Learning Git Through Serious Educational Game

Awni Hamadeh

Computer Science
Bachelor
15 Credits
Spring 2020
Supervisor: Jeanette Eriksson

Abstract

Git is a distributed version control system that tracks changes to a project overtime and is used to save these changes. Today it is being used by millions of people and is becoming a demand on the job market. For this reason it has become important to learn the version control system. Learning Git however may be difficult for beginners and learning it through tutorials may not always be effective. Learning it through a serious educational game (SEG) may be more effective as a SEG can provide motivation and feedback which are two factors for successful learning. This study seeks to assess how effective a SEG is in teaching Git by looking at the amount of knowledge gained from playing a SEG. This study also seeks to assess how much participants learned Git using a tutorial compared to participants who used a serious educational game. From the results, the study found that the SEG expanded the understanding of Git. The study also found that there was no significant difference in the amount of understanding gained from the SEG and the tutorial.

Keywords: git, version control system, tutorial, design science research, interactivity

1. Introduction

With more and more students and non-students learning about software engineering the chances of them using a version control system will be very likely. Git is a distributed version control system [1] that allows multiple people to collaborate on the same project in parallel without having to make a backup or a new version of the project after each change. It tracks changes to a project and is used to save these changes. This can be useful for projects since a project usually consists of more than one person where all are making modifications to the project at the same time.

For beginners learning version control might be difficult. In a study [2] which ran a web development course of 200 students, Git was used as a tool to distribute course material and work with assignments. The purpose of the study was to describe how to introduce the features of Git and how to incorporate them into the course workflow. The students learned Git by first doing tutorials and exercises, and then utilizing it for their assignments. In the feedback survey, the students mentioned difficulties in learning the version control system. While the study does not go so much into why learning Git was difficult for the students, it is possible that doing tutorials was not enough to learn Git. In fact in a study by Ada Kim and Andrew Ko [3] which investigated over 30 popular software engineering tutorial sites, most tutorials were shown not to follow key principles of learning science. For example the tutorials did not answer why and when to use a particular coding concept which could have limited the amount of knowledge gained from doing the tutorial. Due to this reason tutorials might not always be effective in order to learn Git. A new learning approach should be introduced in order to alternatively and effectively learn the version control system. According to Phil Race [4] two factors of successful learning is motivation and feedback, which can be provided by video games [5]. Video games provide instant feedback by informing whether the current game objective was completed or failed. Video games also provide motivation through fun gameplay experiences which in turn can enhance learning [6]. Games have traditionally been used for entertainment purposes however there are games that are primarily used for other reasons than entertainment. These games are referred to as serious games. A Serious educational game has been used to teach science using a framework by Leonard A. Annetta which describes how to design a serious educational game (SEG) [7]. Annetta's framework provides a serious educational game with characteristics that are considered factors to successful learning such as *immersion* and *informed teaching* which is why this study will use the framework.

The purpose of this study is to (a) assess how much understanding about Git was acquired through a serious educational game and (b) evaluate how much participants learned Git using a tutorial compared to participants who used a serious educational game. Since a prototype is required to address the research questions,

the methodology that will be used is Design Science Research Methodology (DSRM) [8]. This research hopes to provide an alternative and effective way in learning Git as a contribution.

2. Related Work

The related work section is divided into three subsections. The first subsection is about Git, why it is relevant, what beginners found difficult and interactive tutorials that teach Git. The second subsection is about the framework that this study will use. And the third subsection presents the research questions that are addressed by this study.

2.1 Git

Git is used today by several popular companies such as Facebook, LinkedIn, Microsoft, Netflix, Google and Twitter. However the version control system is not only used by well known companies. In fact on Github [9] it was reported that there were over 100 million Git projects [10] and more than 40 million users [11] which suggests that Git is widely used even outside these companies. During the following years after this report, Indeed [12] reported that the version control system was one of the most frequently mentioned job qualifications [13]. Due to its wide use on the job market and efficiency, it has become important to learn Git. However learning Git may be considered difficult for beginners. In the research by Haaranen and Lethinen [2], students learned or improved their understanding of Git through exercises and tutorials. The students would then provide their feedback through a survey. In the survey some of the students mentioned difficulties in learning the version control system. One student wrote ”*Understanding Git was quite hard in the beginning.*” and another one wrote ”*Git could always be explained better, because despite the seeming simplicity it is quite a complicated version control system, especially compared to something like SVN.*”. The study does not go so much into detail of what the students considered difficult. It is possible that they considered merge conflict as difficult. This is because in another similar research [14], almost one third of the students mentioned merge conflicts as difficult to understand. This study will be similar to the one carried out by Haaranen & Lehtinen [2] except that a serious educational game will be used to teach Git instead of only a tutorial. The study will also assess the amount of knowledge gained. It is important to mention that in their research, the students learned Git by doing tutorials, however tutorials may not always be effective in learning. This was in fact demonstrated by Kim and Ko [3] where several tutorials lacked immediate feedback and only a few were shown to answer why and when to use a particular concept. Kim and Ko [3] also recommends that teachers should have their students use interactive tutorials and educational games over other tutorials as they fulfilled most of the criteria for effective learning.

For beginners there are interactive tutorials that teach Git. *LearnGitBranching* is a website that helps to learn Git through visualization and interactivity [15]. Its main purpose is to facilitate the learning of Git through a graphical user interface. This means that users of the web application write git commands as input and the result is shown visually as a branch topology. The application also provides immediate feedback which can be useful when learning. *Codecademy* is another similar website that also teaches Git [16]. However unlike *LearnGitBranching*, *Codecademy* does not display a branch topology. While interactive tutorials are an alternative to learning Git other than books and other tutorials, serious educational games (SEGs) may become a new and preferable alternative. Unlike interactive tutorials, SEGs provide immersion which can become useful in teaching. This is what mainly distinguishes SEGs from interactive tutorials. According to Annetta [7], immersion is considered an essential game element in a SEG.

2.2 Serious Educational Game

The "The "I's" Have It: A Framework for Serious Educational Game Design" by Annetta describes six essential game elements that should exist in a serious educational game. These elements are *identity*, *immersion*, *interactivity*, *instructional*, *informed teaching* and an *increasing complexity*. These will be described in detail in the method section. However some of the elements will also be described here for comparison.

Immersion is the idea that the player has to feel like being part of the game environment. This will motivate the player to continue playing. *Identity* means that the player should have a unique identity throughout the game. This immerses the player. And finally, *informed teaching* means that there should be a way to record the players answers so feedback can be provided. These elements are considered to be factors to successful learning according to the Ripple model. The Ripple model is a theory created by Phil Race that describes six factors that leads to successful learning. According to Race [4], the factors for successful learning is *needing/wanting*, *doing*, *feedback*, *making sense*, *verbalising* and *assessing*. It is namely *needing/wanting* and *feedback* that the game elements correspond with. For example *Needing/wanting* means that the learner has to need and want to learn. Annetta mentions *immersion* and *identity* which motivates the player to learn. This corresponds with wanting to learn. Annetta also mentions *informed teaching,* this means that if the players want to receive good feedback they need to learn the game. *Feedback* means that there should be feedback provided after each action. This once again is included in Annetta's framework which is *informed teaching*. By incorporating these game elements into a video game, the video game will contain characteristics that are considered factors to successful learning and therefore improve its teaching capabilities. For this reason the framework by Annetta will be

used to design a prototype. The implementation details of the prototype using the framework is presented in section 4.

3. Research Questions

The research questions in this paper is as follows:

RQ1: How much did the participants learn about Git through a serious educational game?

In order to address this research question, a SEG prototype is required. Using the prototype, the amount of knowledge gained can then be assessed. This prototype will be created and developed through a process known as Design Science Research Methodology [8]. More information about this methodology is provided in section 3.

RQ2: How much did the participants learn about Git through a tutorial compared to participants that used a serious educational game?

In respect to this research question, two different groups will either do a tutorial or play a serious educational game. The group that does the tutorial will act as the control group, while the group that uses the artifact will become the experiment group. The amount of understanding gained will then be measured and compared.

4. Method

4.1 Methodology

Since the research questions require a game, the process that this study will undertake is the *Design Science Research Methodology* (DSRM). In this methodology an artifact, which is a serious educational game in this case, is developed iteratively in order to address the research questions. DSRM is broken down into six steps by Peffers et al. [8]:

1. **Identify Problem & Motivate:** Identify and define the problem and then justify the value of the solution.
2. **Define Objectives of a Solution:** Infer objectives of the problem definition and knowledge based on what is feasible. The objectives can be either qualitative or quantitative.
3. **Design & Development:** Design and create the artifact. In this step the functionality of the artifact is first set and then created.
4. **Demonstration:** Demonstrate the use of the artifact to solve the problem. This can involve using experimentation, simulation, case study, proof or other suitable activity.

