Star Wars

Sprint 1
September 7th, 2023

Name	**Email Address**
Matthew Irizarry	matthew.irizarry745@topper.wku.edu
Zach Vance	zachary.vance141@topper.wku.edu
Keimon Bush	keimon.bush105@topper.wku.edu
Jeremiah Harris	jeremiah.harris978@topper.wku.edu

CS 360
Fall 2023
Project Organization Documentation
Galloway Games, Inc.

Contents

List of Figures

1 Project Team's Organizational Approach

1.1 Sprint 1

For this sprint, Matthew Irizarry, was the project manager. We decided to meet twice a week after class, for approximately 30-45 minutes each time. We ended up going to the Commons to meet in person, as there was always somewhere for us to sit. We generally discussed quick progress updates, and wins to celebrate. However, we did recognize a few areas for improvement around scheduling and organizing documents into one place, but a solution was quickly implemented. Overall, the sprint was very efficient and effective, and allowed us to complete our work fully and in a timely manner. We plan to have a Sprint 1 Retrospective to learn from what we did, and improve on our process in the next sprint.

1.2 Sprint 2

For this sprint, Matthew Irizarry, was the project manager. We decided to meet, once again, twice a week after class, for approximately 30-45 minutes each time. During these meetings, we discussed how we have implemented the solutions we discussed during our sprint 1 retrospective, and if they are working to improve the organization issues that we encountered. Our overall consensus was that yes, GitHub did indeed help us stay more organized, and we were happy with how we quickly got onboarded to GitHub. Next, we would discuss how we were doing on tasks, and if there was anyone that needed help from other team members to complete their task.

1.3 Sprint 3

For this sprint, Matt Irizarry was the project manager. We pretty much followed the same meetings, however we saw the need for more once development of the game was underway. For these meetings, we would meet via Discord whenever two or three of us was available. This allowed everyone to stay mostly up to date on what was going on, but was not hindering development by missing one or two group members on a meeting. Overall, the sprint was a good sprint, and we learned a lot (especially merging and rebasing stuff on GitHub)

1.4 Sprint 4

For this sprint, we were basically only meeting on Discord and some impromptu meetings after the class. Matt was responsible for being the project manager for this sprint. We didn't meet always as a full team, but rather as smaller pods to facilitate more intentional development. The only thing I think we could have improved as far as communication is doing a daily standup of sorts to see where roadblocks were, and if we could easily solve them. For example, our code review process was enforced before a team member was able to commit their code to the main branch, and therefore could be waiting asynchronously for a team member to login and review someone elses code. Had we had more daily standups, this would have been more helpful instead of more of a hinderence.

2 Schedule Organization

2.1 Gantt Chart v1:

A Gantt chart is a project management tool that provides a detailed visual representation of a project's timeline, tasks, and progress. The Gantt chart displays the tasks that were worked on and completed during the first sprint. Tasks completed in the first sprint include: class readings, planning, task distribution, presentation, documentation, and CATME evaluations. These tasks aided the team in the planning and organization of the remainder of the project and allowed for the team to familiarize themselves with each other and work together better in future sprints.

The GANTT chart for this sprint has been included in the submission archive that was uploaded to blackboard.

2.2 Gantt Chart v2:

For this sprint, we primarily focused on familiarizing ourselves with GitHub issues, commits and branching so we can hit the ground running in Sprint 3 once we start programming. However, the primary focus was our UML diagrams, the presentation, and the documentation for everything. Overall, we kept to a timely schedule, and everyone did really great work. The GANTT chart is included in the archive that is submitted to blackboard, as it was last time.

2.3 Gantt Chart v3:

For this sprint, our primary tasks were making sure that we implemented everything as closely as possible to the way it was described in the Technical documentation. There were a few things that changed. For example, we no longer are going to use some kind of Node.js server, but instead are relying on Django for our backend. This gives us some relief with regards to security, as Django already comes with a robust security suite in place. Once we started this, we were quickly on our way to developing an MVP for our client. Finally, we had to make sure that we finished our documentation and CATME eval tasks. Everyone was very diligent and thorough with their assigned tasks.

