applied sciences

Article

Game Development Topics: A Tag-Based Investigation on Game Development Stack Exchange

Farag Almansoury *,†, Segla Kpodjedo *,† and Ghizlane El Boussaidi †

École de Technologie Supérieure (ÉTS), University of Quebec, Montreal, QC H3C 1K3, Canada
* Correspondence: farag.almansoury.1@etsmtl.ca (F.A.); segla.kpodjedo@etsmtl.ca (S.K.)
† These authors contributed equally to this work.

Abstract: Video-game development, despite being a multi-billion-dollar industry, has not attracted sustained attention from software engineering researchers and remains understudied from a software engineering perspective. We aim to uncover, from game developers' perspectives, which video game development topics are the most asked about and which are the most supported, in order to provide insights about technological and conceptual challenges game developers and managers may face on their projects. To do so, we turned to the Game Development Stack Exchange (GDSE), a prominent Question and Answer forum dedicated to game development. On that forum, users ask questions and tag them with keywords recognized as important categories by the community. Our study relies on those tags, which we classify either as technology or concept topics. We then analysed these topics for their levels of community attention (number of questions, views, upvotes, etc.) and community support (whether their questions are answered and how long it takes). Related to community attention, we found that topics with the most questions include concepts such as *2D* and *collision detection* and technologies such as *Unity* and *C#* , whereas questions touching on concepts such as *video* and *augmented reality* and technologies such as *iOS*, *Unreal-4* and *Three.js* generally lack satisfactory answers. Moreover, by pairing topics, we uncovered early clues that, from a community support perspective, (i) the pairing of some technologies appear more challenging (e.g., questions mixing *HLSL* and *MonoGame* receive a relatively lower level of support); (ii) some concepts may be more difficult to handle conjointly (e.g., *rotation* and *movement*); and some technologies may prove more challenging to use to address a given concept (e.g., *Java* for *3D*). Our findings provide insights to video game developers on the topics and challenges they might encounter and highlight tool selection and integration for video game development as a promising research direction.

Keywords: video-game development; empirical study; stack exchange

check for updates

Citation: Almansoury, F.; Kpodjedo, S.; El Boussaidi, G. Game Development Topics: A Tag-Based Investigation on Game Development Stack Exchange. *Appl. Sci.* **2022**, *12*, 10750. https://doi.org/10.3390/app122110750

Academic Editors: Robertas Damaševičius, Sanjay Misra and Bharti Suri

Received: 17 July 2022
Accepted: 20 September 2022
Published: 24 October 2022

1. Introduction

The video-game industry is one that has been thriving and growing, from around USD 1B in revenue in 1971 to USD 43B in 2000 and USD 180B in 2020 (https://www.marketwatch.com/story/videogames-are-a-bigger-industry-than-sports-and-movies-combined-thanks-to-the-pandemic-11608654990 (accessed on 1 October 2021)). The field has steadily grown and evolved with technology: from arcades dominating the 1970s and 1980s to game consoles, handheld devices, and PCs becoming the main platforms in the 1990s and early 2000s. Since 2011, mobile devices are the dominant platform; they represent 48% of the gaming market, followed by Personal Computers (25%), game consoles (19%), and emerging technology such as *Virtual Reality* headsets (3%). Despite such growth and dynamism, the video game industry has yet to receive appropriate and lasting levels of interest from the software engineering research community [1,2].

Prior research from the software engineering community has investigated possible differences between the development of video-games vs. non-video-game software products. Video-games have distinctive features that make them complex to design and test.

They require developing teams with very specific and diverse skills, and their inherent complexity can result in defects which negatively impact players' experience [3,4].

Furthermore, unlike some other types of software projects (e.g., Customer Relationship Management or Enterprise Resource Planning software), game development attracts many indie developers, many of whom have only a partial view of what it entails (https://www.gamesindustry.biz/articles/2018-10-02-its-time-we-stopped-encouraging-indies (accessed on 1 October 2021)). Game development involves diverse skills and a wide variety of concepts to master and technologies to use. Game developers, especially independent ones, thus have to navigate a wide range of concepts (e.g., sound, physics, animation, gameplay, etc.) and technologies (e.g., game engines, libraries, programming languages) in order to build their products. In this context, question and answer forums are privileged places where game developers can find solutions or information relevant to their sometimes unique challenges. One such prominent forum is the Game Development Stack Exchange (GDSE), from the well known the Stack Exchange network of Q&A websites. Much like the more well-known Stack Overflow, GDSE—for which the target audience is "professional and independent game developers"—allows its users to ask questions to which other users may respond. The questions on GDSE are often tagged with keywords that provide crucial information (on the issue, the circumstances, the development environment etc.), which can be analysed to provide insights as to which topics (concepts and technologies) are relevant or challenging for game development.

The present paper reports on the results of an empirical study of GDSE questions that aims at investigating topics (concepts and technologies) encountered in video-game development. In our study, we consider that tags represent topics, and we aim to answer the three following questions:

RQ1: Which video-game development topics received the most community attention? An answer to this question could help developers identify topics they may have to handle on their game development projects. In our study, we measure community attention through metrics related to the number of questions, views, upvotes, downvotes, and bookmarks associated with a tag in GDSE.

RQ2: Which video-game development topics receive the most/least community support? An answer to this question could help developers identify early the topics for which they are more/less likely to receive community help. In our study, we measure community support through metrics related to whether a question receives an answer that is accepted by the person who asked it and how long it took to receive said answer.

RQ3: Which combinations of video-game development topics receive the most/least community attention and support? An answer to this question could inform the choices of game developers, from a community support perspective, when it comes to (i) selecting the "best" technology for a given concept, (ii) selecting the "best" technology to add to their existing technology stack, and (iii) obtaining insights on the challenges raised by the handling of two concepts. For this question, we reuse metrics from RQ2, with a focus on the differences of support when a topic is mixed with another.

To the best of our knowledge, our study is the first to uncover and analyse the common issues reported by video-game developers using Q&A websites. The results of our study provide an empirical look at the landscape of video game development topics as they are perceived and handled by GDSE, one of the most prominent video-game development communities. We found, through our first research question, that prominent topics/tags include frameworks and languages such as *Unity*, *C#*, *OpenGL*, *GLSL*, *Android*, *Game-Maker*, and *Lua* but also concepts such as *2D*, *collision detection*, *mathematics*, and *game design*. Through our second research question, we found that, on average, half the questions are not satisfactorily answered (no accepted answer by the asker), and when they are, it generally takes between 3 and 6 h. Finally, our third research question considered pairs of tags and investigated whether questions with a particular tag pair $X|Y$ generally receive more (or less) support than questions tagged with X or Y. The results expose many such instances and provide early insights as to which pairings of topics gain in community support relatively to the topics taken separately (e.g., the pairs *HLSL|Directx11*, *2D|rotation*, and *3D|XNA*) and which pairings receive reduced support (e.g., *HLSL|XNA*, *3D|collision detection*, and *3D|Java*). We believe these results bear potential as decision helpers for indie game developers and game development teams, particularly those about to embark on a game development project and still deciding on which technologies and techniques meet their needs.

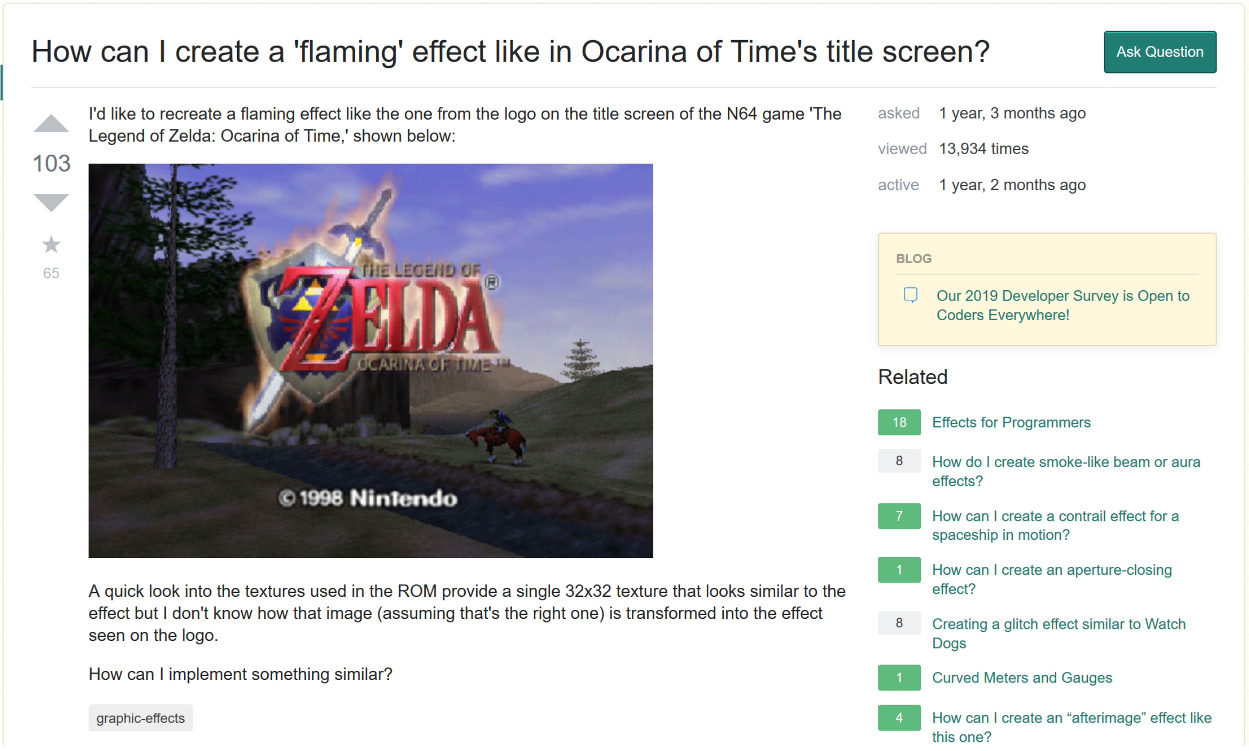
The rest of the paper is organized as follows: Section 2 introduces the background. Section 3 reviews the literature related to our study. Section 4 outlines our empirical study setup. Answers and discussion on our three RQs are provided in Sections 5–7. Section 8 provides additional analysis, after which Section 9 discusses some threats to validity. Finally, Section 10 provides concluding remarks and discusses future work.

2. Background

In this section, we provide general background information on the Game Development Stack Exchange website and its metadata schema that we leverage in the study.

2.1. Game Development Stack Exchange

Stack Exchange is a network of dedicated Q&A websites on which users can ask questions, share their expertise by answering questions, or just leave comments; the sites are self-moderated through a reputation award system that applies to the questions, the answers, the users, and even the comments. In this paper, we focus on Game Development Stack Exchange (GDSE), which is an exclusive Q&A website for video-game developers. When posting a question, the game developer is prompted to tag it to allow other developers or users to find them easily. Figure 1 presents an example of a question posted on GDSE. The title of the post is "*How can I create a 'flaming' effect like in Ocarina of Time's title screen?*". The word "*graphic-effects*" is the only tag provided by the user; the body of the post lies in between and gives a detailed description of the issue. The margins of the post contain some other meta-data relevant for our research, such as the score (103), the number of favourites (65), and the number of views (13,934).


How can I create a 'flaming' effect like in Ocarina of Time's title screen?

I'd like to recreate a flaming effect like the one from the logo on the title screen of the N64 game 'The Legend of Zelda: Ocarina of Time,' shown below:

A quick look into the textures used in the ROM provide a single 32x32 texture that looks similar to the effect but I don't know how that image (assuming that's the right one) is transformed into the effect seen on the logo.

How can I implement something similar?

graphic-effects

asked	1 year, 3 months ago
viewed	13,934 times
active	1 year, 2 months ago

BLOG

Our 2019 Developer Survey is Open to Coders Everywhere!

Related

18	Effects for Programmers
8	How do I create smoke-like beam or aura effects?
7	How can I create a contrail effect for a spaceship in motion?
1	How can I create an aperture-closing effect?
8	Creating a glitch effect similar to Watch Dogs
1	Curved Meters and Gauges
4	How can I create an "afterimage" effect like this one?

Figure 1. Example of a question on Game Development Stack Exchange.

Game Development Stack Exchange users are encouraged to participate as they can earn points towards their reputation and other privileges (e.g., editing), for being actively involved on the site. Users can vote for questions and answers, with both *upvotes* (positive feedback) and *downvotes* (negative feedback) allowed; the number of *upvotes* minus the number of *downvotes* represents the *score*. For a given question, only one answer can receive the status of "*Accepted Answer*", meaning that the user who posted the question designated it as a satisfying answer to his problem. GDSE posts (questions and answers) are archived in a database, primarily in a table Post, the schema of which can be seen in Table 1.

Table 1. Schema of the Post table.

Name	Description
Id	Post ID
PostTypeId	Post type: 1 for questions and 2 for answers
AcceptedAnswerId	ID of the accepted answer post
ParentId	ID of the parent element, if relevant
CreationDate	The date when the post was created
DeletionDate	The date when the post was deleted
Score	Number of upvotes minus number of downvotes.
ViewCount	Number of views for the question post
Body	Content of the question post
Title	Title of the question post
Tags	Tags provided for the post
AnswerCount	Number of answers received for the question post
CommentCount	Number of comments received for the question post
FavoriteCount	Number of users who marked the question post as favorite

3. Related Work

In this section, we provide a short review of game development from a software engineering research perspective. Then, we present some of the previous relevant work conducted on Stack Overflow, the most prominent Stack Exchange website.

3.1. Research on Video Game Development

The study [1] investigated software engineering practices in video-game development and found that video game developers generally adopted and adapted traditional methods of software engineering but would benefit from more elaborate empirical methods. A few years later, ref. [5] carried out a study to pinpoint the differences between software development and game development. They found that video game requirements are substantially different from other software products since fun and entertainment are the sole goal of the former; for instance, maintaining a game means changing some of its content (e.g., new quests, characters, or add-ons) rather than just features in a traditional sense. Furthermore, testing and quality assurance in both industries present significant differences. More recently, the study [4] reported on distinctive features of video-games and non-video game software through the analysis of open source software of both types. They explained that these distinctions lie in the tools used in both industries, the range of expertise found in developers' teams, the testing of the products, the way bugs are dealt with and the release and follow up of the products.

