Final Report

Background pattern

Description automatically generated with medium confidence

That One Company Inc.

Jonathan Mason, Michael Merritt, Omar Rosario, William Siri

12/14/2021

Overview

Project Team Formation

Team Members:

* + Michael Merritt
    - Role: Scrum Master
    - Strengths: Leadership
      * Michael was the scrum master throughout the whole project. He was excellent at making sure that we all got our work done in a timely manner.
    - Weaknesses: Rarely asks for help
      * If Michael ever had any issues with the project, he would rarely ask for help from any of the other team members.
  + Omar Rosario
    - Role: Programmer
    - Strengths: Presentation
      * Omar excelled at presenting the work that he had done over the course of a scrum sprint, especially in terms of the slides that he would make for them.
    - Weaknesses: Consistency
      * While the slides he made for the presentations were amazing, they were rarely in line with the rest of the presentation.
  + Jonathan Mason
    - Role: Programmer
    - Strengths: Humility
      * If a feature of his was not working the way he wanted to work, he would not merge the feature into the master branch.
    - Weaknesses: Time management
      * Jonathan would sometimes not have his features ready until the last minute, leaving little time to fix any bugs with his features.
  + William Siri
    - Role: Programmer
    - Strengths: Coding
      * William was extremely adept at writing code in Java.
    - Weaknesses: Communication
      * William would sometimes not attend/ be late to meetings that the team would schedule.

Bug and Enhancement Prioritization

The way we prioritized bugs and enhancements was based on user data that we gathered before beginning work on the project. We gathered data from both experienced and inexperienced gamers in order to get a wide range of opinions. After this, we decided to prioritize 3 main types of improvements: bugs that were prevalent among both groups, improving the experience for inexperienced users, and improvements that were helpful to any kind of user.

Project Management Plan

* Strategy: We used the scrum method of project management. This meant that we would have two-week periods where we would each be given a task to complete. At the end of this two-week period, we would give a presentation of all of the changes that we made to the project.
* Tools: We used the following tools for this project:
  + Discord
  + Zoom
  + Git
  + OneDrive
  + Google Docs

Testing Plan

A unit test was written for each bug and enhancement. Each team member would fill out a unit test for another team member's bug or enhancement to confirm that it worked on other members' computers. If the test case passed, we would merge the feature into the master branch of the Git repository, as it is at the point where we can consider it finished. If it failed the test case, we would continue to develop the feature and revise the code so that it can pass and be merged in the next meeting.

Process

Scrum Management Process and Schedule

For the development process, we ended up using a scrum model of development in which we divided the year into five two-week sprints. For each sprint, we had an intended goal in mind, and chose the work units and backlog items to work on in accordance with that initial goal. As these backlog items were completed, we updated it to reflect all of the work we had completed and all the work we had yet to complete. The goal for each scrum sprint was as follows:

* **Scrum Sprint 1:** Implementation of high-priority improvements for novice users
* **Scrum Sprint 2:** Implementation of high-priority improvements for experienced users
* **Scrum Sprint 3:** Implementation of medium- and low-priority improvements for all users
* **Scrum Sprint 4:** Creation of brand-new game content

To stay organized and consistent throughout the scrum sprints, we followed a general schedule for meetings, deadlines, and scrum stand-ups. This schedule largely dictated how we utilized our in-class meetings on each Tuesday and Thursday of the week, but we would occasionally plan other meetings outside of class, such as a Tuesday evening merge meeting or a Wednesday presentation meeting. The general schedule that we stuck to each week was as follows:

* **Tuesday, Week 1 - Planning Day**
  + During the first Tuesday of the sprint, we held a meeting to decide on the goal for the sprint as well as the work units we intended to work on. After discussing that as a team, we created a work plan for that sprint and filled out the notable system improvement form.
* **Thursday, Week 1 - Initial Check-in**
  + During the first Thursday of the sprint, the scrum master led the first scrum stand-up of the sprint in which each team member would discuss their progress, show off any implemented code or features, discuss future plans for the project over the weekend, and any obstacles they may have run into.
* **Tuesday, Week 2 - Merge Meeting**
  + During the second Tuesday of the sprint, the scrum master led the second scrum stand-up of the sprint in which each team member showed off their completed code features, discussed any further work we had to do for the feature, and mentioned any obstacles that may have prevented completion. If the features were complete after that stand-up, we would run the corresponding test cases and merge the features into the master branch.
* **Thursday, Week 2 – Presentation Day**
  + During the second Thursday of the sprint, we presented our improvements and had an in-class demo of our project to director of software development and the other teams.

