University of HUDDERSFIELD
University of Huddersfield Press

Journal of Play in Adulthood
Available open access at:
www.journalofplayinadulthood.org.uk

Journal of Play in Adulthood

Git going with LEGO: A playful partnership for making sense of technology

Tami Albin[a], Jamene Brooks-Kieffer[b]

[a]*University of Kansas Libraries, 1425 Jayhawk Boulevard, Lawrence, Kansas, 66045, USA*
[b]*University of Kansas Libraries, 1425 Jayhawk Boulevard, Lawrence, Kansas, 66045, USA*

ARTICLE INFO

Article history:

Keywords:
Git
Version Control Software
LEGO Serious Play
The Carpentries
Teaching
Partnership
Play
Novice

ABSTRACT

Git, an open source version control software, is difficult to both teach and learn because it conceals its work in a hidden directory and is often deployed using a command line interface. Yet, version control is an essential tool of reproducible research. Two librarians formed a playful partnership to experiment with teaching Git using LEGO, an approach inspired by our Carpentries Instructor certifications and one's LEGO Serious Play certification. Our method combines novice-centric Carpentries pedagogy with theories of play, constructionism, constructivism, conceptual metaphor, and the hand-mind connection that are foundational to LEGO Serious Play. We share details of developing, testing, and assessing this Git-LEGO combination. Our preliminary study indicates that our LEGO activity may increase learners' short-term retention of Git commands. We discuss future directions for our research and our experiences as academic librarians in a playful partnership.

Introduction

The playful partnership behind this paper began with an invitation. Jamene, a certified Carpentries instructor, felt inept and frustrated after teaching Git, a broadly practical version control software that is a stinker to learn. She invited Tami and her LEGO to a meeting to explore whether the solid properties of LEGO could help better explain the elusive Git. Because Git is only visible in its effects on files and through its log, presenting it to learners felt like holding up vapor for inspection. With nothing for novices to hang onto, participants in Git lessons reached their cognitive limit before they understood anything useful. But Jamene also knew that her office neighbor was a LEGO trainer of some kind, and what is LEGO if not touchable?

Tami, feeling similarly frustrated, was fresh from LEGO Serious Play Facilitator (LSP) training, but had little opportunity to implement what she had learned. LSP is an established methodology built upon many years of research grounded in "psychological theories of learning" which include play, constructionism, flow, the hand-mind connection, imagination, metaphor, and complex adaptive systems (beliefs)" (McCusker, 2014, p. 28;

Rasmussen, 2012; Barton & James, 2017, p. 252). It is also extremely time-consuming, with the shortest activities taking upwards of three hours. It was difficult to explain to over-extended academic library colleagues why spending three hours in a LEGO activity would contribute to their mission. With gusto, she accepted Jamene's invitation, flung wide her office toy box, and arrived at their meeting with the bounty of LEGO that had not seen the light of day as often as she had hoped.

What followed was an extended tête-à-tête where we spread LEGO all over the conference table, talked about what Git was for, and looked at a computer only to reference the next Git command and concept from the Software Carpentry Git lesson (Figure 1). About one week later, Tami could recall and draw the steps of the LEGO build and the Git concepts they represented despite never having used Git before. Through our play in the conference room, mashing up Git with LEGO pieces, we identified and articulated the problem that the rest of this paper addresses. A gap exists between visual and physical tools that are intended to help people understand Git and the novice learners who most need that help. Theories that form the backbone of LEGO Serious Play inform our approach to filling this gap. One invitation led to many more and resulted in the collaborative intervention that is the subject of this paper, an activity that we call the "Git Stack."

Figure 1
The authors' LEGO minifigure avatars buried in LEGO

Background

The gap between Git representations and the novice learners who need them is a problem that we came to understand by combining our areas of expertise. Here, we parse out some key information about each distinct area to contextualize the problem that we called on LEGO to solve.

Git

Every writer and beginning coder knows the sinking feeling of having deleted text or code yesterday that they wind up needing today. Comics poke fun at multiplying files with names like "Final," "FinalFinal", and "FinalReallyThisTime" (Cham, 2012). Wilson et al. (2017) describe a manual process for managing this chaos, but the most effective solution is to adopt version control software. Version control software creates an unlimited undo feature for project files (Munk et al., 2019). As long as information recorded by the software remains intact, changes all the way back to the beginning of a project can be retrieved. One of these version control tools is Git, first released in 2005 by Linus Torvalds for controlling development of the Linux operating system (Straube, 2016).

Git is powerful, fast, and flexible, enabling complex collaborations and contribution monitoring (Milliken et al., 2021; Straube, 2016). The software's features, market dominance, and free, open source license make it an essential tool of research for purposes of reproducibility, collaboration within and among large teams, and communication of results and products (Beckman et al., 2021; Bryan, 2018; Hall et al., 2018; Himmelstein et al., 2019; Ram, 2013; Sandve et al., 2013; Szitenberg et al., 2015; Wilson et al., 2017; Wittman & Aukema, 2020).

While Git is popular and heavily used, it is also criticized for its complexity and poor usability (Beckman et al., 2021; Bennett, 2012; De Rosso & Jackson, 2016; Straube, 2016). Git's default user interface is the command line (CLI), which requires that users type the correct command word using accurate syntax in order to complete a task. People who use Git encounter unintuitive command terms, inconsistent syntax, and enigmatic documentation (Bennett, 2012; Isomöttönen & Cochez, 2014; Milliken et al., 2021).

University instructors have found teaching Git in for-credit courses to be both difficult and necessary for students' preparedness in fields such as software development, statistics, and data science (Beckman et al., 2021; Bryan, 2018; Fiksel et al., 2019; Glassey, 2019; Isomöttönen & Cochez, 2014). Research conducted by Milliken et al. (2021) finds that scholars learn Git for version control, course requirements, and project collaboration, yet often fail to thoroughly understand the software. To address the tension between academics' need to learn Git and the challenges of doing so, the researchers recommend regular, scaffolded learning opportunities, including workshops taught using the Software Carpentry curriculum.

Software Carpentry

The Carpentries Foundation ("The Carpentries") is a non-profit foundation housing and coordinating the global communities that have grown up around three different openly available curricula: Software Carpentry (Wilson, 2006, 2016), Data Carpentry (Teal et al., 2015), and Library Carpentry (Baker et al., 2016). Among these curricula, Software Carpentry and Library Carpentry include lessons about Git (Erdmann et al., 2019; Munk et al., 2019). In this paper we primarily discuss workshops teaching the Software Carpentry curriculum ("Software Carpentry workshops") and the Software Carpentry Git lesson.

Instructors ("Carpentries instructors") certify to teach Software Carpentry workshops by completing training that introduces "educational psychology, instructional design, and how these things apply to teaching programming" (Wilson, 2016, p. 9). Instructors learn to deliver information in ways that novice learners can take in without becoming overwhelmed. This problem, called cognitive load, describes how learners' limited attention can be swamped by switching focus, overly difficult tasks, or expectations to perform multiple tasks at the same time (Ambrose, 2010, p. 103; Deans for Impact, 2015; Word et al., 2021, "Memory and Cognitive Load").

During a typical two-day, three-topic workshop, an instructor teaches the three-hour Git lesson using a CLI. The instructor delivers the lesson using a method called participatory live coding, which provides learners with continuous hands-on practice during the workshop (Wilson, 2016; Word et al., 2021, "Live Coding is a Skill"). Instructors assume that learners are new to Git; they focus the instruction on basic version control concepts and essential Git commands ("What is a Carpentries Workshop?," n.d.; Munk et al., 2019).

Research conducted by The Carpentries indicates that this environment is effective in helping novices learn Git (Jordan, Marwick, Duckles et al., 2017; Jordan, Marwick, Weaver et al., 2017; Jordan et al., 2018; Jordan & Michonneau, 2020). Even so, Git is still difficult to both teach and learn because of its confusing commands, obscure terminology, and invisibility within the CLI.

