Link Between Gaming Communities in YouTube and Computer Science

Lassi Haaranen and Rodrigo Duran
Department of Computer Science, Aalto University, Helsinki, Finland

Keywords: Computer Science Education, Game-based Learning, Informal Learning.

Abstract: Playing games has become a permanent part of popular culture, and the number of players keeps increasing. Part of this phenomenon is the act of recording gameplay and sharing the videos creating online gaming communities. We describe different types of gaming videos that intersect with learning computer science (CS). In addition, we looked at the discussions those videos spurred and found a rich interaction with CS topics. Since games can act as an engaging environment for informal learning, we conclude that the gaming community has relevant connections to learning CS and we as the research community should pay more attention to this phenomenon.

1 INTRODUCTION

Digital games are an important part of the modern day culture. Whereas once video games were viewed as a pastime for the children, this picture has dramatically shifted in the past decades. In the United States alone, it is estimated that the amount of money consumers spent on games exceed 22 billion dollars, the average age of gamers is 35 years, and approximately 155 million Americans play video games (Entertainment Software Association, 2015).

But just because the average age of a gamer keeps climbing up, it does not mean that children are not playing. New York Times published a long article describing the 'Minecraft Generation' (Thompson, 2016) – a generation building their own visions in virtual block worlds. And the culture of video games is not just contained in the games and the worlds they create. Different fan sites and discussion forums have existed for a long while, but in more recent years video sharing services, YouTube in particular, have become an important part of the cultural phenomenon of video games.

Videos related to gaming have spawned their own industry. Google, owner of YouTube, itself acknowledged the importance of games creating its first spinoff platform from YouTube exclusively dedicated to games, YouTube Gaming (https://gaming.youtube.com/). As an example, there are dozens of YouTube channels focusing on Minecraft that have more than a million subscribers and correspondingly new videos from those channels receive millions of views. Typically these channels focus on gameplay videos, usually in the form of "let's play" where the author records him/herself playing a game and narrating the events at the same time. Whilst majority of these channels and videos are purely for entertainment, there are some that have educational value as well. For example, user SethBling has created a four-part video tutorial series[1] on how to replicate a Turtle-like programming environment in MineCraft. The first part of the series detailing the use of `if-else` statements has gathered over 700 000 views, so the potential impact on learning to program could be substantial.

Given how popular gaming and the videos related to it are, and especially since quite a few of them contain instructional material regarding computing and programming, we wanted to investigate this phenomenon further. Our goals are summarized by the following research questions:

- **RQ1** What kind of gaming videos there are that are related to computer science?
- **RQ2** How are the commenters discussing these videos?

The overall goal is to *describe and examine the phenomena of gaming videos related to learning computing*. We are particularly interested in informal learning from the videos, in other words, their use outside of classrooms and formal education.

[1]https://www.youtube.com/playlist?list=PL2Qvl4gaBge02Eh4AqtDSWg3sojt3jeRO

Haaranen, L. and Duran, R.
Link Between Gaming Communities in YouTube and Computer Science.
DOI: 10.5220/0006267000170024
In *Proceedings of the 9th International Conference on Computer Supported Education (CSEDU 2017) - Volume 2*, pages 17-24
ISBN: 978-989-758-240-0

The rest of the article is structured as follows: In Section 2 we provide relevant research linking games and learning computer science. Section 3 describes our data collection and analysis and Section 4 summarizes our results. Finally, we conclude by discussing our findings in Section 5.

2 BACKGROUND

In this section, we first present relevant literature regarding the use of games in computer science education. After that, we discuss previous work on informal learning contexts. Finally, we explore research done on YouTube videos and comments done in other domains.

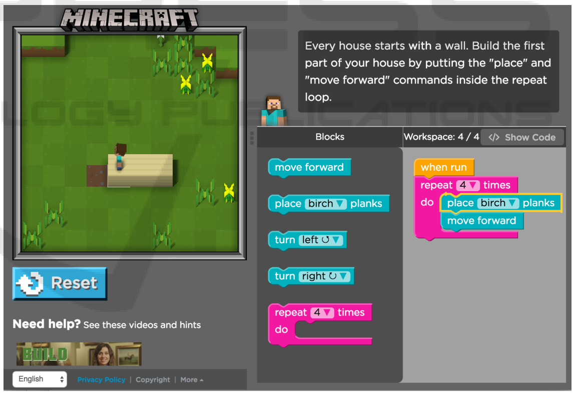
2.1 Games and Computer Science Education

There has been very little research in learning programming or computing in informal contexts. However, in more formal context the research on using games to teach has been focused on programming and computer science. Hayes and Games reviewed the use of games in education and identified four different goals for using games: making games as a context to learn programming, as a way to interest girls in programming, as a way to learn in other academic domains, and as a way to understand design concept (Hayes and Games, 2008). They state that the most commonly sought after learning goal was related to learning programming or concepts related to it.