Collaboration Tools

To assist with management and collaboration, we used a variety of tools and programs to help us work together, including Git, GitHub, Google Docs, Google Slides, Zoom, and Discord. A description of how and why we used each tool is listed below.

* **Git/GitHub:** We used Git and GitHub to manage the code repository in a way that allowed all team members to work on the project asynchronously at the same time. GitHub’s features for merging and easily resolving merge conflicts were especially invaluable, as we were able to quickly resolve the conflicts with nothing more than a simple look at the code and a quick conversation between the two programmers involved.
* **Google Docs/Google Slides:** We used Google Docs and Slides to work on the presentations in real time. By hosting the documents in Google Drive and sharing them will all of the team members, we were able to track work on the presentations and papers in real time and hold a virtual meeting where all of the team members could look at what others were doing and give live feedback before simple mistakes became larger and harder to fix.
* **Zoom:** We used Zoom to meet up for in-class meetings as well as record presentations. We chose to use Zoom for these recordings because it cloud-saves them in a convenient way, allowing us to access the recording using a URL instead of having to work with the massive 10-minute video file.
* **Discord:** We used Discord for communication and meetings outside of class. We chose to use Discord because it was an application everyone was already accustomed to using, so we would not have to learn much about setting up and working with it. We also used Discord for its text and voice chat capabilities, as we used voice chat for meetings and text chat for reminders, questions, and discussion.

Project Work Plan

For the work plan, we tried to split the backlog items down into categories which we could then tackle separately each sprint. For each sprint, we took the backlog items we intended to complete and broke them down thoroughly into work units on the sprint’s workplan so we could track and record our progress. For each work unit, we included the following details:

* **Corresponding backlog item:** We recorded the corresponding backlog item for each workplan task to help us organize topics as well as easily be able to see when all of the parts of an improvement were finished.
* Team members responsible: We included the number and names of the team members working on each work unit of the workplan so that it would be easy for team members to find what tasks they were expected to be working on throughout the sprint without needing to interrupt someone else to ask.
* Brief task description: We included a simple description of what needed to be done for every work unit, as it gave the team a chance to decide on what specifically that work unit should entail.
* Task start and end dates: We added the task start and end dates to the workplan as a way of making deadlines for accomplishing each of the work units. These dates can also help give an idea of how long a specific work unit was expected to take.
* Date of task completion: We also added the date on which the task was finished on, as this gave more insight into if tasks were completed early (meaning they were easier than expected or just dealt with earlier), on time, or late (due to obstacles usually described elsewhere).
* Estimated and actual time to complete: We included the time we expected to take and the time it actually took for each work unit as a way to track and compare how much of a contribution each team member was making each week, as well as a way to make sure that no one was getting stuck and spending too much time on a task that someone else could help with.
* Success criteria: We came up with a specific and actionable task to demonstrate if a work unit was truly completed. If this action was demonstrably completed, the entire work unit could be marked as finished. These included tasks such as presenting ideas to teams, passing test cases, and submitting files to the scrum master.
* Obstacles encountered: Finally, each work unit contained an optional field to describe any potential obstacles encountered as a way of explaining any issues you ran into with a certain task. These were useful as we could avoid making the same mistake twice, and also had an easy way of knowing which team member to ask if you ran into a similar issue.

Results

Final Version

Our goal as a development team was to eliminate as many game-breaking bugs as possible. The issue with the level collisions and level 5 looping (which was our final level in scrum sprint 1) were bugs that greatly deterred the player’s experience. If the player were to miss the level completion box in the attempt to explore the map, the player would either respawn at the beginning of the level or fall under the level’s constructs. Eliminating this bug allows the player to safely navigate the map without getting frustrated or exiting the application entirely. Level 5 looping did not allow the player to return to the main menu after finishing the level, when a game typically allows the player to do so after completing the final level. The code was amended so that the player returns to the main menu while adding a notification screen to let the player know that they have completed the game. After adding additional levels, the team made sure to transfer the code so that the new final level (Level 9) executes in the same manner.