GANTT chart is included in the archive that is submitted to blackboard, the same as it was last time.

2.4 Final Gantt Chart:

In Sprint 4, we adopted a ”see a need, fill a need” approach, assigning tasks based on individual comfort and expertise. This facilitated efficient testing, with experienced members leading smaller code review sessions. Pair programming and smaller groups streamlined the process, while larger calls were reserved for high-level code reviews. Individual responsibilities were crucial; for instance, Jeremiah handled enemy implementation, and Matt oversaw auth and database connection. Collaboration evolved as we finalized the project, with an emphasis on smaller pods for effective progress. A comprehensive review and testing ensured a polished final product. Despite being our busiest sprint, it was the most impactful in terms of learning.

GANTT chart is included in the archive that is submitted to blackboard, the same as it was in previous sprints.

3 Progress Visibility

3.1 Sprint 1 Progress Visibility

For this sprint, we focused on establishing channels of communication, meeting times that worked for everyone, and just generally got to know each other. We successfully divided up the tasks for the technical documentation, the presentation, and the organizational documentation. After that, we got straight to work. There was never any point where we had to remind individuals to stay on task, but I did make sure to give reminders of what was upcoming. Everyone was able to use these quick reminders to stay on task and get their work done, which was really nice. We used our primary channel of communication, Discord, to handle any situations that we could over text, and whenever necessary would utilize Discord or an in person meeting to clarify things that couldn't be clarified over text. Both meetings with Galloway Games went well, and we were able to use these meetings to dive deeper where necessary, and pivot away from work that wasn't within the scope of this sprint, or the project whatsoever.

3.2 Sprint 2 Progress Visibility

For this sprint, we focused on GitHub being the source of progress tracking for this repository. For each line item on the rubric, it was divided up and organized under the epic ¡ story ¡ feature paradigm, and then team members self-assigned issues depending on what was due next. In addition to this, the issues had all the information necessary to complete, and thus it was very easy to start work on the sprint this time around. I, Matt, was responsible for making sure everyone had enough tasks where they didn't feel overwhelmed but were making significant contributions to the team as a whole.

When we were able to meet with Dr. Galloway, it was really easy for us to know where we stood and thus it was easy to tell him what we had accomplished so far. Overall we were really happy with the changes that we implemented this sprint, and I am sure we will discuss it more in our retrospective for this sprint.

3.3 Sprint 3 Progress Visibility

In Sprint 3 tasks were assigned primarily via volunteering and task lists are displayed on both GitHub and Discord. Team members have each created respective branches on GitHub which they upload their work to.The progress is periodically merged and checked for conflicts. The team regularly communicated via discord calls and text messages, particularly as deliverables were due. Pair programming has taken place on discord to help encourage collaboration and help teammates problem solve when issues arise during the development process. The Star Wars game has progressed well and the team has established a solid base for progress to be accomplished throughout Sprint 4 and finish strong. Progress has been shared with the client via weekly team-client meetings and a demo which was delivered via a presentation. The demo showcased our level design and player movement. Some enemy design has taken place, but did not make it into the demo. Additionally, a Django Database has been implemented and is functional, but does not yet interact with the game. The user interface has not yet been completed but the design has been planned. Overall, the team has made good progress and is collaborating well. Looking forward to sprint 4, the team has well defined goals regarding game design, performance, and documentation which will make it easier for the software to be successfully developed.

