of critical issues that cannot wait until the next major update. To find the urgent updates in our data, we used a script to parse through each update in our data to check the following information: whether the update notes contained language referring to the update as a hotfix, the number of days between the current update and the last update, and the number of non-whitespace lines in the update notes.

For the self-identified hotfixes, we searched for updates that contained the word "hotfix" in the first twenty lines of the update text. This was to help avoid false positives, because updates that self-identified as hotfixes would use the language near the beginning of the update text. When the word appeared further into the text, this was typically because the update was referring to another hotfix update, such as a past update or a planned future update. After running our script, we then manually reviewed the resulting updates and filtered out any remaining false positive updates and marked the remaining updates as urgent. For the 0-day updates, we marked as urgent all updates that had a value of 0 days since the last update.

For the last two metrics (number of days and number of non-whitespace lines), we used Double MAD (Median Absolute Deviation) [18], [26] to find the outliers below the median for both the days since the last update and for the update size. Updates that fit either of these criteria were also marked as urgent. We used the same threshold value of 2 that was used by Lin et al. in their paper [18].

TABLE II
TYPES OF BUGS FOUND IN UPDATE NOTES

Category	Description	Category	Description
Action	Errors in the ability/inability to perform actions.	Game Graphics	A certain visual aspect of the game world is being rendered incorrectly.
Artificial Intelligence	An NPC or AI does not behave in the intended manner.	Implementation Response	There are errors regarding the game's interactions with hardware or base software that ends up affecting performance.
Audio	Game audio is not playing correctly.	Information	Information about the game world is not conveyed to the player correctly.
Bounds	An object is outside of the world boundaries or outside the intended game space.	Interaction of Object Properties	Two or more object properties do not behave properly when interacting with each other.
Camera	There is an issue with game camera through which the player views the game world.	Interrupted Event	An action in game that was previously operating is terminated abruptly against expectations, or the action was not interrupted when it should have been.
Collision of Objects	Objects do not behave properly when they collide or make contact with each other.	Object Persistence	Game objects do not correctly enter or exit the game world.
Context State	Objects do not enter and exit states properly.	Position of Object	An object is not in the correct position or orientation within the game space over time.
Crash	The game crashes or is otherwise forced to close.	Triggered Event	Improper triggering of game events (Event A causes Event B).
Event Occurrence	Discrete events do not occur at the correct rate.	User Interface	The appearance/position/behavior of UI elements are incorrect.
Exploit	Players are able to improperly and unfairly gain game benefits through behavior unintended by developers.	Value	An in-game variable is not set to the correct value.

Once we identified all the urgent updates in our data, we flagged all the bug fixes lines from our data that appeared in an urgent update. An urgent update is generally intended to fix "problems that are deemed critical enough to not be left unfixed until a regular-cycle update" [18]. As such, the bug fixes included in these urgent updates are likely to have severe negative impacts on the game experience that require more immediate attention. For our purposes then, if a bug fix was included in an urgent update, we treated it as a severe bug. We used the findings from this process to calculate the severity rate of each bug type, as explained in Part *C* of Section IV.

E. Survey

Protocol: We created a 12-minute survey using the Qualtrics survey system [2] to assess game developer perceptions of the types of bugs that appear in game update notes. First, the survey asked participants certain demographic questions, such as the years of game development experience and the number of people in their organization. We also asked participants if they had experience fixing bugs for updates to video games. The next section asked participants about their experience with the different bug types in our taxonomy. For each of the twenty bug types, we asked participants to rate on a 5-point Likert scale their level of agreement that the bug type was likely to have a severe negative impact on the game experience. Participants were then informed of the three most frequent bug types across all updates and were asked if these results matched with their own experience. They were also offered to explain why they believed these bug types (and other bug types they commonly encounter) appear so frequently in game updates. Similar questions were also asked for the frequently recurring bug types. Finally, participants were asked about the aspects of game development related to bug recurrence and the challenges in identifying and fixing bugs.

Respondents: To find participants, we identified GitHub [1] repositories for games and game-related tools that had been marked as noteworthy by other GitHub users. We sent the survey to GitHub users who had contributed to these repositories and who had made their email addresses visible to other users. Additionally, we sent the survey to developers of the games examined for this study whose email addresses were publicly listed on the developers' web sites. We also made the link to

the survey available to a collection of game developers with the intention that they would apply snowballing to spread the survey to other developers [14].