Other research works [6–11] aimed at uncovering issues in video-game developments through the analysis of postmortem retrospectives of various games. Postmortems of video game projects are often published by video game development managers on gaming forums such as Gamasutra (https://gamasutra.com/ (accessed on 1 October 2021)) and are useful sources to analyse issues arising in game development projects. In the research work [6], the authors concluded from their analysis that many of the reported issues could be traced back to inadequate requirements of engineering. They zoomed in on media and technology integration (interaction of game assets and game engines, artists, and programmers), non-functional requirements, and game play requirements (hard to prototype). They noted how dependent games were on technology, which can be cutting-edge and which game projects often have to pioneer or develop internally. A few years later, ref. [7] proposed a comparative analysis of video game development issues versus those in traditional software development. The paper surveyed development project problems from research literature and postmortems. Even though the paper appears to highlight management problems stemming from inappropriate handling of needed features (unrealistic scope, features creep, or feature reduction), various problems linked to technology and tools (notably from third parties) account for more than 60% of issues reported in the analysed postmortems. Interestingly, ref. [8] propose a similar perspective, with an emphasis on management problems but extensive discussion on the selection and integration of third-party technology, notably game engines. Most recently, ref. [11] proposed a dataset of video game development problems extracted from 200 game project postmortems spanning over 20 years (1998 to 2018). They found 1035 problems related to software engineering, which they grouped into 20 different types such as Tools, Testing, Monetization, and Marketing, which in turn are classified into three groups (Production, Management, and Business). A subsequent study by the same authors [12] further analyzed that database; the top five most common types of issues were technical (11%), related to game design (11%), team (8%), tools (7%), and planning (7%). Production issues, on the whole, account for 46% of the problems, even though technical and game design problems have been decreasing over the years.

3.2. Research Studies Based on Stack Overflow

As stated above, the Stack Exchange network regroups different Q&A websites, of which Stack Overflow (SO) is the most popular and the top Q&A website for software developers. SO contains more posts, covers a variety of programming topics, and has attracted a much more substantial quantity of software engineering research work. In the following, we review research studies centered on SO that are the most relevant to our current study.

The study by [13] considered questions asked over a period of 15 days and found that Stack Overflow seemed effective at providing help for code reviews, while how-to questions and questions about unexpected behaviors were the most popular. A later study by [14] used Latent Dirichlet Allocation (LDA) to identify topics, then proceeded to analyze their popularity and trends. They found that popular topics range from jobs to version control to specific language syntax, with web and mobile development trending up the most.

Many research studies have narrowed their analysis of Stack Overflow on specific areas, such as *mobile development* [15–17], *web development* [18–20], *web3D technologies (Three.js and WebGL)* [21], *testing* [22,23], and *security* [24–27].

On mobile development, we can cite [15], which used LDA and reported notably on the popularity of general questions and device compatibility issues, with more specific topics such as crash reports and database connection reported as attracting less attention. The study [16] performed a manual analysis on the top 450 Android-related posts on SO and found that most common questions could be classified as “*How to?*” and “*What is the problem?*”, while “User Interface” and “Core Elements” are the most discussed issues. More recently, [17] took, in addition to questions, interest in provided answers and analyzed over 13 million posts. They found that most popular questions involved app distribution, mobile tools, and user interface development, and also found that mobile-related questions were relatively harder to answer than non-mobile-related questions.

On web development, a first paper of note would be [18], which used data from Stack Overflow to obtain a better understanding of the challenges faced by web developers. Their results show, among other things, that there was an uptrend in the number of questions related to web-development, concurrently with a downtrend for cross-browser related posts. Another study by [19] collected data from Stack Overflow to investigate web developers concerns pertaining to Web APIs. They found that ’Known issue/bug’ is a particularly dominant topic of discussion, and observed that discussions are usually (three times out of four) about occasional concerns that disappear quickly, which would suggest that "*Web API providers tend to timely address most problems encountered by client developers*".

Various other concepts were explored using SO. Testing-related issues were studied, notably by [22,23], with a focus on Selenium-related questions. Security-related issues were investigated by [24], which classified these issues into five categories (i.e., web security, mobile security, cryptography, software security, and system security), with web security questions being the most common ones. Questions on passwords, hashes, signatures, and SQL injection were reported as the most frequent ones when it comes to security-related discussions on Stack Overflow.

A great body of research that used data from Stack Exchange can be found in [28]. To the best of our knowledge, with the exception of a recent 2019 paper on GDSE with a narrow focus on "serious games" [29], there have been no studies that used Stack Exchange Game Development forum, nor has there been any research that investigated game development topics using Stack Overflow.

4. Empirical Study Setup

4.1. Study Overview

Briefly put, our study is centered on identifying video-game development topics with the highest (or lowest) levels of community attention and support.

Our goal is to give game developers and managers insights about the technological and conceptual challenges they may face on their project. To achieve that goal, we conducted a study on GameDev and aim to answer three research questions: (i) which video-games topics receive the most attention, (ii) which video-games topics receive the most/the least support, and (iii) which combinations of topics receive the most/least community attention and support. Each research question is answered through various metrics, detailed in Section 4.2.

Our community of interest is GDSE, the question and answer website dedicated to video-game developers in the popular Stack Exchange network. As previously presented in Section 2.1, questions in GDSE come with user-defined tags that help to categorize them. For this first study, we rely on these tags as a way to aggregate questions into more generic topics. Doing so allows for a clean and reliable way to assess the topics of interest for game developers using GDSE. Furthermore, a cursory analysis of the tags present in the GDSE site reveals that they generally fall into two categories: (i) general topics or concepts (e.g., *animation*) and (ii) specific technology choices (e.g., *Unity*, the popular game engine). Preliminary analysis of GDSE questions pointed to a need to make such distinction for the tags or otherwise end up mostly with tags about technologies that dominate the game development market. Thus, in the following, we will refer to concepts and techs to denote these two categories. Furthermore, we will use italics for tags and reserve capital letters for techs. Examples 1 and 2 display questions centered on concept tags and tech tags, respectively.

Example 1. *(https://gamedev.stackexchange.com/questions/149031/ (accessed on 1 October 2021))*

Title: How do I get players to say "no" when they are afraid of missing out on sidequests or XP?

Body: In my RPG, I have a companion npc who is overconfident in his abilities and lacks self-control. ... So how do I get players to say "no" when they are afraid of missing out on sidequests or XP?

Tags: game-design, rpg, npc

Viewed 156k times, score: 156, favorite: 36

Example 2. *(https://gamedev.stackexchange.com/questions/96014/ (accessed on 1 October 2021))*

Title: What is Vulkan and how does it differ from OpenGL?

Body: Khronos Group (the standards body behind OpenGL) has just announced Vulkan: Vulkan is the new generation, open standard API for high-efficiency access to graphics .... What *exactly is Vulkan's relationship to OpenGL? Its previous name "glNext" (short for "Next Generation OpenGL Initiative") makes it sound like a replacement.*

Tags: OpenGL, Vulkan

Viewed 130k times, score: 156, favourite: 45

4.2. Research Questions and Metrics

In this section, we present our research questions, their motivation, and the metrics associated with each of them.

RQ1: Which video-game development topics receive the most community attention?

Motivation: An answer to this question gives developers an overview of the topics they are likely to encounter when they embark on developing a game.

Metrics: This first research question (RQ1) explores various popularity metrics (*view counts*, *favourite* counts, *score*, number of *questions*) for a single tag (single tag here does not refer to a tag used alone for a question, but rather in opposition to the tag pair analysis of RQ3), with consideration to *tech* tags and *concept* tags. More specifically, we retrieved top *concept* and *tech* tags according to metrics such as the number of questions in which they were present, the cumulative (and average) number of views these questions generated, the cumulative (and average) score of these questions and the cumulative (and average) the number of times these questions were bookmarked as favourites.

RQ2: Which video-game development topics receive the most/least community support?

Motivation: An answer to this question could help developers identify early topics for which they are more/less likely to receive community help. In particular, this RQ aims to provide insights on issues that are more likely to stall a project or require more resources, in the sense that community help (in this instance, from GDSE) is lacking. In short, and somewhat colloquially, game developers facing these kinds of issues may be on their own.

Metrics: This second research question (RQ2) explores various community support metrics associated to a single tag, also separately for tech tags and concept tags. We consider that, from the perspective of an asker, a question that does not receive an accepted answer is a failure. Furthermore, in successful cases, we take interest in the wait time between the moment a question is asked and the moment an accepted answer is provided. To evaluate the community support for a topic, we thus consider all the questions associated with its corresponding tag and collect two metrics:

1. Its *failure rate*: the percentage of questions that do not have an accepted answer;
2. Its median (success) *wait time*: in other words, the median time for satisfactory answers (in these cases, where the question received an answer that its asker accepted).

More specifically, we considered, for each tag, (i) the number of questions that use it, (ii) its failure rate (the percentage of these questions that do not have accepted answer), and (iii) its wait time (the median time for an accepted answer when there is indeed one).

RQ3: Which combinations of video-game development topics receive the most/least community attention and support?

Motivation: An answer to this question could inform the choices of game developers when it comes to, for instance, selecting technologies that have high support for a given concept or technologies that work well together or not. This third research question (RQ3) goes further in analysing the community support of tags, by focusing on two tags at a time instead of one. A key motivation for such investigation comes from the observation that some key tags, such as "*Unity*" (https://unity.com/ (accessed on 1 October 2021)), are present in a lot of questions. Thus, we have an interest in unveiling which questions within that *Unity* "universe" may be more popular or difficult. More broadly, studying pairs of tags can provide valuable information for both researchers and game developers, when it comes to community support, about (i) which technologies are the most/the least challenging to combine (*tech–tech*), (ii) which concepts are harder/easier to conjointly address (*concept–concept*), and (iii) the worst/best choices when choosing a technology to handle a given concept (*tech–concept*).

Metrics: RQ3 reuses the same metrics as RQ1 and RQ2. To account for computational issues related to query execution success (see Section 4.3), we limited the tags considered in this RQ to the top 1000 tags per number of questions in our dataset.

For all RQs, we considered only tags and pairs of tags for which there were at least 30 questions. We settled on 30 based on the view that a sample size of 30 is valid for most research investigations [30].

4.3. Data Collection and Processing

The data used in our study is extracted through T-SQL queries executed on the Stack Exchange Data Explorer (https://data.stackexchange.com/ (accessed on 1 October 2021)), which is an open source tool for running queries against public data available from the Stack Exchange network.

A key table in that database is the Post table (see Table 1), which proposes data (title, body, tags, dates, and various popularity counts) for both questions (*PostTypeId* = 1) and answers (*PostTypeId* = 2). In particular, each question can be linked to an *accepted answer* through the field AcceptedAnswerId, with a null value indicating that the question does not have any accepted answer. For the purpose of our study, we considered data over a five year period, from 1 January 2014 to 31 December 2018, thus collecting a total of 25,534 questions, of which 22016 received at least one answer and 12,912 received an accepted answer. The data set is available at [31]

Over the five-year span, 982 distinct tags were used by the GDSE community. For the purpose of this study, we only considered tags that occurred at least 30 times, which is 344 tags (96 tech tags and 248 concept tags). The tags in our study relate to a variety of areas. There were 248 concept tags, referring to generic areas of interest for game developers (concepts such as *animation*, *textures*, etc.), and 96 *tech* tags, referring to a specific technology/library offering, be it frameworks, languages, libraries (such as *Unity*, *C#*, or *DirectX*). As stated before, the distinction between tech and concept tags is an important one to separate tags that reflect game development concerns ("*concepts*") from those that may be used to specify the development environment, and which "popularity" may merely reflect market dominance.

5. Results of RQ.1: Community Attention per Topic

We took particular interest in the top 10 tags for each of the metrics in RQ1, which resulted in 27 concept tags and 26 tech tags. As announced in Section 4, we consider the results for techs and concepts separately. Tables 2 and 3 summarize our results for techs and concepts, respectively, and present the tags found in the top 10 for at least one of the metrics. In general, for a given tag, the more questions associated with it there are, the more views and the better the cumulative score or number of stars (Pearson correlations between these metrics range from 0.91 to 0.99). So, for concision, our tables only present the averages for views, scores, and favourites. For each of the tags, we indicate the raw value of each of the considered metrics, as well as the relative ranking (with respect to the category: tech or concept). We use, in the table, Q, AvgV, AvgS, AvgF for the number of questions, the average number of views, the average score, and the average number of favourites, respectively, while R-Q, R-AvgV, R-AvgS, and R-AvgF stand for their respective ranks. In both tables, tags are sorted based on the number of questions associated with them: the ones with the most questions are presented first.

Looking at Tables 2 and 3, we can see that some tags present significant differences depending on which popularity metric is examined. A simple Pearson correlation analysis reveals that the averages do not correlate with the number of questions (correlation values ranging from 0.02 to 0.08), but there is some small correlation between the average number of views and the average score (-0.44) and a large correlation between the average score and the average number of favorites (0.86). From our perspective, the numbers of questions or views associated with a tag can be seen as audience metrics (implicit relevance), while score and stars would be favourability indices (explicit endorsement).

Table 2. Top 10 tech tags for all the considered metrics.

