Why Are We Permanently Stuck in an Elevator?
A Software Engineering Perspective on Game Bugs

Iris Zhang
Columbia University, New York, NY, USA

Abstract — **In the past decade, the complexity of video games have increased dramatically and so have the complexity of software systems behind them [7]. The difficulty in designing and testing games invariably leads to bugs that manifest themselves across funny video reels on graphical glitches and millions of submitted support tickets [8] [12]. This paper presents an analysis of game developers and their teams who have knowingly released bugs to see what factors may motivate them in doing so. It examines different development environments as well as inquiring into varied types of game platforms and play-style. Above all, it seeks out how established research on software development best practices and challenges should inform understanding of these bugs. These findings may lead to targeted efforts to mitigate some of the factors leading to glitches, tailored to the specific needs of the game development team.**

I. Introduction

As a consumer of games, the author of this paper has had personal experiences with a game's buggy interface, from suddenly disconnecting in a middle of an intense multiplayer game to watching their player character fall seemingly forever. As a game developer, the author has seen her team finding bugs during the testing phase and letting some go while flagging others for immediate fixes. Some bugs seem to be "acceptable" to the team to have upon release but others are seen as dire and need to be addressed immediately. Why, if developers know of bugs in their code, do they willingly release glitchy games?

The reader will learn about the formal process for categorizing bugs in different types of video game development teams, gathering current research on current software development challenges, and categorizing games based on their context of team size and gameplay style. They will see a survey designed and the teams assessed using these parameters. One emphasis will be on different types of teams based on personal past experience working in various sized companies and games. Another factor that will undoubtedly affect the nature of bugs will be the structure of the gameplay and intended end platform. Most likely, smaller game development teams will not differ from larger established studio games in that both teams knowingly release bugs. Complex game systems backed by large studios may tend to suffer more from issues due to networking, game balance, and visual bugs. Smaller development teams may suffer across the board but may not tackle more complicated games and thus have less complex bugs. The paper will seek to confirm and expand upon this hypothesis.

II. Background and Related Works

Many books and articles exist that prescribe how one can debug their game or merely documents the existence of them. The web is rife with patch logs, humor reels, and articles describing the myriad of released glitches [8] [9]. The difficulty lay in finding existing research that delve deeply into the topic of video game bugs. One particular reading ("Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development?") provides a cursory answer to its titled question [1]. While it provided a unique view into game developers' personal opinions on how their work differs from traditional software development in the specific context of Microsoft Xbox studios, it did not perform a statistically rigorous survey across different types of game development or break down the types of games the respondents worked on. Another paper provided a detailed look at how Finnish game development teams approach testing before a game is released [4]. It outlined in detail the types of bugs that development teams will typically test for, as well as summarizing the different roles on the development team.

Also sought were works that would inform the survey question on game bug types and reason for their persistence. The paper "What Went Wrong: A Taxonomy of Video Game Bugs" broke down the types of video game bugs by specifying the subcategories *fault*, *error*, and *failure* [3]. While not lacking in complexity, the categories were ill-suited for the purposes of designing a game developer-facing survey so only the examples given in the different categories were used. A much better aligned work was used to design the survey question around concrete, software and game development-specific reasons things went wrong [2]. No shortage of web articles speculated as to the reason games seem to be so buggy these days [14][15]. Although they hypothesized issues such as shorter development cycles and more complex systems, all lacked the detail and rigorous empirical research needed to come to satisfying conclusions.

TABLE I: Terminology used in the Survey and Findings

Indie	Short for "independent." Refers to a type of studio owned by independent game developers. [10]
AAA	Known as "Triple A." Refers to a corporate-type game development team with large, hierarchical structure [10]
MMO	Massive Multiplayer Online. Games that allow a large number of players to participate simultaneously over an internet connection. These games usually take place in a shared world that the gamer can access after purchasing or installing the game software [16].
MOBA	Massive Online Battle Arena, similar to MMO in complexity
S*ingle-player*	Only one player can play the game at a time
Multiplayer/Co-op	Multiple people can play the game at once on a single or split screen, e.g. Mario Kart. Coop specifically refers to gameplay where players are put on one team and work collaboratively instead of versus each other.

NPC Non-player character. Refers to opponents or characters in the game controlled by AI.

III. Methodology and Survey Design

The goal of the survey is to understand why bugs are released by game development teams. It attempts to address several questions:

- *Do game developers knowingly release buggy games?* One should not just take as reality it is "now par for the course that games release with bugs and glitches" [15]. I must demonstrate that it is supported by empirical evidence.
- *Were there significant differences between types of bugs reported based on the types of games that developers worked on or the types of development team?* The second question aims at researching if self-reported bugs are more common based on differing contexts, such as intended end platforms or the size of the supporting game studio. It is also hypothesized that the nature of the bugs will differ based on these parameters.
- *Why did the bugs go unaddressed before release?* This final question gets to the heart of the issue, which asks in what contexts do bugs go unaddressed? It will attempt to validate common perceptions such as development schedule being a major factor. It will also examine different contexts as per the second research question to see if any significant differences arose between Indie and AAA teams, or the types of games being developed.