Use of game elements (achievements, badges, rankings) in non-game environments (such as projects and classroom lessons) has attracted a great interest of education community in recent years embedded in the concept known as gamification (Deterding et al., 2011). Although being used in many diverse aspects, gamification applied to programming attempted to introduce computing skills, especially to younger students and novice programmers, in a more palatable fashion with different interfaces, objectives, and outcomes. Tillmann et al. describe how their new gamified platform, Code Hunt (http://www.codehunt.com) can help beginners to understand computing concepts in the form of numerical output puzzles using Java and C# (Tillman et al., 2014) . Engagement to the platform comes from the introduction of a ranking system that rewards programmers that create elegant programs with fewer attempts.

Using full games, instead of gamified applications, as an engaging environment to promote learning of computer skills has been discussed for decades (Kelleher and Pausch, 2005). When it comes to formal education and computer science courses, four ways of utilizing computer games as part of the learning process has been described (Wallace et al., 2010). In the first approach, students create their own games in order to learn (1). And relatedly, in the second group they implement a critical part of a given game (2). In the third approach, students learn by implementing an agent to a game (3). This group is of particular interest given that this can be done in many of the current games meant for entertainment. In the last category, students learn by playing a serious game that has been designed to teach particular concepts (4).

Code.org (https://code.org/learn) iniative hosts several games that support programming learning through games. Most of these games use mechanics similar to the popular Lightbot (https://lightbot.com) game where users need to provide a sequence of commands so a robot can reach a specific point on the map. To address syntax issues that young users might have most of these games use a block-based interface. Figure 1 shows an example of a Minecraft-themed game by Code.org, where users solve specific tasks like constructing or moving in the map using blocks interface.


MINECRAFT
Reset
Need help? See these videos and hints
BUILD
English
Privacy Policy | Copyright | More
Blocks
Workspace: 4 / 4
Show Code

Every house starts with a wall. Build the first part of your house by putting the "place" and "move forward" commands inside the repeat loop.

```
move forward
place birch planks
turn left
turn right
repeat 4 times
do
```

```
when run
repeat 4 times
do  place birch planks
    move forward
```

Figure 1: Code.org Minecraft themed sequence commands game.

2.2 Games Enhanced by Programming

Although the literature of using games to teach programming in a formal setting as a development tool is rich, in recent years a new approach has emerged: using programming inside games as a way to provide context to programming, as well as, to enhance the play itself. This approach is distinct from the previous experiences using games not as a development environment or an educational game to teach a specific subject, but as a skill to improve features on the act of play itself.

Minecraft, from Mojang, now the 3rd most sold game of all time, allows deep modifications of the game so researchers use its features as building blocks to provide a constructionist experience to learn programming. A tailored version of Minecraft called ComputerCraftEdu provides insight on how block-based programming performs versus textual programming in a game environment, showing that block-based programming increases interest in programming compared to textual in the same setting (Saito et al., 2015).

Zorn et al. developed their own block-representations of programming constructs that function similarly to normal Minecraft blocks (Zorn et al., 2013). This enabled users to create code in the same manner they construct originally in Minecraft. As in ComputerCraftEdu, the goal of the research was to better understand how block-based versus textual language performs in terms of students appreciation and if learning style affected this appreciation. Results show that programming appreciation increased with the use of their blocks programming tool.

It is interesting to highlight that those features described in Saito et al. (2015) and Zorn et al. (2013), although studied as teaching tools, could be used to automatize tasks executed extensively by players, such as mining and constructing. This intrinsic correlation with in-game functions creates the potential use of those features in diverse aspect from the experiences previously described in the literature, promoting more engagement and even reaching a distinct audience that incidentally discovered computing subjects.