Tag Names	Q	R-Q	AvgV	R-AvgV	AvgS	R-AvgS	AvgF	R-AvgF
Unity	9029	1	1202	10	1.17	49	0.29	32
C#	4212	2	1074	18	0.97	73	0.25	45
OpenGL	2185	3	870	27	1.62	13	0.4	15
C++	2013	4	837	30	1.4	20	0.4	15
Java	1930	5	734	41	1.06	59	0.25	45
libGDX	1798	6	821	32	1.28	36	0.29	32
Android	1134	7	1100	14	1.2	44	0.29	32
XNA	787	8	512	77	1.1	55	0.22	55
Javascript	738	9	711	47	1.08	57	0.26	39
MonoGame	572	10	556	72	1.15	52	0.19	68
GLSL	491	11	842	29	1.64	9	0.42	12
Game-Maker	328	15	678	52	2.47	5	0.86	3
Unityscript	302	16	1436	7	0.81	86	0.22	55
Windows	126	35	652	56	1.64	9	0.29	32
Phaser	114	37	1121	11	1.84	6	0.43	11
Lua	110	38	831	31	1.64	9	0.84	4
SpriteKit	91	41	696	50	1.66	8	0.45	9
Google-Play-Services	89	42	1365	8	1.21	41	0.38	17
Three.js	75	49	1440	6	1.23	39	0.6	7
Google-Play	72	51	3136	2	3.17	4	0.78	6
Steam	65	57	1582	5	3.55	3	0.83	5
Godot	54	65	588	66	1.67	7	0.09	91
Facebook	45	70	1223	9	0.8	87	0.38	17
JSON	42	74	1008	20	1.36	22	0.45	9
UDP	40	75	1717	4	3.77	2	1.27	2
Eclipse	34	85	1812	3	1.06	59	0.26	39
Oculus	33	89	612	62	1.64	9	0.3	28
Vulkan	32	90	3910	1	5.41	1	1.63	1

^{Q}: Number of Questions. $^{R\text{-}Q}$: Rank per Questions. ^{AvgV}: Average number of Views. $^{R\text{-}AvgV}$: Rank per Average number of Views. ^{AvgS}: Average Score. $^{R\text{-}AvgS}$: Rank per Average Score. ^{AvgF}: Average number of Favourites. $^{R\text{-}AVgF}$: Rank per Average number of Favourites.

5.1. Tech Tags: Technology/Library Choices

Looking at the tech tags in Table 2, we observe that, with respect to the number of questions, the game engine *Unity* is, with 9029 questions, the top tag by far, followed by the programming language 'C#' (4212 questions), which is unsurprising, since it is the language of choice for *Unity*. Aside from *Unity*, frameworks and technologies such as the API for high performance graphics *OpenGL* (https://www.OpenGL.org/ (accessed on 1 October 2021)), the game development framework *libGDX* (https://libgdx.badlogicgames.com/ (accessed on 1 October 2021)), Microsoft's game framework *XNA*, and the platform *Android* generate the most questions. Aside from *C#*, languages such as *C++*, *Java*, and, to a lesser extent, *Javascript* lead for the number of questions.

A ranking by average number of views presents a very different picture, with *Unity* barely making it into the top 10 (1202 views on average) while a narrower aspect of it, *UnityScript*, its now deprecated scripting language, is ranked seventh (1436 views on average). It should be noted that, although the average number of views is not correlated with the number of questions, 8 out of 10 of the top tags per average number of views have under 100 questions. We used average as a metric aggregate, following [17,24] in that regard, but high averages could be the result of one or two exceptionally popular questions. For instance, the top tech in this category is *Vulkan*, a "*low-level, cross-platform graphics API*";

it has one of the top viewed question in GDSE (130K), but most of its questions have less than 1K views.

When it comes to average score and favorites, *Vulkan* is again the top tech, also in part because of the same very popular question. Other notable tags with high average score and favourability include *UDP* (we decided to classify *UDP* as a tech tag, but it is such a standard that a classification as a concept could be valid too), *Steam*, *Google-Play*, *Lua*, *Game-Maker* and *SpriteKit*.

Beyond the specific technologies that appear to be receiving the most attention from game developers, our data suggest that game engines and development frameworks (e.g., *Unity*) are the most common topics attached to questions in GDSE. Then come graphics APIs and programming languages, with the most prominent being the ones associated wtih the top game engines. Aside from *Android*, specific platforms on which the games will be played are rarely mentioned. A possible explanation is that very popular game engines such as Unity offer advanced cross-platform support, making it less of an issue for game developers.

Table 3. Top 10 of all concepts tags for all the considered metrics.

Tag Names	Q	R-Q	AvgV	R-AvgV	AvgS	R-AvgS	AvgF	R-AvgF
2D	1593	1	1142	55	1.75	129	0.52	110
collision-detection	1173	2	774	160	1.31	185	0.34	161
physics	1021	3	746	166	1.64	142	0.46	128
shaders	980	4	957	94	1.91	110	0.58	96
mathematics	824	5	800	154	2	92	0.62	85
3D	779	6	850	136	1.62	143	0.47	124
textures	745	7	894	120	1.44	168	0.3	178
animation	685	8	1011	84	1.32	184	0.34	161
game-design	664	9	1476	21	5.79	9	1.88	11
rendering	610	10	909	111	2.15	72	0.52	110
terminology	122	66	3039	2	8.75	3	2.09	7
graphic-effects	101	85	1419	28	5.38	12	2.46	3
level-design	89	96	1371	34	5.49	11	1.96	9
user-experience	55	142	1805	9	7.2	5	1.36	18
publishing	51	151	1760	10	1.98	95	0.45	129
balance	50	154	2892	3	12.42	1	3.82	1
levels	49	156	1192	47	5.2	13	1.9	10
marketing	47	164	1257	41	7.11	6	1.15	30
licensing	46	168	1862	8	3.83	21	0.72	61
file-format	43	183	905	114	6.3	7	2.07	8
puzzle	41	188	1396	32	5.59	10	1.61	12
cross-platform	36	208	2122	5	5.83	8	1.22	26
file	36	208	2767	4	2.11	78	0.64	80
strategy	34	217	1432	25	5.12	14	2.35	4
aspect-ratio	31	236	4483	1	4.06	16	2.61	2
npc	31	236	2010	6	8.35	4	2.1	6
monetization	30	243	1913	7	8.93	2	2.23	5

^{Q}: Number of Questions. $^{R\text{-}Q}$: Rank per Questions. ^{AvgV}: Average number of Views. $^{R\text{-}AvgV}$: Rank per Average number of Views. ^{AvgS}: Average Score. $^{R\text{-}AvgS}$: Rank per Average Score. ^{AvgF}: Average number of Favourites. $^{R\text{-}AVgF}$: Rank per Average number of Favourites.

5.2. Concept Tags: Game Development Aspects

Table 3 presents the top 10 concept tags relative to the metrics described earlier. As stated before, these tags refer to generic terms encompassing various aspects relevant to game development: handling of *shaders* (shaders are computer programs that run on graphics hardware and provide a high degree of control over how scenes are rendered), *textures*, *sprites*, *rendering*, and *graphics* in general, as well as *collision detection*, *animation*, and *physics* in general.

With respect to the number of questions, the tag *2D* is the most used (1593 questions) and dominates *3D* (sixth with 779 questions) across all metrics. It is followed by *collision-detection* (second with 1173 questions), *physics* (third with 1021 questions), and *shaders* (fourth with 980 questions).

Here again, using the average number of views helps to highlight a different set of tags, mostly with fewer questions. The top tag here is *aspect-ratio* (with an average of 4483 views), followed by *terminology* (second with an average of 3039 views), *balance*, *file*, and *cross-platform*.

When it comes to favourability and score, the top tags include *balance*, *monetization*, *terminology*, *file-format*, and *npc* (non-player characters). In addition, the average metrics bring into light a number of tags related to publishing and advertising the games, with tags such as *publishing*, *marketing*, and *monetization*.

Most of the concept tags in our dataset are related to the rendering and animation of 2D (or 3D) characters or scenes, along with fundamental notions from physics and mathematics. However, some concepts are exclusively about gaming: mostly game genres (card game, puzzle, role playing game, etc.) but also aspects such as game design, leaderboards, and controllers. Among game genres, role playing games (rpg) are the most asked about (154 questions) followed by massively multiplayer online and real-time strategy games. Overall, concepts exclusively about gaming appear as tags for 3203 questions, which is only about 12.5% of our dataset. These questions have an average score of 3 and are viewed 1292 times on average, which is representative of concept tags as a whole.

The findings from RQ1 can be summarised as follows:

- Topics that receive the most attention include technologies such as *Unity*, *C#*, *Android*, and *Vulkan* and concepts such as *2D*, *3D*, *collision detection*, *animation*, *game design*, and *terminology*.
- The average number of questions per technology is higher than that of concepts, but questions attached to concepts are twice more upvoted or marked as favourite.
- Most game development questions are not about gaming elements per se; in fact, only about one question out of eight is tagged with a concept that refers to some gaming aspect, with game genres appearing the most.

6. Results of RQ.2: Community Support per Topic

It is important to first note that, when considering all the questions in GDSE (from 2014 to 2018), we found that only about 50% of the questions had an accepted answer, with a median wait time of 189 minutes. These numbers appear worse than what is reported in [17] for mobile-related questions in Stack Overflow (55%, 55 min average wait time), and even more so when compared to all of Stack Overflow (70%, 21 min average wait time) [17,32] (This is not a perfect comparison, as the years and aggregate metric somewhat differ.).

As in RQ1, we investigated tech tags and concept tags (as stated earlier, we only considered tags with 30 questions or more) separately, but this time through two main metrics: (i) the failure rate, which refers to the percentage of questions that do not have an accepted answer, and (ii) the median wait time, which refers to the median time for receiving satisfactory answers. Tech tags have, on average, a failure rate of 51.5% and a median wait time of 359 min. Concept tags also have, on average, a 51% failure rate but a better median wait time of 194 min.

Our findings are reported in Figures 2–7. The x-axis represents the median wait time it took, for a given tag, to receive an accepted answer for questions it is associated with. The y-axis represents, for a given tag, its percentage of questions without an accepted answer: the failure rate. Each tag is represented as a bubble, with the size of the bubble indicating its number of questions; the bigger the bubble, the more questions there are with the tag. An intuitive way to read these figures, relative to the x and y axes, can be summed up as follows: *"For a given tag, y% of its questions do not have an accepted answer, but when they do, it takes about x min for an accepted answer to be submitted."* Furthermore, we use in Figures 2 and 6 reference lines to indicate medians for both the failure rate and the wait time. Thus, we define four zones of relative difficulty of receiving answers, with respect to median values (the reference line in the figure): Easier-Faster (Bottom Left), Easier-Slower (Bottom Right), Harder-Faster (Top Left), and Harder-Slower (Top Right). The results reported in these figures will be discussed in detail in the following subsections.

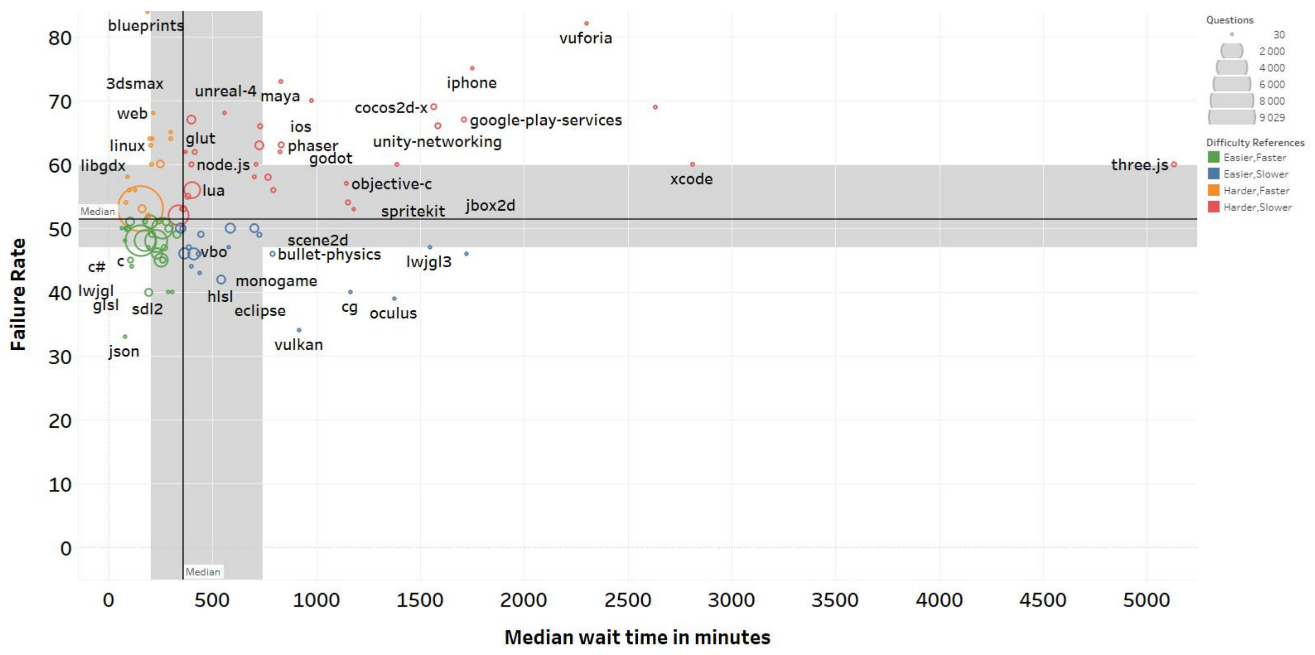


Figure 2. Community support for tech tags. Reference point is based on median values (51% for the Failure Rate, 351 min for the Wait Time). Circle sizes indicate the number of questions of the tag.

6.1. Community Support for Tech Tags

Figure 2 presents the community support metrics for tech tags. We can clearly see on the figure that the tag with the highest failure rate is, with around 85%, *Blueprints*, which is described in GDSE as "*a visual, node-based scripting model used by Unreal Engine-4.*" *Three.js*, a *JavaScript* 3D library, is by far the tag for which the wait time (5129 min) is the longest. (This is why it is not displayed in the figure.) In contrast, *Unity*, the most popular tag, bears a relatively low wait time of 152 min and a failure rate of 53%.