The survey was sent to 630 contacts. 16 of these invitations were not delivered successfully. Three invitations generated automatic replies suggesting that the recipient was unable to receive the survey, and one recipient indicated that he did not possess a game development background. Overall, 610 survey invites were delivered, and we received 47 applicable responses. This gave us a response rate of 7.70%. Other important studies in the software engineering field have reported response rates ranging between 5.7% [24] and 7.9% [20], indicating that our response rate for this study is satisfactory.

IV. Results

A. Frequent Bug Types in All Updates

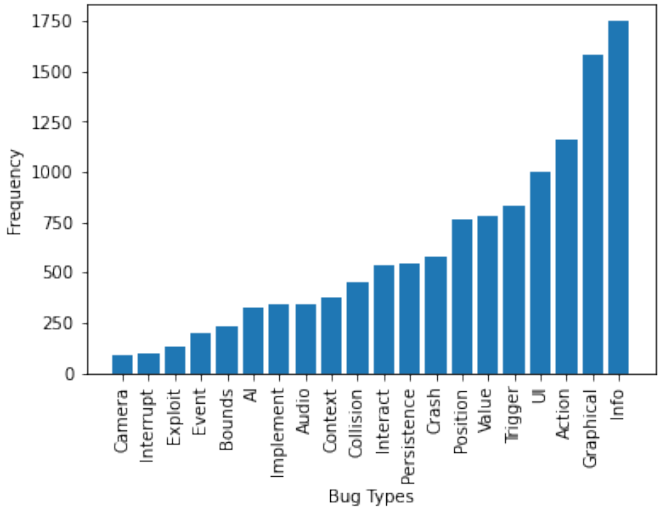

Fig. 1. Frequency of Bug Types Across All Updates

Figure 1 shows the frequencies at which each of the 20 bug types appeared as a fix in any game update from our data.

Observation 1: The most frequently occurring bug types were *Information*, *Game Graphics*, and *Action*.

The three least frequent bug types were *Camera*, *Interrupted Event*, and *Exploit*. All three of the most frequent bug types were bug types identified by Lewis et al. [17]. Meanwhile, for the three least frequent bug types, only *Interrupted Event* was one of the bug types identified by Lewis et al. [17].

The two most frequent bug types, *Information* and *Game Graphics* cover largely non-interactive elements of the game. This runs somewhat contrary to findings found by Santos et al. that suggest some of the difficulties in testing games are due to challenges in testing interactive elements [28]. If the most frequent bug types are not necessarily interactive in nature, perhaps automated testing might be more effective at catching these more common types of bugs.

For each of the three most frequent bug types, survey respondents were asked if they frequently fixed bugs of that type in game updates. 57% of respondents said they frequently fixed *Information* bugs, 40% said they frequently fixed *Game Graphics* bugs, and 67% said they frequently fixed *Action* bugs.

When asked why certain bug types might appear frequently, respondents gave a variety of responses. For *Information* bugs, some respondents referenced priority concerns.

- *"Developing mechanics is the main focus and usually information related components are pushed down to the last of the pipeline and as deadline approaches, they are not done right."*

Other respondents noted that the correct information may be difficult to identify if that information is the result of various calculations involving *"complex states and interactions"*. This information could also be *"buried in data files"* that testers may not be able to locate. Incorrect information might be missed, because the correct information may not be accessible.

Observation 2: Survey explanations for the frequency of *Information* bugs include: low priority compared to other aspects of the game and difficulties in finding the correct information.

With respect to *Game Graphics* bugs, the most cited reason by respondents to their frequency in updates was due to the complexity of the subject.

- *"Computer graphics is one of the most complex disciplines to work with and need highly experienced programmers to make sure that everything just work."*

Respondents also cited the large number of cases that need to be tested.

- *"It's fairly simple to miss visual corner cases in testing."*

Additionally, some responses seemed to speak to the difficulties in properly assessing the quality of game graphics. One respondent pointed out that game graphics can be subjective. What looks like a graphical bug to one tester may not look like a bug to another tester. Another respondent mentioned that the correct graphical qualities of the game might be highly specific and require fine-tuning to get them right.