A. Protocol

I designed a 5 minute Google Forms survey with enough detail to glean what I needed. I asked the audience to consider a single game that they have released or helped release in the past 5 years when answering these survey questions. This was to narrow down the scope of the questions so that they would have the bugs of one specific representative game in mind, and not all the bugs they have ever seen in their career. It would also give specific context to the nature of these bugs in terms of type of game and development environment.

First, the respondent had to check that they fulfilled the qualifications for taking the survey. The first two questions were demographics-based, establishing the role of the survey taker in their team and the nature of their team. The second half dealt with their knowledge of released bugs, their nature, and the reasons for the bugs remaining. All questions were required except for the last two, which dealt with the types of bugs and the reasons why bugs went unfixed. If a respondent answered no to the question about releasing a game with known bugs, they could leave those two questions blank.

B. Audience

The survey was aimed at a technical audience, although one does not necessarily need to have extensive game development experience, merely enough knowledge of technical details of bugs to answer. The audience must have worked on a game developed recently (released in the past 5 years.) In disseminating my survey, I reached out across several social media platforms:

through the Reddit community /r/gamedev, a game development group on Facebook, my personal Twitter and Facebook account, and word of mouth through friends.

In total there were 34 respondents, with the majority choosing to answer questions about their experience on an Indie team (64.7%) and the remaining on a AAA team (35.3%), as seen in Table II. There is no hard and fast rule about what is considered Indie and what is considered AAA. In the industry these are loose and fast terms, but generally it is a difference in management approach, creative control, and/or finances [10]. “Indie,” industry colloquial for “independent,” gets its designation from teams that have few owners who usually work closely in development. Members of the team generally contribute to more than one area of the game, with the designer, the developer and the artist often blending roles or simply being a single role. Usually this means a general difference in size of development team, since Indie teams fall on the small side. AAA refers to large studios with deep pockets that often have specialized teams led by specific executives [10]. Large, complex games, usually 3D and with multiplayer elements offering hours of gameplay are often published by AAA studios, although certain large games such as League of Legends were published by indie developers before they became AAA [13]. After considering the format for this type of question from differing angle, such as size or management style, the survey left this question purposefully vague so as to not bias the respondent. They simply picked the best fit based on common industry knowledge.

TABLE II: Indie and AAA development teams surveyed

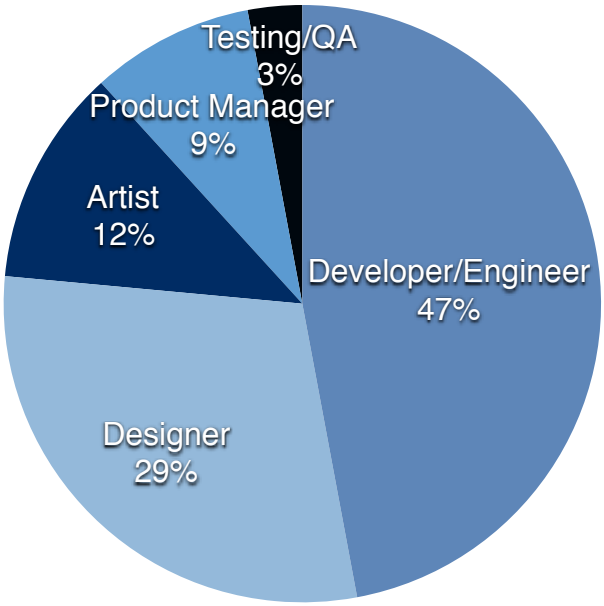
	Indie	AAA
What best describes your development team?	22	12

The survey also asked about the role of the survey taker on the development team (Appendix I, question 2.) The roles themselves are taken from a mixture of established prescribed roles for larger game studios as well as other research that had use for these categorizations [4] [6]. A brief description of each of these roles follows, but going into detail is out of this paper's scope.

- *Developer/Engineer*: This member is responsible for “creating the technical design document, building the game and delivering a high quality game that follows the requirements set in the game design document and the technical design document” [4]. The focus of this survey will be in the responses of developers.
- *Testing/QA:* This role is “responsible for developing the testing plan and managing the testers during a game project” [4]. For most smaller Indie studios, quality assurance may be handled by everyone on the team, and not as a separate role.
- *Designer:* The design team typically consists of “level designers, scripters, interface designers, writers, researchers and game tuners” [7].
- *Artist:* This member works with visual aspects of the game, from illustrations to menu interfaces to button textures.

- *Product Manager/Producer:* "Their job is to report progress, … solve different issues as they arise and to ensure that the game is done on schedule, within budget, and as close as possible to the vision presented in the game design document" [4].
- *Audio/Music Producer:* They are "responsible for all of the audio in the game from sound effects, music to dialogue" [4].
- *Other:* I included this free-form category for survey respondents who felt a single category did not accurately represent their role.

Figure 1: Role of survey respondents in game development team



By far the biggest plurality were developers or engineers (47%), with designers a large portion of the rest (29%). A small number were artists, product managers, and one respondent was a member of the QA team. One also identified as Other and clarified they were a mix of artist and developer, but I grouped them with the developers for purposes of analysis (Figure 1).