Vuforia (https://developer.vuforia.com/ (accessed on 1 October 2021)), an "*augmented reality system developed by Qualcomm*", not only has the second highest failure rate (above 80%) but also bears, at about 40 h, one of the longest median wait times for accepted answers. The few questions about *Vuforia* are mostly not receiving answers, and in the few instances when they do, it takes about 2 days. This may signal, with some caveats (i.e., a quick check of *Vuforia* tagged questions in Stack Overflow does show a less severe failure rate (at 55%), and the developer forums hosted by *Vuforia* seem quite active), to *Vuforia* project managers the need to allocate some resources to community support for game developers on GDSE. More generally, the **Harder-Faster** quadrant includes *Unity*, some desktop OSs (such as *MacOS*, *Linux*), some 3D software (*Blender*, *3DS Max*), and various programming languages (such as *Python*, *HTML5*, *PHP*, *.NET*). The **Easier-Faster** quadrant, especially in the lower tier, includes various languages and notation formats (*C*, *C#*, *XML*, *JSON*) as well as graphics libraries such as *GLSL* and *XNA-4.0*. On the **Easier-Slower** side, there are fewer elements, and they are relatively diverse (from *Windows* to *Eclipse*, *Oculus*, and *jMonkeyEngine*). Finally, the **Harder-Slower** quadrant, which stands for tags with higher failure rate and higher wait time, includes tags such as *iOS*, *Facebook*, *Google-Play-Service*, etc.

In the following, we contextualize our analysis of the data with respect to various categories and the top tech tags identified in RQ1.

Game Engine and Development Frameworks: Figure 3 presents data on game engines, game development frameworks, and libraries. Questions about *Cocos2d*, including *Cocos2d-X* (a cross-platform port) and *Cocos2d-iPhone* (*iOS* only), seem significantly more challenging to answer on all fronts, with a failure rate around 70% and median wait-times of 1567 min (*Cocos2d-x*) and 2634 min (*Cocos2d-iPhone*). Also outstanding are questions about *Unreal-4*, with a very high failure rate (around 68%) but a relatively low median time (around 400 min), and *jMonkeyEngine*, with a high median wait time (around 30 h) but a relatively low failure rate (around 46%). On the definitely easier side are questions about Simple DirectMedia Layer (*SDL*) (https://www.libsdl.org/ (accessed on 1 October 2021)), which appears to be the least challenging both in failure rate and median wait-times. The most popular engine, *Unity*, is unsurprisingly among the ones with the best community support, with an average failure rate and a lower wait time.

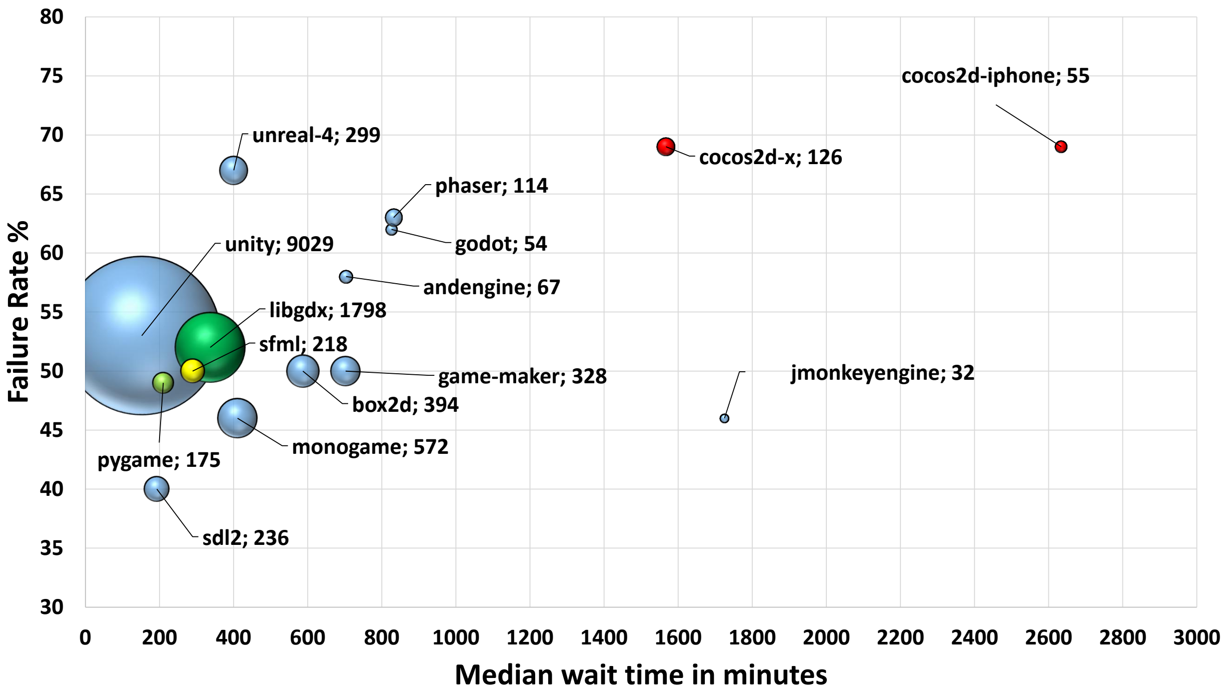


Figure 3. Failure Rate and Median Wait Time for Game Engines and Development Frameworks.

Three-dimensional software for asset creation: When it comes to 3D software used to create assets, established software such as *Blender*, *3DS Max*, and *Maya* all have a failure rate upwards 60%. In particular, nearly 75% of questions tagged with *Maya* do not have accepted answers.

Graphics APIs: Among graphics APIs, *WebGL* seems to be the most challenging when it comes to receiving community support, with a 56% failure rate and a wait-time around 13 h. Comparatively, *DirectX*, which is a set of multimedia APIs from Microsoft, and its versions *DirectX 9.0* and *DirectX 11* fare slightly better, with respective failure rates of 46% and 51% and median wait times around 4 and 6 h. As for the low-level, cross-platform graphics API *Vulkan*, its failure rate is only 34%, but its median wait time stands at more than 15 h.

Desktop and Mobile OS: In the mobile OS category, *iOS* (63%, 726 min) has the worst community support. Comparatively, *Android*-related issues fare better, with 56% of failure and a median wait time of 403 minutes. *Windows Phone* fares better than either *iOS* or *Android*, but at 34 questions, its data sample is way lower. As for desktop OS, *Windows* questions have a lower failure rate but take more time to answer while *Linux* and *MacOS* questions have virtually the same level of support, with a relatively high failure rate (low to mid-60s) but lower wait-times (around 200 min). A deeper analysis of tags specifically tied to these OSs suggests that *iOS*-related (more generally Apple technology) issues lag in terms of community support on GDSE. It starts with the tag *iOS* (63%, 726 min) but extends to tags such as *Swift* (70%, 979 min), *iPhone* (75%, 1753 min), *Xcode* (60%, 2811 min), *Cocos2d-iPhone* ("a free open-source framework for building 2D games, demos, and other graphical/interactive applications for iOS") (69%, 2634 min), etc. Most questions related to *iOS* stand out negatively in terms of effective and fast community support on GDSE.

Programming Languages: Figure 4 presents data on the programming languages. *C#* is the dominant language and benefits from the second best support (after *C*), with lower failure rate (48%) and median wait time (155 min). *Swift* and *Objective-C* are the least popular languages and present the highest failure rates (70% and 57%, respectively) and longest wait times (979 and 1145 minutes, respectively). The popularity level does not always translate to equivalent support level; however, as we can see, *C* has a modest number of questions but the best support.

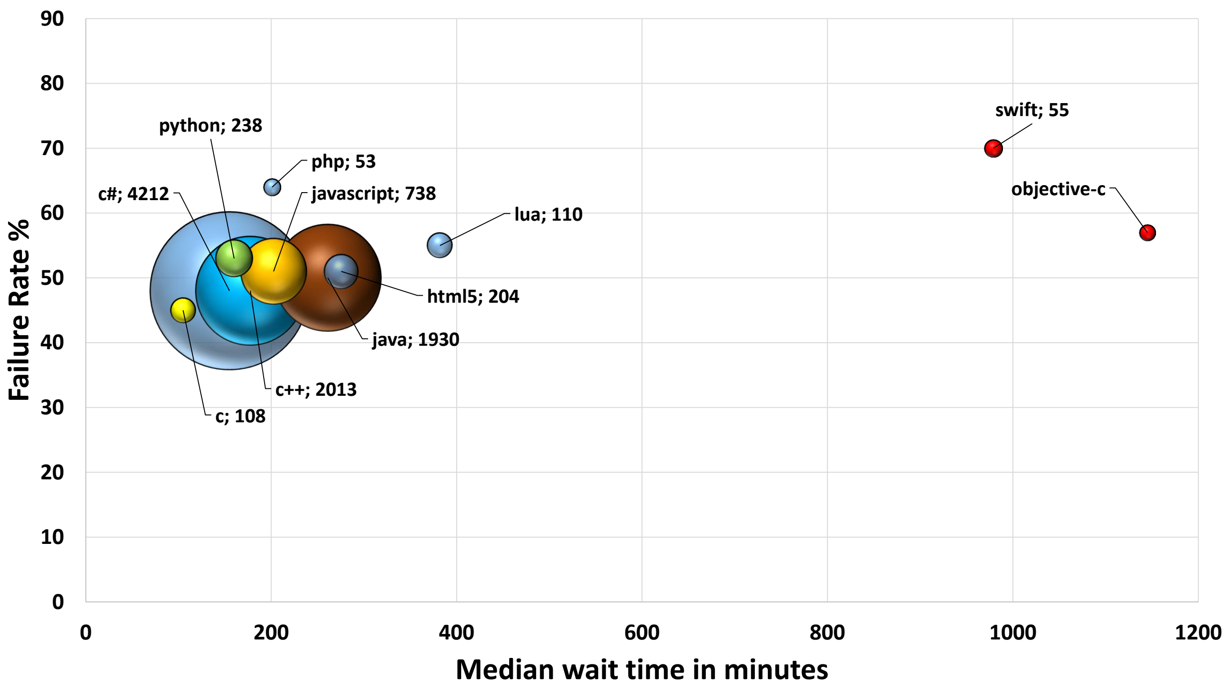


Figure 4. Failure Rate and Median Wait Time for Programming Languages.

As for data formats, *JSON* seems to have more solid support (33% failure rate, wait time of 79 min) than *XML* (40% failure rate, wait time of 289 min.)

Integrated Development Environments: In this category, *Visual Studio* (https://visualstudio.microsoft.com/ (accessed on 1 October 2021)) is quite popular (138 questions) and arguably the IDE with the highest support (50%, 94 min). *Eclipse* is somewhat close support-wise, but the number of questions associated with it is low (34). *Xcode*-related questions fare the worst.

Support for top tech tags: Looking at Figure 5, which features the top techs, we note that most of these tags have a failure rate around 50% (± 10%), with *Unreal-4*, *iOS*, and *Phaser* as notable exceptions. The wait time spreads over a wide range (from around 2 to 7 h). Questions related to programming languages (*C#*, *C++*, *Python*, *Javascript*, *Java*) seem to receive higher support than, for instance, questions related to operating systems (*Android* and *iOS*).

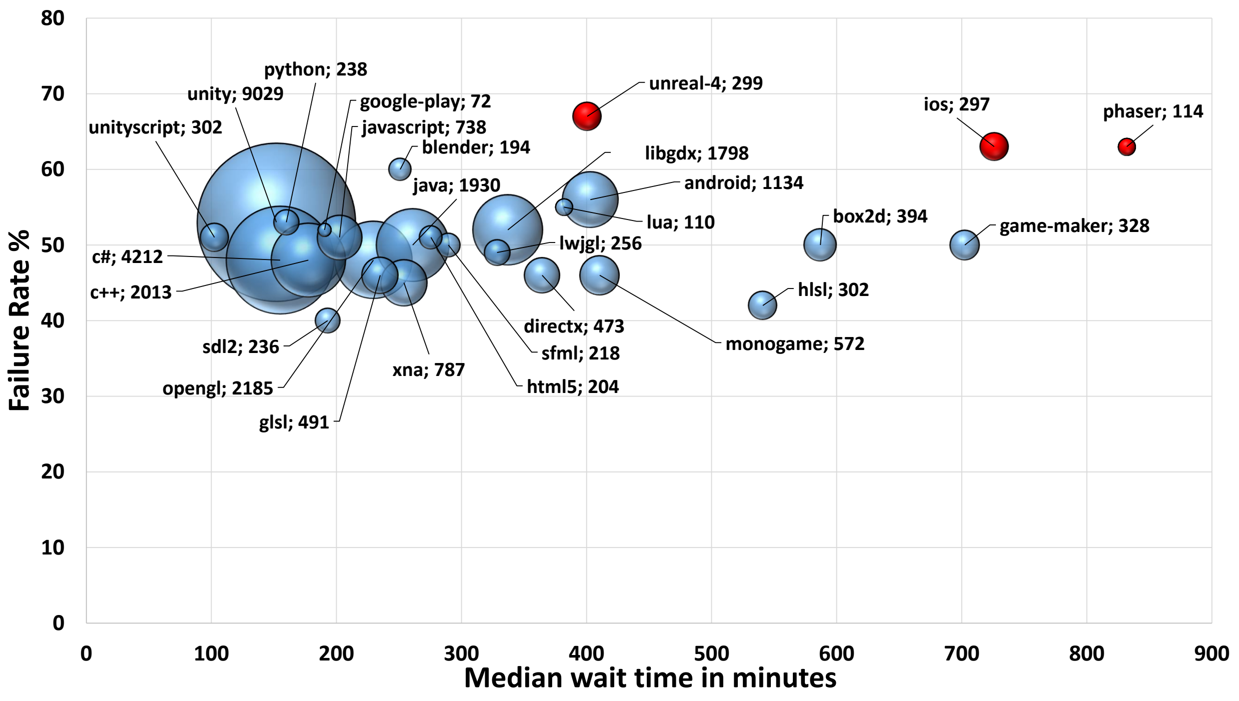


Figure 5. Failure Rate and Median Wait Time of most popular techs.

6.2. Community Support for Concept Tags

Figure 6 presents the distribution of concept tags along the failure rate and wait-time axes of analysis. A few tags, such as *licensing* (32%, 35 min), *storage* (33%, 85 min), and *security* (39%, 72 min), stand out as particularly well supported, but many more stand out as challenging.