Another example of games that use programming as an intrinsic part of its mechanics is CodeSpells (https://codespells.org/). Starting as a research project and later evolving into a full commercial game, the concept relies on creating powerful magic spells using programming (Esper et al., 2014). A programming language is used to describe the properties and actions performed by a spell that users can freely create and manipulate. In that sense, learning programming happens due to the built-in game mechanics.

2.3 Learning Computer Science in Informal Settings

The previous examples are from a formal context, i.e. courses or outreach activities specifically aimed at teaching or improving the perception of programming and computer science. However, we are specifically interested in informal learning in computer science but previous research on the subject is very sparse. One example where informal learning in computing has been studied comes from a short report that looked at how elementary school students acquired programming knowledge (Boyle and McDougall, 2003). Although this report looked at the informal sources used to gain programming knowledge, the knowledge was used and practiced in a formal classroom setting as well.

McCartney et al. investigated what motivates computing students to learn in informal and self-directed settings. Using questionnaires with 17 selected students, a thematic analysis was performed and five different topics emerged (social influence, projects, joy of learning and fear) to understand why students engaged in this particular learning setting (McCartney et al., 2016). Evidence points to that learning with projects, in different nuances, is pivotal in self-directed learning and could have an impact in classroom pedagogies.

Whilst the actual learning process in informal settings has not been studied in detail, there is at least evidence that gaming communities and forums can host a high-level discussion and scientific argumentation. Investigation on discussion forum of a popular massively multiplayer online game World of Warcraft and found that at least in that one particular forum, more than 80 percent of the discussion was centered around social knowledge construction (Steinkuehler and Duncan, 2008). And importantly, over a half of the forum posts contained systems based reasoning, as well as, a few posts containing model-based reasoning of game mechanics that the players were trying to uncover.

2.4 YouTube as a Community for Learning

One of the important features of YouTube is the low barrier to entry to produce and consume videos (Chau, 2010). Videos and comments to those can be easily added and circulated amongst peer groups. And one of the main categories of videos he identified was the 'informal mentorship' in the form of how-to videos.

In other domains, there is some research on using YouTube as an educational channel. In medicine, publishing videos on clinical skill training on YouTube instead of peer-reviewed services has been investigated, in hopes of attracting a considerably larger audience (Topps et al., 2013). They found the results promising and the number of views encouraging, but naturally assessing educational impact is difficult with anonymous viewers.

3 METHODS

To gain an understanding on how gaming videos and learning programming are connected we used a mixed method approach. We first collected and categorized videos based on their content to gauge the prevalence of game videos that related to programming or computer science in some way. After this step, we selected two videos and used a grounded theory based analysis to have a more detailed picture of the phenomena. This section first describes the classification of the videos and then explains the method of the comment analysis in more detail.

3.1 Classification of Videos

Video categorization began by both authors reviewing together an initial set of 50 gaming videos selected using the 'programming' query. Through discussion of the titles and content of the videos, five categories of gaming videos that pertain to programming were created. This categorization scheme was used to categorize a larger sample of videos. The reliability of this categorization was tested by evaluating inter-rater reliability. The five categories to classify the larger set of videos were:

- **Gaming enhanced by programming** Using programming and/or computer science concepts inside a specific game, to improve gameplay
- **Game programming tutorial** General tutorials for game programming, teaching game development whilst not being related to any specific game
- **Modding tutorials** Tutorials teaching how to create a modification (mod) for a specific existing game
- **Game programming discussion** Discussions on general level what it means to be a game programmer and what the field is like
- **Other** Unrelated videos or videos that did not fit any of the above categories

We used the query 'programming' and gathered the first 400 videos for classification. Whilst searching we used the default parameters available (sorted by relevance and no limits to when the video was uploaded). However, it should be noted that the information on how exactly YouTube orders the search results is not available.

Both authors independently labeled the 400 preselected videos. After labeling, we resolved any conflicts through discussion. Consensus and the results of the categorization are presented in Section 4.1.

3.2 Grounded Theory and Comments

In addition to looking at the videos as a whole, we analyzed the discussions in the comment sections to provide a richer description of the phenomena. To do this we adopted a grounded theory approach. Grounded theory as a method aims at providing a rich description and explanation of a phenomenon and it is also useful for generating hypotheses. At the core is the idea of building a theory that is grounded in the gathered data. The procedure of grounded theory is to analyze the data by firstly coding it openly and then returning to the codes and doing axial coding. In axial coding, the codes are categorized to extract themes and concepts of interest. For a discussion on grounded theory specifically in computing education, see (Kinnunen and Simon, 2010).