C. Data Specification

The scope of identifying my target audience was not limited to demographics. I also asked about the types of games that were developed in order to narrow down their context (see Appendix I, Full Survey, question 4.) While games have a myriad of categories [6] [17], all categories presented as options on the survey were picked to pinpoint the differences of developing for types of games on based on three criteria:

1) *Platform*. The platform refers to the end platform that the user will play the game on. While hybridization tools and game engines such as Unity have made it easier for developers to have one codebase while releasing for multiple platforms, the difference between console, desktop/laptop and mobile games is still vast by nature of their differing UI and native needs.
2) *Framework*. 2D and 3D games were considered on the basis that they face differing needs in terms of testing and fixing glitches. While completely text-based multiplayer online games do exist, they fall outside the scope of this paper.

3) *Networking needs and complexity of gameplay.* While not a perfect indicator of complexity, whether a game is single player or massive multiplayer should give clues as to how the complexity of state must be within a game at a given point in time.

The second consideration to data specification was to the nature of the bugs surveyed. The survey taker was asked to consider the bugs released as a whole for the singular game so as to not bias them towards choosing a bug that they had personally worked on, or favoring a particular bug that they wish had been addressed. The categories themselves were narrowed down from numerous sources, of which the Taxonomy paper had emphasis on 3D complex games with networking elements. The purpose of the paper was to help establish abstract types of game bug categories to help guide human testing and "provide a framework to validate the coverage of new testing paradigms that may emerge" [7]. As a result, there was much more emphasis on the technical root cause of the bugs themselves, and not how they manifest to the player. The survey designed for the purposes of this paper did take into consideration the category of bugs related to Information and Actions, as well as the concrete examples they gave in Table 1 and 2. (Action refers to bugs in which something a player can normally do or access cannot be done.) In contrast the Lahti paper offered some broad category of bugs that were more accessible to the end user, such as Visual, Audio, Artificial Intelligence, Stability, Performance, Compatibility and Physics, all of which made its way into the final survey [4]. It also helped that it provided some subcategories for those broad categories which allowed the survey to provide extra detail in helping the survey taker determine the best choice to describe their type of bug. A full list of categories of bugs can be seen in Appendix I, question 6.

The final question related to the reasons bugs went unfixed was designed mostly around a paper called "What Went Wrong? A Survey of Problems in Game Development." The premise was to collect data on game bugs from game postmortems (document that summarizes the project development experience) and "exploring their similarities and differences to well-known problems in traditional information systems" [12]. Therefore, I found it very useful to take from this paper the reasons bugs may go unfixed from both a software engineering perspective as well as a narrower game development scope. The problems of *scheduling*, *budget*, and *management* (later categorized in the survey with *communication*) were common to both industries so they were included as reasons that could be chosen [12]. Among unique problems in the game industry, *unrealistic scope* and *feature creep* seemed to have the most relevance to software development, as well as *technology tools*, *testing/QA*, and *failures in 3rd party APIs*. There were some categories, such as *crunch time* and *lack of documentation* which I felt either fit better as a subcategory or were altogether too narrow. Overall, there were 10 categories that survey takers could select from that identified reasons bugs went fixed, as well as an Other category where they could fill in the blank. To see more detailed descriptions and examples of these reasons, see Appendix I, question 7.

IV. Analysis of Results

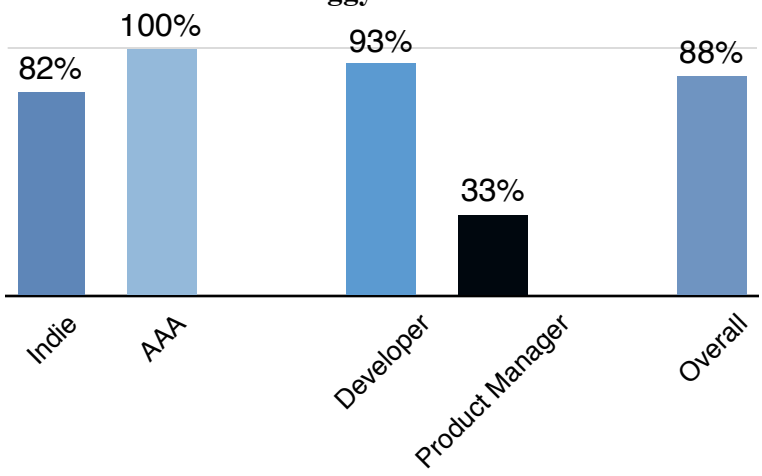
Here I revisit the questions proposed in the methodology. Do game developers knowingly release buggy games? Did that change based on what role that developer played, or what kind of

team they were on? Were there significant differences between types of bugs reported based on the types of games that developers worked on or the types of development team? I evaluated this in the context of the types of games, contrasting 2D with 3D games, across multiple platforms, and also game complexity using format of players. I also evaluated the same question looking at indie versus AAA developer. Why did the bugs go unaddressed before release? This question was addressed looking at aggregate data across the board, as well as team type and the role of the respondent on the game team.