3.4 Sprint 4 Progress Visibility

In Sprint 4, we took a see a need fill a need approach. We pretty much let whoever was comfortable with a task perform it, and we feel like this worked pretty well for us. This was especially helpful when we needed to start doing various types of testing, as some group members were more experienced than others. This was a great opportunity for us to pair program and do code reviews as small one or two person calls, and allowed the testing process to feel very streamlined. Smaller groups allowed for an easier back and forth, but also we did larger calls when we were reviewing the project code from a higher level. For example, Jeremiah was the main group member responsible for implementing enemies, so it wouldn't make sense to try and code review enemies without him there. Matt was responsible for the implementation of auth and the database connection, so anytime we did code reviews of that, he would be the primary driver of the call. If we needed to work on a feature that was implemented with multiple features, it was beneficial to work in a larger group than as a smaller one. Overall, as we finalized the work on this project and implemented the last features, we found new and different ways to collaborate that would help us move along the progress quicker and with a higher quality by working in smaller pods. Finally, once we believed that all the development features of the game were done, we would review the game as a whole together and test it together to make sure that nothing was forgotten. Sprint 4 was one of our busiest sprints, but I feel like everyone learned the most here.

4 Software Process Model

For this project, we find it paramount that we be able to effectively plan in the long term, but also be able to make changes on the fly if requested by the client, or if it is demanded by a change in software or something out of our control. As such, we will be adopting the Agile organizational model. Comparing this to something like the waterfall approach, which is a linear software development model, agile allows a team to make iterative changes over several sprints. This allows for better pivoting when necessary, and encourages small iterative development over large swaths of changes all at once. Doing smaller, iterative changes like this allows bugs to be caught earlier, and squashed with less effort. This will have a significant impact in the long run, not just in the quality of the software, but especially in the amount of time that will be saved from this iterative approach.

5 Risk Management

5.1 Risk Identification

The understanding of when a risk occurs can be crucial to making progress as a team in developing software. This is because if you let a risk go unnoticed it can cause other problems to occur beyond the initial risk and something that was once small is now a burden to the entire team and all resources and time are now committed to solving these other problems instead of progressing in development. The key solution to identifying risks is tight communication between members whether it is through code review, communication schedules, or setting up meetings. More ways in identifying risks is to not look over the small things that appear. If something small happens it is best to write it down for later discussion as issues most of the time start small but grow in size. As well as identifying how bad an unexpected event will be on the project. If we can identify the risk an issue poses on the project we can more easily combat said issue.

5.2 Risk Planning

Risk planning is the process of looking in the future and examining possible risks because if you can identify a risk before it comes you can plan your workflow around said risk to minimize any conflict. You can separate these risks into 3 categories: Known Knowns, Known Unknowns, and Unknown Unknowns. The first 2 options are more feasible to plan around as the team knows to some degree what and/or when to expect the risks. However, the most dangerous is Unknown Unknown as there is no telling when it will hit your team or what the exact issue will be. For unknown unknowns these are ones we take as we see them. As they can not be planned for and generally happen when least expected we can perform damage control. This will happen by a discussion of the situation and how to tackle said issue. Depending on what the group feels is the best course for action will depend on how these are handled. For the known knowns we can easily plan for these and have predetermined steps to fix said issues when they arise as these are expected issues. For the known unknowns we can't have a set plan for these as even though we know of them we do not know if they will arise and when. For this though we have more basic plans of being ahead. So if someone unexpectedly becomes sick we have some plans that we discuss with the group quickly and take the best course of action.

5.3 Risk Monitoring

Monitoring the risk is quite simple with the software architecture that the team will implement. GitHub has an 'Issues' tab that we can use to keep track of any problems software-wise that can happen where teammates can comment on the issue and work together to resolve the risk. Team dynamics risks such as the possibility of losing a member, scheduling issues, or disagreement can be solved with meetings and double-checking that work is split fairly amongst members so that tasks can be done as efficiently as possible. As well as having a check up where we ask all members how they feel about current situations. We do these checks at most every meeting as it is a great way to get an idea of how everyone is feeling at the current time. If someone says something that they believe to be a risk it is also taken into account and depending on the situation we will have an immediate discussion or a discussion for later. This can help with us solving said issues that appear and seeing if a risk is actually a real risk or a momentary issue.