Visualizations

Milliken et al. (2021) note one potential way to address this challenge: create and steer learners toward visual and physical representations of Git. Indeed, case studies from the literature advocate for teaching, learning, and using Git through graphical user interfaces (GUIs) (Beckman et al., 2021; Bryan, 2018; Fiksel et al., 2019). GUIs for using Git include dedicated clients and add-ons for Integrated Development Environments (IDEs). Web visualizations diagram the interplay between Git commands and particular states of a project over time (Lodato, 2010; Wang, 2015). Schwern (2013) took Git into the physical realm by building a representation of Git's internal processes out of Tinker Toy pieces.

We find, however, that most of these interpretations of Git defy effective practices for teaching novices:

managing learners' cognitive load by limiting distractions, slowing the introduction of new information, and expecting no prior knowledge. GUI clients present a great deal of information and many choices all at once. IDEs expect a user to be at least conversant in a relevant programming language. Web visualizations require some familiarity with Git (Lodato, 2010). Even Schwern's (2013) Tinker Toy build assumes prior knowledge of 12 Git commands.

These attempts to visualize Git are likely to *increase* novice learners' cognitive load at a time when the piece of software they are trying to learn is already pushing the limits of their working memory. Learners who are close to cognitive overload will withdraw their attention if they are faced with additional expectations to understand a complex image or if they do not know something the instructor assumes is common knowledge. Novice learners of Git deserve representations of the tool that do not demand anything extra from them.

LEGO Serious Play

The genesis of Lego Serious Play (LSP) dates to the mid-1990s with Johan Roos and Bart Victor, professors of leadership and strategy at the International Institute for Management Development (IMD) in Lausanne, Switzerland. Through their research, they realized that leadership management teams in the business world viewed objective, data-driven analysis as a superior approach for making strategic decisions. Teams did not value or view as relevant the subjective and experiential knowledge of leadership managers during strategic planning processes. Roos and Victor found two possible reasons for this dismissal of subjective and experiential knowledge: managers feared trying to legitimise and justify their subjective experiences against objective data, and/or managers encountered difficulty fitting their embodied experiences into strategic frameworks or models in use within the organization. Uncovering the ineffectiveness of strategic development leadership teams gave Roos and Victor an opportunity to envision a new framework that would result in the methodology we now know as Lego Serious Play (Roos & Victor, 2018).

LSP methodology "offers a sophisticated means for a group to share ideas, assumptions and understandings; to engage in rich dialogue and discussion; and to work out meaningful solutions to real problems" (LEGO Group, 2010, p. 10). According to Frick et al. (2013), LSP follows a set of basic values that include: "The answer is in the system. Everyone has to express his/her reflections. There is no ONE right answer" (p. 3). LSP facilitators lead and guide participants through the four-step Core Process which, in essence, involves a facilitator "asking a question," followed by "participants Building, Sharing and Reflecting" (McCusker, 2014, p. 28). Certified facilitators attend a four day course for LSP v. 2.0. This course, led by members of the Association of Master Trainers, includes the "4 step Core Process and 7 Application Techniques" (Rasmussen & Kristiansen, 2022).

For years, LEGO has been used as a tool for exploring and teaching complex concepts without explicitly engaging in the theoretical framework of LEGO Serious Play (Gridley, 2018; Hood & Hood, 2005, 2006; Ornes,

2015; Paasivaara et al., 2014; Steghöfer et al., 2017). LEGO Serious Play has been implemented and integrated directly into some for-credit courses (Barton & James, 2017; James & Brookfield, 2013; Kurkovsky, 2015; Nerantzi & James, 2019; Schulz & Geithner, 2011). There are very few examples of LEGO or LSP being used in library or technology instruction (Bond, 2018; Buckley, 2015).

Theoretical Framework

It is important to note that our collaboration — at least in this moment of our Git-LEGO explorations — is far from a one-to-one mash-up of Software Carpentries Git lessons and LEGO Serious Play workshops. Instead, we used LEGO and selected theories underpinning LEGO Serious Play to develop an additional activity for a Software Carpentry Git Lesson in hopes of concretizing the abstract (we anticipate gasps from Software Carpentries and LEGO Serious Play practitioners alike, as we veer from the pedagogies of each, even as we put them in productive play). The theories we engage from LSP include: play, constructionism, constructivism, conceptual metaphor, and the hand-mind connection. These theories complement the learning theories embedded within the pedagogical framework used in Software Carpentry lessons and are valuable toward the creation of our intervention. For this intervention, we did not use flow or complex adaptive systems theories of learning, though these are also critical grounding within LSP.

The concept of play is a cornerstone for LSP. Play, according to many scholars does not have a single definition (James, 2021; Sicart, 2014). Some believe that it is enigmatic (Bateson & Martin, 2013), while others view it as ambiguous (Sutton-Smith, 2006). Play has a range of meanings, contradictions, and lacks any sort of universal consensus (James, 2021). However, there does seem to be agreement among scholars that play can be defined by specific characteristics. It is a "limited, structured and voluntary activity that involves the imaginary" (Rasmussen, 2012, para. 7). Play for adults is almost always intentional, with a particular purpose in mind (Kristiansen & Rasmussen, 2014; Roos et al., 2001).

Constructionism, developed by Seymore Papert (Papert & Harel, 1991), is key to LSP and is rooted in Jean Piaget's work on Constructivism. Constructivism is "a theory of learning as an active process, in which people actively construct knowledge from their personal experience of the world" (Monga et al., 2018, p. 1). According to Ackerman (2001), constructionism takes constructivism a step further and posits that self-directed, tactile experiences of creating and making are dynamic, unique, and situational. In these moments of tensions and transitions, the learner must adjust, stretch, and expand their understanding of the situation. This creates a conflict and incompatibility between expectation and experience, which according to Papert, is a "crucial part of learning" (Ackermann, 2001, p. 10). Eventually, the individual makes sense of the situation and what they learn, and transitions into knowledge. Once this happens, the individual will develop a "deeper connectedness and understanding" which will enable them to take this new knowledge and move forward to the next

situation (Ackermann, 2001, p. 10).

Metaphor is a figure of speech, often used to help people make sense of a situation or experience. Metaphor provides a way of framing a story or experience by connecting or carrying over "frames or perspectives from one domain to another" (Schön, 1993, p. 138). In Kövecses's work on conceptual metaphor theory, there are two different conceptual domains at play: source domain (familiar, concrete, experiences) and target domain (complex, abstract) (2010, p. 3-4). By linking these two domains using metaphor, intangible concepts are connected to a familiar image or object. The imagery and concretization through the metaphor creates a make-sense moment and a mental model for future reference. When working with novices, however, conceptual metaphor theory may not provide sufficient support for making connections between the target and source domains. To ensure that novices have the necessary references, Schön's (1993) concept of the generative metaphor comes into play.

According to Schön (1993), generative metaphor occurs when individuals in a situation - often a problem or dilemma to solve - move beyond describing the situation to immersing themselves in the situation. This immersion is often tactile and impacted by each individual's previous personal experiences and sensorial triggers. The combination of immersion, experiences, and senses generates new perspectives and innovations, which Schön (1993) refers to as "seeing-as" (pp. 137-138). The challenge inherent within generative metaphor arises when a problem or situation is not well-structured. In these cases, "seeing-as" can introduce a breadth of new perspectives that reach beyond the problem or situation at hand. To use generative metaphor well, Schön suggests employing metaphor to set up a particular problem or situation. This approach focuses on problem-setting instead of problem-solving. Problem-setting acknowledges the complexity of a situation, where problem-solving seeks to solve a given problem; the solution may not consider potential ramifications or additional problems.

By setting up and framing the problem or question at hand for novices, followed by mapping and naming the meaning from the abstract target domain to the familiar source domain, the abstract becomes concrete. Solidifying the abstract for novices reduces the amount of cognitive load necessary for understanding the metaphor. The novice experiences less stress and can engage more fully in their activity.