We started our open coding by selecting four videos from the games enhanced by programming category and one video from the modding category. The selection of these two categories highlights the informal setting we are investigating. These types of videos enable a diverse type of content where learning is incidental to the task. The initial open coding resulted in ten different categories for comments, such as 'joke', 'reflect', 'question' etc. Whilst this open coding did describe the contents of the discussions it did so in very broad terms. However, it did not highlight the importance of discussing computer science and programming concepts which were common in the comments.

To improve the coding and produce more useful axial coding, we selected two videos where programming and games were mixed and that had a large number of comments. The key difference this time was that we only included comments that were directly related to computer science or programming, thus enabling us to focus on coding the relevant content. Based on the original set of codes and the codes generated during this step, we created the axial coding that is presented in Section 4.2.

After the open and axial coding we had the key themes of the discussions, or more precisely in our case we had identified different types of discussions. The third analysis phase of grounded theory is the selective coding, where a core category is selected that describes the phenomena and the rest of the categories are integrated and reflected in light of this core category. However, since the categories of discussions that we found were so different from each other we decide to forgo this step and focus on providing a description communication in the comments.

4 RESULTS

4.1 Categorizing Game Videos Related to Computing

Both authors coded the list of 400 videos individually, whenever the title of the video clearly showed the category it belonged to it was used. In the cases, where the title was not enough to classify the video, the content of the video was reviewed. After the individual coding (Cohen's $\kappa = 0.619$) the conflicts were resolved through discussion. Table 1 summarizes the distribution of the videos.

Table 1: The distribution of the videos.

Category	#	%
Games enhanced by Programming	38	9.50
Game programming tutorial	242	60.50
Modding tutorials	7	1.75
Game programming discussion	56	14.00
Other	57	14.25

As can be seen in Table 1, the majority of the videos are in game programming tutorials which usually come in longer series having multiple videos. However, almost tenth of the videos were about commercial games meant for entertainment (not serious games) that incorporated programming in some form.

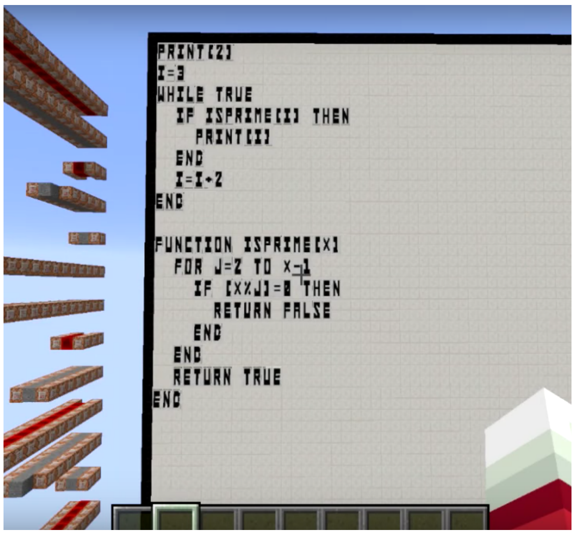
4.2 Discussions Related to The Videos

To provide a richer description of the gaming community intersecting with computer science, we looked at the discussions in the comment sections of two different videos. The videos were chosen to highlight two different types of interaction with computing. Any of the popular videos could have been selected but we felt that these two videos provided a good variety in both discussion and the content of the video itself. The content of the selected videos are detailed next.

V1 BASIC Interpreter in Minecraft

The first video chosen is called "BASIC Programming Language in Minecraft" from the channel SethBling, seen in Figure 2. At the time of writing (July 2016), the video has garnered over half a million views and hundreds of comments. The running time of the video is 13'45" and it focuses on how the author implemented an interpreter for BASIC programming language inside MineCraft.