A. The Modern Reality of Buggy Games

The overwhelming majority of survey respondents said yes, they released bugs knowingly. However, a breakdown of this answer by the role of the survey taker and their team reveals some differences in how developers and other members of the team may perceive the glitches of their game. What immediately stood out was that the majority of product managers denied releasing buggy games (67%), which goes against the overwhelming majority of developers, designers and artists. Of interesting note, 100% of AAA team members said they released buggy games, whereas only 82% of indie developers did. Either AAA studios are more aware of the bugs they release compared to their indie counterparts, or some other compounding factor inflates those numbers. Certainly, this seems to confirm one article's speculation that releasing buggy games has become the norm among studios [14]. One lone indie developer stated they released a non-buggy game, but later in the survey clarified that the bug was unknown until after release, when customers pointed it out.

Figure 2: Percentage of Game Developers Admitting to Buggy Games



B. Platform, Format, and Gameplay Wars: Nobody Wins

To accurately assess the prevalence of certain bugs, I pulled aggregate data on number of times types of bugs were reported for each type of game. Then I mapped it to the total number of games of that type reported, giving a percentage. For example, of all twelve 2D games surveyed, only three reported having bugs related to performance, giving it a 25% occurrence for that game type. I had suspected that there will be many overlaps between the types of games and bug types, but there were noticeable differences (see Table 3). In aggregate it seems as though bugs related to *performance* and *stability* were the most prevalent upon release, with about half of all

respondents identifying them as being issues. This was a bit unexpected as these are not among the bugs most complained about among game players. One would think that bugs of such nature are given priority and addressed prior to release, although such bugs may be harder to test for and address from a technical standpoint.

Table III: Percentage of aggregate game bugs over game types

	Graphics	Assets	Audio	Physics	Information	AI	Action	Networking	Performance	Stability	Compatibility
2D	50.0%	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	41.7%	33.3%
3D	33.3%	20.0%	6.7%	20.0%	6.7%	26.7%	26.7%	40.0%	66.7%	60.0%	6.7%
Mobile	33.3%	33.3%	6.7%	33.3%	6.7%	0.0%	0.0%	13.3%	40.0%	40.0%	20.0%
Console	12.5%	0.0%	12.5%	0.0%	12.5%	37.5%	37.5%	37.5%	37.5%	50.0%	0.0%
PC	43.8%	18.8%	18.8%	18.8%	18.8%	37.5%	37.5%	31.3%	56.3%	50.0%	25.0%
Single	35.3%	11.8%	11.8%	11.8%	11.8%	35.3%	35.3%	11.8%	47.1%	47.1%	17.6%
Multi	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	25.0%	50.0%	50.0%	62.5%	12.5%
MMO	50.0%	50.0%	0.0%	50.0%	0.0%	50.0%	50.0%	50.0%	75.0%	50.0%	0.0%
Overall	37.5%	25%	9.4%	21.9%	18.8%	28.1%	12.5%	18.8%	46.9%	50.0%	15.6%

When looking strictly at 2D and 3D games, I found it surprising that *performance* and *stability* were significantly worse for 3D releases than 2D ones. I would have thought much higher instances of graphics, assets, or physics related bugs for 3D, but 2D seems to have just as much if not more. In addition, about a third of 2D games suffered from compatibility issues whereas only 1 of the 15 3D games identified did. Since many 3D games these days are made with game engine tools such as Unity or Unreal allowing ease of transport to multiple platforms, it may have contributed this low rate. It's unclear why 2D games have a higher rate of compatibility issues given the prevalence of 2D game tools equivalent to Unity or Unreal.

In examining releases to different platforms, not much difference existed in the prevalence of performance or stability bugs, which hovered in the 40's and 50's percentage across the board. Of note none of the console games surveyed had any asset nor compatibility issues. While the lack of asset bugs seems baffling, I can see why games designed explicitly and natively for the console would not suffer from compatibility, especially if it's only released for one platform. In addition console games had a 50% occurrence of physics-related glitches, which is double that of PC/Mac and dwarfs mobile's. While I would have had to delve deeper into