The hand-mind connection is seen as a way to "manipulate and construct the world" and a way for the brain to "construct its own knowledge of the world (Rasmussen, 2012, para. 55). As James and Brookfield (2013) point out, "we learn through our senses and our thinking is formed by sensory engagement" (p. 2). Engaging with one's "senses in different ways shifts how people respond to their situation and the world around them" (James & Brookfield, 2013, p. 2). Hands-on active engagement with objects is key to constructivism and constructionism and as stated earlier leads to deeper learning (Bürgi et al., 2005; Hadida, 2013, p. 3; Kurkovsky,

2015, p. 13).

The Intervention

During the summer of 2018, we spent many lengthy meetings in a conference room, buried in LEGO, teaching each other Git and elements of LEGO Serious Play. The activity that we developed and refined in those meetings maps individual LEGO pieces to specific Git commands. The learner builds the LEGO, piece by piece, into a tower while simultaneously building up a history of changes to a file in Git's record of the project, called a repository. We call this activity the Git Stack.

The Git Stack models how five essential Git commands work together to record and review changes to a project's history. Learners in a Software Carpentry Git lesson practice these commands repeatedly through participatory live coding. The five commands are listed and defined in Table 1:

Table 1
Definitions of Git commands present in the Git Stack Activity

Command	Definition
`git init`	Initialize a Git repository to record changes to a project.
`git add`	Record changes to one or more files to the repository.
`git commit`	Assign a unique ID to the changes added to the repository. Include a message describing the changes.
`git diff`	Compare differences between file versions recorded to different commits in the repository.
`git checkout`	Review or restore a previous file version committed to the repository.

We chose to structure the Git Stack activity around three early episodes in the Git lesson: "Creating a Repository," "Tracking Changes," and "Exploring History" (Munk et al., 2019). These episodes teach the commands defined in Table 1 as well as the command `git status`, which checks on the state of the Git repository. During our playful meetings, we discovered that LEGO pieces can model these key commands and their actions with reasonable accuracy.

The Git Stack uses commonly available LEGO pieces that are inexpensive, easy to source, and easy to stack and pull apart. We selected plates to represent Git commands and bricks to represent the creation and modified versions of a file. The colors of the bricks and plates have some significance in the activity, but we find that consistency is more important than the actual colors. Once we determined the correct number and type of LEGO for the activity, we purchased the pieces and bagged them into individual LEGO kits that learners can use during the Git lesson. Figure 2 shows the inventory of bricks and plates in one LEGO kit. We detail the process of building the Git Stack in the sections below.

Figure 2
Inventory of LEGO bricks and plates in one kit for the Git Stack activity

1 – 2x4 purple brick		1 – 2x4 purple plate	
1 – 2x4 yellow brick		1 – 2x4 yellow plate	
1 – 2x4 white brick		1 – 2x4 white plate	
1 – 2x4 green brick		1 – 2x4 green plate	
1 – 2x4 orange brick		1 – 2x4 orange plate	
5 – 2x4 black plates		1 – 8x8 grey plate	

Lay the Foundation

In the "Creating a Repository" episode of the Git lesson, learners create the Git repository that will store the history of their project (Munk et al., 2019). The instructor begins this episode by demonstrating creation of a dedicated project directory while learners follow along on their laptops. Inside the new project directory, the instructor initializes, or creates, a Git repository using the `git init` command. Once learners have issued this command on their own computers, the instructor directs them to take the 8x8 grey plate from their LEGO kit and place it on their work surface. This large plate represents the Git repository. Figure 3 shows the plate that represents the command `git init`.

Figure 3

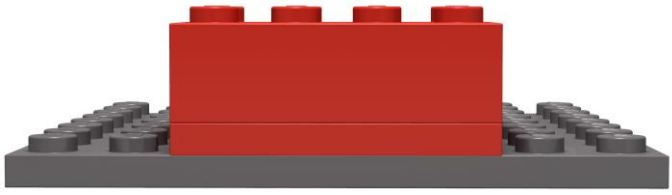
LEGO 8x8 plate representing `git init`

Practice the Pattern

In the "Tracking Changes" episode, learners repeatedly practice the two-stage process of saving project information in their Git repository (Munk et al., 2019). The instructor begins this section by creating a new plain text file in the project directory as learners follow along. The instructor and learners then type the command `git add` plus the name of the new file: `git add filename.txt`. Learners select one 2x4 brick and the matching 2x4 plate from their LEGO kits. The instructor specifies that the brick represents the new file and the plate represents the `git add` command. When learners execute the command, they stack first their brick and

then the matching plate onto the large grey plate on their work surface. These actions represent adding the new file to the Git repository. Figure 4 shows how the brick and plate stack up.

Figure 4
LEGO 2x4 plate and brick stacked to represent `git add` plus the file

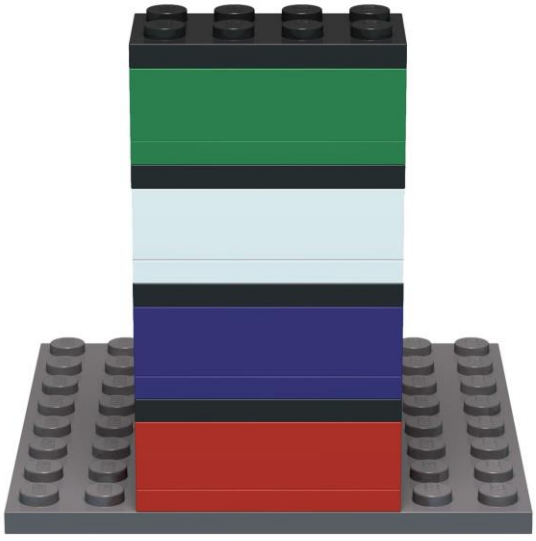


Git's record of this change is not complete until the change has been issued a unique ID, called a commit ID. To complete this second step, the instructor demonstrates the command `git commit`. Git requires a description of the changes being committed; this is called a commit message. Along with the instructor, learners type `git commit -m "Create new file"`. When this command executes, learners stack a black 2x4 plate on top of the plate representing `git add`. This black plate stands for the action of committing changes to the repository with a unique commit ID. Within this project, Git can always recall this set of changes using the commit ID. Figure 5 shows the Git Stack once the first commit is complete.

Figure 5
Black LEGO 2x4 plate added to the stack to represent `git commit`

The instructor and learners work through this process three more times during this episode: modify the text file, `git add` the changed file to the repository, and `git commit` the change to create a commit ID. Each time the learners issue these commands, they build up the Git Stack using a matching brick and plate to add the changed file, then a black plate to commit the change. By the end of this episode, learners have a Git Stack that represents the Git repository and four distinct changes to their project, with the most recent change at the top of the stack. Figure 6 shows the Git Stack at the end of the "Tracking Changes" episode.

Figure 6
Git Stack representing four distinct commits to the Git repository



The Ah-Hah Moment

The "Exploring History" episode disrupts the modify-add-commit rhythm learners found in "Tracking Changes." In this episode, learners undo recent changes to their file in order to experience the purpose and power of Git (Munk et al., 2019). After making edits to their file, the instructor and learners use the `git diff` command to examine differences between current changes and a previous version. Learners can pick up a LEGO brick, representing the changed file, and compare it to a brick in any layer of the Git Stack. Because the bricks are different colors, learners will immediately see a difference in the model.

A typical version control workflow involves finding differences between versions of a file and, if possible, restoring text that had been deleted and is now desired. The instructor demonstrates how to do this using Git's command `git checkout`, restoring the file to the way it was at the beginning of the episode. Learners can check their file and their Git Stack to see that their project is back to the same state it was at the end of "Tracking Changes." They discover that they can rely on Git to remove undesired changes; they do not have to re-edit the file or depend on an unpredictable undo feature.