5. **Evaluation:** Evaluate how well the artifact supports a solution to the problem. This is done by comparing the results of the artifact to the objectives of the solution.
6. **Communication:** Communicate the problem, the artifact and its effectiveness to relevant stakeholders.

Using this methodology in our study:

1. **Identify Problem & Motivate:** Learning Git might be difficult for beginners and learning it through tutorials might not always be effective. Using a serious educational game to teach Git instead of a tutorial may be more effective as games can inherit characteristics that are considered factors to successful learning.
2. **Define Objectives of a Solution:** The objective is to (a) assess how much understanding about Git was acquired through a serious educational game and (b) evaluate how much participants learned about Git using a tutorial compared to participants who used a serious educational game.
3. **Design & Development:** A framework was chosen when considering the game design of the prototype. The framework that was chosen is titled "The "I's" Have It: A Framework for Serious Educational Game Design". This framework was chosen because it provides a game with characteristics that are considered factors to successful learning. According to Annetta, there should be six elements present in a serious educational game. These elements are:

    1. *Identity*: The player should have a unique identity in the game. An example of a game incorporating this feature is World of Warcraft [17]. In World of Warcraft you have to set the name of your character at the character creation screen. The naming of the character helps to provide a unique identity to the player. Players have shown to be more immersed in a game when allowed to form their own unique identities [7].
    2. *Immersion*: The player should be able to feel being part of the environment in the game. When players become immersed they will continue to spend more time on the game. This element is essential in SEG as it motivates the player to continue learning.
    3. *Interactivity*: There should be some sort of interaction in the game whether it be with non-player characters (NPCs) or other players. Interaction has shown to improve the player's communication skills [18]. Communication is important for learning as it helps improve the player's understanding.

4. *Increasing complexity*: The complexity should increase as the player progresses further. Increasing the complexity will make the game more challenging and thus motivate the player to continue playing.
5. *Informed Teaching*: The game should log the answers of the players so that feedback can be provided. Logging the answers of the players can also be used for analysis and adjustment of the game difficulty so that the game does not become impossible to finish.
6. *Instructional:* There should be instructions on a similar game objective or instructional scaffolding for players who are not able to finish the current game objective. By providing similar and easier game objectives to solve, the player might be able to use that to solve the more difficult game objectives.

In the artifact, these features were:

1. *Identity*: The player is hired as a Git expert who helps coworkers in case they have any questions on the system. This provides a unique identity or role to the player as the player is the only one who can help others on Git.
2. *Immersion*: The player becomes immersed by answering the given questions in time. The game also includes several scenes of dialogues. This becomes a motivating factor in continuing the game. This is because the player becomes interested in the outcome.
3. *Interactivity*: The player interacts with other characters in order to assist them on Git.
4. *Increasing complexity*: While the player is helping other workers that require help, the questions become more complex as the game heads towards the next sprint.
5. *Informed Teaching*: Game would log the questions along with the provided solution by the player and the correct solution. This would then be displayed in the end. This is so that the player could view them and compare them with his or her solution with respect to the question. The game would also provide instant feedback after each answer to the question.
6. *Instructional:* If the player was not able to solve a question, the player could request instructional scaffolding by clicking a "Show Tips" button.

4. **Demonstration:** As the artifact was created and finished, the artifact was tested and developed iteratively in order to fix bugs that became present so the artifact could be demonstrated. Once no bug was found in the artifact anymore it was deemed ready for demonstration. The effectiveness of the prototype was demonstrated by 5 participants. Participants were assigned to

one of two groups depending on whether they were gonna use the artifact or a tutorial. For the sake of simplicity, these groups will be referred to as group 1 and 2 throughout this study. Group 1 learned Git by following a tutorial while group 2 learned Git using the artifact. Group 1 also consisted of only four participants while group 2 consisted of five participants. The tutorial chosen was actually a combination of two tutorials titled "Learn Git In 15 Minutes" [19] and "Learn Github in 20 minutes" [20] both by Colt Steele. These two tutorials were chosen for this study because they were popular and highly favorable. In fact both tutorials had over 50 000 views and thousands of likes on Youtube. The reasoning behind using and combining these two tutorials instead of using just one of them was because (1) the second tutorial [20] was a sequel to the first tutorial [19] and (2) with both combined, the tutorials were able to cover more fundamental Git commands such as commit, add, branches, merges, push and pull. Participants of both groups were given a pre questionnaire prior to the tutorial or the artifact and a post questionnaire afterwards. Participants of both groups used Parsec [21], a software application, to gain remote access to a computer so the artifact or the tutorial could be played digitally while Discord [22] was used for voice communication.

5. **Evaluation:** To address the first objective, the results of the pre and post questionnaires from group 2 were compared in order to assess how much understanding was gained on Git prior to and after using the artifact. An in-depth interview was also carried out on the group in order to provide an insight into the effectiveness of the solution. For the second objective which was evaluating how much participants learned Git using a tutorial compared to participants who used a SEG; the percentage increase of understanding of Git from before to after using the artifact or tutorial were compared between participants. This was done by also comparing the pre and post questionnaire results. However unlike the evaluation for the first research question, this was done only for the commands that both the artifact and tutorial covered.
6. **Communication:** The results were then discussed and future research was considered. See the results and conclusion section below.

4.2 Limitation

Due to the COVID-19 pandemic, the study had to be carried out digitally. This created limitations, mainly that observation could not be made during the questionnaires. An in-depth interview had to also be carried out digitally using Discord [22]. Another limitation was that the results depend on how good the prototype is. A bad game will not give the same result as a good game. What might be considered a good game will vary from participant to participant. This will affect the amount of knowledge gained by the participant and thus affect the result. The COVID-19 pandemic also reduced the amount of participants the study could have.

Before the study was conducted, it was agreed that the study would be conducted at Malmö University since it would provide more participants but due to the COVID-19 pandemic, the university was closed [23] and this greatly limited the amount of participants the study could acquire.

Lastly, an important limitation in this study was the idea of comparing a prototype to a tutorial where the prototype had to be ready in less than four months. A tutorial might have been developed iteratively over time before it was published with the intention to make it fully complete, while the prototype created in this study was still at a prototype stage and therefore lacked additional features that would make it fully complete because there was not enough time. For example the prototype lacked optimized graphics settings, more sound effects, graphical user interface elements such as a start menu and additional game mechanics that provided longer game time. This could have affected the results in a way where players perceive the game as being short and not fun because it is not a finished product. And as a consequence this might have decreased the effect of the prototype in teaching. Therefore comparing the prototype to a tutorial might be difficult and the tutorial might have an advantage that the prototype does not due to a time constraint.

5. The Artifact

The artifact is titled Giteducatus and will be referred to as such throughout the text.

5.1 Game Overview

Giteducatus is a serious educational game where the player is hired as a Git expert with the task to help coworkers on using the version control system. The player has to interact with the coworkers who give a quest that the player needs to solve using an in-game terminal. The game uses a real Git command line interpreter. This means that after the player writes a Git command using the in-game terminal, the output will be the same as a real Git bash output.