- *"Graphics can be delicate. It can take a lot of tweaking to get something to look just right"*

One response noted that graphics are considered a lower priority to fix before production than other game elements.

Observation 3: Survey explanations for the frequency of *Game Graphics* bugs include: complexity, large size of testing state space and evaluation of graphics quality.

For *Action* bugs, many of the respondents spoke to complexity in testing as a reason these bugs might appear frequently in updates. Each possible action gives the player a new way to interact with the game world. As the number of possible interactions increases, there is a greater chance that these interactions might conflict with each other. Testing all possible conflicts can be difficult to do within the development timeline, especially when dealing with actions that may or may not be executable depending on the game state.

- *"Games often have a large number of available actions, only some of which are available at a given time; the results of actions often depend on context (and sometimes randomness); and actions may depend on or be changed by previously-performed actions. While testing every possible action might be feasible, testing every possible sequence of actions in every possible sequence of contexts is usually combinatorially impossible."*

Observation 4: Survey explanations for the frequency of *Action* bugs include: increasing complexity with additional possible actions and large size of testing state space.

Respondents were also asked if they frequently fixed bugs of any other type from our taxonomy and why these bugs might be frequent. *Crash* bugs were identified as frequent by two respondents, with one respondent attributing this frequency to *"not properly checking for input sanity"*. Another respondent mentioned *Exploit* bugs, stating *"players are good at finding them"*. Another respondent who worked with multiplayer games referred to bugs related to desynchronization of players. Under our taxonomy, this bug would fall under the *Implementation Response* label. Another user mentioned *Bounds* bugs as a frequent concern, especially for 3D games, explaining that the collision and physics systems in a game tend to produce edge cases somewhere that leads to bounds issues.

B. Recurring Bugs

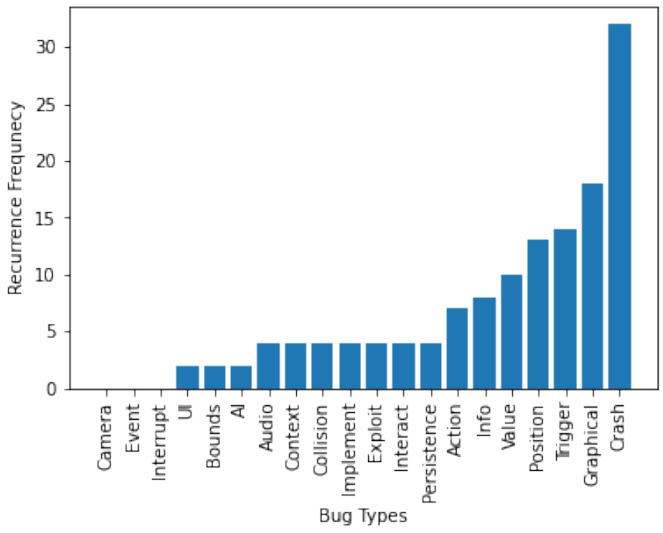

Fig. 2. Frequency of Recurring Bug Types

Figure 2 shows the frequency in which bugs of a certain type would recur over multiple updates.

Observation 5: The bug types that recurred the most frequently over multiple updates were *Crash*, *Game Graphics*, and *Triggered Event*.

The bug types that recurred over multiple updates the least frequently were *Camera*, *Event Occurrence*, and *Interrupted Event*; none of these bug types had any instances of bugs that recurred over multiple updates. One of the most frequently recurring bug types–*Game Graphics*–and two of the least frequently recurring bug types–*Interrupted Event* and *Event Occurrence*–came from the taxonomy of Lewis et al. [17].

Regarding the top three frequently recurring bug types, survey respondents were asked if they frequently fixed recurring bugs that fell into each of these categories. 48% of respondents said they frequently fixed recurring *Crash* bugs, 21% said they frequently fixed recurring *Game Graphics* bugs, and 24% said they frequently fixed recurring *Triggered Event* bugs.

For each of these top three bug types, survey respondents were offered to explain why these bug types recur over multiple updates more frequently. When it came to *Crash* bugs, multiple respondents cited the difficulties in identifying the actual root cause of the crash.