Learners then experience the erasure of all the changes they made to their file during the "Tracking Changes" episode. Invoking the commit ID, the instructor and learners type `git checkout g94e501 filename.txt` to roll back the file's contents to the very first commit. Learners then take down three layers of the Git Stack, leaving them with a tower that matches the example shown in Figure 5. When they check the contents of their file, learners see only the first line of text they wrote and saved. The instructor then helps learners bring back all of the text written in the file during the "Tracking Changes" episode by guiding learners through the command `git checkout HEAD filename.txt` (Git's `HEAD` designation specifies the most recent commit).

When learners see that their file once again contains four lines of text, they put the three layers of LEGO pieces back onto their Git Stack. Un-building and then re-building their Git Stack help learners comprehend that Git commands both record new project information and restore old information. Figure 6 shows the state of the Git Stack at the end of the "Exploring History" episode.

Git Stack Summary

The LEGO tower that results from the Git Stack activity, with its layers of intentionally chosen bricks and plates, models the layers of commits built up in the project's history and recorded in the Git repository. The stack of LEGO pieces is an accurate and tangible representation of Git's essential functions that novice learners need to know first. Unlike Schwern's (2013) Tinker Toy representation of Git, the Git Stack requires no prior knowledge. Where Schwern's Tinker Toy build was a demonstration for an audience, we designed the Git Stack as kits that instructors can distribute to learners. Each learner, then, has the opportunity to not only see a model of Git's commands and actions, but also to touch the pieces, build, un-build, and re-build the model for themselves.

A stack of LEGO bricks and plates is a particularly good fit for tangiblizing Git. As explored by Straube (2016), "the stack" underpins the design, implementation, understanding, and investigation of digital technologies, including invisible infrastructures such as Git. The stack consists of hierarchically organized layers that form conceptual models and technical realities of many aspects of computing, including communication protocols, software-to-hardware translations, and server configurations. Straube reveals Git interacting with time and multiple spaces in actual, rather than merely metaphorical, ways that exceed human understanding. "It is … imperative to be prepared for encounters with time-spaces that are difficult or impossible to relate to, without reducing them to some more intuitive framing" (p. 8). Straube argues for the stack as that "more intuitive framing;" a means of grounding the incomprehensible in the physical realm (p. 8).

Relevance to the Problem

As mentioned previously, Git is problematic for novice learners; its complexity, steep learning curve, terminology, and intangibility create considerable challenges and frustration. We were not interested in creating Git super-users; rather, we wanted to take an incredibly valuable, yet problematic, tool and create new, tangible entry points into understanding and using Git.

It was important to us, as trainers, to ensure that the Git Stack activity, as well as the space we were inhabiting during the Git lessons: 1) were playful, and as instructors create an affect that encouraged a playfulness mindset that included openness, understanding, positivity, and encouragement, especially with making mistakes; 2) involved the notion of play via LEGO, a toy that is familiar to many and relatively easy to use even for novice users; and 3) were grounded in some LEGO Serious Play foundational theories.

Introducing play and playful behavior for both the learners and instructors in the Git lesson is not a typical approach in a Carpentries workshop; rather, it is disruptive. By modeling and encouraging people to think creatively, be curious, embrace a sense of humor, enjoy the moment, and experiment with making mistakes, we are inviting our participants to take a calculated risk with us in this new approach (Guitard et al., 2005).

The Git Stack activity most likely causes some cognitive overload because it interrupts the live coding, but this is not necessarily a bad thing. It is necessary for novice learners to physically hold, see, and feel the Git Stack as they work through the live coding. The hands-on build engages the learners' other senses with the shifting back and forth between live coding and hand building, from the abstract to the concrete, creating a deeper understanding and embodiment of Git. The disconnect and fuzziness of Git comes into focus and begins makes more sense to learners. Learners can add or remove pieces as necessary as they work through specific Git commands on the command line.

Assigning novice Git learners the task of creating their own Git metaphor and mental model from the LEGO pieces would be incredibly challenging and cause significant cognitive load on their working memory. There is no doubt that learners would create some sort of metaphor and that each creation would have a different Git interpretation, meaning each metaphorical Git Stack model (and mental model) would look very different. Most likely, these models would also be incorrect. This would require learners to undo what they have just taught themselves. It would be a very frustrating scenario and add to their cognitive load.

By naming the problem, naming, and mapping each Git concept directly onto particular pieces of LEGO, we take on the responsibility to set up the necessary framework and terminology for the learner. We create a temporal space that invites learners to embody their experiences and understanding of Git differently than solely through live coding. It is through embodiment of the Git Stack experience that the intangible becomes tangible.

Testing the Intervention

After we created the Git Stack activity and assembled kits containing the correct LEGO pieces, we tested the activity on actual learners at Software Carpentry workshops. Tami certified as a Software Carpentry instructor so that either of us could teach the Git lesson. Our study design, instrument, consent form, and recruitment strategy were approved by the University of Kansas (KU) Institutional Review Board under IRB # STUDY0014274.

Survey Instrument

We used a pre- and post-workshop survey design. The pre-workshop survey asked learners about their previous experiences with either Git or LEGO. The post-workshop survey asked learners about their experience learning Git during the workshop and their personal assessment of whether the Git Stack activity

helped or hindered their learning. The post-workshop survey also included a factual question asking participants to match a Git command to an image of a particular stage of the Git Stack. We chose the commands `git init`, `git add`, and `git commit` for this quiz question because they are critical steps for starting a Git repository and regularly recording changes using this software. The surveys did not collect any identifiable information about learners. The Carpentries' pre- and post-workshop surveys (Becker, 2020b, 2020a) informed the questions and answer choices we included in our surveys. We revised our post-workshop survey after one deployment. The contents of the pre-workshop survey and both versions of the post-workshop survey are available in the Appendix.

Study Setting and Population

The University of Kansas (KU) Libraries co-sponsor between two and five Carpentries workshops per year. Participants include KU faculty, staff, postdoctoral researchers, graduate students, and community and regional learners. Jamene arranged to teach the Git lesson with the Git Stack activity and deploy the survey to learners during three in-person workshops that took place between August 2018 and January 2019. A summary of the workshop topics and attendance is available in Table 2.

Table 2

Summary of information about workshops included in this study and their survey response rates

Curriculum	Date	Length	Lesson(s)	Attendance	Pre-workshop Responses	Post-workshop Responses
Software Carpentry	August 2018	two days	Unix Shell, Git, Python	28	28	25
Library Carpentry	November 2018	half day	Git	7	7	6
Software Carpentry	January 2019	two days	Unix Shell, Git, R	27	21	21

Since the Software Carpentry and Library Carpentry Git lessons introduce Git concepts and commands in very similar ways, we used the Git Stack in the same way in both lessons (Munk et al., 2019; Erdmann et al., 2019).

Survey Deployment and Responses

Participants were recruited from workshop attendees during the check-in process for the Git lesson. Attendees were given an information statement outlining the research project, the pre-workshop survey, information on opting out of the surveys, and LEGO kits for building the Git Stack. Learners completed the pre-workshop survey or left it blank; we then collected all surveys. The instructor then delivered the Git lesson, including the three modified sections incorporating the Git Stack. All learners had access to LEGO kits, regardless of their survey participation. At the end of the Git lesson, post-workshop surveys were distributed to learners to complete or leave blank and were collected along with the LEGO kits.

Both pre- and post-workshop paper survey results were entered into a spreadsheet. Attributes for workshop date and workshop type were added. Any identifying information included by respondents was not recorded.

Across the three workshops, 56 learners responded to the pre-workshop survey and 52 learners responded to the post-workshop survey. A breakdown of the responses per workshop is detailed in Table 2.

Findings

According to the survey data, these three workshops reached novice Git learners and introduced them to new material, both related to Git and in the use of LEGO for professional learning and problem-solving. Learners demonstrated short-term retention of essential Git commands when these commands were linked to specific stages of the Git Stack activity. Specific findings from the survey responses are detailed below.