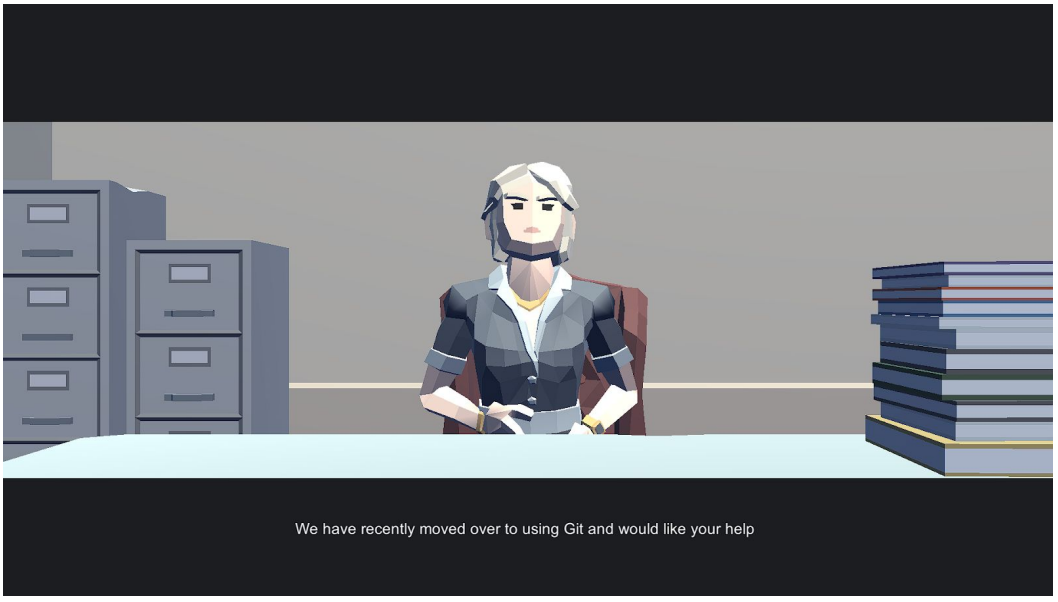


Figure 1: The introduction of the game

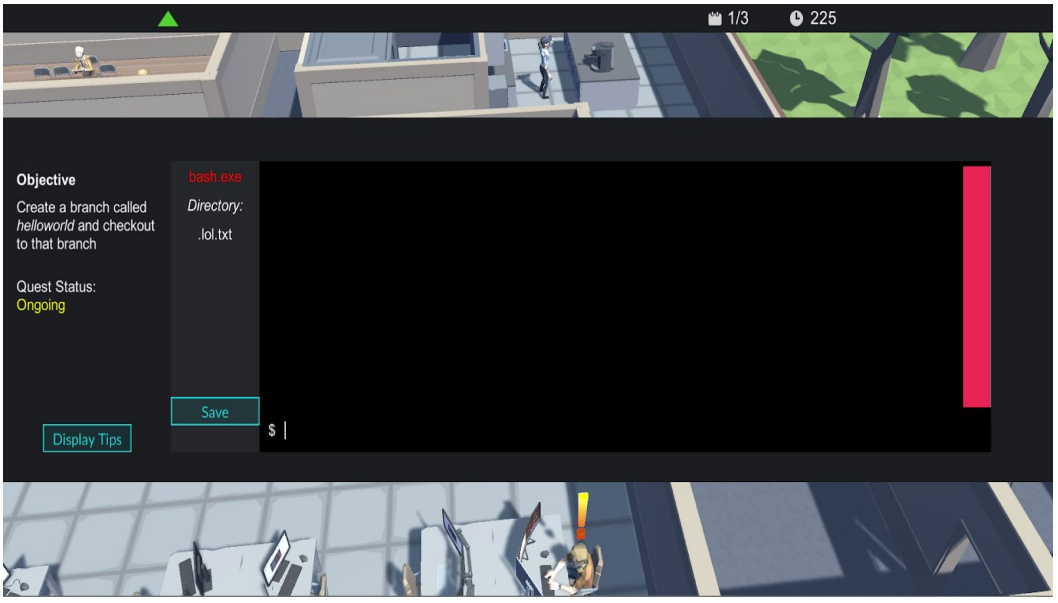


Figure 2: A quest in progress

5.2 Gameplay

At certain times the coworkers will stumble across an issue when using Git and will require help. An exclamation mark will then be displayed above them which represents an available quest. The player can click on the coworker to start the quest. Clicking on the coworker will start a dialogue and then open a panel with a command line interpreter (CLI) which can be seen in Figure 2. The CLI is named 'bash.exe' and is used by the player to write git commands. The player also has a 'directory' text under bash.exe. This text displays a list of the current files in the directory. In this case it is a text file named ".lol.txt". The player runs git commands for these files.

When the quest starts, the player is presented with an objective and has a certain amount of time to finish the objective. If the player wants instructional scaffolding the player can click on the 'Display Tips' button which provides hints and information about git commands relevant for the quest. As the player progresses, more difficult quests start to become available. The text to the right of the calendar icon displays the current game progress. 1/3 means that the game has begun, while 2/3 means that the game is halfway to the ending and 3/3 means that the player is close to finishing the game.

The game has quests that covers the following commands and terminologies: *git checkout* [24], *git branch* [25], *git merge* [26], *merge conflict* [27], *git add* [28], *git commit* [29], *git push* [30], *git fetch* [31] and *git pull* [32]. However these are only some commands. Git has over a hundred commands [33] and these commands may even be considered basic. Hence the game may not be suitable for those who have a long history of using Git.

5.3 End Conditions

The outcome of the game can be either victory or defeat. For the player to win the player has to avoid failing a number of quests. In Figure 2 there is a green icon near the upper left corner which represents the health bar of the player. Every time the player fails a quest, the health bar will slowly change its color. If the health bar finally reaches a red color then the game transitions into a defeat scene. Failing a quest happens if the player is not able to finish the objective in time.

6. Results and Analysis

6.1 Data Presentation

The data is presented in a form of two column charts and a summary of the interview. Section 6.2 addresses RQ1 which is how much the participants learned Git using Giteducatus while section 6.3 addresses RQ2.

6.2 Before and after Giteducatus

6.2.1 Amount of Understanding Gained

Results of the participants who played Giteducatus is shown in Figure 3. The y-axis represents the amount of points that they scored on the pre- and post questionnaire. Scoring the right answer on each question awards the participant with one point. The x-axis represents the participants who are denoted by a unique id. Almost all the participants increased their understanding of Git after using Giteducatus with the exception of P3 who did not experience any change. However P3 already had over a year experience of using Git. Since Giteducatus only covered basic commands then P3 might not have benefited from using the artifact. Most participants were however able to gain at least twice the amount of understanding after playing Giteducatus. See Appendix C for more information.

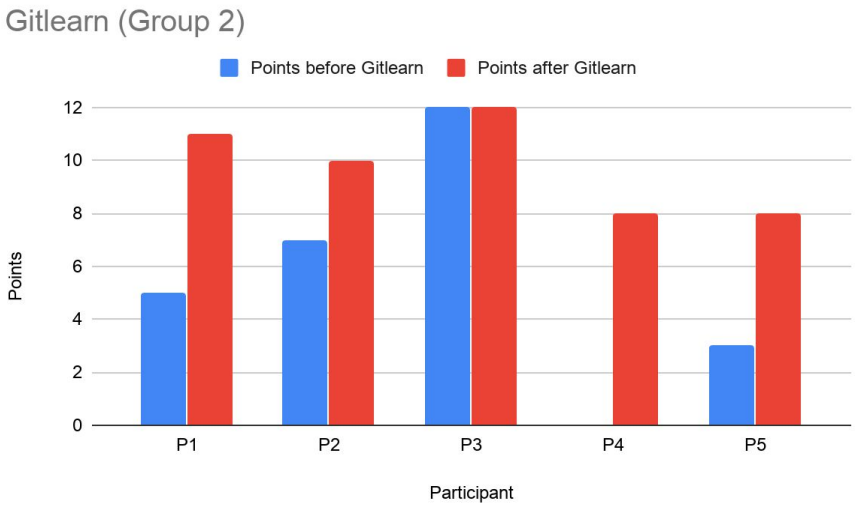


Figure 3: Participants who filled in the questionnaires before and after playing Giteducatus.

An interesting observation made during the study is that those who did not have any experience of using Git found it difficult to learn Git through the artifact. It is possible that those who have not used Git before are not interested in software engineering and may therefore find it more difficult in learning the version control system especially through a game. This is not only because they may not not know what Git is but also may not understand what a command terminal or a directory is. The artifact assumes that you have a basic understanding of a command line interpreter (CLI) program. However all the participants were still able to learn Git through the artifact regardless of their previous experience of using the version control system.

A commonly mentioned git command that was learned through the artifact was git fetch. In fact when the participants were asked whether they learned a lot about the version control system during the interview, some replied:

"Yes more than what I've learned from reading tutorials on the net. I've done the most basic things on Git such as git pull, git push, git commit and something like that.. I understand a bit better what git fetch and branch means now. " (Participant P1, see section 10.4.1)

"Not a lot but a little bit like git checkout and git fetch." (Participant P2, see section 10.4.2)

"Not much but I learned new things such as git fetch." (Participant P3, see section 10.4.3)

This can be confirmed by the questionnaire results because all the participants were able to score the correct answer on the git fetch question after using the artifact. The questions that the majority were not able to score the right answer on was regarding merge conflicts. Only two of the participants were able to score the right answer on one of the merge conflict questions. A possible explanation is that the artifact lacked various merge conflict quests. Also the explanation for when a merge conflict occurs was not so detailed according to the two participants. In fact the two participants, P2 and P3 said during the interview:

"Yes on merging. I think the game doesn't mention whether the histories also get merged during git merge." (Participant P2, see section 10.4.2)

"Yes a bit on merging. When merging two branches if the two git histories were also merged. The game did not cover that so much. " (Participant P3, see section 10.4.3)

This would affect the understanding of how a git merge works. This could partly explain why the majority participants were not able to grasp merge conflict. This is not what this study hoped to achieve because in the previous mentioned study [14], almost one third of the students found merge conflict to be difficult to understand. This study hoped to provide an understanding especially on merge conflicts considering that almost a majority found it difficult in the previous mentioned study.