- *"When fixing a crash bug, it is easier to find the single caller that passed bad data in the crash report than it is to ensure that all callers pass good data."*

Other respondents spoke to how other changes to the game can easily cause the crash to reappear even after fixing it.

- *"Due to how interconnected most things in games are, it is easy to reintroduce a crash after initially fixing it"*

Observation 6: Survey explanations for the frequent recurrence of *Crash* bugs include: reproducing bugs and reappearance after other changes.

With respect to why *Game Graphics* bugs would recur frequently, the responses were more mixed. One respondent mentioned that game graphics can be *"inter-dependent and cascade through fixes"*, while another expressed experience with having to deal with unforeseen edge cases.

A different respondent reiterated the delicate nature of game graphics. A previous fix to a graphical component of the game might not have been sufficient, and further tweaking might be required to get the graphics just right. Another user pointed out that *"Graphics bugs are the most apparent thing for the player to notice"*. Building off these responses, it is possible that–since the quality of game graphics can be subjective, and because players more readily notice the quality of game graphics–these kinds of bugs recur more frequently because players are more likely to notice graphics that do not meet their personal quality standards, which could mean they are more likely to express this dissatisfaction to developers.

Observation 7: Survey explanations for the frequent recurrence of *Game Graphics* bugs include: interdependence of graphics, unforeseen edge cases, and the specific nature of graphical components.

Because few respondents claimed to see frequently recurring *Triggered Event* bugs, there were not many explanations as to why these bugs might recur frequently over multiple updates. Still, some responses speak to the large size of the test space.

- *"Events can often be triggered in many different ways; making sure every way works is tricky and time-consuming."*

Observation 8: Survey explanations for the frequent recurrence of *Triggered Event* bugs include: large size of testing state space.

When asked if respondents had seen any other frequently recurring bug types, one respondent pointed to *Action* bugs, stating that these bugs recur often for the same reasons as *Crash* bugs. Other respondents spoke more broadly, identifying reasons why bugs in general might recur over updates.

- *"Sometimes bugs recur because some careless guy removes your fix when merging his/her code."*
- *"It has to do with whether management allows us the time for testing before they push out the newest code."*

C. Bug Severity

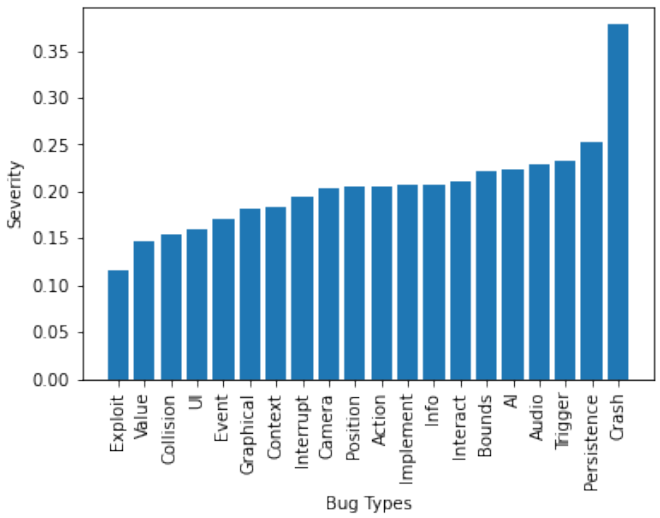

Fig. 3. Severity of Bug Types

We calculated the severity of each bug type by finding the proportion of bug fixes from the data that appeared in an update marked as urgent. For each bug type, we divided the frequency of the bug type in an urgent update by the frequency of the bug type in all updates, as expressed below:

$$BugTypeSeverity = \frac{BugTypeFreq(UrgentUpdates)}{BugTypeFreq(AllUpdates)} \quad (1)$$

Figure 3 shows the resulting severity rates of each bug type.

Observation 9: Based on the update data, the bug type with the highest severity *Crash*. The next severe bug types were *Object Persistence* and *Triggered Event*.

Crash bugs appeared 580 times across all updates and 219 times in urgent updates. 38% of the occurrences were in urgent updates, the greatest proportion for any bug type.