The first part of the video focuses on how the interpreter was created in the game and how to use it to

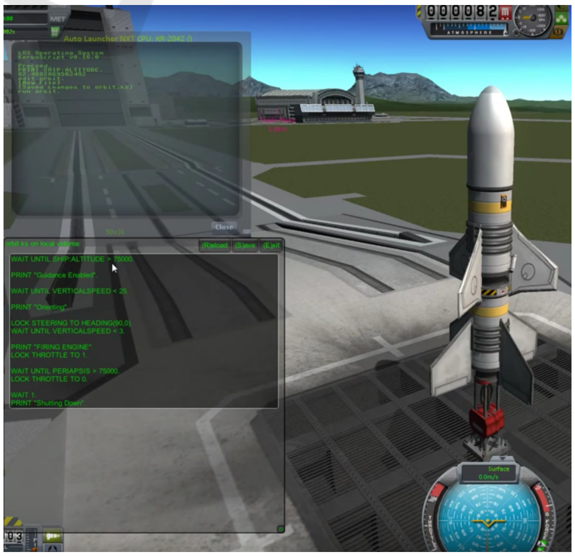

```
PRINT(2)
I=3
WHILE TRUE
  IF ISPRIME(I) THEN
    PRINT(I)
  END
  I=I+2
END

FUNCTION ISPRIME(X)
  FOR J=2 TO X-1
    IF (X%J)=0 THEN
      RETURN FALSE
    END
  END
  RETURN TRUE
END
```

Figure 2: BASIC Interpreter in Minecraft.

run programs. The author showcases the interpreter by writing a simple function that calculates whether a number is prime and also touches some advanced topics, such as abstract syntax tree, the difference between compiled and interpreted language. The video also contains a section with the author explaining how this BASIC interpreter can be used to control a turtle inside MineCraft, demonstrated by a loop with commands for the turtle to move in a particular pattern and mine a route through the virtual blocks of MineCraft.

V2 Programming Rockets in Kerbal Space Program



```
WAIT UNTIL SHIP:ALTITUDE > 75000.

PRINT "Guidance Enabled".

WAIT UNTIL VERTICALSPEED < 25.

PRINT "Orienting".

LOCK STEERING TO HEADING(90,0).
WAIT UNTIL VERTICALSPEED < 3.

PRINT "FIRING ENGINE".
LOCK THROTTLE TO 1.

WAIT UNTIL PERIAPSIS > 75000.
LOCK THROTTLE TO 0.

WAIT 1.
PRINT "Shutting Down".
```

Figure 3: Programming Rockets in Kerbal Space Program

In the second video (seen in Figure 3) the author, Scott Manley, describes the usage of kOS mod for Kerbal Space Program. The game consists of designing and flying rockets and kOS is a modification for the game which brings a programmable computer that can be used to automate tasks. The total length of the video is 8'11", and at the time of writing (July 2016), it has gathered over 120 thousand views and more than 400 comments.

The video showcases a simple script written for the kOS that guides the rocket to an orbit around the planet. First, in the video, the rocket is launched. After a while, it reaches the highest point in the ballistic trajectory, and the script takes control of the ship. The script then finalizes the launch by guiding the rocket into a stable orbit. The contents of the script are shown in the video and the various lines of codes are explained as they happen. In the scripting language various commands are given that wait until certain conditions (e.g. high enough altitude) are met and then another command is given (e.g. turn the rocket to face a particular direction or turn the engines on). The author also talks about how the mod can be used for more complex things such as automated rovers and discusses the similarities with the system and the use of computers in real life during space missions.

Overview of The Discussion

After the open coding of the first 350 comments from both videos, we started axial coding and found that the majority of comments abstracted to three categories: programming languages, efficiency, and learning experience. We use V1 and V2 to denote from which video the comment came from. The number of relevant, pertaining to computer science, comments for V1 was 139 (ca. 40%) and for V2 73 (ca. 21%).

From the comments, it is clear that some of the commenters were already familiar with computer science and programming. How big this population is compared to those who are novices is hard to estimate, and the only method for doing so is to infer from the contents of the comments.

Programming Languages

Referring to Douglas Adam's Hitchhiker's Guide to Galaxy, someone started a posted a comment: *Now we need a command block that can calculate the answer to life the universe and everything. (V1).* This gathered several replies, where people showed their knowledge and shared simple programs in many languages to show how printing number 42 was achieved, e.g. *" this is it in HTML <p>42 </p>" (V1)*, *"In scratch: When green flag clicked: Say 42" (V1)*, *"Here is in python print(42)" (V1)*, and other commenters point out a way to replicate this in multiple languages, including for example Java, and JavaScript.