While the artifact teaches what certain git commands mean, there were some git commands that the game did not cover so much. In fact participant P1 mentions in

the interview that the artifact did not cover git add so much. But git add was not part of the questionnaire so it is unclear whether all of the participants felt the same way.

Apart from how much the participants learned using the artifact, all of the participants said that the artifact conveys why and when to use commands. By providing contextual information such as when and why, the artifact is shown to be effective in teaching the version control system.

6.2.2 Summary of the In-depth Interview

This section presents a summary of the in-depth interview that was carried out in respect to RQ1. The results of the interview is:

Three of five mentioned that they learned a lot about Git while the two others mentioned that they learned a little about the system. One participant mentioned git fetch and git branch as examples. Another participant also mentioned git fetch as an example.

Four of five participants said that they would use Giteducatus as an alternative to tutorial while one was uncertain.

One of five participants mentioned that they know why Git is used after playing Giteducatus. Four of five already knew prior to playing Giteducatus on why Git is used.

Five of five participants mentioned that the game conveys why and when to use git commands. See Appendix D for more information about the interview.

6.3 Amount of Understanding Gained from Tutorial and Giteducatus

The results of group 1 and group 2 is presented in Figure 4. The chart shows the percentage increase in understanding of Git after the artifact or the tutorial. The participant who learned the most in group 1 was T4 while in group 2 it was P4. The knowledge of T4 increased by 67% compared to P4 group 2 who also increased by 67%. Both T4 and P4 did not have any previous experience of using Git which could explain why they acquired a large amount of understanding.

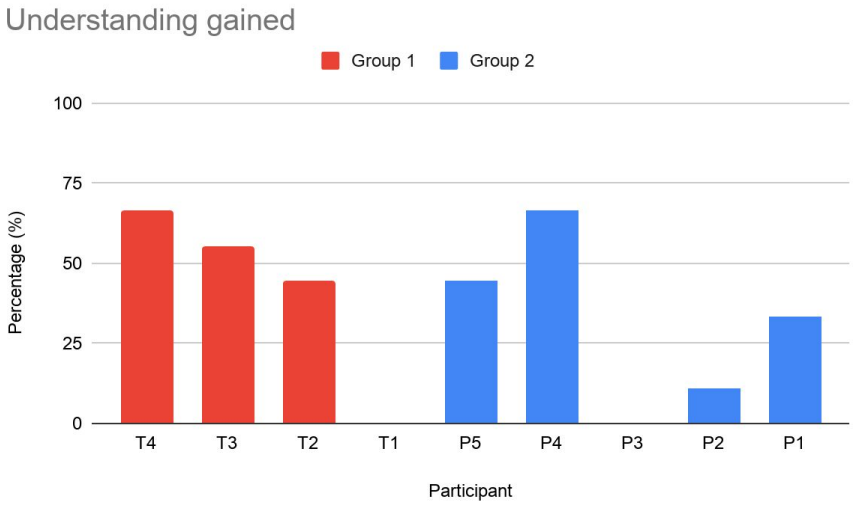


Figure 4: The percentage increase in understanding of Git after the tutorial or the artifact

The results are not what this study expected. Before the study was carried out what was expected was that all of the participants who learned through the artifact would have a higher understanding gained than those who did the tutorial with the exception of those who already had previous experience of Git. However in Figure 4 some participants from both groups were able to acquire more understanding than participants from the other group with the same amount of experience. For example T4 and T3 had higher than P5. P4 had higher than T2 and T3. Also the gain difference between P5 and T3 was not so significantly different either. And there was no difference in the understanding gained between P4 and T4, or P5 and T2. This would suggest that there was no significant difference in the understanding acquired between the participants who used the tutorial or the artifact.

A possible explanation is that some participants might not have considered the game to be fun. This is because one participant mentioned that he would not play the game because it is fun as he did not consider it such but rather because he has to learn Git. One limitation of the artifact itself is that what may be considered fun

will vary between participants. The fact that one participant stated that he did not find the game fun could have been the same for the majority participants. This is important because it could affect the results as video games are known for providing motivation through fun gameplay experiences which in turn can enhance learning [6]. It can be argued that because a SEG contains *immersion* and *identity* it should provide motivation to learn. But these two factors alone might not provide as significant motivation to learn as fun gameplay experience could.

It is relevant to mention that almost everyone in group 2 prefered the artifact over tutorials they have previously done regardless of motivation. But the interview does not ask whether they preferred the artifact over the tutorial used in this study which makes it difficult to know what they would think if the artifact was ever compared to the tutorial used in this study. But the reasoning for preferring the artifact over tutorials is interesting because it points out quality attributes that all tutorials do or do not have. For example participant P1 mentions interactivity as reason for preferring the SEG over tutorials in general. To quote the response of all the participants:

"Yes although I haven't done so many tutorials. But I wouldn't play the game because it is fun as it is not but rather because I want to learn. I also think this game is good because of the interactivity." (Participant P1, see section 10.4.1)

"In some cases. If what I want to learn on Git is too specific then I will choose tutorial, but if I want to learn more commands then I will choose the game. " (Participant P2, see section 10.4.2)

"Yes." (Participant P3, see section 10.4.3)

"Haven't done tutorials on Git but I would still prefer the game. Because you learn more when you do practical things than to watch a tutorial that tells you do this and that. And thing is that if you watch a video that tells you do this to achieve a result then you learn how to get the result but you don't learn what it means and why. " (Participant P4, see section 10.4.4)

"Possibly after reading more about Git on Stackoverflow. Once I would have a better understanding I would use the game similar to Codeacademy to test my knowledge on Git. I think the game is more like a tool that can be added together with say a tutorial. " (Participant P5, see section 10.4.5)

This is interesting because unlike SEGs, tutorials do not provide interactivity unless they are interactive tutorials. This could suggest that while there was no difference in the amount of understanding gained by tutorial and the artifact, different quality

attributes such as interactivity could make the artifact more preferable in learning. An interesting response is provided by P2 who says that it depends on what he wants to learn. If there are too broad concepts that he wishes to learn then he would choose the artifact and if he wishes to learn something specific then he would choose a tutorial. This would suggest that some might still choose a tutorial over a SEG depending on the amount of information they would want to learn regardless of the quality attributes of a SEG.

The only participant that was unsure of whether he would use the artifact over tutorials was P5. But the response of P5 is also interesting because he states that he would use the artifact as a tool in learning Git together with a tutorial. The participant also mentions that he would first read more on other websites before using the artifact. This might suggest that the information provided by the artifact was not sufficient. It is also possible that because the participant did not have any previous experience of using Git he would not understand what a command line interpreter is and as a way to compensate for that knowledge he would use websites to read more about Git first.

While the artifact provides interactivity that tutorials do not, most participants in group 2 such as P1, P2 and P3 did not seem to acquire as significant understanding as those in group 1. But it is important to mention that most who played Giteducatus also had previous experience of using Git which could explain why. Only two of the participants from group 2 have not used Git which could explain why they learned more than the rest in the group. Meanwhile in group 1 the majority did not have any previous experience of using the version control system which explains the large learning percentage in Figure 4. Only T1 had previous experience of using Git in the group and his experience was ranging between 4 to 8 months which could explain why the participant did not learn anything. It is possible that the set of questions from the questionnaires might not have been suitable for those who have at least 4 months of experience. In this case it would be better to have more participants in the study with less than 4 months of experience. The results would then be more clear.