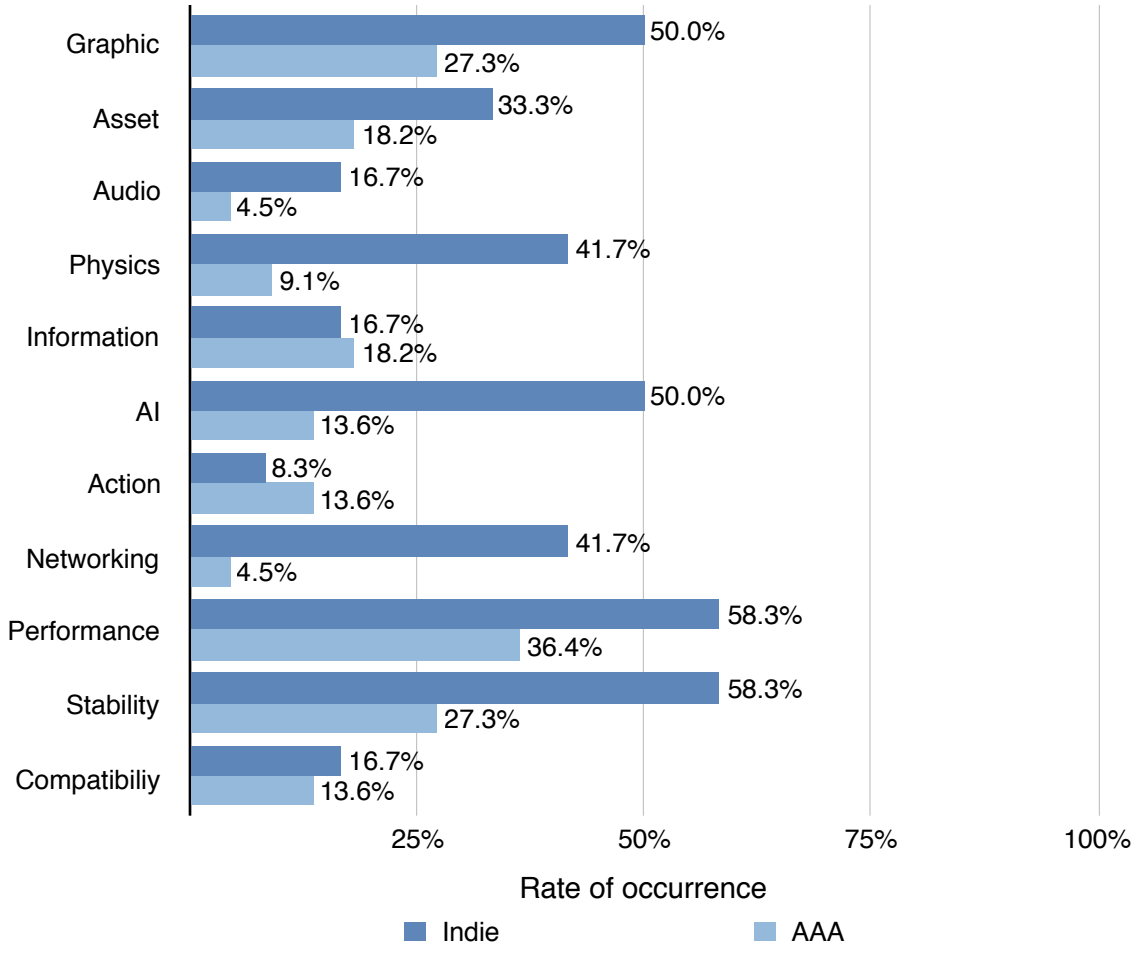
game types to understand what causes this definitively, one can see why console and desktop games would lend itself better to more complex, 3D games rather than mobile.

In the land of gameplay format, only single-player vs. multiplayer vs. MMO were considered. One would expect that MMO's have the highest rates of bugs given their complexity, and this reflects itself in the data. The single highest aggregate rate of bugs for any given category was performance in MMO's (75%), followed by stability in Multiplayer (62%). While single player games did have lower rates of bugs compared to their counterparts, they made a significant showing in the rate of performance and stability bugs as well, hovering right around the overall average. The only single higher bug rate for single player was in the existence of graphics-related bugs.

C. Indie vs AAA: Final Showdown

Overall the data confirms that bugs related to networking, performance, and stability were higher for AAA teams than indie, which would make sense as they attempt to publish more ambitious game systems. However, this belies the trend that AAA teams suffered from higher prevalence of bugs across all categories except for Information and Action, where the rates were comparable. Regardless, both types of teams released bugs across all categories (Figure 4).