Experience with Git

82% of participants in the pre-workshop survey indicated little to no experience with Git (n = 56), while 85% of participants in the post-workshop survey indicated that most or all of the lesson material was new to them (n = 52). This comparison confirms that the Git lessons within these three workshops reached novice learners.

Feelings about Git

Coming into the Git lesson, nearly half (46%) of pre-workshop survey respondents indicated feeling slightly or very intimidated by Git (n = 56). After the lesson, 68% of post-workshop respondents reported that they found Git less confusing after completing the LEGO activity (n = 44).

Knowledge of Git

After the lesson, most respondents gave correct answers to the quiz asking them to match a Git command to an image of a stage of the Git Stack. Results are detailed in Table 3.

Table 3

Results of the post-workshop quiz matching Git commands to stages of the LEGO build

Git command	LEGO image	% correct
`git init`		90%
`git add`		94%
`git commit`		94%
n = 52		

Use and Role of LEGO

86% of respondents to the pre-workshop survey indicated that they had never used LEGO in a professional capacity. Likewise, 91% reported that they had not used LEGO to understand a complex topic (n = 56).

Responses indicating which elements of the instruction were the most helpful stayed consistent from

version one to version two of the post-workshop survey. “Building with LEGO bricks” and “Hearing other participants’ questions” received mixed results in the middle-bottom of the ratings. “Typing Git commands” and “Hearing directions from the instructor” helped the most. “Discussing Git with other participants” was consistently rated last. Detailed percentages for both versions of the survey are included in Figures 7 and 8.

Figure 7
Detailed results of learners’ rankings of instruction methods in version one of the post-workshop survey

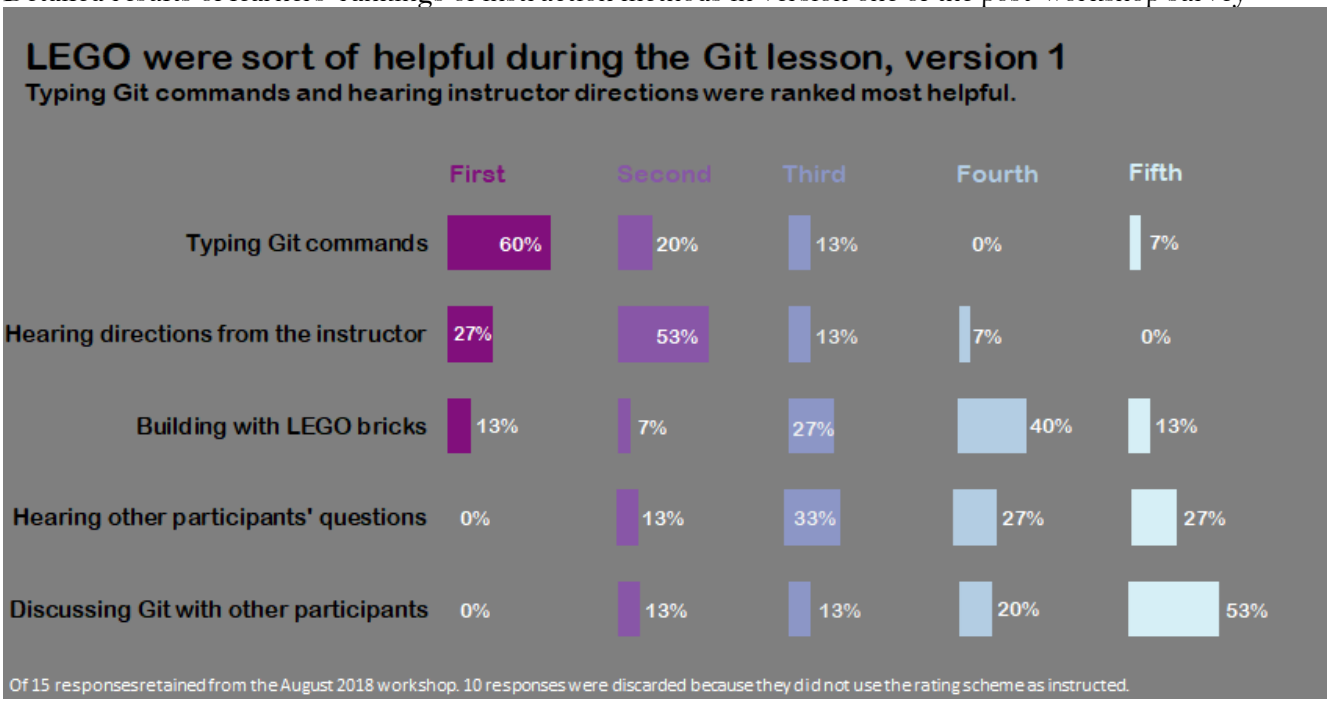
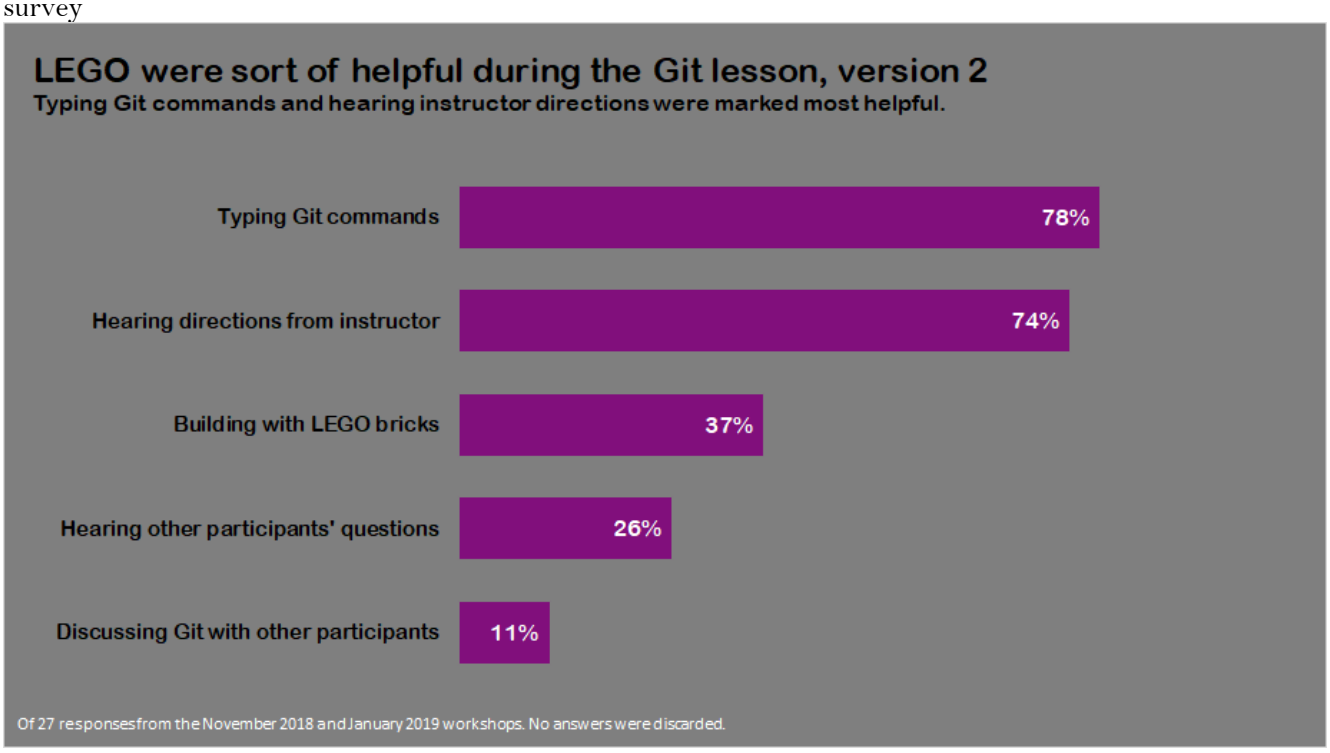


Figure 8
Detailed results of instruction methods learners marked as most helpful in version two of the post-workshop survey



Significance

Our survey is a small, preliminary study that needs further data collection. However, we find these results encouraging, since they demonstrate that the Git Stack helps novice learners retain essential Git commands, at least in the short term. Our current study design does not include surveying learners who attended a Git lesson where no Git Stack activity was used. However, The Carpentries has released reports analyzing results of their own pre- and post-workshop surveys (Jordan, Marwick, Duckles et al., 2017; Jordan et al., 2018). Respondents to The Carpentries' surveys presumably did not experience a tangible addition to the Git lesson.