7. Discussion

By using Annetta's framework [7] to design a serious educational game I was able to create a game that expanded the understanding of Git. Most participants prefered the artifact over tutorials they have previously done with one mentioning interactivity as reason. While the artifact did not seem to perform differently to tutorial in providing an amount of understanding, I still propose that a SEG should be used to teach Git instead of tutorials. This is for four reasons:

- SEGs provide interactivity and there seems to be a preference for interactivity as pointed out in the interview.
- SEGs allow for connecting real-world scenarios with the usage of git commands. For example they can display common Git problems that a person may encounter everyday at work. This would help to answer the 'why' and 'when' question for the git commands. The results of this paper reinforces Kim's and Ko's research that the investigated educational games were more effective than the investigated tutorials. Because if the interview in this paper is taken into consideration, the SEG provided instant feedback and all the participants mentioned that the game conveys why and when to use git commands.
- SEGs can provide fun gameplay experiences which would motivate learning. However the artifact that was used in this study was not considered fun by one participant which could have been the same for the majority participants. But it is also possible that the majority might have found the game to be fun. It is important to mention that the process of creating and developing the artifact had a time constraint. In fact the artifact had to become ready in less than four months which might have been a strong factor to why the game was not considered fun by one participant. What was noticed about the game during the study was that it was fairly short. With more time, more content could have been added to the game and this would possibly make it fun.
- SEGs can become an alternative for those that do not like tutorials. In the study by Haaranen & Lehtinen [2], students found it difficult to learn Git. They used tutorials and exercises to learn the version control system. However they would perhaps prefer learning Git through a SEG instead as the participants in this study acquired a greater understanding through the SEG. The participants also prefered the SEG over tutorials they have done possibly due to interactivity.

An interesting point is the 'why' and 'when' questions in the second bullet point. When the SEG in this study was designed and developed the 'why' and 'when' questions for different git commands were not taken into consideration. Instead they were naturally added to the game as the game was designed and developed. This is understandable because in the SEG used by this paper the player interacts with non-player characters and is then presented with an objective. The interaction helps to address the 'when' question and the objective helps to address the 'why'. This could suggest that SEGs usually address the 'why' and 'when' question and unlike tutorials they can do so through graphics such as a virtual world. What also makes SEGs different from tutorials is guidance for common errors. In fact in the research by Kim and Ko [3], only a few tutorials provided guidance when common errors were made. In a SEG the error would be highlighted in a form of instant

feedback. To give an example: In the SEG used by this study, if a git command was misspelled the command line interpreter would output an error message letting the player know what caused the error. It can be argued that tutorials also offer this type of error message as the player would receive an error message if he or she mistyped a command in the command line interpreter. However unlike tutorials, in a SEG the player has to provide input in order to pass to the next level while in a tutorial the player does not necessarily have to provide anything in order to watch or read the tutorial. In fact an interesting observation made among the group who did the tutorial was that one participant did not do everything that was done in the tutorial which suggests that there are people who do not do everything that is shown in a tutorial. This would mean that the person doing the tutorial would be less likely to stumble across an error and have less mistakes to reflect on. While in a SEG player input is required which increases the chance of making mistakes and thus provides the player with more errors to reflect on.

When it came to the information conveyed there was a difference between the tutorial and the SEG. In some cases the tutorial provided additional information that the SEG did not. For example the tutorial goes through how to set up Git and also slightly explains what the CLI application Git Bash is which provides a better introduction on the version control system while the SEG did not. Instead the SEG assumed that there was a basic understanding of how a command line interpreter works. This could explain why beginners found it difficult to learn Git through the artifact. In fact one participant said that he would read more about the version control system before using the artifact and this could be a reason why. That one participant would first read more on other websites before using the artifact could also suggest that the artifact did not provide sufficient information on different git commands. Interestingly the same participant pointed out that he would use the game as a tool together with a tutorial in learning Git. Even though this paper wants to provide an alternative and an effective way in learning Git, tutorials do not necessarily have to be replaced. Instead they can even be combined together with a SEG to strengthen learning.

7.1 Additional Limitations

In this paper there were some limitations that were noted after carrying out the study. First limitation was that there was no way to know whether the interest in Git correlated with how fun the SEG was. What could have been done differently is to assess whether the players found the SEG to be fun and then to check whether there was any interest in learning Git. In the post questionnaire there was only a question of whether the participants increased their interest in learning Git after using the artifact. However this does not necessarily have to be because the SEG was fun. Instead it could be for other reasons such as finding Git to be an effective tool for projects. By assessing whether the players found the game to be fun, a correlation

between fun and the increase of interest in learning Git could have been possibly formed. The assessment would also provide clearer results.

The second limitation was the amount of people that participated in this study was not considered sufficient for RQ2. However getting people to participate in the study was not an easy task to do and this ended up with only being able to get 9 participants. It would be interesting to see the outcome of the study with a larger population size. Perhaps the results would then become more clear.

Third limitation was that participants were not evenly assigned to the two different groups. In fact there were more in group 2 who had previous experience of Git compared to group 1 which would affect the results for RQ2 and therefore make comparisons difficult. That participants were not evenly assigned to groups creates bias and is the biggest weakness of this study. What could have been done differently is to assign evenly so that both groups have the same amount of people with the same amount of experience. A better solution would be to restrict groups to a certain amount of experience using exclusion and inclusion criteria. For example participants who had previous experience of Git should not have been allowed to join either group. However exclusion and inclusion criteria was not used for two reasons which was that (a) there was a lack of participants and (b) having a variety of participants with different amounts of experience was not expected so exclusion and inclusion criteria was not taken into consideration at the time.

If Inclusion and exclusion criteria were to be used in this study there is a chance that this would be very difficult to implement. This is because there would be the risk that the results presented in Figure 4 would cover less participants which would not give as clear results and analysis. If participants were excluded on the basis that they did not have an opposing participant with the same amount of experience in the other group then that leaves out participants who provided interesting input from the interview such as P1 or P2. Hence it would require more participants in the study to fill up the space. Finding participants was the biggest difficulty in this study and COVID-19 hindered the ability to acquire participants since it prompted measures throughout the country [34]. Finding participants with a specified amount of experience would even prove to be more difficult. Also, it is important to mention that while some participants did have previous experience of Git, this study points that out when addressing the second research question. See the end of section 6.3.

A final weakness of this study is during the interview when participants are asked whether they preferred the artifact over tutorials. The problem is that the question does not take account of the tutorial used in this study. What could have been done differently instead is to see if the participants in group 2 preferred the artifact over

the tutorial used in this study. However this was not done because getting the participants to play the artifact was time consuming enough. In fact the playtime of the artifact took an average between one to two hours. By letting them watch the tutorial after using the artifact would add an additional 35 minutes and therefore consume more time that the participants might not have. Biggest drawback was the assumption that the more time the participants spent, the less detailed the results would be as they would become exhausted once they answered the questionnaire and the interview questions. An alternative solution would be to instead create a new group of participants that is used to determine what the participants preferred. But this would then require more participants which this study could not provide due to a limitation. Another reason for why the study did not take the tutorial into account was because when the in-depth interview questions were created, they were created in respect to RQ1 and not RQ2.

8. Conclusion and Future Work

The results suggest that a SEG may expand the understanding of Git. The results did not seem to provide a significant difference in the amount of understanding gained between those who used a tutorial and those who used the artifact. This was possibly because the game might not have been considered fun. However most participants prefered the serious educational game over previous tutorials they have done with one mentioning interactivity as reason.

In regards to future work: this study only covered certain git commands and terminologies such as git checkout, git branch, git merge, merge conflict, git add, git commit, git push, git fetch and git pull. It would be interesting to see whether an artificial intelligence can be made to generate a quest or exercise for each command since Git has over hundred commands. Git also has a history of updating and adding new commands [35] which is why it would be better to have an AI that generates quest or exercise for each command. However it does not necessarily have to be for each command, it can also be for the same command. This means that the game should create different quests teaching the same command.

Other future work includes assessing whether most participants found the game to be fun as that would be interesting to know. Right now there is no way to know whether most participants found the game to be enjoyable since the questionnaire did not ask that. So it is difficult to know whether fun was a motivating factor in what led to the results. Besides assessing whether the participants found the game to be fun, it would also be interesting to know whether most participants prefered the artifact over the chosen tutorial. Perhaps assessing this question can be done in future work as well.

A final note, this study only covered a specific case of comparing a SEG to a tutorial which was by looking at the amount of knowledge gained between a specific SEG and a specific tutorial. What could also be expanded upon for future work is the validity of the artifact itself as an actual SEG and how representative the tutorial is compared to other tutorials. This would provide more weight in the discussion of a SEG versus a tutorial. What was noticed after carrying out the study was that there was no guarantee that every participant would feel immersion as it was up to the participants themselves. Perhaps there could be a questionnaire that would validate the elements from the framework "The I's have it" [7] used in the artifact in order to assess how much the participant experienced these elements during gameplay. As for the tutorial, the choice of picking the tutorial was very specific. In fact it was chosen based on popularity. But the tutorial can perhaps be compared to other popular tutorials and then from there pick the tutorial with the most quality attributes so it can be compared to the SEG as well. A better solution would be to compare multiple popular tutorials to the SEG in order to see how well all performed in comparison to each other in providing an amount of understanding.