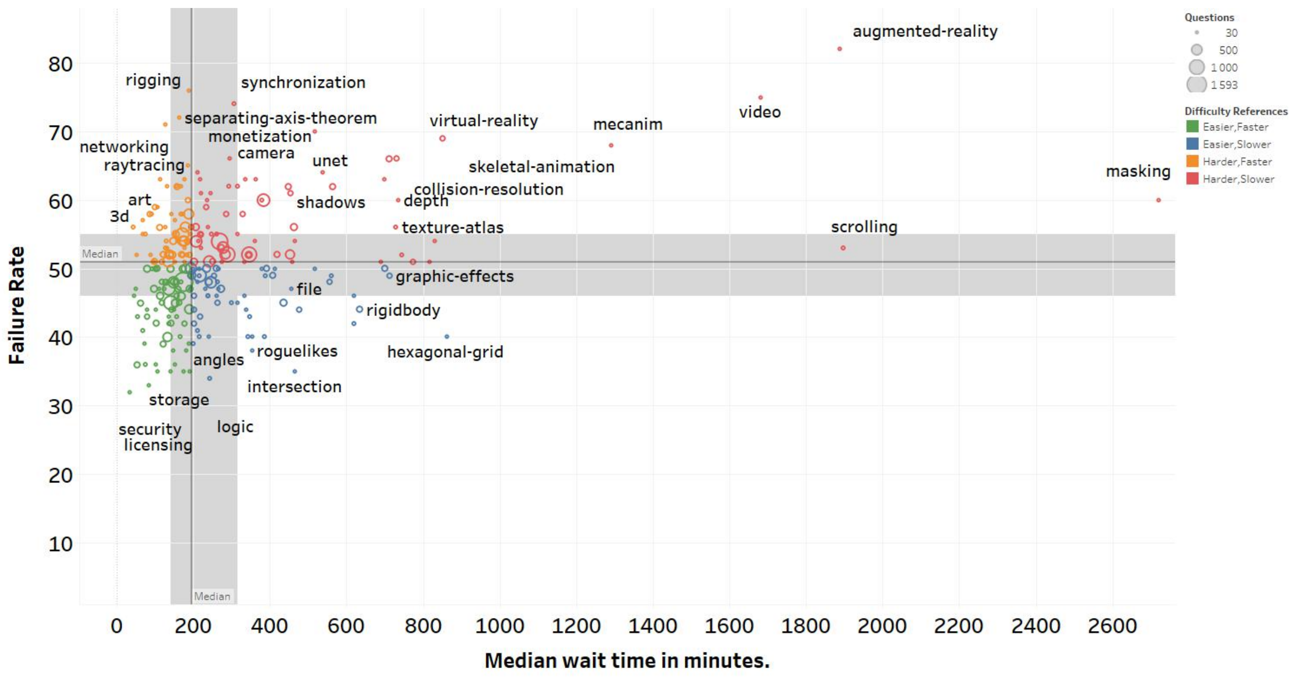


Figure 6. Community support for concept Tags. Reference point is based on median values (51% for the Failure Rate, 195 minutes for the Wait Time). Circle sizes indicate the number of questions of the tag.

Some tags, such as *controllers*, *combat*, and *leaderboards*, refer to concepts that are specifically about gaming, not merely 2D or 3D design. Among these, *modding* (66%, 731 min) stands out as the least supported. (Interestingly minecraft-modding (58%, 330 min) is a tag of its own and is the fourth most supported concept per failure rate.) Other weakly supported concepts include *character* (60%, 187 min) and *multiplayer* (56%, 179 min). On the other end of the spectrum, questions tagged with concepts such as *savegame* (43%, 79 min), and *npc* (35%, 190 min) appear easier to answer.

Some other tags explicitly refer to the handling and rendering of 2D or 3D elements that are featured in any typical game. In this category, the tags *video* (75%, 1682 min) and *blending* (70%, 517 min) stand out as lacking support in contrast to *side-scroller* (35%, 140 min), which has the best support. Unsurprisingly, *3D* (55%, 172 min) is harder than *2D* (48%, 175 min).

The rendering of game scenes and such often requires concepts taken from disciplines such as mathematics and physics. Among the tags that are heavier on mathematics or physics, questions on *rigging* (76%, 189 min) or *separating-axis-theorem* (72%, 164 min) are clearly the least supported. Also lacking support are questions on *skeletal-animation* (66%, 711 min) and *collider* (63%, 698 min). Other tags of note include *animation* (60%, 383 min), *physics* (52%, 290 min), *collision-resolution* (56%, 464 min), and *collision-detection* (54%, 270 min). It is worth noting that some of the more purely mathematical concepts seem to have relatively good support: *geometry* (48%, 150 min), *movement* (48%, 149 min), *rotation* (48%, 246 min), and *mathematics* (45%, 141 min). In particular, *trigonometry* (36%, 76 min) has one of the highest community support.

Finally, some tags are quite generic, whether they come from larger computer/science terminology (e.g., *inheritance*, *bug*, *debugging*) or even broader (technology-related or not) vocabulary (e.g., *art*, *marketing*). According to our data, the least supported tag here is *augmented reality* (82%, 1887 min); another trending technology that is not well supported yet is*virtual reality* (69%, 851 min). These are, with *synchronization* (74%, 307 min), the top 3, by far, least supported tags. Also lacking support are questions about *advertisements* (63%, 364 min) and *monetization* (63%, 218 min). Conversely, some related tags such as *copyright* (59%, 100 min) and *licensing* (32%, 35 min) show stark differences, with *licensing* questions being way more supported.

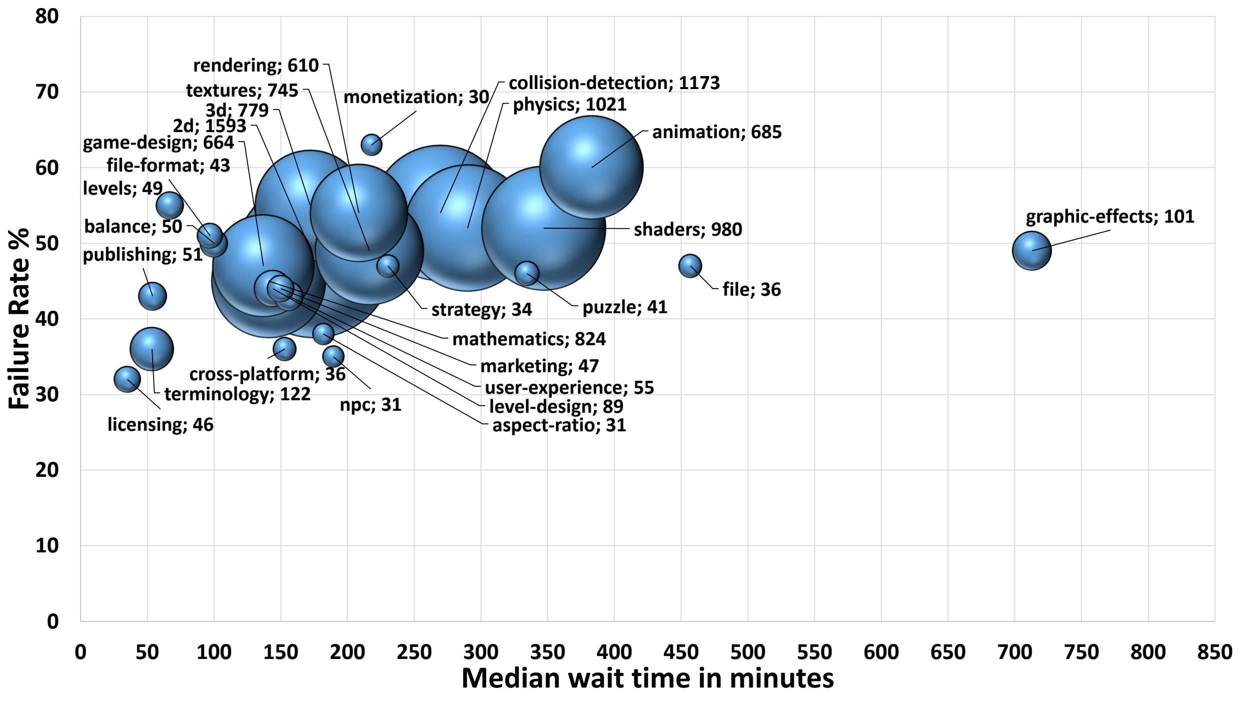


Figure 7. Failure Rate and Median Wait Time for most popular concepts.

Support for top concept tags: Looking at Figure 7, which features the concepts that command the most attention, we note that most of these tags have a failure rate around 50% (± 10%) and a median wait time from 30 min to 6 h. The tag *graphic-effect* stands out with a wait time of around 20h. Although many of the most supported concepts (*licensing*, *terminology*, *publishing*, etc.) seem peripheral to game development activities, there is no clear pattern; for instance *monetization* (63%, 218 min) has a fairly high failure rate compared to more technical tags such as *collision-detection* (54%, 270 min).

The findings from RQ2 can be summarised as follows:

- For both concept and tech tags, popular tags generally have a failure rate around 50% (± 10%) and a median wait time between 30 min and 6 h (for concept tags) and 2 and 7 h (for tech tags).
- We uncovered some clear differences of support between technologies occupying the same space: for instance, *Unreal-4* has a much lower support than its rival *Unity*, while *Android*-related tags are generally better supported than *iOS*-related tags.
- For concept tags, we found that 3D questions have, unsurprisingly, less support than 2D, whereas purely mathematical concepts generally have good support; additionally, relatively new paradigms such as *virtual reality* and *augmented reality* seem to be trailing in community support.

7. Results of RQ.3

A first result of this RQ relates to community attention and stems from the analysis of the number of questions covered by the pairs of tags found on GDSE. Such numbers provide a window into which tags appear most often together, that is, which topics are most associated with one another in GDSE questions. It is not uncommon to see some tags reoccurring together. In fact, 67 pairs of tags appear at least 100 times in GDSE questions. The pair *Unity*|*C#* is the most frequent, with more than 3000 questions, followed by pairs such as *Java*|*libGDX*, *2D*|*Unity*, *C++*|*OpenGL*, and *C#*|*XNA*. With respect to the distinction between concept and tech tags, we can report that, out of the top 100 pairs of tags per number of questions, thirty-four (34) involve only tech tags, twelve (12) involve only concepts, and fifty-four (54) were a mix. Moreover, some specific insights can be gained with respect to a given technology or concept. For instance, when considering the number of questions associated to pairs of tags, Unity's top 10 topics are *C#*, *2D*, *shaders*, *animation*, *physics*, *collision detection*, *gui*, *Android*, *UnityScript*, and *camera*, whereas *OpenGL* co-occurs with *shaders*, *Java*, *textures*, *LWJGL*, *rendering*, *3D*, *graphics*, *DirectX*, *matrix* and *lighting*. Similarly, we can uncover that a concept tag such as *3D* is most associated with *Unity*, *OpenGL*, *C#*, *mathematics*, *2d*, *rotation*, *C++*, *Java*, *rendering*, and *XNA*, in that order.

Our focus in this RQ is on community support. For each pair of tags, we take interest in the differences between failure rates and wait times when considering the pair versus one of its constituents. In particular, to simplify our analysis, we focus on failure rates and highlight cases where a pair $X \mid Y$ has a failure rate that is five (5) points higher or lower than both X and Y. In other words, we focus on cases where a pair receives notably higher or lower community support than its constituents, thus indicating that it is easier or harder to receive support for these situations when both tags are present.

We report the results of our analysis in Figures 8–10. In these figures, upwards (respectively, downwards) triangles indicate cases where the pair has a failure rate at least 5 points higher (respectively, lower) than both of its constituents. The size of the triangles is indicative of the number of questions for the tag pair. Colours indicate the relative support of the pair in the set of pairs, as presented in RQ2.

Note that, in the following, we will sometimes use the notation ($\pm F\%$, $\pm W$) as a shorthand, where $\pm F\%$ represents the average difference between the failure rate of a pair and those of its constituents and $\pm W$ represents the average difference between the median wait time of a pair and those of its constituents.

7.1. Tech–Tech Pairs: Integrating Two Technologies

Looking at Figure 8 and considering failure rates, we can see that pairs with notably lower failure rates than both their constituents are almost all below the median line and with failure rates between 30% and 50%. They also have relatively low wait times (within 2 and 11 h). On the other hand, pairs with notably higher failure rates fit within a 50–70% failure rate but are way more spread out in terms of wait time (from 1 to 26+ hours). Pairs with better support include *C++*|*HLSL*, *Java*|*OpenGL*, and *Android*|*iOS*.