Figure 4: Bug Types, Indie vs. AAA

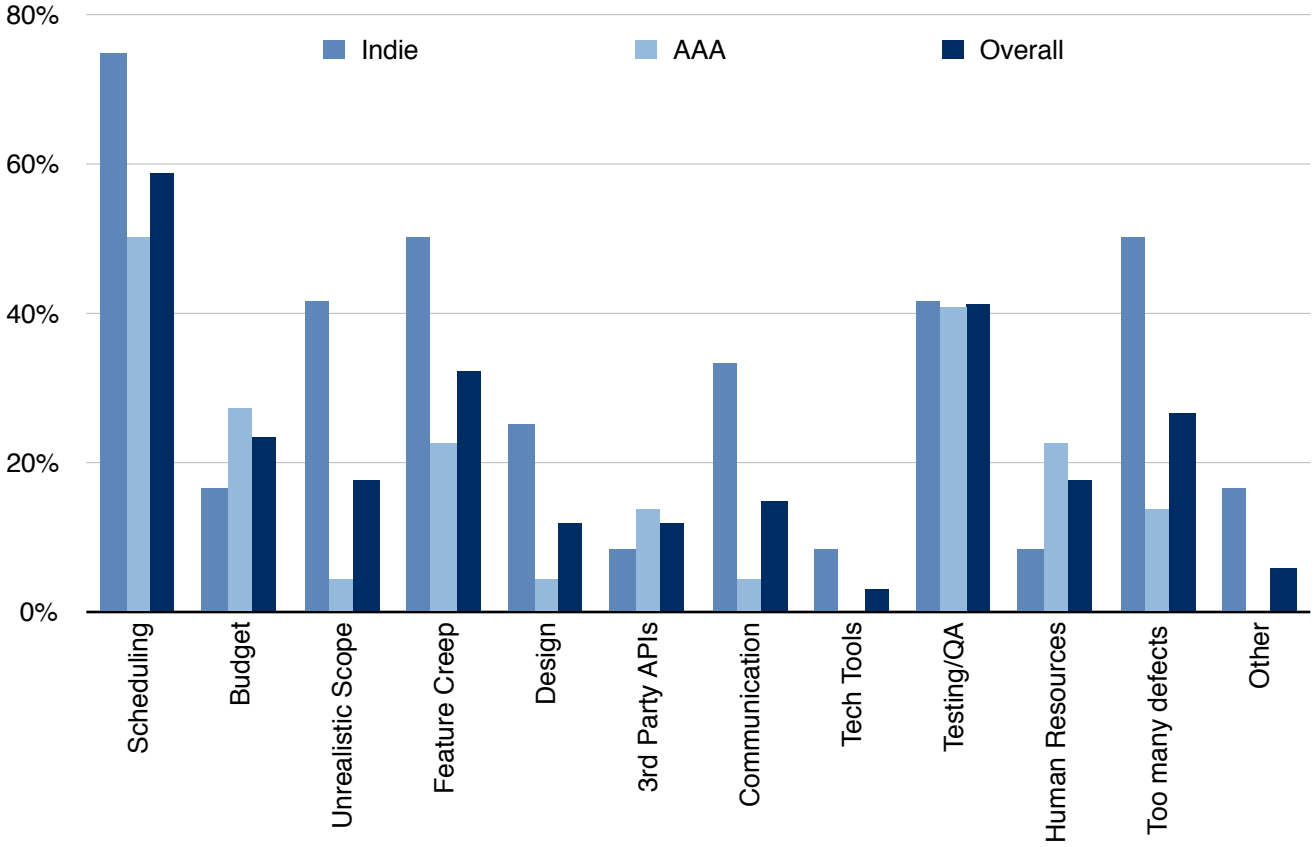


Of note, Physics, AI, Networking , Audio, and Stability bug rates for AAA were more than double than those for indie. Information, Action, and Compatibility bugs were more or less the same for both.

D. A Bug's Causes

Why do game developers knowingly release buggy games? As the data tells us, reasons are numerous but somewhat predictable: scheduling and lack of adequate time and effort given to testing and quality assurance were the top one and two reasons overall, respectively. This aligns with previous surveys that point out that "crunch time"—an arbitrary date a game has to ship either for production reasons or to be on budget— and delays leads to having to bring an imperfect product to market [2]. While I expected budget to make it into the top reasons, it was behind feature creep and the abundance of defects, which were the next highest rate. There were much smaller occurrences of companies that pointed out problems in the design phase or communication, which I had thought might have contributed to more bugs given that common knowledge leads one to believe that more complex systems contribute to more chances of faulty code [15].

Figure 5: Reasons for Bugs, Indie vs. AAA vs. All



When broken down by team type, the reasons for game glitches reveals more about the challenges faced in these unique environments. Both indie and AAA teams shared scheduling as the primary reasons bugs went unfixed. However, indie teams suffered from feature creep and too many defects as their second highest reason, perhaps because poor planning of code architecture from the onset lead to confusion over which issues should be given priority when it came down to game release. This makes sense given the higher instance of "design" and "unrealistic scope" issues cited by indie teams. By contrast AAA teams had feature creep as their forth most commonly cited cause of bugs and a very low rate of design-related and unrealistic scope problems, so while poor code architecture was a factor, poor planning or lack of vision as a whole didn't seem to affect these teams as much.

Interesting of note is that indie teams suffered from scheduling issues at a significantly higher rate than their AAA counter parts, even though common knowledge dictates that due to their independent nature they should not be beholden to outside influences such as a board of directors or third party publishers [10]. In addition, AAA teams suffering more from budget issues seems to directly counter the claim that AAA teams have "deep pockets" [10]. One possible scenario that leads to this contradiction might be that many indie developers put their livelihoods on the line when making games and prioritize getting more games to market rather than thinking of the money they could they making doing another job as a "budget" issue. Indie teams also suffered from communication problems and not enough human resources, which given the less hierarchical nature of such teams may mean less direction from project managers and leaders [10].

Both teams cited time and effort given toward testing as being a significant reason for buggy games (around 40%.) Given that some AAA teams have dedicated testing units, it's unclear why this rate is so close to that of indie teams'. Perhaps telling, the single respondent who identified as being from QA did not cite testing as a reason for bugs.

V. Limitations and Looking Ahead

Given the paper we read in class ("Cowboys") much effort was taken to avoid falling into the same trap of doing a very broad, cursory survey from which only flimsy conclusions could be drawn [1]. Nevertheless this paper suffers from the same sampling bias due to the limited number of respondents and the self-selecting nature of those who took the survey. One cannot confidently say that those who took this survey accurately reflect the game industry as a whole. Another concerns is the lack of survey respondents were not enough in certain categories. For example, to draw conclusions about MMO games as a whole given that only 4 games of that type were identified seems misleading.

There were also weaknesses in survey design. Some of the questions may have confused survey takers in their wording. Although care was taken to cover all types of game bugs, some filled in the "Other" category with bugs related to UI or public API calls, showing that not all the categories covered its scope in entirety. I suspect that many also confused the Asset bug category with Graphics/Vision and Audio, given the limited examples supplied. Overall, the bug categories could have been better specified and fine-tuned so as to not muddy the data. Question 4 could have better addressed the type of games that the survey respondent released (Appendix

I.) Instead of being a *check all that applies* question, it would have been better if the survey were designed with the three distinct axis along which analysis would be performed (platform, gameplay, framework) in mind. If I could go back in time, this would be a three-part question with multiple choice answers so that individuals must pick from all different specifications for their game. Thus entries where only "2D" was picked as a game category would be populated with much better data, as this does not reveal the game's platform or networking capabilities, although every game does possess such.