Since the study only covered a specific case of comparing a SEG to a tutorial, it is important to mention that the amount of knowledge gained does not necessarily suggest that one method of learning is better than the other but it may provide an indication. Perhaps a SEG and a tutorial can be compared in other cases. For example how fast does a SEG teach certain concepts compared to a tutorial. Or which of the two conveys low-level concepts better.

9. References

1. S. Chacon and J. Long, "*Git*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/. [Accessed: 04- Mar- 2020].
2. L. Haaranen and T. Lehtinen, "Teaching Git on the Side: Version Control System as a Course Platform", in *ITiCSE: Innovation and Technology in Computer Science Education*, Vilnius, Lithuania, 2015, pp. 87-92.
3. A. Kim and A. Ko, "A Pedagogical Analysis of Online Coding Tutorials", in *SIGCSE '17: The 48th ACM Technical Symposium on Computer Science Education*, Washington, 2017, pp. 321-326.
4. P. Race, "*Updated 'Ripples Model'*", Phil Race. 2020 [Online]. Available: https://phil-race.co.uk/2012/03/updated-ripples-model/. [Accessed: 06- Jun- 2020].
5. A. Mitchell and C. Savill-Smith, *The use of computer and video games for learning*. London: Learning and Skills Development Agency, 2004, p. 17.
6. R. Teed, "*Why Use Games to Teach?*", SERC: Science Education Resource Center at Carleton College, n.d. [Online]. Available: https://serc.carleton.edu/introgeo/games/whygames.html. [Accessed: 06- May- 2020].
7. L. Annetta, "The "I's" Have It: A Framework for Serious Educational Game Design", *Review of General Psychology*, vol. 14, no. 2, pp. 105-113, 2010. Available: 10.1037/a0018985.
8. K. Peffers, T. Tuunanen, M. Rothenberger and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45-77, 2007. Available: 10.2753/mis0742-1222240302.
9. "*Build software better, together*", GitHub, n.d. [Online]. Available: https://github.com/. [Accessed: 06- May- 2020].
10. K. Johnson, "*GitHub passes 100 million repositories*", VentureBeat, 2018. [Online]. Available: https://venturebeat.com/2018/11/08/github-passes-100-million-repositories/. [Accessed: 06- Jul- 2020].
11. "*Showing 42,116,266 Available Users*", GitHub, n.d. [Online]. Available: https://github.com/search?q=type:user&type=Users. [Accessed: 06- Jul- 2020].
12. "*Job Search | Indeed*", Indeed, n.d. [Online]. Available: https://www.indeed.com/. [Accessed: 06- Aug- 2020].
13. R. Chan and B. Sapra, "*The 20 top tech skills that employers want and that can help you find a job, according to recruiting site Indeed*", Business Insider, 2019. [Online]. Available:

https://www.businessinsider.com/20-top-tech-skills-indeed-2019-11. [Accessed: 01- Mar- 2020].
14. V. Isomöttönen and M. Cochez, "Challenges and Confusions in Learning Version Control with Git", in *International Conference on Information and Communication Technologies in Education, Research, and Industrial Applications*, Kherson, Ukraine, 2020, pp. 178-193 [Online]. Available: https://doi.org/10.1007/978-3-319-13206-8_9. [Accessed: 06- Apr- 2020].
15. P. Cottle, "*Learn Git Branching*", LearnGitBranching, 2012. [Online]. Available: https://learngitbranching.js.org. [Accessed: 28- Jul- 2020].
16. "*Learn Git*", Codecademy. [Online]. Available: https://www.codecademy.com/learn/learn-git. [Accessed: 06- Jul- 2020].
17. K. Bessière, A. Seay and S. Kiesler, "The Ideal Elf: Identity Exploration in World of Warcraft", *CyberPsychology & Behavior*, vol. 10, no. 4, pp. 530-535, 2007. Available: 10.1089/cpb.2007.9994.
18. N. Ducheneaut and R. Moore, "More than just 'XP': learning social skills in massively multiplayer online games", *Interactive Technology and Smart Education*, vol. 2, no. 2, pp. 89-100, 2005. Available: 10.1108/17415650580000035.
19. C. Steele, "*Learn Git In 15 Minutes*", Youtube, 2019. [Online]. Available: https://www.youtube.com/watch?v=USjZcfj8yxE. [Accessed: 30- Feb- 2020].
20. C. Steele, "*Learn Github in 20 Minutes*", Youtube, 2019. [Online]. Available: https://www.youtube.com/watch?v=nhNq2kIvi9s. [Accessed: 23- Aug- 2020].
21. *Parsec*. Parsec Gaming, 2016. Accessed: Jul. 12, 2020. [Online]. Available: https://parsecgaming.com/
22. *Discord*. Discord Inc., 2015. Accessed: Jul. 12, 2020. [Online]. Available: https://discord.com/
23. "*Information to Students Regarding the Coronavirus and COVID-19*", Mau.se, 2020. [Online]. Available: https://student.mau.se/en/startpage/recommendations-regarding-the-coronavirus-and-covid-19/#accordion-46949. [Accessed: 17- Jul- 2020].
24. "*Git - git-checkout Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-checkout. [Accessed: 06- Jul- 2020].
25. "*Git - git-branch Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-branch. [Accessed: 25- Jun- 2020].
26. "*Git - Basic Branching and Merging*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging. [Accessed: 27- Jun- 2020].
27. "*Resolving a merge conflict using the command line - GitHub Docs*", Github, n.d. [Online]. Available:

https://docs.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line. [Accessed: 17- Jul- 2020].
28. "*Git - git-add Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-add. [Accessed: 28- Jun- 2020].
29. "*Git - git-commit Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-commit. [Accessed: 28- Jun- 2020].
30. "*Git - git-push Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-push. [Accessed: 28- Jun- 2020].
31. "*Git - git-fetch Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-fetch. [Accessed: 28- Jun- 2020].
32. "*Git - git-pull Documentation*", Git-scm.com, n.d. [Online]. Available: https://git-scm.com/docs/git-pull. [Accessed: 28- Jun- 2020].
33. "*How many commands does Git have?*", Stack Overflow, 2012. [Online]. Available: https://stackoverflow.com/questions/11719013/how-many-commands-does-git-have. [Accessed: 28- Jun- 2020].
34. "*The Government's work in response to the virus responsible for COVID-19*", Regeringskansliet, 2020. [Online]. Available: https://www.government.se/government-policy/the-governments-work-in-response-to-the-virus-responsible-for-covid-19/. [Accessed: 06- Jul- 2020].
35. "*Git 2.23 released with two new commands 'git switch' and 'git restore', and more!*", Packt Hub, 2020. [Online]. Available: https://hub.packtpub.com/git-2-23-released-with-two-new-commands-git-switch-and-git-restore-a-new-tutorial-and-much-more/. [Accessed: 28- Jul- 2020].

10. Appendices

10.1 Appendix A - Pre Questionnaire

10.1.1 Pre Questionnaire for Giteducatus

This survey is being conducted for a study that is about learning Git through a serious game. The survey will be used to determine the current knowledge you have on Git. You will get to play a serious educational game afterwards. After you have finished the game you will be given a post questionnaire which almost has the same questions as this pre questionnaire in order to assess the knowledge you gained on Git. Thank you for your participation.