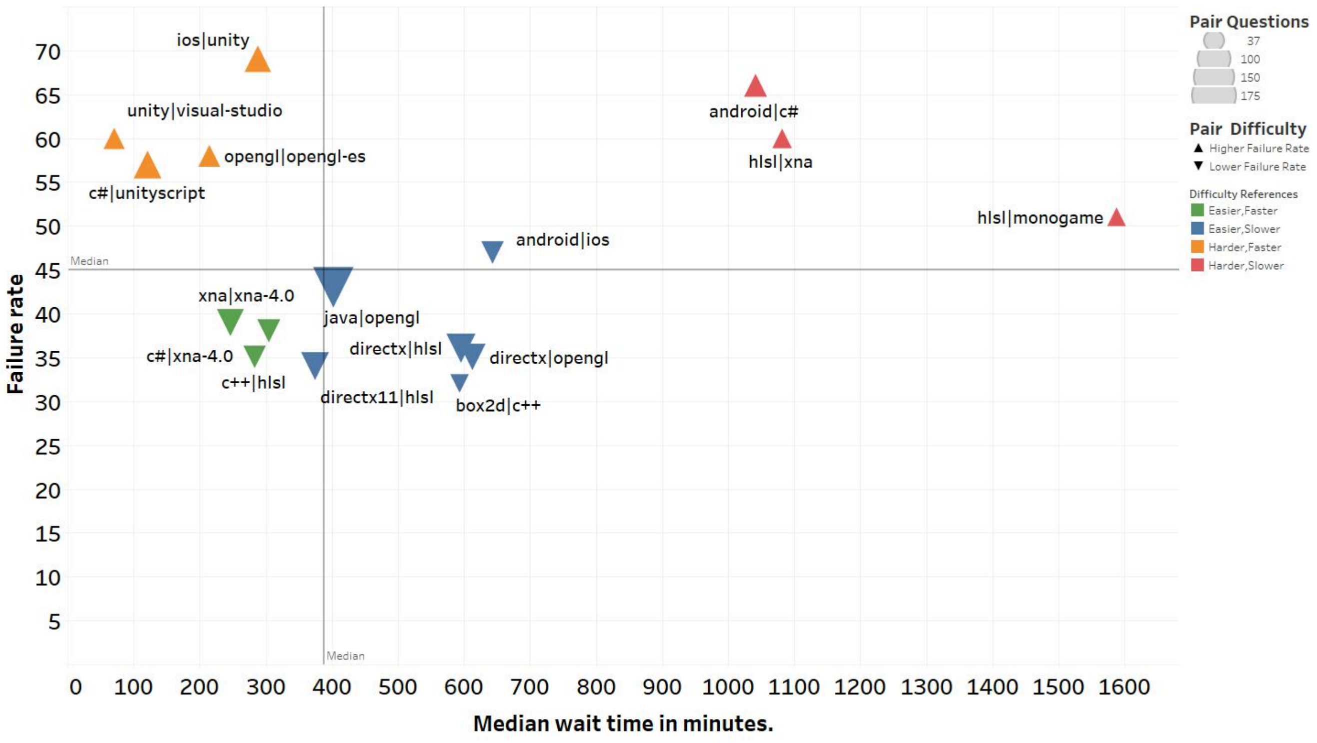


Figure 8. Community support for Tech|Tech pairs. Upwards (resp. downwards) triangles indicate cases where the pair has a failure rate at least 5 points higher (resp. lower) than any of its constituents. The size of the triangles is indicative of the number of questions for the tag pair. Colors indicate the relative support of the pair in the set of tech tech pairs.

Beyond the figure, we analysed failure rate differences to obtain a better sense of the data. The combination with the most improved support is *Box2D* and *C++*, with a failure rate of only 32%, which improves on that of *C++* (48%) and *Box2D* (50%), albeit with much worse time (594 min) relative to *C++* (178 min). A similar phenomenon occurs for questions tagged with both *Android* and *iOS*, with a failure rate reduced by 12.5 on average (−9 relative to *Android*, −16 relative to *iOS*) but slightly higher (+4 h) wait times relative to *Android*. Staying with *Android*, its pairing with *C#* is the second worst in terms of failure rates (+10 relative to *Android*, +18 relative to *C#*), and the wait times are also inflated (+638 min relative to *Android*, +886 min relative to *C#*). Also worth mentioning, Unity questions mixed with some other tech tend to receive lesser support; such is the case for when Unity is mixed with *OpenGL* (+5.5%, −113 min) or *Visual Studio* (+8.5%, −52 min).

The most noteworthy case here is that of the library *HLSL*, which receives significantly better support when combined with *DirectX11* (on average −12% and −71 min) or *C++* (on average −12% and −76 min) but receives significantly degraded support when combined with *XNA* (+16.5%, +684 min; the worst degradation) and *MonoGame* (+7%, +1113 min). Except for *C++*, all these technologies are firmly in the Microsoft universe. Considering that *HLSL* is a shading language developed for use with *Direct 3D*, a component of *DirectX*, it is unsurprising that *HLSL* meshes well with *DirectX* and even *C++*, the most prominent language for *DirectX*. *MonoGame* is the open source implementation of *XNA*, which was often viewed as the *.NET* analog to *DirectX*. Our results suggest there is a deficit in support for projects trying to mix *HLSL* with *C#/MonoGame* vs. *DirectX/C++*.

Manual analysis showed that most *tech|tech* tag pairs come from questions actually addressing both technologies. However, it is worth noting that this is not always the case. In the case of *Android|iOS*, most questions were not cross-platform questions but generic questions about mobile game development, with *Android* and *iOS* used together because of their duopoly on that market. As for the pair *C#|UnityScript*, it comes more often than not from questions where the asker wants to perform a specific task (with *Unity*) and is mentioning both languages, maybe out of ignorance of what to precisely do. Only a few questions are really about translations or focus on both languages. Finally, some cases involve pairings of versions/variants of the same technology. Such is the case for the pair *OpenGL-ES* and *OpenGL*, for which it is not clear why both tags would be used, since, as the *OpenGL* variant for embedded systems, *OpenGL-ES* already involves *OpenGL*. We could only speculate that these are attempts to get answers from the bigger (more generic) *OpenGL* crowd rather than just people interested into *OpenGL-ES*. This could also indicate a lower level of confidence of the asker as to which specific variant his question is more relevant for. Note that there were, however, some questions that are really and explicitly about both *OpenGL* and *OpenGL-ES*.

7.2. Concept–Concept Pairs: Handling Conjointly Two Concepts

When it comes to pairs of concept tags (Figure 9), the most obvious observation is that *3D* is often involved for cases in which support is degraded. The worst combination is *3D* and *collision-detection*, which results in increases in failure rates of about 16.5% for both tags, albeit with some upside in wait time (−99 min). This holds for tags as varied as *rendering* (+14.5%, −130 min), *mathematics* (+10%, +22 min), *physics* (+9.5%, +97 min), and *graphics* (+6.5%, +19.5 min), with the notable exception of *camera*, with *camera|3D* ranking as one of the most improved pairs (−11%, +227 min), although with worse wait times. Alternatively, many of the most improved combinations involve the tag *2D*, be it with *sprites* (−7%, +0 min), *vector* (−9%, −30 min), *movement* (−10%, −11 min), or *rotation* (−14%, −28.5 min).

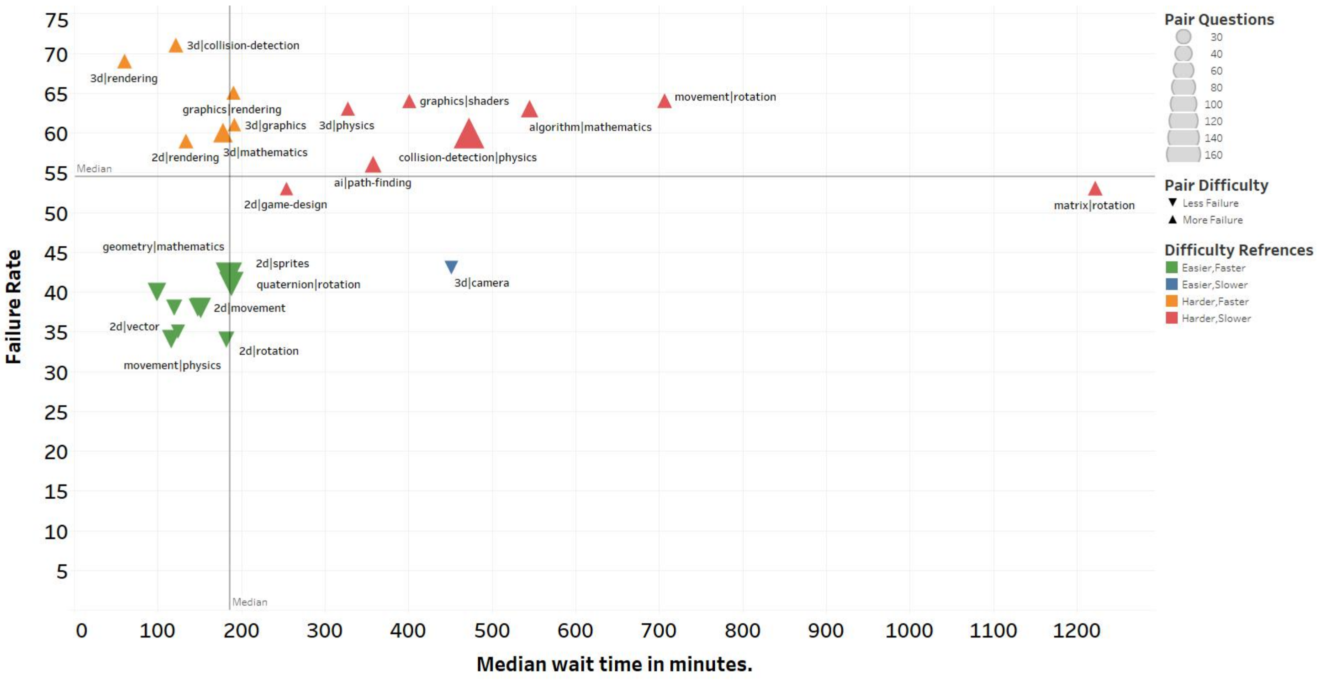


Figure 9. Community support for concept–concept pairs. Upwards (resp. downwards) triangles indicate cases where the pair has a failure rate at least 5 points higher (resp. lower) than any of its constituents. The size of the triangles is indicative of the number of questions for the tag pair. Colours indicate the relative support of the pair in the set of concept–concept pairs.

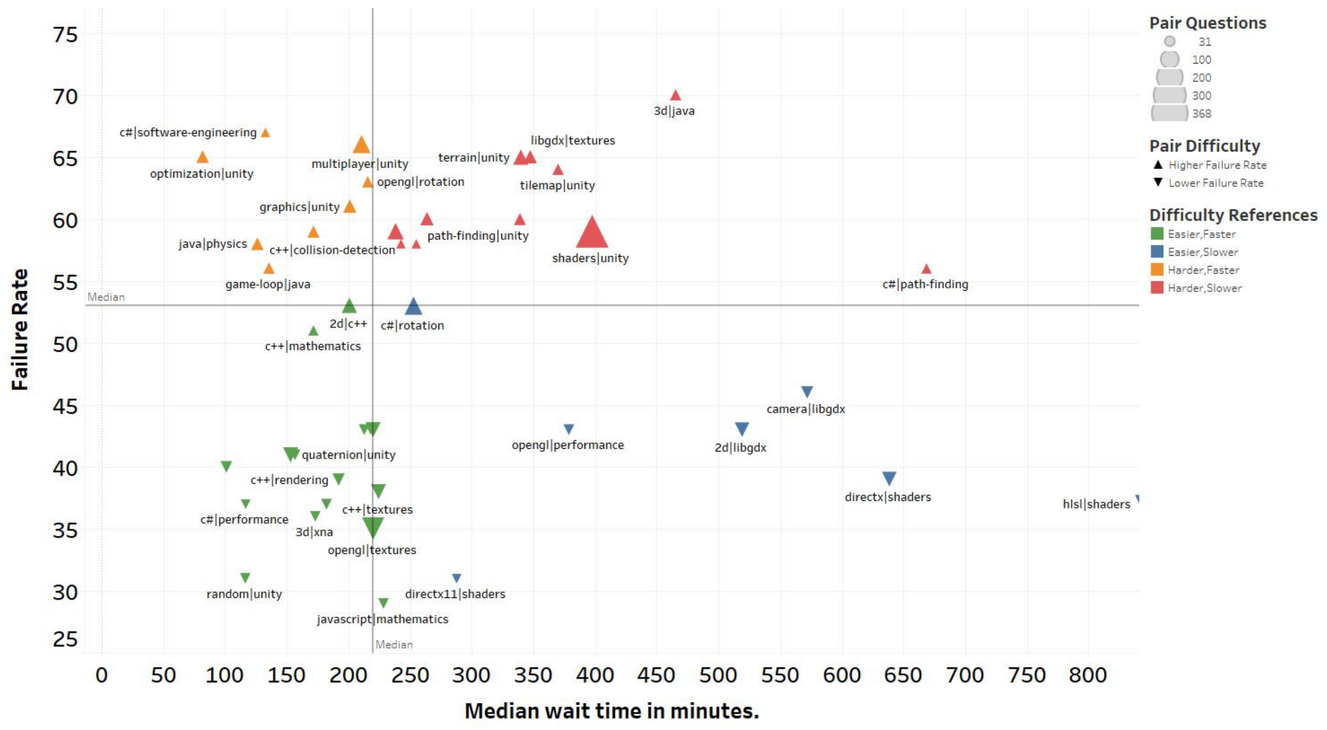


Figure 10. Community support for tech–concept pairs. Upwards (resp. downwards) triangles indicate cases where the pair has a failure rate at least 5 points higher (resp. lower) than any of its constituents. The size of the triangles is indicative of the number of questions for the tag pair. Colours indicate the relative support of the pair in the set of tech–concept pairs.

Also worth highlighting is the combination of *movement* and *rotation*, which is the second worst combination; it increases by 16% the failure rate of both tags and augments their wait times by 510 min on average. In GDSE, *movement* stands for translation ("*the change of position of an entity typically on the route of going from point A to point B*"). The combination of such translations and rotations appears to be, unsurprisingly, more challenging. The tag *rotation* is also involved in another worse-off combination: with *matrix*, it results in a 5.5% additional failure rate and 964 more minutes to wait for an answer.

As we did for *tech–tech* tags, we investigated cases where the tags are very related, with one implying the other. The only case here is *collision-detection* and *physics*: an analysis of the questions suggests that these are deeper questions involving *physics* engine and more advanced *collision detection* techniques.

7.3. Tech–Concept Pairs: Handling a Concept with a Given Technology

Figure 10 displays our results. The most difficult pair here is *3D*|*Java* (+17.5%, +249 min), much worse than *3D*|*C++* (+4.5%, + 315 min) and on the opposite side of *3D*|*C#* (−2.5%, −5.5 min) or *3D*|*XNA* (−14%, −40 min), which is one of the pair with the most improved support. These observations are undoubtedly indicators of interest for game developers about to embark on a *3D* game project and looking for the best languages/technologies. The same kind of observations can be made for concepts such as (i) *rotation*, which seems easier to perform with *Unity* (−5.5%, −45 min) than with *OpenGL* (+15%, −22 min) or (ii) *textures*, seemingly way easier with *OpenGL* (−13.5%, −3 min) than *Unity* (+6%, +52 min) or *libGDX* (+14.5%, +71 min).