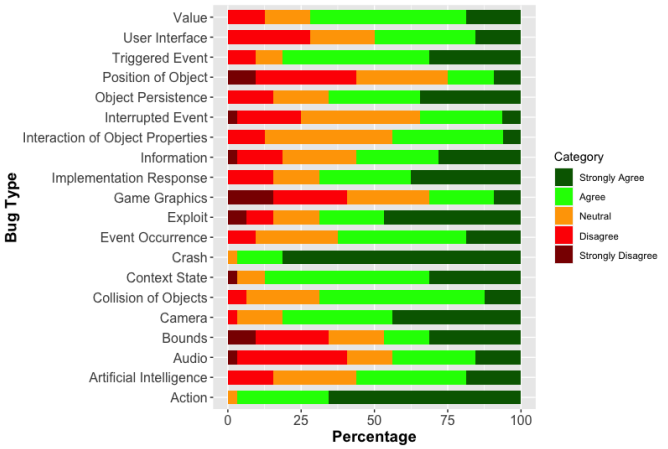

Fig. 4. Agreement Level to Whether Selected Bug Types are Likely to Have a Severe Negative Impact on Game Experience

Figure 4 depicts the perceptions of bug severity among survey participants. For each bug type *X*, respondents were asked to state the level of agreement with the statement "*X* bugs are likely to have a severe negative impact on the game experience."

Observation 10: The bug types that generated the largest number of *Strongly Agree* responses were *Crash*, *Action*, and *Exploit*.

With respect to *Crash* bugs, the results from the update data and from the survey appear to match; *Crash* bugs appear to be the most severe bugs in both data sources. *Action* bugs, the second-most severe type from the survey, do not rank as highly when looking at the mined data seen in Figure 3, however. Interestingly, *Exploit* was the bug type that received the third-most *Strongly Agree* responses, yet it had the lowest severity rate from the mined update data.

D. Bug Fix Association with Game Development Processes

Survey respondents were asked *"In your opinion, what aspects of game development (i.e. testing, designing, etc.) are associated with bug recurrence in your current project or in other game projects in which you participated?"* 16 respondents mentioned testing as a component of game development that had a strong association with bug recurrence. Some of these responses mentioned how a lack of testing in general was linked to bug recurrence, while other responses spoke about the lack of certain kinds of testing, such as automated testing, integration testing, and cross-platform testing.

- *"Lack of automated testing definitely increases bugs in general"*
- *"The testing only checks the current feature/bugfix and not how it interacts with everything else"*

Four respondents claimed that game design had a strong link to bug recurrence.

- *"Good design helps avoid bug recurrence in the first place. Testing can be used as a fail-safe"*

Four respondents pointed to code quality/coding as an important component connected with bug recurrence.

- *"Code quality is often a huge factor for bugs. Badly designed code interacts badly with other pieces of code, and conflicts can result easily, causing bugs."*

Other aspects of game development provided in the responses included the game architecture, quality assurance, a lack of consistency checks, and the addition of new features.

- *"In my experience, new gameplay features often lead to bug recurrence, when a previous bug was "solved" by addressing the particular feature interaction that caused the bug rather than making those features more robust in general."*

Observation 11: The aspects of game development most frequently linked to bug recurrence were: testing, game design, and code quality.

Finally, respondents were asked what they believed were the main challenges in identifying and fixing bugs in video game development. Like the previous question, 22 respondents provided answers. Eight responses pointed to challenges with conducting adequate testing.

- *"too many possibilities of interaction/combination to test them all thoroughly"*

Seven responses mentioned difficulties in identifying the origin of bugs or reproducing bugs.

- *"Reproducing a bug often requires a large time investment to reach the state in which it is apparent. Games specifically are huge loosely connected systems; bugs often depend on more than one of these systems having a certain state."*

Six responses focused on code quality. There was a particular emphasis on the importance of well-written code that does not cause conflicts elsewhere in the game.

- *"The hardest part is finding an appropriate solution to a bug that isn't hacky and is future-proof to later additions to the code that might break the bugfix again."*

Two responses mentioned development schedules.

- *"Lack of QA, faulty development schedule, low focus on code quality."*

Two responses mentioned unforeseen bugs that only appear in particular situations, such as bugs arising out of specific hardware setups.

- *"Bugs that occur only with some OSes or hardware"*

Two responses explained that the choice of programming language can lead to issues.