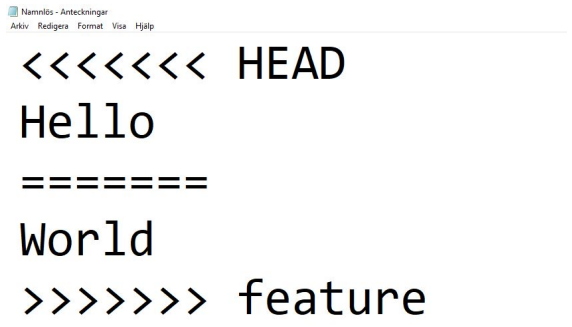
1. Have you used Git?
    - Yes
    - No
2. If you answered yes in the previous question, please name the other version control systems that you have used
3. How long have you used Git? (Answer this question if you answered ‘yes’ on question 1)
    - Less than a month
    - 1-3 Months
    - 4-8 Months
    - 9-12 Months
    - Over a year
4. When do you use Git? (Answer this question if you answered ‘yes’ on question 1)
    - University Assignments
    - Work
    - Own Projects
    - Other
5. Why have you not used Git? (Answer this question if you answered ‘no’ on question 1)
    - Don't know what it is
    - Not required for the work I do
    - It is difficult to learn
6. What is a repository?
    A) It is a folder where all your project files are stored and tracks changes in your files
    B) It is a special folder that is used for Git hooks

C) It is a folder where all your project files are stored but doesn't track changes
D) No idea

7. What is a branch?
   A) A pointer to a commit
   B) A container of commits
   C) A clone of the original project
   D) No idea
8. When should branches be used? (Multiple answers are allowed)
   A) When wanting to experiment on the project without affecting the original project
   B) Whenever cloning project
   C) When wanting to create additional features
   D) No idea
9. What does git checkout -b do?
   A) Retrieves the latest changes to the repository
   B) Switches to and creates a new branch
   C) Creates a new branch
   D) No idea
10. What does git fetch do?
    A) Downloads files from a different repository
    B) Retrieves the latest changes from the remote repository
    C) Clones a repository
    D) No idea
11. What does git commit do?
    A) Adds staged changes to the local repository
    B) Adds staged changes to the remote repository
    C) Adds staged changes to both repositories
    D) No idea
12. What does git push do?
    A) Appends staged changes to the current commit
    B) Takes the current commits from the current branch to another branch
    C) Takes commits from the local repository and sends them to the remote repository
    D) No idea
13. What does git pull do?
    A) Retrieves the latest content from the remote repository but doesn't update the local repository
    B) Retrieves the latest content from the remote repository and updates the local repository to match that content.
    C) Clones a repository
    D) No idea

14. What does git merge do?
    A) Joins two branches together into one branch. The git history of both branches are also joined together.
    B) Adds a new change to the local repository
    C) Joins two branches together into one branch. The history of one branch is not joined together with the other.
    D) No idea
15. When does a merge conflict occur in Git?
    A) When two separate branches that are to be merged have modified the same file
    B) When merging two branches that only contain changes to different files
    C) When writing git push
    D) No idea
16. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep both changes from a merge, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.
17. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep the changes from the current branch you're in, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.
18. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep the changes from the other branch that you're merging with, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.

M1



10.1.2 Pre Questionnaire for Tutorial

This survey is being conducted for a study that is about learning Git through a serious game. The survey will be used to determine the current knowledge you have on Git. You will get to do a tutorial afterwards. After you have finished the tutorial you will be given a post questionnaire which almost has the same questions as this pre questionnaire in order to assess the knowledge you gained on Git. Thank you for your participation.

1. Have you used Git?
    - Yes
    - No
2. If you answered yes in the previous question, please name the other version control systems that you have used
3. How long have you used Git? (Answer this question if you answered ‘yes’ on question 1)
    - Less than a month
    - 1-3 Months
    - 4-8 Months
    - 9-12 Months
    - Over a year
4. When do you use Git? (Answer this question if you answered ‘yes’ on question 1)
    - University Assignments
    - Work
    - Own Projects
    - Other
5. Why have you not used Git? (Answer this question if you answered ‘no’ on question 1)
    - Don't know what it is
    - Not required for the work I do
    - It is difficult to learn
6. What is a repository?
    A) It is a folder where all your project files are stored and tracks changes in your files
    B) It is a special folder that is used for Git hooks
    C) It is a folder where all your project files are stored but doesn't track changes
    D) No idea
7. What is a branch?
    A) A pointer to a commit
    B) A container of commits
    C) A clone of the original project
    D) No idea

8. When should branches be used? (Multiple answers are allowed)
    A) When wanting to experiment on the project without affecting the original project
    B) Whenever cloning project
    C) When wanting to create additional features
    D) No idea
9. What does git checkout do?
    A) Retrieves the latest changes to the repository
    B) Switches to a branch
    C) Creates a new branch
    D) No idea
10. What does git commit do?
    A) Adds staged changes to the local repository
    B) Adds staged changes to the remote repository
    C) Adds staged changes to both repositories
    D) No idea
11. What does git push do?
    A) Appends staged changes to the current commit
    B) Takes the current commits from the current branch to another branch
    C) Takes commits from the local repository and sends them to the remote repository
    D) No idea
12. What does git pull do?
    A) Retrieves the latest content from the remote repository but doesn't update the local repository
    B) Retrieves the latest content from the remote repository and updates the local repository to match that content.
    C) Clones a repository
    D) No idea
13. What does git merge do?
    A) Joins two branches together into one branch. The git history of both branches are also joined together.
    B) Adds a new change to the local repository
    C) Joins two branches together into one branch. The history of one branch is not joined together with the other.
    D) No idea

10.2 Appendix B - Post Questionnaire

10.2.1 Post Questionnaire for Giteducatus

1. Did you feel like this game was a good way to learn Git?
    - Yes
    - No

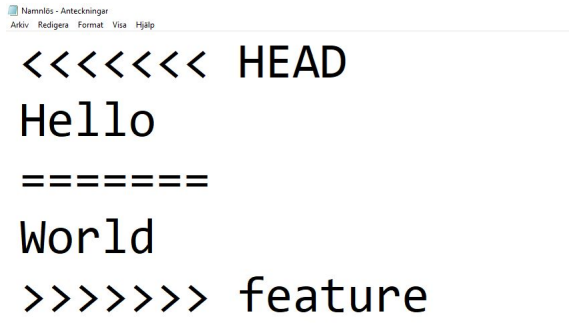
- Maybe

2. Did your interest in Git increase after playing this game?
    - Yes
    - No
    - Maybe
3. What is a repository?
    A) It is a folder where all your project files are stored and tracks changes in your files
    B) It is a special folder that is used for Git hooks
    C) It is a folder where all your project files are stored but doesn't track changes
    D) No idea
4. What is a branch?
    A) A pointer to a commit
    B) A container of commits
    C) A clone of the original project
    D) No idea
5. When should branches be used? (Multiple answers are allowed)
    A) When wanting to experiment on the project without affecting the original project
    B) Whenever cloning project
    C) When wanting to create additional features
    D) No idea
6. What does git checkout -b do?
    A) Retrieves the latest changes to the repository
    B) Switches to and creates a new branch
    C) Creates a new branch
    D) No idea
7. What does git fetch do?
    A) Downloads files from a different repository
    B) Retrieves the latest changes from the remote repository
    C) Clones a repository
    D) No idea
8. What does git commit do?
    A) Adds staged changes to the local repository
    B) Adds staged changes to the remote repository
    C) Adds staged changes to both repositories
    D) No idea
9. What does git push do?
    A) Appends staged changes to the current commit
    B) Takes the current commits from the current branch to another branch

C) Takes commits from the local repository and sends them to the remote repository
D) No idea

10. What does git pull do?
    A) Retrieves the latest content from the remote repository but doesn't update the local repository
    B) Retrieves the latest content from the remote repository and updates the local repository to match that content.
    C) Clones a repository
    D) No idea
11. What does git merge do?
    A) Joins two branches together into one branch. The git history of both branches are also joined together.
    B) Adds a new change to the local repository
    C) Joins two branches together into one branch. The history of one branch is not joined together with the other.
    D) No idea
12. When does a merge conflict occur in Git?
    A) When two separate branches that are to be merged have modified the same file
    B) When merging two branches that only contain changes to different files
    C) When writing git push
    D) No idea
13. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep both changes from a merge, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.
14. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep the changes from the current branch you're in, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.
15. Look at the image below with the title M1 that shows a merge conflict inside a text file. Say that you would want to keep the changes from the other branch that you're merging with, how would you change the text file in order to solve the merge conflict in Git? No Git command is required to solve this question. Only modify the text file.