Conversely, we can infer from our data some strengths and weaknesses of the various technologies. For instance *Unity* questions related to *optimization*, *multiplayer*, *tilemap*, and *terrain* appear way more challenging than those related to mathematical concepts such as *random*, *quaternion*, *transformation*, and *rotation*. Similarly, *C++* questions related to *collision-detection* have significantly less support than those related to *shaders*, *textures*, and *rendering*.

The findings from RQ3 can be summarised as follows:

- We uncovered technologies that, from a community support standpoint, seem to work well together (e.g., *C++* and *Box2D*, *HLSL* and *C++*) and some others that receive less support when present together (e.g., *Android* and *C#*, *HLSL* and *XNA*).
- We found that, from a community support standpoint, some game development concepts mesh well (e.g., *2D* and *movement*, *2D* and *rotation*) and some others not at all (e.g., *movement* and *rotation*, *3D* and *collision detection*).
- Finally, we uncover, from a community support standpoint, hints about which technologies may work better (e.g., *XNA* for *3D*, *OpenGL* for *textures*) or worse (e.g., *Java* for *3D*, *libGDX* for *textures*) when used for a particular concept.

8. Discussion

8.1. RQ1 Discussion

A first obvious line of inquiry relates to the comparison of concepts and technologies in the attention they command from the GDSE community. Our data show that when it comes to the number of questions, top tech tags are way ahead of top concept tags. This is unsurprising and suggests that most questions on that Q&A website are grounded in specific struggles with respect to a technological offer. In particular, one has to go down to the seventh top tech tag (*Android*) to find a tech tag with less questions than the top concept tag (*2D*). When it comes to averages, top concept tags fare way better, with a slight edge on average views and a very clear lead on favorites and score. This suggests that while many questions contain information about the environment or technology being used, the concepts (what people are trying to achieve) tend to attract more endorsements. In fact, based on all the 344 tags, techs have an average of 354 questions vs. 137 for concepts, but they lag in terms of average number of views (812 vs. 958 for concepts) and are about twice less favoured (1.28 vs. 2.15 for the score and 0.3 vs. 0.61 for the favorites).

In this RQ, we computed, for each tag, metrics that, we believe, inform about the community attention they command. An alternative way of investigating community attention could have consisted of analysing top questions and taking note of which tags appear the most often. Therefore, to complement our analysis, we considered the **top 100 questions** per number of views, score, and favourites and analysed the top 10 tags for both techs and concepts. We found that techs (such as *Unity* and *C#*) dominate on the ranking by number of views, while concepts (such as *game design*, *terminology*, *level design*, *mathematics*) dominate on the rankings by score and favorites. It should be noted that only a few concepts such as *gui*, *architecture*, *game mechanics*, and *performance* were not already present in the top concepts identified in Table 3 by our analysis.

Furthermore, we analysed in detail the **top 10 questions** per number of views, score, and favourites. Questions with the most views can be classified into (i) general inquiries ("*what is*", "*is it possible*") about a technology (e.g., "*What is Vulkan?*" or "*Is it possible to use C++ and Java for Unity*") or a concept (e.g., "*What is the difference between alpha and beta release*") and (ii) specific questions ("*where is*", "*how to*") about some specific task (e.g., "*Where is the Android sdk folder?*" or "*How to rotate the camera?*"). Somewhat similar to most viewed questions, questions with the highest scores often (4 cases out of 10) deal with inquiries inviting discussion and clarifications about some element ("*what are*", "*why is*") but also ask for help about how to accomplish some task (6 cases out of 10). Questions most marked as favourite overlap significantly with the most upvoted questions, but in our sample, they seem to have a higher share of "*how to*" questions (80%) than questions inviting discussions.

8.2. RQ2 Discussion

The results presented for RQ2 may suggest that the more questions a tag has attached to it, the higher its community support. However a correlation analysis found no such phenomenon. In fact, the various figures above also show many cases where tags with a small number of questions are actually well supported.

Beyond tags and their number of questions, we considered individual questions and the relation between their score and views and whether they receive accepted answers or not. We divided the questions into two sets (i.e., those with an accepted answer and those without) and computed their 5-number summary (minimum, first quartile, median, third quartile, maximum) plus the mean and standard deviation. Table 4 suggests that questions with accepted answers (*AA*) have slightly higher scores and number of views than those without (*noAA*).

To obtain some statistical confidence, we use the Mann–Whitney–Wilcoxon test (MWW), which is a non-parametric test able, in short, to tell us whether scores (or view counts) of questions with accepted answers are higher than those of questions without accepted answers. Additionally, to quantify the magnitude of such possible difference, we compute the effect size using another non parametric test, the Vargha–Delaney A (VD.A) [33]. We found that both scores and view counts of questions with accepted answers were indeed statistically higher but with negligible effect (VD.A of 0.51 for scores and 0.52 for views). This suggests that questions without accepted answers are not just the result of a lack of views or fewer upvotes.

Table 4. Scores and views for questions with accepted answers vs. those without accepted answers.

		min	**Q1**	**Median**	**Q3**	**max**	**avg**	**std**
Score	noAA	−18	0	1	2	132	1.52	3.93
	AA	−10	0	1	2	209	2.14	6.83
Views	noAA	17	163	424	1046	77,003	1110.79	2581.83
	AA	15	168	472	1301	129,994	1547.20	4423.81

noAA: No Accepted Answer, AA: Accepted Answer.

To better understand the data, we further analysed questions without accepted answers by considering their top 10 with respect to score and views.

Top scoring questions without accepted answers tend to be open-ended or asking for explanations, as illustrated below:

- *What are "affordances" in game design?*
- *What is an optimum failure rate that will keep people coming back to my game?*
- *How to avoid players getting lost in and/or bored by the meta game?*
- *Why do some games persistently have mostly one viable strategy, while others can have many?*

Out of the top 10 (score-wise) questions without accepted answers, only one referred to a specific technology : "*How do you handle aspect ratio differences with Unity 2D?*". (That question also stands out because, unlike the others, which generally receive an answer within the first two hours, that one took almost 6 days before a first answer.) The same question is also the most viewed question without an accepted answer. In fact, this fits into a larger pattern: unlike top scoring unanswered questions, **top viewed questions without accepted answers** are tied to a specific technology (*Unity* in most cases but also *Python* and *C#*). A few examples below:

- *Pygame for python 3.5?*
- *How to detect that user has touched UI canvas in Unity 4.6?*
- *How can i export Unity games to Android?*
- *How do I have a camera follow my object in Unity?*

Conversely, we looked at **questions with accepted answers**. For both score and views, the observations made above hold: questions with the highest scores tend to be high-level, open-ended, and centered on concept tags while questions with the most views are tied to a specific technology. This is unsurprising, as popular technologies tend to obtain more views but may be ultimately less specific to users' needs.

8.3. RQ3 Discussion

In a software development context where software is often the meeting of demands/requirements for a domain with the available technological offers, it is particularly important to analyse how well different requirements or concerns mesh (concept–concept), how well different technologies answer these demands (tech–concept), and, finally, how well these technologies can work together (tech–tech). Our analyses of pairs of tags and their findings provide a first step towards tooling that could help project developers, maintainers, and managers decide on which choices provide better community support.

To further our understanding, from an academic research perspective, of the interactions between tags, we conducted a preliminary additional investigation into combinations of tags. We looked into cases where questions were tagged only with concepts, only with techs, or with a mix of both. We found that about 22% were tagged with only concepts, 22% were tagged with only techs, and 56% had a mix of tech and concept tags. Questions tagged with only concepts are outstanding in a number of ways: in comparison to the other questions, they are slightly less viewed but have scores that are on average 2–3 times higher. Moreover, their failure rate is slightly lower.

To obtain a better understanding of the data and the tag combinations, we took interest in the top 100 questions by score. There were six tech-only questions, and they all received accepted answers; a few examples: "*What is Vulkan and how does it differ from OpenGL?*", and "*What happens when Time.time gets very large in Unity?*". Only seventeen questions have a mix of concept and tech tags, two of them without accepted answers. Examples of answered mixed questions include: "*How can I create a "see behind walls" effect?*" [tags: *Unity*, *shaders*, *graphics effects*], "*Is UDP still better than TCP for data-heavy realtime games?*" [tags: *c++*, *networking*, *udp*, *realtime*]. The two mixed questions without accepted answers are: "*How do you handle aspect ratio differences with Unity 2D?*", "*Why do tutorials use different approaches to OpenGL rendering?*". Regarding concept-only questions, 60 questions out of the 77 received accepted answers, including "*How can I store game metadata in a .png file?*", "*Why is it so bad to optimize too early?*". Concept-only questions without accepted answers include: "*How to avoid players getting lost in and/or bored by the meta game?*" and "*What is an optimum failure rate that will keep people coming back to my game?*". The vast majority of top scoring questions are relatively open-ended or asking for opinions, and we could not uncover any clear pattern that distinguishes questions that receive accepted answers from those that do not.

The above analysis highlights that questions that are detached from specific technology offerings generally score higher than those which are tethered to a given technology. In a way, it is not really surprising since answers to these questions may help developers, regardless of the different technology choices they may have already made. Nonetheless, we could not find in the data strong indications that questions focusing on concepts get significantly higher community support.

8.4. Alternative Classification

To further our analysis, we considered grouping the tags according to the classification in groups and types provided in [12], which is the most recent paper on video game development problems, as mined from postmortems. We used the categories Design, Implementation, and Business, which we derived from the papers, as follows. Design, which refers to game design without technical details, is taken from the group Production as is. Implementation is primarily about tools but also includes the types Technical and Testing from the group Production in [12]; we decided to have that single category for technical aspects because it was not feasible to separate those without looking at every single question. Our category Business is a group in [12], which includes the types Monetization and Marketing. We went through each tag in our data and assigned it to the above categories. Some tags belong to more than a single category. In particular, game genres were often a mix of Design and Implementation questions. The vast majority of tags (95%) are related to Implementation, including virtually all the tech tags. They account for 97% of the questions, are viewed on average 1301 times, and have an average score of 1.7 and a failure rate of 49.45%.

Around 10% of the tags (the percentages of the different categories add up to more than 100, since some tags belong to more than one category), such as *balance*, *character*, *game design*, and *level design*, are related to Design; these include tags whose questions often blend Design and Implementation questions, typically game genres. They account for 11% of the questions, are viewed on average 1334 times, and have an average score of 3.22 and a failure rate of 47.73%. Examples of the top scoring questions in that category include: "*How do I get players to say "no" when they are afraid of missing out on sidequests or XP?*", "*Is it unethical to make a game AI that is secretly non-competitive?*", and "*What is the design rationale behind melee retaliations in turn-based games?*". Top scoring questions with no accepted answers include questions such as "*How to avoid players getting lost in and/or bored by the meta game?*", "*What are the advantages of putting cheat codes into a game?*", and "*What is an optimum failure rate that will keep people coming back to my game?*"

Around 3% of the tags, such as *advertisements*, *copyright*, *marketing*, *monetization*, and *Steam*, are related to Business. They account for 2.2% of the questions, are viewed on average 1696 times, and have an average score of 3.35 and a failure rate of 49.3%. Examples of the top scoring questions in that category include: "*Should I worry about Youtube Let's Plays when I'm creating a story-heavy game?*", "*Making an indie with friends: Legal considerations*", and "*Do I need an Indie Studio Name?*". All the above questions received accepted answers. Looking specifically at top-scoring Business questions without accepted answers, we can highlight questions such as "*Why are microtransactions more or less universally hated?*" or "*What do I need to legally use copyrighted music in my game?*".

The analysis above shows that questions from GDSE are, perhaps unsurprisingly, mostly about technical aspects and implementation. Unlike in the postmortems analysed in [12], management issues, whether about features or people, are rarely asked about in GDSE. Nonetheless, roughly 10% of the questions are related to aspects that focus on the design of the game, and 2–3% are about business aspects. Notably, the questions about Design are more popular and receive satisfactory answers at a higher rate than technical questions. Overall, we believe that our study provides a complementary lens to the work conducted on postmortems, such as in [12], from the perspective of Q&A forums. Our emphasis is clearly more on technical aspects and at a lower granularity, but it is noteworthy that game design aspects in both [12] and this study account for roughly 1 out of 10 issues.

8.5. Implications

Implications for Video Game developers: Video Game development is attractive to many software programmers or engineers, who sometimes try it out without a clear picture of the topics they may encounter. Our empirical study focuses on a prominent game development Q&A forum (GDSE) and conveys, from that perspective, insights to new and seasoned game developers on concepts they may have to understand and technologies they may have to use. While around 10% of the questions are about game design aspects that are not necessarily technical, most questions on GDSE involve game engines, libraries, and languages. In that sense, the careful selection of the tools and technologies, for any given game development project, is thus of the utmost importance. Our identification of concepts, technologies, and their co-occurrences provide game developers a preliminary understanding of the challenges they may face when they will be trying to implement a given concept using a given technology or when they will be trying to integrate different technologies. In this paper, we highlighted some cases that we found most meaningful, but we made our dataset available, so that game developers can use it to seek insights relatively to their specific context.

Implications for Video Game Researchers: To the best of our knowledge, post-mortems of video game development projects have, so far, been the only source of empirical data used to understand video game development issues from a software engineering perspective. Our study establishes Q&A websites as valuable data sources that can be used for insights on game development topics and issues. While such data heavily skew technical, we believe they are very much needed for analyses at a finer granularity, with details about technologies used and specific concepts implemented by game developers. Our study provides the video game research community with some statistics that can inform future research as to which tools and concepts are worth dedicated studies.

We uncovered or confirmed the dominance of technologies such as *Unity*, but we also put the spotlight on concepts such as *collision detection*, and *animation*. We notably found that concepts that can only be found in a gaming context, such as *leaderboards*, or *non-player character*, only appear in about one-eighth of the questions. Nevertheless, we believe that, most importantly and from a software engineering perspective, our study documents the significant differences in community attention and support for related concepts or rival technologies. We believe that our study lays the foundation for future research work on recommendation approaches for tool selection and tool integration in a video game development context. Furthermore, our data on degraded community support when some specific technologies are involved (such as *HLSL* and *MonoGame*) could be the starting point of studies dedicated to further exploration of tool integration issues in a video game development context, and possibly beyond.

9. Threats to Validity

We now outline the potential threats to the validity of our results.