- *"Using languages with a weak memory model such as C++ makes it hard to identify bugs caused by memory corruption errors such as use-after-free."*

Observation 12: The most frequently mentioned main challenges to identifying and fixing bugs in video games were: inadequate testing, reproducing bugs, and code quality.

V. Discussion

A. For Researchers

Prioritizing of Bug Types: Our results identify which types of bugs appear in updates more frequently than others, which bugs recur over multiple updates more frequently, and which bug types are more severe in negatively impacting the game experience for players. When comparing frequency to severity, these results could help illuminate which types of bugs researchers might prioritize when devising techniques to prevent or fix certain types of bugs. For example, though *Information* bugs were the most frequently occurring bug type across all the update notes, this bug type was not as highly ranked with respect to recurrence and severity, as seen in Figures 3 and 4. The results seem to indicate that even if *Information* bugs are common in game updates, they do not seem to recur as often as other bugs, and they are not seen as having as much of a severe negative impact on the game experience. our results imply that *Crash* bugs could probably use a greater focus from researchers. Though these bugs were not the most frequent bug type in general, this was the bug type that had the highest recurrence rate and the highest severity rate in our data. Additionally, the survey results also indicate that this bug has a high level of severity. Using the findings from this paper, researchers can develop specialized tools and methods to specifically target the more problematic bug types.

Our survey results also provide some explanations as to why some of these certain bug types stand out more than others. From our results, we have found that some explanations appear multiple times and in multiple contexts for different bug types. The large size of the testing state space, for example, is provided as an explanation for why both *Game Graphics* bugs and *Action* bugs are fixed frequently in game updates and for why *Triggered Event* bugs recur frequently over multiple updates. A deeper investigation into these common causes of bug issues and approaches into remedying them might be a worthwhile subject for future research.

Game Development Processes: The findings regarding bug frequency, recurrence, and severity also have implications relating to the game development process. Testing, code quality, game design, and bug reproduction were all aspects of game development that seemed to have a strong association with bugs, according to the survey results. While some research exists on the testing process in game development [28], there does not appear to be much research into the other three processes within the game development context. Future research can focus on devising approaches and methods focused on improving these aspects of the game development process so that they are less likely to lead to the appearance of bugs.

For example, reproducing bugs was one of the most widely cited challenges in identifying and fixing bugs in video games. Reproduction is difficult, because bugs might be caused by a specific interaction of game states and properties that may not be evident to whomever experiences the bug. The lack of information about the bug's true cause leads to challenges in fixing the bug and a greater chance that the bug will reappear

in a future release. Indeed, this challenge with reproducing bugs was one of the major reasons from respondents for why *Crash* bugs recur frequently over updates. To combat this problem, researchers could focus on devising tools to better collect information about the game state so that developers are better equipped to find the true cause of these bugs.

B. For Practitioners

Prioritizing of Bug Types: The results can help game developers identify which types of bugs may need more attention than others. As described above, developers can evaluate the frequency, recurrence, and severity information from this paper to decide which bugs may need more attention when testing or fixing bugs.

Game Development Processes: Similar to researchers, game developers can take advantage of the results relating to aspects of the development process that have a link to the bug occurrence. For instance, the results suggest that difficulties in the testing process are linked with the appearance of bugs. While the size of the game state space can make comprehensive testing difficult, the responses also seem to indicate that testing may also be too narrow in scope. Some respondents mentioned issues where testing does not involve broader interactions with other components of the system. Furthermore, the interactions between system components was was given as a reason for the recurrence of *Crash* bugs. These results suggest that developers might benefit by adjusting their testing. The game space might be too large to test everything, but developers can focus on creating more tests targeting high-priority bug types. Additionally, they could emphasize using tests that incorporate multiple interacting components of the system, as opposed to testing components in isolation.

Code quality was also provided both as an aspect of game design associated with bug recurrence and as a challenge in identifying and fixing bugs in video games. Poor code quality increases complexity of the program, and it also can lead to conflicts with other components of the system, ultimately resulting in bugs. The interaction of code modules in games seems to be an important component of game development that is related to both the testing process and the coding process. Implementing practices to enforce higher code quality could end up benefiting both of these processes.

Our results point to different possible parts of the game development process associated with bugs. By implementing practices to improve these processes, developers might be able to mitigate the rate at which bugs appear or recur.