Since V2 featured a custom language created for the game that is not in use elsewhere, it received plenty of comments reflecting on learning a new and highly specific language versus more general programming languages. This was reflected in comments such as: *"I just love the idea of kOS, and would gladly spend humongous amount of time playing with it... but, i also hate the programming language itself. It would be so awesome to see something like kOS which is using JavaScript/Python/Ruby/C# .. or any sensible programming language out there (thus object orientation would probably be very useful feature)" (V2)* and *"Yeah, a C or Python (maybe even Lua) version would have been nicer, but basic isn't hard to learn, it is very basic" (V2)*

Some of the commenters also pointed out the limitations of the Kerboscript (the name of the language used in the kOS mod): *"Yeah I really wish they had a more mainstream language, but then again it's still a beta. We might see something like that later hopefully, having functions and classes would be very cool." (V2)*. This, in turn, prompted a discussion on how languages work on a lower level: *"Whatever programming language and style you like, it still gets turned into machine code which doesn't inherently support functions and classes" (V2)*.

Overall, the discussion on both videos was focused on the *authenticity* of programming languages. This was especially prevalent on V2 since the video portrayed a custom programming language that was then used to complete 'real' programming tasks.

Efficiency

Much of the discussion related to computer science in comments to V1 revolved around the topic of algorithmic efficiency. In the video, the author presented a simple `ISPRIME` function which computed whether a number was prime in a suboptimal way. This was pointed out in many comments, often with suggestions on how to improve, for example: *"Instead of X-1 wouldn't 1/2 X be a better cutoff point for finding primes? (Since no number can be evenly divided by a number greater than 1/2 itself?) That would save some cycles, especially as the numbers get higher. ..." (V1)* , and *"Yeah, computing the square root of something is usually 'expensive', but dividing by two is usually cheap. It'd be an easy optimization." (V1)*

As a classic algorithm efficiency problem, some experienced users provided the best cutoff helping to improve the author's solution. However, some users also reflected about the problem and discussed the methods. In V2, efficiency discussion centered around certain parameters in the code, which some users felt could be optimized further.

Learning Experience

The third and perhaps the most interesting category is the discussion on learning experience regarding programming. One long thread in the comments involved the use of these types of games and their affordances in teaching younger population: *"Does this have the potential to help kids learn to program? IE if it was directly applicable in the world like you showed the turtle? :-)" (V1)*

This spurred many replies, giving suggestions on how to start learning programming using various approaches, such as game modding: *"Could try learning to mod the game; modding can range from needing almost no programming experience to advance programming, depending on how far you take it." (V1)* and watching programming tutorials on video: *"You can watch video tutorials though this is the hardest way to learn programming." (V1)*

The different tools and aspects of learning programming were discussed as well. With one commenter noting that block-based programming has gained popularity: *To be honest, logical-block-style programming is taught very very early in some places now. Because it teaches logic without needing to learn a programming language syntax." (V2)*. There were thoughts also on the difficulties of using professional tools when starting out in programming: *Well, the main reason it was hard for me to get into programming was because the IDE was made for more experienced people and was packed with features ... (V1)*

Other Observations

We initially suspected, based on the literature (Ramalingam et al., 2004), that there would be numerous comments portraying poor self-efficacy when confronted with programming and computer science. However, to our surprise these comments were almost completely missing, with only one concrete example: *"Now i feel stupid ..... This is insane! :D Wow" (V1)*. While it is difficult to verify, we suspect that there are more feelings of poor self-efficacy but those individual just do not comment, or even tune out the video as soon as something complex is presented.

5 DISCUSSION

In this article, we have examined the intersection of computer science and computer games in an informal setting. We approached this by describing the gaming content that includes programming as well as discussion related to it. While there has been many different approaches to incorporate games into classrooms, especially programming ones, very little discussion has taken place regarding informal learning from games. We approached this phenomenon through analyzing gaming videos that relate to programming – a popular way of disseminating content and ideas in the gaming community.

Looking at different kind of gaming videos that relate to computer science (**RQ1**), we discovered a plethora of programming-related videos associated with a variety of different games. Additionally, we sought to provide a more comprehensive picture of the phenomena by analyzing the discussions on these videos (**RQ2**). We conclude that a fair portion of the discussion is relevant to computer science and programming. The vast majority of programming related discussion can be described to belong to one of the three categories: programming languages and their capabilities, efficiency in algorithms and programs, and the experience of learning programming and computer science.