Internal Validity: For this study on the topics and issues faced by game developers, we relied on tags assigned to questions posted on the online forum Game Development Stack Exchange. Our reliance on tags makes this study vulnerable to possible tag omissions by question askers. However, we believe alternatives such as topic analysis through techniques such as LDA come with their own pitfalls, notably an oversized dependence on chosen parameters and human post-processing. Overall, we were willing to trade the possibly higher recall of advanced topic discovery techniques for the better precision and reduced noise that can be delivered through tag analysis.

Construct Validity: Our study initially set out to investigate popularity and difficulty of game development issues by focusing on various metrics computed on tags assigned to questions by posters. However, we ultimately decided against using prominently either term: popularity or difficulty. First, popularity may not be the best term as we are in a context of developers encountering problems. However, aside from that semantic problem, there could be questions about whether the metrics we used and proposed really measure popularity and difficulty. To mitigate these concerns, we substituted popularity for community attention and difficulty for community support. About community attention, we believe we used a wide variety of metrics that paint an accurate picture of which issues come up the most (number of questions, views) and are the most relevant (score, number of times the questions are marked as favourite). As for community support, we used the failure rate and median wait time for questions related to a given tag (or pair of tags). Still, lack of community support may occur for specific reasons pertaining more to the questions than the tags they involve: the questions could be badly formulated or duplicates, etc. However, the statistical analysis we conducted in Section 8.2 shows that questions without accepted answers are only marginally less endorsed (lower score) than those with accepted answers, suggesting that bad community support goes beyond "bad" questions.

External Validity: A potential external validity threat for our study is that we exclusively relied on data from the GDSE community. Such a choice allows for more control and certainty about the data analysed, but the GDSE community is relatively small compared to the Stack Exchange flagship that is Stack Overflow. However, game development is a relatively orphaned topic in software engineering, and we wanted this preliminary study to avoid the noise that would come with the (even targeted) analysis of more general-purpose forums. It is also worth stressing that as a self-described “question and answer site for professional and independent game developers”, GDSE represents a community more committed to game development than users on Stack Overflow. That being said, we plan on conducting a follow-up study on other programming forums, in particular Stack Overflow.

10. Conclusions and Future Work

10.1. Conclusions

Video-game development has been reported by software engineering (SE) research to be of a different nature than the traditional software development in terms of design, build, testing, and release. Even though video-game development is a big and expanding segment of the development market and a 100-plus billion-dollar industry, it has attracted only marginal attention from SE researchers. In this work, we analysed data from the Stack Exchange Q&A website dedicated to game development (GDSE), to identify the topics which receive the most attention and support from a prominent community of video-game developers. Our framework analysis is based on tags explicitly assigned to questions and the distinction between technologies used (techs) and game development concerns (concepts). Our study seeks to shed more light on video game development by highlighting the topics game developers address on Q&A websites dedicated to game development. We believe our findings can help game developers on what to expect and on which concepts and technologies are the most popular and the most supported. As for researchers, this study aims to provide them with complementary perspective and data, aside from game development postmortems, to investigate this understudied domain and address topics of interest for game development communities.

Our analysis revealed that community attention goes to technologies and languages such as *Unity*, *C#*, *OpenGL*, *Windows*, and *Lua*,as well as concepts such as *2D*, *3D*, *collision detection*, *shaders*, *level design*, and *monetisation*. As for community support, we found, for instance, that the game engine *Unreal-4* has a much lower support than its rival *Unity*, while Android-related tags are generally better supported than *iOS*-related tags. As for concept tags, we found that 3D questions have unsurprisingly less support than 2D, whereas purely mathematical concepts generally have good support; additionally, relatively new paradigms such as virtual reality and augmented reality lack community support. For both concept and tech tags, popular tags generally have a failure rate around 50% ($\pm$ 10%) and a median wait time between 30 min and 6 h (for concept tags) and 2 and 7 h (for tech tags). Finally, we considered interactions between tags and analysed pairs of tags involving two tech tags, two concept tags, or mixing techs and concepts. We found that analysis particularly insightful and consider this particular look a major contribution of the present paper. In the context of community support for video game development, we were able to highlight technologies that work well together (e.g., *C++* and *Box2D*, *HLSL* and *C++*) or not (e.g., *Android* and *C#*, *HLSL* and *XNA*), game development concerns that mesh well (e.g., *2D* and *movement*, *2D* and *rotation*) or not (e.g., *movement* and *rotation*, *3D* and *collision detection*) but also hints about which technology may work better (e.g., *XNA* for *3D*, *OpenGL* for *textures*) or worse (e.g., *Java* for *3D*, *libGDX* for *textures*) with a particular concern.

10.2. Future Work

We plan to extend our work as follows. First, we intend to work on a fine-grained mapping of concepts and technologies relevant to game development. We would use this as a first step towards a recommendation approach able to orient, from a community support perspective, game developers towards technologies that correspond to their needs and integrate well. Longer term, our vision is that of a tool which would take input from game developers and managers as to the concepts they want to handle and the technologies they are eyeing and which would return the additional concepts they may have to consider and the tools that would complement their existing set. Such a tool could make use of advanced machine learning techniques to predict the likelihood of fast and effective community support for a given software development stack.

With respect to next-generation computing and emerging areas, as outlined in [34], such as cloud computing, artificial intelligence, and virtual reality, the data from GDSE do not validate them as major concerns among game developers. With the possible exception of artificial intelligence, which does appear in many questions, these areas are not very prominent in the GDSE forum. A possible reason is that these emerging technologies are still either very niche and/or require resources and investments beyond those of most gaming projects.

More broadly, and not restricted to game development, we plan to conduct similar tag-based analyses on Stack Overflow and investigate what kinds of insights it can yield, particularly with respect to tag interactions and what they suggest about technology and concept compatibility.

Author Contributions: Conceptualization, methodology, investigation, data curation, writing—original draft preparation, F.A.; conceptualization, methodology, investigation, data curation, writing the manuscript, reviewing, editing, supervision, S.K.; conceptualization, supervision, reviewing and editing, G.E.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) RGPIN-2015-03858 .

Data Availability Statement: The data that support the findings of this study are available at [31].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ampatzoglou, A.; Stamelos, I. Software engineering research for computer games: A systematic review. *Inf. Softw. Technol.* **2010**, *52*, 888–901. [CrossRef]
2. Aleem, S.; Capretz, L.F.; Ahmed, F. Game development software engineering process life cycle: A systematic review. *J. Softw. Eng. Res. Dev.* **2016**, *4*, 1–30. [CrossRef]
3. Lewis, C.; Whitehead, J.; Wardrip-Fruin, N. What went wrong: A taxonomy of video game bugs. In Proceedings of the Fifth International Conference on the Foundations of Digital Games, Monterey, CA, USA, 19–21 June 2010; pp. 108–115.
4. Pascarella, L.; Palomba, F.; Di Penta, M.; Bacchelli, A. How Is Video Game Development Different from Software Development in Open Source? In Proceedings of the 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), Gothenburg, Sweden, 27 May–3 June 2018; pp. 392–402.
5. Murphy-Hill, E.; Zimmermann, T.; Nagappan, N. 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, Hyderabad, India, 31 May–7 June 2014; pp. 1–11.
6. Callele, D.; Neufeld, E.; Schneider, K. Requirements engineering and the creative process in the video game industry. In Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05), 29 August–2 September 2005; pp. 240–250.
7. Petrillo, F.; Pimenta, M.; Trindade, F.; Dietrich, C. What went wrong? A survey of problems in game development. *Comput. Entertain. (CIE)* **2009**, *7*, 1–22. [CrossRef]
8. Kanode, C.M.; Haddad, H.M. Software engineering challenges in game development. In Proceedings of the 2009 Sixth International Conference on Information Technology, New Generations, Las Vegas, NV, USA, 27–29 April 2009; pp. 260–265.
9. Washburn, M., Jr.; Sathiyanarayanan, P.; Nagappan, M.; Zimmermann, T.; Bird, C. What went right and what went wrong: An analysis of 155 postmortems from game development. In Proceedings of the 38th International Conference on Software Engineering Companion, Austin, TX, USA, 14–22 May 2016; pp. 280–289.
10. Politowski, C.; Fontoura, L.; Petrillo, F.; Guéhéneuc, Y.G. Are the old days gone? A survey on actual software engineering processes in video game industry. In Proceedings of the 5th International Workshop on Games and Software Engineering, Austin, TX, USA, 16 May 2016; pp. 22–28.
11. Politowski, C.; Petrillo, F.; Ullmann, G.C.; de Andrade Werly, J.; Guéhéneuc, Y.G. Dataset of video game development problems. In Proceedings of 17th International Conference on Mining Software Repositories, Seoul, Korea, 29–30 June 2020; pp. 553–557.
12. Politowski, C.; Petrillo, F.; Ullmann, G.C.; Guéhéneuc, Y.G. Game industry problems: An extensive analysis of the gray literature. *Inf. Softw. Technol.* **2021**, *134*, 106538. [CrossRef]
13. Treude, C.; Barzilay, O.; Storey, M.A. How do programmers ask and answer questions on the web?: Nier track. In Proceedings of the 2011 33rd International Conference on Software Engineering (ICSE), Honolulu, HI, USA, 21–28 May 2011; pp. 804–807.
14. Barua, A.; Thomas, S.W.; Hassan, A.E. What are developers talking about? an analysis of topics and trends in stack overflow. *Empir. Softw. Eng.* **2014**, *19*, 619–654. [CrossRef]
15. Linares-Vásquez, M.; Dit, B.; Poshyvanyk, D. An exploratory analysis of mobile development issues using stack overflow. In Proceedings of the 10th Working Conference on Mining Software Repositories, San Francisco, CA, USA, 18–19 May 2013; pp. 93–96.

16. Beyer, S.; Pinzger, M. A manual categorization of android app development issues on stack overflow. In Proceedings of the International Conference on Software Maintenance and Evolution (ICSME), Victoria, BC, Canada, 29 September–3 October 2014; pp. 531–535.
17. Rosen, C.; Shihab, E. What are mobile developers asking about? a large scale study using stack overflow. *Empir. Softw. Eng.* **2016**, *21*, 1192–1223. [CrossRef]
18. Bajaj, K.; Pattabiraman, K.; Mesbah, A. Mining questions asked by web developers. In Proceedings of the 11th Working Conference on Mining Software Repositories, Hyderabad, India, 31 May–1 June 2014; pp. 112–121.
19. Venkatesh, P.K.; Wang, S.; Zhang, F.; Zou, Y.; Hassan, A.E. What do client developers concern when using web apis? an empirical study on developer forums and stack overflow. In Proceedings of the 2016 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 27 June–2 July 2016; pp. 131–138.
20. Mehrab, Z.; Yousuf, R.B.; Tahmid, I.A.; Shahriyar, R. *Mining Developer Questions about Major Web Frameworks*; Sience and Technology Publications: LdaSetùbal, Portugal, 2017; pp. 191–198.
21. Almansoury, F.; Kpodjedo, S.; Boussaidi, G.E. Investigating Web3D topics on StackOverflow: A preliminary study of WebGL and Three. js. In Proceedings of the The 25th International Conference on 3D Web Technology, Virtual Event, Korea, 9–13 November 2020; pp. 1–2.
22. Kochhar, P.S. Mining testing questions on stack overflow. In Proceedings of the 5th International Workshop on Software Mining, Singapore, 3 September 2016; pp. 32–38.
23. Shariff, S.M. Investigating Selenium Usage Challenges and Reducing the Performance Overhead of Selenium-Based Load Tests. Doctoral Dissertation, Queen's University, Kingston, ON, Canada, 2019.
24. Yang, X.L.; Lo, D.; Xia, X.; Wan, Z.Y.; Sun, J.L. What security questions do developers ask? a large-scale study of stack overflow posts. *J. Comput. Sci. Technol.* **2016**, *31*, 910–924. [CrossRef]
25. Fischer, F.; Böttinger, K.; Xiao, H.; Stransky, C.; Acar, Y.; Backes, M.; Fahl, S. Stack overflow considered harmful? the impact of copy&paste on android application security. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 25 May 2017; pp. 121–136.
26. Meng, N.; Nagy, S.; Yao, D.; Zhuang, W.; Argoty, G.A. Secure coding practices in java: Challenges and vulnerabilities. In Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, 27 May 2018; pp. 372–383.
27. Lopez, T.; Tun, T.T.; Bandara, A.; Levine, M.; Nuseibeh, B.; Sharp, H. An investigation of security conversations in stack overflow: perceptions of security and community involvement. In Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment, Gothenburg, Sweden, 27 May 2018–3 June 2018; pp. 26–32.
28. Vasilescu, B. Academic Papers Using Stack Exchange Data. 2020. Available online: https://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data (accessed on 26 June 2020).
29. Tamla, P.; Böhm, T.; Nawroth, C.; Hemmje, M.; Fuchs, M. What Do Serious Games Developers Search Online? A Study of GameDev StackExchange. In Proceedings of the CERC, Darmstadt, Germany, 29–30 March 2019; pp. 131–142.
30. Sekaran, U.; Bougie, R. *Research Methods for Business: A Skill Building Approach*; John Wiley & Sons: Hoboken, NJ, USA, 2016.
31. Almansoury, F.; Kpodjedo, S. Game Developers Issues. 2020. Available online: https://zenodo.org/record/3908627#.Yy7U_ORBxPY (accessed on 1 January 2020).
32. Mamykina, L.; Manoim, B.; Mittal, M.; Hripcsak, G.; Hartmann, B. Design lessons from the fastest q&a site in the west. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada, 7–12 May 2011; pp. 2857–2866.
33. Vargha, A.; Delaney, H.D. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *J. Educ. Behav. Stat.* **2000**, *25*, 101–132.
34. Singh Gill, S.; Xu, M.; Ottaviani, C.; Patros, P.; Bahsoon, R.; Shaghaghi, A.; Golec, M.; Stankovski, V.; Wu, H.; Abraham, A.; et al. AI for Next Generation Computing: Emerging Trends and Future Directions. *Internet Things* **2022**, *19*, 100514. [CrossRef]