Apart from adding variable jump functionality and additional levels, all other enhancements are considered requirements for this type of game. This does not discredit the team’s hard work or deems the enhancements as insignificant, but to explicitly say these changes are features that should have been implemented from the game’s inception. Although not every bug was addressed, the team believes, with the changes made, that this project is in a satisfactory state to receive features that expand its affordances and constraints in a way that make it feel more Super-Mario-World-esque.

Contributions Per Team member

|  |  |  |  |
| --- | --- | --- | --- |
|  | Role |  | Work Hours |
| Michael Merritt | Scrum Master | 1. Unlooped Level 5 2. Restructured Level 3 3. Improved Completion Feedback 4. Adjusted Attack Spawn Location 5. Created Level 9 6. Improved Old Levels 7. Implemented Sound Effects | 80+ |
| Jonathan Mason | Programmer | 1. Improved Instructions Screen 2. Enabled Attacking While Jumping 3. Enabled the Flipping of the Cat Sprite While Jumping 4. Fixed Attack/Walk Animation 5. Created Level 8 6. Implemented New Music | 46+ |
| Omar Rosario | Programmer | 1. Implemented Level Collisions 2. Implemented Mute Feature 3. Fixed Double-Bound Key 4. Relocated Narrative Screen 5. Created Level 6 6. Made Minor Changes to Previous Enhancements | 47.5+ |
| William Siri | Programmer | 1. Improved Menu Navigation 2. Implemented Variable Jump Height 3. Enabled Attacking While Jumping 4. Enabled Single Shot Attack 5. Created Level 7 6. Modified Level Selection Screen | 42.5+ |

Fixed Bugs

|  |  |  |
| --- | --- | --- |
|  | Detail | Scrum Sprint |
| Level Collisions | The player was able to walk off the level through the right boundary resulting in the player spawning at the beginning of the level or falling under the level’s constructs. | 1 |
| Looping Level 5 | Clearing level 5 puts the player back at the beginning of level 5, leading to them getting stuck until they lose and quit or close the application. | 1 |
| Mid-air Sprite Flip | When the player is in the air, the cat sprite does not change orientation when the player moves in the opposite direction. | 3 |
| Walk/Attack Animation | If the player presses the attack key while walking, the animation of the cat sprite will remain in the walking state instead switching to the standing state. | 3 |

Completed Enhancements

|  |  |  |
| --- | --- | --- |
|  | Detail | Scrum Sprint |
| Instruction Screen | Instruction screen was updated to explain the correct controls and changed the color of the text for visibility and legibility. | 1 |
| Menu Navigation | The player is able to navigate the menu in all directions as opposed to only up and down. | 1 |
| Mute Feature | Allows the player to immediately mute the in-game sound without muting their system’s volume, potentially allowing players to use their own music or enjoy the game in silence. | 2 |
| Restructure Level 3 | Restructured Level 3 in order for the player to complete it. Was previously unbeatable due to the design of the level. | 2 |
| Single Shot Attack Mechanic | Changed the attack animation to emit a single shot. | 3 |
| Variable Jump | The player was limited to a singular jump height, despite there being smaller obstacles throughout each level. | 3 |
| Relocate Narrative Screen | Moved the Narrative Screen to appear before the player starts Level 1. | 3 |
| Level Completion Feedback | Changed the animation and sprite of the level completion box and added sound effects. | 3 |
| Attack Spawn Location | Lowered the spawn location of the attack projectile to hit smaller enemies. | 3 |
| Increase Game Duration | Added five new levels. | 4 |
| Improve and Restructure Old Levels | Redesigned old levels for variation. | 4 |
| New Sound Effects | Added sound effects for jumping, attacking, enemy deaths. | 4 |
| Level Selection Options | Added selection options for new levels. | 4 |

Remaining Improvements

|  |  |  |
| --- | --- | --- |
| Name | Type | Detail |
| Cat Color Selection | Bug | When choosing a cat color from the options screen, the game immediately starts Level 1. |
| Double Jump | Bug | Implementing a variable jump inadvertently enables the player to perform two small jumps. |
| Player Lives | Enhancement | Add lives to add challenges |
| Timer | Enhancement | Add a timer for competitiveness. |
| Scoreboard | Enhancement | Add a scoreboard for competitiveness. |
| Difficulty Setting | Enhancement | Add difficulties for the expert players. |
| Power-Ups | Enhancement | Implement power-ups for variability in gameplay. |

Repository Link

The project repository can be found here: <https://github.com/mikepmerritt/SER225-TeamA3>