We conducted an initial comparison between our results and The Carpentries' results specific to Git lessons (see Table 4). Our results show a very slight increase in novice learners' ability to complete a specific Git task after the workshop compared to The Carpentries' results. We cannot say for certain that this improvement is statistically significant or is caused by the Git Stack activity, but we believe this difference warrants further study.

Table 4
Comparison of select pre- and post-workshop survey results from The Carpentries versus this study

Survey Question	Carpentries finding (no LEGO)	This study finding (LEGO)
Pre-workshop: Knowledge of Git	78% reported little to none	82% reported little to none
Post-workshop: Knowledge of Git	63% reported at least some increase	68% reported finding Git less confusing
Pre-workshop: Perception of Git	57% reported feeling intimidated	46% reported feeling intimidated
Post-workshop: Successfully initialize a Git repository	88% reported increased confidence in completing this task	90% correctly matched `git init` to its LEGO build image
Carpentries: n ≥ 3200 (not all questions completed) All Carpentries results (Jordan, Marwick, Duckles et al., 2017, pp. 7-9, 13)		
This study: pre: n = 56; post: n = 52 (not all questions completed)		

Limitations

Our survey design ensured the anonymity of respondents but prevented us from being able to compare individuals' pre- and post-workshop responses. We can only make these comparisons in general terms where large majorities of responses on either side of the lesson must logically include some overlap. Our original post-workshop survey design would have benefited from testing to identify and correct issues we encountered during its first use at the August 2018 workshop.

Future Directions and Conclusion

This project presents us with many possible future directions for collecting more data to understand the significance of the Git Stack's effects. We can further investigate the difference the Git Stack seems to make for novice learners remembering Git commands by redesigning our survey method and collecting data from learners who both did and did not experience the Git Stack activity. We can also incorporate qualitative methods by interviewing workshop participants about their experiences using the Git Stack and any subsequent use of Git after a workshop. We can use participant observation methods to investigate changes in affect that may occur for learners during the workshop. This inquiry may be particularly relevant during the "Exploring History" episode, which disrupts a comfortable pattern of Git commands by un-doing and re-doing previous work. All of these future directions require us to be able to get back to offering in-person Carpentries workshops, something that KU Libraries have not done since January 2020.

As a first-time collaboration, our partnership could have gone wrong in many ways. We had not worked closely together in the past and needed to intentionally negotiate the scope of our involvement. We structured our venture using the playful methodology of LSP and the language of Git, committing to:

- Respecting your colleague. Honor their working and learning style, especially when it doesn't match your own.

- Doing what you say you will do.
- The mess of learning new things.
- Having fun while you learn.
- Being honest, particularly when you don't understand, get frustrated, or have a brain melt.
- Vulnerability and sharing what you do and don't do well.
- A healthy process, not only a final project.

Our gambit worked in part because we each bring a sense of play and experimentation to what we do. In our organization, we are both notorious for doing something generative to see what will happen. Although we work and think very differently, we both put mismatched things together, try ideas on, negotiate and push on the parameters of what academic librarians are "supposed" to do. We create environments where experimentation is safe and welcome. We hold space to get messy without judgment and learn something new without pressure. When we bring our whole selves to a playful partnership, we can find out what happens when a Carpentries instructor and a LEGO Serious Play facilitator walk into a bar (Figure 9).

Figure 9
The authors' LEGO Minifigure avatars walk into a bar

Acknowledgements

The authors contributed equally to the creation of this paper and choose to list our names in alphabetical order. We wish to thank Dr. Sherrie Tucker, Dr. Hannah Britton, and our reviewers for their suggestions that vastly improved this paper. Any remaining errors are ours alone.

All Figures containing LEGO and/or LEGO Minifigure images were created by Tami Albin.

This research was supported by a University of Kansas Libraries Research Fund 2018 grant.

References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference? *Future of Learning Group Publication*, 1–11. https://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., Norman, M. K., & Mayer, R. E. (2010). *How learning works: Seven research-based principles for smart teaching*. Wiley.

Baker, J., Moore, C., Priego, E., Alegre, R., Cope, J., Price, L., Stephens, O., van Strien, D., & Wilson, G. (2016). Library carpentry: Software skills training for library professionals. *LIBER Quarterly*, *26*(3), 141–162. https://doi.org/10.18352/lq.10176

Barton, G., & James, A. (2017). Threshold concepts, LEGO® SERIOUS PLAY® and whole systems thinking: Towards a combined methodology. *Practice and Evidence of the Scholarship of Teaching and Learning in Higher Education*, *12*(2), 249–271. https://www.pestlhe.org/index.php/pestlhe/article/view/171

Bateson, P., & Martin, P. (2013). *Play, playfulness, creativity and innovation*. Cambridge University Press.

Becker, E. (2020a, February 6). *The Carpentries post-workshop survey*. The Carpentries Survey Archives. https://carpentries.github.io/assessment-archives/post-workshop/post-workshop-2020-02-06-2b63c0.html

Becker, E. (2020b, February 6). *The Carpentries pre-workshop survey*. The Carpentries Survey Archives. https://carpentries.github.io/assessment-archives/pre-workshop/pre-workshop-2020-02-06-34facf.html

Beckman, M. D., Cetinkaya-Rundel, M., Horton, N. J., Rundel, C. W., Sullivan, A. J., & Tackett, M. (2021). Implementing version control with Git and Github as a learning objective in statistics and data science courses. *Journal of Statistics and Data Science Education*, *29*, S132–S144. https://doi.org/10.1080/10691898.2020.1848485

Bennett, S. (2012, February 24). 10 things I hate about Git. *Steve Bennett Blogs*. https://stevebennett.me/2012/02/24/10-things-i-hate-about-git/

Bond, M. (2018). Teaching referencing and plagiarism awareness using LEGO® SERIOUS PLAY®. *International Journal of Management and Applied Research*, *5*(4), 232–237. https://doi.org/10.18646/2056.54.18-017

Bryan, J. (2018). Excuse me, do you have a moment to talk about version control? *The American Statistician*, *72*(1), 20–27. https://doi.org/10.1080/00031305.2017.1399928

Buckley, C. (2015). Conceptualising plagiarism: Using LEGO to construct students' understanding of authorship and citation. *Teaching in Higher Education*, *20*(3), 352–358. https://doi.org/10.1080/13562517.2015.1016418

Bürgi, P. T., Jacobs, C. D., & Roos, J. (2005). From metaphor to practice: In the crafting of strategy. *Journal of Management Inquiry*, *14*(1), 78–94. https://doi.org/10.1177/1056492604270802

Cham, J. (2012). *PHD comics: NotFinal.doc*. https://phdcomics.com/comics/archive.php?comicid=1531

De Rosso, S. P., & Jackson, D. (2016). Purposes, concepts, misfits, and a redesign of Git. *ACM SIGPLAN Notices*, *51*(10), 292–310. https://doi.org/10.1145/3022671.2984018

Deans for Impact. (2015). *The science of learning*. https://deansforimpact.org/wp-content/uploads/2016/12/The_Science_of_Learning.pdf