VI. Threats to Validity

With the intent to make sure our results could be generalized, we tried to collect a large amount of update notes from a high number of games. We collected the top 30 games with the highest player counts on the Steam platform in order to collect a sample of games that represents the range of video games currently available. It is possible however, that the games studied and the update notes analyzed are not representative to the entire body of game update notes.

It is possible that survey respondents may not have understood the different bug types when answering questions regarding their experiences with these bugs. To avoid this, we provided an explanation of each bug type in the survey that described the bug and included an example bug fix from the update notes that belonged to that category.

There was a risk that the bug types in our taxonomy might be influenced by bias or that they might not account for all types of bugs. To reduce this bias, we created a taxonomy that was based on prior work [17]. Additionally, the large number of bug fixes (12,122) we analyzed should help ensure that our final taxonomy is complete.

In identifying severe bug fixes in the update notes, there is a possibility that our mechanisms to identify urgent updates might not have captured the correct number of relevant updates. Our methods could have been under-inclusive or over-inclusive. To avoid this, we based our approaches off of approaches used in prior research [18].

Additionally, our script to identify the bug fixes in our update notes might not have identified all bug fixes. From our own manual analysis, we found common repeating words and phrases that were used in the lines containing bug fixes. We accounted for these terms when creating our script. In order to test the script's efficiency, we inspected the code of our script, ran tests with the script, and tested the script on update notes with known results. It is possible, however, that some bug fix lines may not have included these terms.

VII. Conclusions and Future Work

In this paper, we collected the update notes from popular games on the Steam platform. We created a taxonomy of bug types that we used to label the bug fixes included in the update notes. We found that the bug types that appeared the most frequently in update notes were *Information*, *Game Graphics*, and *Action*. The bug types that recurred the most often over multiple updates were *Crash*, *Game Graphics*, and *Triggered Event*, and the bug types with the highest rate of severity were *Crash*, *Object Persistence*, and *Triggered Event*. We surveyed game developers who corroborated that *Crash* bugs were likely to have a severe negative impact on the game experience. The survey responses also suggested that challenges in testing, bug reproduction, and code quality had a strong association with the occurrence of bugs in game updates.

These results can help game developers identify which types of bugs to pay more attention to when testing and fixing bugs. Developers can also use these results to help adjust practices related to game development process in order to better prevent, identify, and fix bugs.

Researchers can take advantage of these results in order to develop tools or methods that can target specific bug types that are more likely to severely impact the game. There is room for future work that can identify aspects of game development that might benefit from specialized tools or methods that address some of the challenges provided in the survey. Our study takes the first step towards fulfilling that goal.

References

[1] "Github," https://github.com/, accessed: Aug. 1, 2020.

[2] "Qualtrics," https://www.qualtrics.com/, accessed: Aug. 5, 2020.

[3] "Steam," https://store.steampowered.com/, accessed: Jun. 20, 2020.

[4] "Steam & game stats," https://store.steampowered.com/stats/, accessed: Jun. 23, 2020.

[5] "Wallpaper engine," https://store.steampowered.com/app/431960/Wallpaper_Engine/, accessed: Jul. 7, 2020.

[6] S. L. Abebe, N. Ali, and A. E. Hassan, "An empirical study of software release notes," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1107–1142, 2016.

[7] A. Ampatzoglou, A. Kritikos, E.-M. Arvanitou, A. Gortzis, F. Chatzi-asimidis, and I. Stamelos, "An empirical investigation on the impact of design pattern application on computer game defects," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 2011, pp. 214–221.

[8] M. Beckwith, "Apex legends game breaking revive bug returns in season 6," https://gamerant.com/apex-legends-season-6-revive-bug/, Aug. 2020, accessed: Aug. 24, 2020.

[9] Blue Mammoth, "Brawlhalla's new battle pass season 1 - patch 4.00," https://www.brawlhalla.com/news/brawlhallas-new-battle-pass-season-1-patch-4-00/, May 2020, accessed: Jul. 2, 2020.

[10] A. Borrelli, V. Nardone, G. A. Di Lucca, G. Canfora, and M. Di Penta, "Detecting video game-specific bad smells in unity projects," in *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea*, 2020.

