**Tool Learning Package**

**Introduction**

GitHub is a web-based [Git](https://en.wikipedia.org/wiki/Git_(software)) repository [hosting service](https://en.wikipedia.org/wiki/Hosting_service). Git is a [version control system](https://en.wikipedia.org/wiki/Version_control_system) that is widely used for [software development](https://en.wikipedia.org/wiki/Software_development)and other version control tasks. It is a [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) system with an emphasis on speed data integrity and support for distributed, non-linear workflows. It offers all of the [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of [Git](https://en.wikipedia.org/wiki/Git_(software)) as well as adding its own features. Unlike Git, which is strictly a [command-line](https://en.wikipedia.org/wiki/Command-line) tool, GitHub provides a [Web-based graphical interface](https://en.wikipedia.org/wiki/Web_application) and desktop as well as mobile integration. It also provides [access control](https://en.wikipedia.org/wiki/Access_control) and several collaboration features such as [bug tracking](https://en.wikipedia.org/wiki/Bug_tracking_system), [feature requests](https://en.wikipedia.org/wiki/Software_feature), [task management](https://en.wikipedia.org/wiki/Task_management), and [wikis](https://en.wikipedia.org/wiki/Wiki) for every project.

**Goals**

The aim of GitHub is that it acts as a version control system. When developers are creating an application or a game they are continuously changing their code and releasing new versions of their program. Version control systems help to maintain modifications and store them in a central repository. Due to having this function developers can easily act as a team, as they can download edited versions of the software, make changes, and upload the revised version. Every developer can see these new changes, download them, and contribute to the project.

Not only developers but anyone who is associated with the project can collaborate with what is stored on GitHub. They can download and edit files as well since not all files that are related to the project will be the code files of the project.

Pull Requests is the foremost way in which collaboration on GitHub takes place. When a pull request is opened, it means that the user is proposing their changes and requesting that someone review the content and merge them into the repository. Pull requests show differences of the content. The changes, additions, and subtractions are shown in green and red.

As soon as you make a commit, you can open a pull request and start a discussion, even before the code is finished.

By using GitHub’s [@mention system](https://help.github.com/articles/about-writing-and-formatting-on-github/#text-formatting-toolbar) in the pull request message, the user can request for feedback from specific people or teams, regardless of where they are.

**Target Audience**

The target audience for this project is the Small indie game development start up team. Members of this team might include developers, designers, testers and salesman. Since GitHub is not just restricted to developers using the program, it is a good option for this team. They can also continue to use GitHub in future projects. Developers can use GitHub to store and collaborate on all the code files, and testers will be able to test them. Due to GitHub having a function of a pull request all files have to be overlooked by one or more members in order for it to be stored in the repository. So for example when a developer loads a file on to GitHub they will share it with a tester who will accept it in order to be stored in the repository. It also makes sure that no files get overlooked and get into the repository by mistake, as a result all material is reviewed.

The same thing works for all types of files stored in GitHub and is not restricted to just the files that contain the game code. Therefore all 4 users have access and can review all material and edit it accordingly. GitHubwas chosen as a tool because of its convenience and easy method. Since all material is stored in one place there is less chance of anything going missing and its collaborative feature suits this team particularly well.

**Learning resources**

In order to help the team quickly upskill and help them understand not only how to use GitHub but also familiarize with the concept of Scrum; a few activities have been compiled. These activities will utilize all 4 team members and are a fun way to get them all engaged and increase energy whilst learning at the same time (Energize Learning! Blog 2011).

### **Activity 1- Paper Folding**

1. Give everyone a sheet of paper.
2. Tell everyone to close their eyes and follow your instructions.
3. Start giving instructions about what to do with the piece of paper examples :

* Fold it in half.
* Fold the lower left corner over the upper right corner.
* Turn it 90 degrees to the left.
* Fold it again.
* Rip a half-circle in the middle of the right side, etc.

1. Once you have given quite a few instructions (more than 10 at least for a great success), tell everyone to open their eyes and unfold their piece of paper.
2. Even though they all received the same instructions and had the same starting material, pretty much everyone will have a different result.

Conclusions

* They don’t all start with the same base (some held their piece of paper vertically or horizontally) so they don’t all have the same results
* Some interpreted to rip a piece of paper as removing a big piece, some as a small piece
* Having eyes closed = not receiving feedback on their performance
* No communication so if they are doing it incorrectly they will continue to do so due to no feedback.
* Some instructions appear vague to some and clear to others.

### **Activity 2- Pass the Ball**

This is an exercise where the class participants had to pass a ball back and forth to one another. It started out that the group was completely blindfolded and could not speak, next they were still blindfolded but were allowed to speak to one another, and lastly all obstacles to communication were removed, and of course this allowed them to pass the ball freely to one another, even clear across the room with great success.  It serves as a great demo for why communication is important, and also gets people charged for a participative session (Energize Learning! Blog 2011).

Conclusions

* When obstacles to communication were removed, better results are achieved. Discuss this idea with the team using an example of their software development project. Would anything get done correctly if there was no communication among the team?

### **Activity 3- Beginners Tutorial video on how to use GitHub**

[**https://www.youtube.com/watch?v=73I5dRucCds**](https://www.youtube.com/watch?v=73I5dRucCds)

**Learning activity**

### **Activity 4- Build a Bridge**

In this activity, two teams must work together to build a bridge using materials that are supplied to them. They each build half of the bridge and then "connect" the two pieces to make a complete one, made up of two similar designs.

The activity is challenging because the room is divided, so no team is able to see how the other constructs its bridge. Teams have to communicate verbally through a sheet (tablecloth or bed spread) that divides the room, as they work.

At the end a marble will be run through the bridge to see it connects well. The marble has to make it way from one end to the other.

Conclusion

* The Build a Bridge activity strengthens communication skills because each team has to discuss ideas and brainstorm ways to use their materials to build half a bridge.
* This activity also improves group problem solving and creative thinking. Furthermore the team can use it to develop people's leadership skills if they decide to elect a team leader for each group, or if people naturally take the lead.

**Learning plan**

The team has 4 hours in total to upskill. The learning plan details what activities the team will undertake, when they will undertake them and the duration of each learning session. These learning activities will not only help the team learn how to use GitHub, but will also familiarize them with the concepts of scrum.

**Week 1 - Day 1 – 1 Hour**

1. Try out Activity 1 for 15 minutes
2. Discuss conclusions for 15 minutes
3. Try out activity again for 15 minutes- this time let the team discuss if they wish for 30 seconds.
4. Discuss conclusions for 15 minutes
5. Do the results differ due to team communication beforehand?

**Week 2 - Day 1 – 1 Hour**

1. Watch the tutorial video 35 mins
2. Practice what you learnt for rest of the hour, e.g download GitHub, create repository, etc. They are free to ask questions.

**Week 3 - Day 1– 1 Hour**

1. Try out Activity 2 for 15 minutes
2. Discuss results for 15 minutes
3. Get feedback from team about what they think about the upskilling taken place so far for 15-20 minutes. Reflect of the concepts of scrum they have learned so far
4. Do the activity again for the remainder of the hour.

**Week 4 - Day 1- 1 Hour**

1. Team has 30 minutes to refresh on GitHub skills and ask questions
2. Try out Activity 4 for 15 minutes
3. Discuss conclusions for 15 minutes.

**References**

Energize Learning! Blog. (2011, April 5) [Blog post]. Retrieved from http://blog.trainerswarehouse.com/communication-and-listening-exercises/

How to Get Started with GitHub - Beginner Tutorial (2014, March 13). Retrieved from https://www.youtube.com/watch?v=73I5dRucCds