Erdmann, C., Leinweber, K., Weaver, B., Baker, J., McGregor, N., Heggø, D. M. O., Duckles, J., Mendes, A., Brooks-Kieffer, J., Dennis, T., Oliver, J., McMillin, B., DSTraining, Williamson, E., Wick, R., Atwood, T., 222064h, Mendes, A., Marwaha, K., … Åkerström, W. N. (2019, July 2). *Library Carpentry: Introduction to Git for librarians, Version v2019.06.1*. Zenodo. https://doi.org/10.5281/zenodo.3265772

Fiksel, J., Jager, L. R., Hardin, J. S., & Taub, M. A. (2019). Using Github classroom to teach statistics. *Journal of Statistics Education*, *27*(2), 110–119. https://doi.org/10.1080/10691898.2019.1617089

Frick, E., Tardini, S., & Cantoni, L. (2013). *White Paper on LEGO ® SERIOUS PLAY: A state of the art of its applications in Europe*. Università della Svizzera Italiana. https://www.researchgate.net/publication/262636559_White_Paper_on_LEGO_R_SERIOUS_PLAY_A_state_of_the_art_of_its_applications_in_Europe

Glassey, R. (2019). Adopting Git/Github within teaching: A survey of tool support. *Proceedings of the ACM Conference on Global Computing Education*, 143–149. https://doi.org/10.1145/3300115.3309518

Gridley, A. (2018). Building LEGO® models to teach three-dimensional, mechanical concepts in optometry.

International Journal of Management and Applied Research, *5*(4), 238–244. https://doi.org/10.18646/2056.54.18-018

Guitard, P., Ferland, F., & Dutil, É. (2005). Toward a better understanding of playfulness in adults. *OTJR: Occupation, Participation and Health*, *25*(1), 9–22. https://doi.org/10.1177/153944920502500103

Hadida, A. L. (2013). Let your hands do the thinking! Lego bricks, strategic thinking and ideas generation within organizations. *Strategic Direction*, *29*(2), 2–5. https://doi.org/10.1108/02580541311297976

Hall, K. L., Vogel, A. L., Huang, G. C., Serrano, K. J., Rice, E. L., Tsakraklides, S. P., & Fiore, S. M. (2018). The science of team science: A review of the empirical evidence and research gaps on collaboration in science. *The American Psychologist*, *73*(4), 532–548. https://doi.org/10.1037/amp0000319

Himmelstein, D. S., Rubinetti, V., Slochower, D. R., Hu, D., Malladi, V. S., Greene, C. S., & Gitter, A. (2019). Open collaborative writing with Manubot. *Plos Computational Biology*, *15*(6), e1007128. https://doi.org/10.1371/journal.pcbi.1007128

Hood, C. S., & Hood, D. J. (2005). Teaching programming and language concepts using LEGOs®. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 19–23. https://doi.org/10.1145/1067445.1067454

Hood, D. J., & Hood, C. S. (2006). Teaching software project management using simulations. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 289–293. https://doi.org/10.1145/1140124.1140201

Isomöttönen, V., & Cochez, M. (2014). Challenges and confusions in learning version control with Git. In V. Ermolayev, H. C. Mayr, M. Nikitchenko, A. Spivakovsky, & G. Zholtkevych (Eds.), *Information and Communication Technologies in Education, Research, and Industrial Applications* (pp. 178–193). Springer. https://doi.org/10.1007/978-3-319-13206-8_9

James, A. (2021). Play in research? Yes, it is "proper" practice. *The Journal of Play in Adulthood*, *3*(1). https://doi.org/10.5920/jpa.864

James, A., & Brookfield, S. (2013). The serious use of play and metaphor: Legos and labyrinths. *International Journal of Adult Vocational Education and Technology (IJAVET)*, *4*(3), 1–12. https://doi.org/10.4018/ijavet.2013070101

Jordan, K. L., Marwick, B., Duckles, J., Zimmerman, N., & Becker, E. (2017). *Analysis of Software Carpentry's post-workshop surveys*. Zenodo. https://doi.org/10.5281/zenodo.1043533

Jordan, K. L., Marwick, B., Weaver, B., Zimmerman, N., Williams, J., Teal, T., Becker, E., Duckles, J., Duckles, B.,

& Wickes, E. (2017). *Analysis of the Carpentries' long-term feedback survey*. Zenodo. https://doi.org/10.5281/zenodo.1039944

Jordan, K. L., & Michonneau, F. (2020). *Analysis of the Carpentries long-term surveys (April 2020) v2*. Zenodo. https://doi.org/10.5281/zenodo.3753528

Jordan, K., Michonneau, F., & Weaver, B. (2018). *Analysis of Software and Data Carpentry's pre- and post-workshop surveys*. Zenodo. https://doi.org/10.5281/zenodo.1325464

Kövecses, Z. (2010). *Metaphor: A practical introduction* (2nd ed.). Oxford University Press.

Kristiansen, P., & Rasmussen, R. (2014). *Building a better business using the LEGO SERIOUS PLAY method*. Wiley.

Kurkovsky, S. (2015). Teaching software engineering with LEGO Serious Play. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 213–218. https://doi.org/10.1145/2729094.2742604

LEGO Group. (2010). *LEGO® SERIOUS PLAY® open-source*. http://seriousplaypro.com/about/open-source/

Lodato, M. (2010). *A visual Git reference*. https://marklodato.github.io/visual-git-guide/index-en.html

McCusker, S. (2014). LEGO® Serious Play™: Thinking about teaching and learning. *International Journal of Knowledge, Innovation and Entrepreneurship*, *2*(1), 27–37. http://www.journal.ijkie.org/IJKIE_August2014_SEAN%20MCCUSKERv3.pdf

Milliken, G., Nguyen, S., & Steeves, V. (2021). A behavioral approach to understanding the Git experience. In T. X. Bui (Ed.), *Proceedings of the 54th Annual Hawaii International Conference on System Sciences* (pp. 7239–7248). https://doi.org/10.24251/HICSS.2021.872

Monga, M., Lodi, M., Malchiodi, D., Morpurgo, A., & Spieler, B. (2018). Learning to program in a constructionist way. *Proceedings of Constructionism 2018*, Vilnius, Lithuania. https://hal.inria.fr/hal-01913065/document

Munk, M., Koziar, K., Leinweber, K., Silva, R., Michonneau, F., McCue, R., Hejazi, N., Waldman, S., Emonet, R., Harris, R. M., Olex, A. L., Becker, E. A., Lever, J., Burle, M.-H., Moore, B., Hoshijima, U., Maji, A., Topçuoğlu, B. D., Junghans, C., … Åkerström, W. N. (2019, July 1). *Software Carpentry: Version control with Git, Version v2019.06.1*. Zenodo. https://doi.org/10.5281/zenodo.3264950

Nerantzi, C., & James, A. (2019). *LEGO® for university learning: Inspiring academic practice in higher education* (Version 1). Zenodo. https://doi.org/10.5281/zenodo.2813448

Ornes, S. (2015). Inner workings: LEGOs in the lab. *Proceedings of the National Academy of Sciences*, *112*(42),

12901–12902. https://doi.org/10.1073/pnas.1515439112

Papert, S., & Harel, I. (1991). *Constructionism*. Ablex.