[11] [DE]Megan, "Prime vault: Hotfix 25.7.5," https://forums.warframe.com/topic/1127427-prime-vault-hotfix-2575/, Sep. 2019, accessed: Jul. 11, 2020.

[12] ——, "Saint of altra: Update 25.7.0," https://forums.warframe.com/topic/1123841-saint-of-altra-update-2570/, Aug. 2019, accessed: Jul. 10, 2020.

[13] R. Gilliam, "Paper mario: The origami king has a game-breaking bug near the end," https://tinyurl.com/yydjp7ey, Jul. 2020, accessed: Aug. 28, 2020.

[14] K. Kelley, B. Clark, V. Brown, and J. Sitzia, "Good practice in the conduct and reporting of survey research," *International Journal for Quality in health care*, vol. 15, no. 3, pp. 261–266, 2003.

[15] I. Khan, "Bethesda facing possible class-action lawsuit over fallout 76," https://www.gameinformer.com/2018/11/27/bethesda-facing-possible-class-action-lawsuit-over-fallout-76, Nov. 2018, accessed: Aug. 22, 2020.

[16] S. Lantz, "Call of duty: Warzone demon gun glitch makes an unfortunate return," https://gamerant.com/call-of-duty-warzone-demon-gun-texture-glitch/, Aug. 2020, accessed: Aug. 25, 2020.

[17] C. Lewis, J. Whitehead, and N. Wardrip-Fruin, "What went wrong: a taxonomy of video game bugs," in *Proceedings of the fifth international conference on the foundations of digital games*, 2010, pp. 108–115.

[18] D. Lin, C.-P. Bezemer, and A. E. Hassan, "Studying the urgent updates of popular games on the steam platform," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2095–2126, 2017.

[19] E. Makuch, "Battlefield 4's rocky launch "absolutely" damaged player trust, producer says," https://www.gamespot.com/articles/battlefield-4s-rocky-launch-absolutely-damaged-pla/1100-6422805/, Oct. 2014, accessed: Aug. 20, 2020.

[20] F. Medeiros, M. Ribeiro, R. Gheyi, S. Apel, C. Kästner, B. Ferreira, L. Carvalho, and B. Fonseca, "Discipline matters: Refactoring of preprocessor directives in the #ifdef hell," *IEEE Transactions on Software Engineering*, vol. 44, no. 5, pp. 453–469, 2018.

[21] E. Murphy-Hill, T. Zimmermann, and N. Nagappan, "Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development?" in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 1–11.

[22] E. Narcisse, "More than a third of 2014's big-budget games got day-one patches," https://kotaku.com/more-than-a-third-of-2014-s-big-budget-games-got-day-on-1686398805, Feb. 2015, accessed: Aug. 27, 2020.

[23] L. Pascarella, F. Palomba, M. Di Penta, and A. Bacchelli, "How is video game development different from software development in open source?" in *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE, 2018, pp. 392–402.

[24] L. Passos, R. Queiroz, M. Mukelabai, T. Berger, S. Apel, K. Czarnecki, and J. Padilla, "A study of feature scattering in the linux kernel," *IEEE Transactions on Software Engineering*, 2018.

[25] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 155–165.

[26] P. Rosenmai, "Using the median absolute deviation to find outliers," http://eurekastatistics.com/using-the-median-absolute-deviation-to-find-outliers/, Nov. 2013, accessed: Aug. 17, 2020.

[27] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[28] R. E. Santos, C. V. Magalhães, L. F. Capretz, J. S. Correia-Neto, F. Q. da Silva, and A. Saher, "Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[29] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[30] J. Švelch, "Resisting the perpetual update: Struggles against protocological power in video games," *New Media & Society*, vol. 21, no. 7, pp. 1594–1612, 2019.

[31] E. J. Toy, J. V. Kummaragunta, and J. S. Yoo, "Large-scale cross-country analysis of steam popularity," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2018, pp. 1054–1058.

[32] ZOS_GinaBruno, "Pc/mac patch notes v6.0.5 - greymoor update 26," https://forums.elderscrollsonline.com/en/discussion/528633/pc-mac-patch-notes-v6-0-5-greymoor-update-26, May 2020, accessed: Jul. 8, 2020.