Further research would delve deeper into the types of testing done at game development companies, although the Finnish game paper does go into quite a bit of depth about this topic [4]. It would stand to benefit from performing analysis across types of tests with differing categories of games and game teams. Another direction to go in would be to see how game companies approach addressing bugs after release, and whether or not a formal process for submitting support tickets from players correlates to the prevalence of game bug release rates.

VI. Conclusion

Game developers are faced with many dilemmas these days to produce a perfect product. On one hand, regardless of whether their team is independent or AAA, they face tremendous pressure to bring their product to market in a timely manner. On the other hand, they face scrutiny from players and journalist who publicly complain about and document their buggy games for the world to see [11] [12] [14] [15]. I learned that in the fight to balance the two, glitches win out. Through a survey disseminated to self-identified game developers, I determined that game teams overwhelmingly release buggy games, and they do so knowingly. The developers cite scheduling as a primary reason, followed by lack of dedicated testing time and adding additional features without proper planning of software architecture.

When analyzing the challenges faced by game teams, differences in the prevalence of different bug types and the reasons for their existence became apparent. In general I learned that performance and stability related bugs occurred regardless of game type, but as games becomes more complicated (3D, MMO elements, multi-platform) bugs such as networking or compatibility become more common. In addition, indie teams tend to suffer more from issues related to lack of vision and leadership, such as poor project scoping and feature creep. Both types of teams could benefit from a longer, more fine-tuned testing phase dedicated to game polish. Product and project managers could stand to understand their audience better in the quality of games when deciding upon a schedule for release. Although in the case for certain studios with earned reputations for glitches and doggedly refuse to increase their teams and budget [11], it seems as though players must continue to coexist with bugs.

Works Cited

[1] E. Murphy-Hill, T. Zimmermann and N. Nagappan, “Cowboys, Ankle Sprains, and Keepers of Quality: How Is Video Game Development Different from Software Development?” Proceedings of the 36th International Conference on Software Engineering (pp. 1-11). ACM May 2014.

[2] F. Petrillo, M. Pimenta, F. Trindade and C. Dietrich, “What went wrong: A survey of problems in game development,” Comput. Entertain., vol. 7, Feb 2009.

[3] R. Ramadan and Y. Widyani, “Game Development Life Cycle Guidelines,” ICACSIS 2013.

[4] M. Lahti, “Game Testing in Finnish Game Companies,” Master’s Thesis, Aalto University. School of Science. November 2014.

[5] A. Ampatzoglou and I. Stamelos, "Software engineering research for computer games: A systematic review,” Information and Software Technology, vol. 52, no. 9, pp. 888-901, 2010.

[6] H. M. Chandler, The Game Production Handbook, 2008. Second edition, Hingham, Mass.: Infinity Siene Press.

[7] C. Lewis, J. Whitehead, and N. Wardrip-Fruin. “What Went Wrong: A Taxonomy of Video Game Bugs.” Proceedings of the Fifth International Conference on the Foundations of Digital Games. Pages 108-115. ACM. 2010.

[8] Author Unknown. Patch Notes, Riot Games. [Web]. http://na.leagueoflegends.com/en/news/game-updates/patch/ Visited March 2016.

[9] C. Plante. “How not to get permanently stuck in Fallout 4’s elevator like this schmuck.” The Verge. TL;DR. 9 November 2015. [Web]. http://www.theverge.com/2015/11/9/9696186/fallout-4-bugs Visited March 2016.

[10] Author Unknown. “Anonymous' Answer to What is the difference between working for an indie gaming studio and working for an AAA gaming company?" Quora. 30 October 2014. Web. https://www.quora.com/What-is-the-difference-between-working-for-an-indie-gaming-studio-and-working-for-an-AAA-gaming-company Visited 12 March 2016.

[11] McClendon, Z. “Fallout 4 is Full of Bugs, but Fixing Them Could Ruin It.” Wired. 18 November 2015. [Web]. http://www.wired.com/2015/11/fallout-4-bugs/ Visited March 2016.

[12] McVinnie, James. The beauty of a GTAV bug. 14 April 2015. [Video file]. Retrieved from https://www.youtube.com/watch?v=plKRcrDTd50

[13] Brice, K. “Blizzard developers join Riot Games’ online title.” 25 June 2009. [Web]. http://www.gamesindustry.biz/articles/blizzard-developers-join-riot-games-online-title Visited March 2016.

[14] Makuch, E. “Why are some games so buggy?” 6 March 2014. [Web]. http://www.gamespot.com/articles/why-are-some-games-so-buggy/1100-6418151/ Visited March 2016.

[15] Birch, Aaron. “Have bugged video games now become the norm?” 14 January 2015. [Web]. http://www.denofgeek.com/games/bugged-videogames/33602/have-bugged-videogames-now-become-the-norm Visited March 2016.