Paasivaara, M., Heikkilä, V., Lassenius, C., & Toivola, T. (2014). Teaching students Scrum using LEGO blocks. *Companion Proceedings of the 36th International Conference on Software Engineering*, 382–391. https://doi.org/10.1145/2591062.2591169

Ram, K. (2013). Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, *8*(1), 1-8. https://doi.org/10.1186/1751-0473-8-7

Rasmussen, R. (2012). *The Science behind the LEGO SERIOUS PLAY method*. Rasmussen Consulting. https://www.rasmussenconsulting.dk/s/TheScienceBehindtheLEGOSERIOUSPLAYMethod.pdf

Rasmussen, R., & Kristiansen, P. (n.d.). *Association of Master Trainers in THE LEGO® SERIOUS PLAY® method*. Serious Play Training. Retrieved February 6, 2022, from https://seriousplay.training/

Roos, J., Linder, M. O., & Victor, B. (2001). *Play in organizations* (Working Paper 2). Imagination Lab Foundation. https://imagilab.org/wp-content/uploads/2021/01/WP2.pdf

Roos, J., & Victor, B. (2018). How it all began: The origins of LEGO® SERIOUS PLAY®. *International Journal of Management and Applied Research*, *5*(4), 326–343. https://doi.org/10.18646/2056.54.18-025

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLOS Computational Biology*, *9*(10), e1003285. https://doi.org/10.1371/journal.pcbi.1003285

Schön, D. A. (1993). Generative metaphor: A perspective on problem-setting in social policy. In A. Ortony (Ed.), *Metaphor and thought* (2nd ed., pp. 137–163). Cambridge University Press.

Schulz, K. P., & Geithner, S. (2011, April 12). The development of shared understandings and innovation through metaphorical methods such as LEGO Serious Play™. *Making Waves*: *International Conference on Organizational Learning, Knowledge, and Capabilities (OLKC)*, Hull University Business School, Hull UK. http://www2.warwick.ac.uk/fac/soc/wbs/conf/olkc/archive/olkc6/papers/id_127.pdf

Schwern, M. (2013, January 31). *Git for ages 4 and up* [Conference session video from Linux.conf.au, Australian National University, Canberra, AU]. YouTube. https://www.youtube.com/watch?v=3m7BgIvC-uQ

Sicart, M. (2014). *Play matters*. MIT Press.

Steghöfer, J. P., Burden, H., Alahyari, H., & Haneberg, D. (2017). No silver brick: Opportunities and limitations of teaching Scrum with Lego workshops. *Journal of Systems and Software*, *131*, 230–247. https://doi.org/10.1016/j.jss.2017.06.019

Straube, T. (2016). Stacked spaces: Mapping digital infrastructures. *Big Data & Society*, *3*(2), 1-12. https://doi.org/10.1177/2053951716642456

Sutton-Smith, B. (2006). Play and ambiguity. In K. S. Tekinbas & E. Zimmerman (Eds.), *The game design reader: A rules of play anthology* (pp. 296–313). MIT Press.

Szitenberg, A., John, M., Blaxter, M. L., & Lunt, D. H. (2015). ReproPhylo: An environment for reproducible phylogenomics. *PLOS Computational Biology*, *11*(9), e1004447. https://doi.org/10.1371/journal.pcbi.1004447

Teal, T. K., Cranston, K. A., Lapp, H., White, E., Wilson, G., Ram, K., & Pawlik, A. (2015). Data Carpentry: Workshops to increase data literacy for researchers. *International Journal of Digital Curation*, *10*, 135–143. https://doi.org/10.2218/ijdc.v10i1.351

Wang, W. (2015). *Explain Git with D3*. GitHub. http://onlywei.github.io/explain-git-with-d3/

What is a Carpentries Workshop? (n.d.). The Carpentries. Retrieved December 23, 2019, from https://carpentries.org/workshops/

Wilson, G. (2006). Software Carpentry: Getting scientists to write better code by making them more productive. *Computing in Science Engineering*, *8*(6), 66–69. https://doi.org/10.1109/MCSE.2006.122

Wilson, G. (2016). Software Carpentry: Lessons learned. *F1000Research*, *3*(62), 1-24. https://doi.org/10.12688/f1000research.3-62.v2

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, *13*(6), e1005510. https://doi.org/10.1371/journal.pcbi.1005510

Wittman, J. T., & Aukema, B. H. (2020). A guide and toolbox to replicability and open science in entomology. *Journal of Insect Science*, *20*(3), 1-9. https://doi.org/10.1093/jisesa/ieaa036

Word, K., Sane, M., Barnes, K., Brown, S. M., Michonneau, F., Koch, C., Harris, R. M., Becker, E., Ballsun-Stanton, B., Capes, G., Hodges, T., Njambi, S., Stevens, S., Kamvar, Z. N., Li, A., Deardorff, A., Eranti, P., Hurt, S., Nenadic, A., … Mohanan, A. V. (2021, November 18). *The Carpentries instructor training, Version v2021.11.18-1*. Zenodo. https://doi.org/10.5281/zenodo.5709383

Appendix

Pre-Workshop Survey Questions

1. Please rate your level of experience with Git: (Circle one.)
   Little to none
   Some
   Proficient

2. Please rate your perception of Git: (Circle one.)
   Very intimidating to me
   Slightly intimidating to me
   Neither intimidating nor unintimidating
   Not very intimidating
   Not at all intimidating
   N/A or no opinion

3. How soon do you plan to implement Git into your work? (Circle one.)
   Immediately
   Within the next month
   Within the next 6 months
   More than 6 months from now
   Not sure
   Do not plan to implement

4. Please rate how often you have used Lego bricks in a professional capacity: (Circle one.)
   Never
   A few times
   Frequently

5. Have you ever used Lego bricks to understand a complex topic? (Circle one.)
   Yes
   No

Post-Workshop Survey Questions, version one

1. How much of the information presented about Git was new to you? (Circle one.)
   All of it Most of it About half of it Some of it None of it

2. What part of the instruction on Git helped you the most?
   (Please rank in order from 1 - more useful to 6 - less useful.)
   ____ Discussing Git with other participants
   ____ Hearing other participants' questions
   ____ Typing Git commands
   ____ Building with LEGO bricks
   ____ Hearing directions from instructor
   ____ Other: _____

3. Please label the three LEGO images with their corresponding Git commands.
   The relevant Git commands are: **git init**, **git add**, and **git commit.**

Image 1

Git command: _____

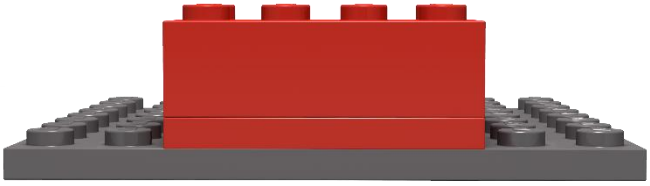


Image 2

Git command: _____

Image 3

Git command: _____

4. To what extent did using the Lego bricks encourage you to discuss Git commands and functions with other workshop participants? (Circle one.)
   Not at all Some Quite a lot

5. After doing this activity with Legos, do you find Git: (Circle one.)
   More confusing About the same Less confusing

Post-Workshop Survey Questions, version two

1. How much of the information presented about Git was new to you? (Circle one.)
   All of it Most of it About half of it Some of it None of it

2. What part of the instruction on Git helped you the most?
   (Please put an X beside statements that you found to be useful.)
   ____ Discussing Git with other participants
   ____ Hearing other participants' questions
   ____ Typing Git commands
   ____ Building with LEGO bricks
   ____ Hearing directions from instructor

3. Please label the three LEGO images with their corresponding Git commands.
   The relevant Git commands are: **git init**, **git add**, and **git commit.**

Image 1

Git command: _____

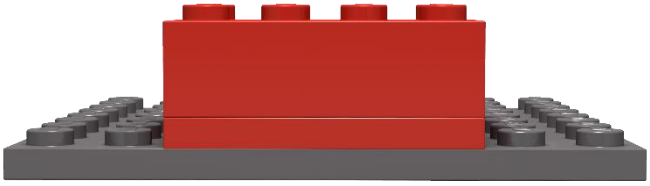


Image 2

Git command: _____

Image 3

Git command: _____

4. To what extent did using the Lego bricks encourage you to discuss Git commands and functions with other workshop participants? (Circle one.)
   Not at all Some Quite a lot

5. After doing this activity with Legos, do you find Git: (Circle one.)
   More confusing About the same Less confusing

Comments (optional):