M1



10.2.2 Post Questionnaire for Tutorial

1. Did you feel like this was a good way to learn Git?
    - Yes
    - No
    - Maybe
2. What is a repository?
    A) It is a folder where all your project files are stored and tracks changes in your files
    B) It is a special folder that is used for Git hooks
    C) It is a folder where all your project files are stored but doesn't track changes
    D) No idea
3. What is a branch?
    A) A pointer to a commit
    B) A container of commits
    C) A clone of the original project
    D) No idea
4. When should branches be used? (Multiple answers are allowed)
    A) When wanting to experiment on the project without affecting the original project
    B) Whenever cloning project
    C) When wanting to create additional features
    D) No idea
5. What does git checkout do?
    A) Retrieves the latest changes to the repository
    B) Switches to a branch
    C) Creates a new branch
    D) No idea

6. What does git commit do?
    A) Adds staged changes to the local repository
    B) Adds staged changes to the remote repository
    C) Adds staged changes to both repositories
    D) No idea
7. What does git push do?
    A) Appends staged changes to the current commit
    B) Takes the current commits from the current branch to another branch
    C) Takes commits from the local repository and sends them to the remote repository
    D) No idea
8. What does git pull do?
    A) Retrieves the latest content from the remote repository but doesn't update the local repository
    B) Retrieves the latest content from the remote repository and updates the local repository to match that content.
    C) Clones a repository
    D) No idea
9. What does git merge do?
    A) Joins two branches together into one branch. The git history of both branches are also joined together.
    B) Adds a new change to the local repository
    C) Joins two branches together into one branch. The history of one branch is not joined together with the other.
    D) No idea

10.3 Appendix C - Questionnaire Results

10.3.1 Pre Questionnaire Results for Giteducatus:

Question	P1 (Answer)	P2 (Answer)	P3 (Answer)	P4 (Answer)	P5 (Answer)	Correct Answer
1	Yes	Yes	Yes	No	No	-
2	No	No	No	No	No	-
3	1-3 Months	1-3 Months	Over a year	-	-	-
4	Assignments (University)	Assignments (University)	Assignments (University), Own Project	-	-	-
5	-	-	-	Do not know what it is	It is too difficult to learn	-
6	A	A	A	D	A	A
7	C	A	B	D	D	A
8	A, B	A, B, C	A, C	D	C	A, C
9	D	B	B	D	D	B
10	D	A	A	D	D	B
11	D	C	A	D	B	A
12	D	A	C	D	B	C
13	B	B	B	D	A	B
14	A	A	A	D	C	A
15	A	A	A	D	A	A
16	No Idea	-	Hello World	-	No Idea	Hello World
17	No Idea	-	Hello	-	No Idea	Hello
18	No Idea	-	World	-	No Idea	World

Score	5	7	12	0	3	14

10.3.2 Post Questionnaire Results for Giteducatus:

Question	P1 (Answer)	P2 (Answer)	P3 (Answer)	P4 (Answer)	P5 (Answer)	Correct Answers
1	Yes	Yes	Yes	Yes	Yes	-
2	No	Maybe	Yes	Yes	Maybe	-
3	A	A	A	A	A	A
4	C	C	B	C	B	A
5	A, B	A, B, C	A, C	A	A	A, C
6	B	B	B	B	B	B
7	B	B	A	B	B	B
8	A	A	A	A	A	A
9	C	C	C	C	B	C
10	B	B	B	B	B	B
11	A	A	A	B	A	A
12	A	A	A	A	A	A
13	-	remove < > =	Hello World	Modify the text and save	remove <<<<<<< , ======= & >>>>>>>	Hello World
14	Remove everything except "hello"	-	Hello	Delete the other branch text and save	remove ======= & >>>>>>>	Hello
15	Remove everything except "World"	-	World	Delete the current branch text and save	remove <<<<<<< & =======	World

Score	11	10	12	8	8	14

10.3.3 Pre Questionnaire Results for Tutorial:

Question	T1 (Answer)	T2 (Answer)	T3 (Answer)	T4 (Answer)	Correct Answer
1	Yes	No	No	No	-
2	No	No	No	No	-
3	4-8 Months	-	-	-	-
4	Assignments (University), Own Projects	-	-	-	-
5	-	Do not know what it is	Do not know what it is	Do not know what it is	-
6	A	D	D	D	A
7	C	D	D	D	A
8	A, C	D	D	D	A, C
9	B	D	D	D	B
10	A	D	D	D	A
11	C	D	D	D	C
12	B	D	D	D	B
13	C	D	D	D	A
Score	7	0	0	0	9

10.3.4 Post Questionnaire Results for Tutorial:

Question	T1 (Answer)	T2 (Answer)	T3 (Answer)	T4 (Answer)	Correct Answer
1	Yes	Yes	Yes	Yes	-
2	C	C	C	A	A
3	B	B	B	C	A
4	A, C	A	A	A, C	A, C
5	B	B	B	C	B
6	A	B	A	C	A
7	C	C	C	C	C
8	B	B	B	B	B
9	C	C	C	A	A
Score	6	4	5	6	9

10.4 Appendix D - Interview Results

10.4.1 Interview P1 (Translated from Swedish to English)

Q1: Did you feel like you learned a lot about the version control system?

A: Yes more than what I've learned from reading Tutorials on the net. I've done the most basic things on Git such as git pull, git push, git commit and something like that.. I understand a bit better what git fetch and branch means now.

Q2: Was there difficulty with any Git commands that you didn't feel like the game covered so much?

A: Yes I would say git checkout and git add. And the flag parameter of git commands such as git commit -m, perhaps there can be a quest of these flags.

Q3: Assuming you have done tutorials on Git, would you use the game as an alternative to tutorial?
A: Yes although I haven't done so many tutorials. But I wouldn't play the game because it is fun as it is not but rather because I want to learn. I also think this game is good because of the interactivity.

Q4: Would you say you know why Git is being used after playing the game?
A: I already knew prior to the game

Q5: Would you also say that the game conveys why and when to use the git commands that you used?
A: Yes

10.4.2 Interview P2 (Translated from Swedish to English)

Q1: Did you feel like you learned a lot about the version control system?

A: Not a lot but a little bit like git checkout and git fetch.

Q2: Was there difficulty with any Git commands that you didn't feel like the game covered so much?
A: Yes a bit on merging. When merging two branches if the two git histories were also merged. The game did not cover that so much.

Q3: Assuming you have done tutorials on Git, would you use the game as an alternative to tutorial?
A: In some cases. If what I want to learn on Git is too specific then I will choose tutorial, but if I want to learn more commands then I will choose the game.

Q4: Would you say you know why Git is being used after playing the game?
A: Already knew prior to the game.

Q5: Would you also say that the game conveys why and when to use the git commands that you used?
A: Yes except on git add. But on the rest of the commands the game conveyed it really well.

10.4.3 Interview P3 (Translated from Swedish to English)

Q1: Did you feel like you learned a lot about the version control system?

A: Not much but I learned new things such as git fetch.

Q2: Was there difficulty with any Git commands that you didn't feel like the game covered so much?
A: Yes on merging. I think the game doesn’t mention whether the histories also get merged during git merge.

Q3: Assuming you have done tutorials on Git, would you use the game as an alternative to tutorial?
A: Yes.

Q4: Would you say you know why Git is being used after playing the game?
A: I already knew before playing the game.

Q5: Would you also say that the game conveys why and when to use the git commands that you used?
A: Yes

10.4.4 Interview P4 (Translated from Swedish to English)

Q1: Did you feel like you learned a lot about the version control system?

A: Yes. You get a good overview on how Git works and a basic foundation.

Q2: Was there difficulty with any Git commands that you didn't feel like the game covered so much?
A: Hmm no.

Q3: Assuming you have done tutorials on Git, would you use the game as an alternative to tutorial?
A: Haven’t done tutorials on Git but I would still prefer the game. Because you learn more when you do practical things than to watch a tutorial that tells you do this and that. And thing is that if you watch a video that tells you do this to achieve a result then you learn how to get the result but you don’t learn what it means and why.

Q4: Would you say you know why Git is being used after playing the game?
A: Yes.

Q5: Would you also say that the game conveys why and when to use the git commands that you used?
A: Yes it does that.

10.4.5 Interview P5 (Translated from Swedish to English)

Q1: Did you feel like you learned a lot about the version control system?

A: Yes. I'd say I got more familiar with using Git. But it was very difficult for me who have not used Git before.

Q2: Was there difficulty with any Git commands that you didn't feel like the game covered so much?
A: It was perhaps how git commands are explained and differentiated between each other. Like when to use git add before another command.

Q3: Assuming you have done tutorials on Git, would you use the game as an alternative to tutorial?
A: Possibly after reading more about Git on Stackoverflow. Once I would have a better understanding I would use the game similar to Codeacademy to test my knowledge on Git. I think the game is more like a tool that can be added together with say a tutorial.

Q4: Would you say you know why Git is being used after playing the game?
A: Already knew prior to the game.

Q5: Would you also say that the game conveys why and when to use the git commands that you used?
A: Yes and very clearly.