Incorporating computing and programming education into the K-12 curriculum has been heavily debated and discussed all over the world. At the same time, games and the whole gaming culture have risen to be a massive cultural phenomenon. And it at least seems that the number of games that contain programming within them is increasing. These games, where programming is a way to interact with the game world, could provide a rich and engaging environment to learn programming. Because the need to program arises from a problem within the game, they provide a motivating environment to learn to program.

However, based on the data, we have no way of knowing who the commenters are and what is their previous experience with computer science. Similarly, based on the discussion it is hard to tell how much actual learning is going on, and whether any of the possible learning is transferable to other contexts.

As we have shown, some of these exchanges are relevant to learning programming and computing skills. Given that, so many, especially young, people do play these games, we argue that the informal side warrants more discussion and research. We hold the view that it is crucial to understand when, where, and how the first contact with programming happens.

5.1 Future Work

In the future, we plan to do a more detailed investigation of the users involved in this kind of informal way to learn programming through gaming videos. Since it is the background of users' is not available and the fact that most of the comments are very short, the phenomena is difficult to analyze using methods applied to other discussion contexts (e.g. forums). Developing methodology to investigate informal learning in general is required.

Additionally, it would be interesting to find out whether similar phenomena is happening in other subjects. While it was not the focus of this research, we did note a portion of the discussion related to physics happening in the comments of V2.

ACKNOWLEDGEMENTS

Rodrigo Duran was supported by CNPq–Brazil.

REFERENCES

Boyle, M. and McDougall, A. (2003). Formal and informal environments for the learning and teaching of computer programming. In *Proceedings of the 3.1 and 3.3 working groups conference on International federation for information processing: ICT and the teacher of the future-Volume 23*, pages 11–12. Australian Computer Society, Inc.

Chau, C. (2010). Youtube as a participatory culture. *New directions for youth development*, 2010(128):65–74.

Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: defining gamification. In *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, pages 9–15. ACM.

Entertainment Software Association (2015). Essential facts about the computer and video game industry. Available Online: http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf.

Esper, S., Foster, S. R., Griswold, W. G., Herrera, C., and Snyder, W. (2014). Codespells: bridging educational language features with industry-standard languages. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*, pages 05–14. ACM.

Hayes, B. and Games, I. (2008). Making computer games and design thinking: A review of current software and strategies. *Games and Culture*.

Kelleher, C. and Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2):83–137.

Kinnunen, P. and Simon, B. (2010). Building theory about computing education phenomena: a discussion of grounded theory. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, pages 37–42. ACM.

McCartney, R., Boustedt, J., Eckerdal, A., Sanders, K., Thomas, L., and Zander, C. (2016). Why computing students learn on their own: Motivation for self-directed learning of computing. *ACM Transactions on Computing Education (TOCE)*, 16(1):2.

Ramalingam, V., LaBelle, D., and Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. In *ACM SIGCSE Bulletin*, volume 36, pages 171–175. ACM.

Saito, D., Washizaki, H., and Fukazawa, Y. (2015). Work in progress: A comparison of programming way: Illustration-based programming and text-based programming. In *Teaching, Assessment, and Learning for Engineering (TALE), 2015 IEEE International Conference on*, pages 220–223. IEEE.

Steinkuehler, C. and Duncan, S. (2008). Scientific habits of mind in virtual worlds. *Journal of Science Education and Technology*, 17(6):530–543.

Thompson, C. (2016). The Minecraft Generation. *The New York Times*. Available online: http://www.nytimes.com/2016/04/17/magazine/the-minecraft-generation.html.

Tillman, N., Halleux, J., Xie, T., and Bishop, J. (2014). Code hunt: Gamifying teaching and learning of computer science at scale. *L@S '14 Proceedings of the first ACM conference on Learning @ scale conference*, pages 221–222.

Topps, D., Helmer, J., and Ellaway, R. (2013). Youtube as a platform for publishing clinical skills training videos. *Academic Medicine*, 88(2):192–197.

Wallace, S. A., McCartney, R., and Russell, I. (2010). Games and machine learning: a powerful combination in an artificial intelligence course. *Computer Science Education*, 20(1):17–36.

Zorn, C., Wingrave, C. A., Charbonneau, E., and LaViola Jr, J. J. (2013). Exploring minecraft as a conduit for increasing interest in programming. In *FDG*, pages 352–359. Citeseer.