[16] Author unknown. “Massively Multiplayer Online Game (MMOG).” Def. 1. *Technopedia Online.* [Web]. https://www.techopedia.com/definition/27054/massively-multiplayer-online-game-mmog Visited March 2016.

[17] J. Hight and J. Novak, *Game Development Essentials.* Game Project Management, New York: Delmar. 2007.

Appendix I. Survey

Game Developers and Bugs Survey

This survey's intended audience is for anyone with at least .5+ years experience developing or designing games in a professional setting. You must have published a game within the past 5 years. Although you are not required to have experience developing the game, you must have enough knowledge about the technical details of the game itself to answer questions related to it. The results of this survey will be completely anonymized. If you have any questions, please contact the survey designer at iz2140@columbia.edu.

1. I have read the description and agree. *
   [] Yes

2. Consider a game you released within the past 5 years. Think of the one that you know the most technical details about, and were most closely involved with. What was your main role on the game development team? *
   - Developer/Engineer
   - Testing/QA
   - Designer (level, gameplay, map etc.)
   - Artist
   - Audio/Music Producer
   - Product Manager/Producer
   - Other: [fill in the blank]

3. Pick the description that best described your game development team.*

(help text) : Indie refers to small game companies where the owners typically are involved in the development. AAA are typically larger companies with hierarchical structures. Owners may not necessarily be involved in designing or developing.

   - Indie
   - AAA

4. What was the nature of this game you worked on? Check all that apply.*
   - MOBA/MMO or any game with massive multiplayer elements
   - 2D
   - 3D
   - Mobile
   - Console
   - PC/Mac
   - Single-player
   - Multiplayer or Co-op (4 max)
   - Other: [fill in the blank]

5. Have you or your team released this game with known bugs/glitches? * (Yes/No)

6. What were the nature of these bugs? Check all that apply.
   - ☐ Graphical/Visual (clipping, z-position mismatch, missing textures)
   - ☐ Audio (skipping, distortion, audio drop)
   - ☐ AI (NPC bugs, pathfinding bugs)
   - ☐ Performance (Installation bugs, loading time, frame rate)
   - ☐ Stability (crashes, freezing)
   - ☐ Networking (Lag, dropped connections, invisible players)
   - ☐ Compatibility (Operating System or console-specific bugs)
   - ☐ Information-related (presented out of order, missing, or invalid presented to player)
   - ☐ Physics
   - ☐ Action not possible or not allowed
   - ☐ Asset (missing or wrong art, missing or wrong sound effect, audio/text mismatch, typos)

7. What are the reasons the bug went unfixed? Check all that apply.
   - ☐ Scheduling (e.g., underestimating how long features take, having to meet a hard deadline)
   - ☐ Budget
   - ☐ Unrealistic Scope (game was too ambitious from the onset)
   - ☐ Feature Creep (New features are added without planning software architecture)
   - ☐ Problems in Design Phase (Design was unrealistic or impossible to implement from technology standpoint)
   - ☐ Failures in 3rd party APIs for platform or hardware
   - ☐ Communication problems - (e.g., management, cross-team collaboration, etc.)
   - ☐ Technology tools: (e.g.developers did not use version control or used it incorrectly, build times too high)
   - ☐ Testing/QA (Testing was not robust enough or not given enough time nor emphasis)
   - ☐ Loss or lack of human resources (Not enough people with experience who have actually released a game, not enough developers or artists.)
   - ☐ Too many defects (Bugs were just too numerous to fix all)

*Required questions

Appendix II. Game Developer Role and Buggy Games, Raw Data

	No	Yes	Grand Total
Artist		4	4
Designer (level, game play, map etc.)	1	9	10
Developer/Engineer	1	14	15
Mix art, animation and programming		1	1

	No	Yes	Grand Total
Product Manager/Producer	2	1	3
Testing/QA		1	1
Grand Total	4	30	34

Appendix III. Game Types and Game Bug Types, Raw Data

	Graphic	Audio	Asset	AI	Performance	Stability	Networking	Compatibiliy	Information	Physics	Action
Mobile	5	1	5	0	6	6	2	3	3	1	1
Console	1	1	0	3	3	4	3	0	1	4	2
PC/Mac	7	3	3	6	9	8	5	4	3	4	2
Single Player	6	2	2	6	8	8	2	3	4	4	3
Multi	2	2	2	2	4	5	4	1	2	4	1
MMO	2	0	2	2	3	2	2	0	0	1	0

Appendix IV. Game Team Types and Game Bug Types, Raw Data

	Graphic	Asset	Audio	Physics	Information	AI	Action	Networking	Performance	Stability	Compatibiliy
AAA	6	4	2	5	2	6	1	5	7	7	2
Indie	6	4	1	2	4	3	3	1	8	6	3

Appendix V. Game Team Types and Bug Reasons, Raw Data

	Scheduling	Budget	Unrealistic Scope	Feature Creep	Design	3rd Party APIs	Communication	Techn tools	Testing/QA	human resources	Too many defects	Other
AAA	9	2	5	6	3	1	4	1	5	1	6	2
Indie	11	6	1	5	1	3	1	0	9	5	3	0
Overall	20	8	6	11	4	4	5	1	14	6	9	2