Module	SEPR
Year	2019/20
Assessment	3
Team	Early Bird
Members	James Little, Ryan Vint, Adam Lynch, Georgina Martin, Kheng Yeoh, Tanay Malde
Deliverable	Change Report

Section A

Change Management

At the start of this Assessment, the group took some time to understand and summarise the previous team's approach to creating the product. Each team member examined the project so that we all had a thorough understanding of the way they had structured it, any problems it might have, and an idea of how we wanted to continue with development. In the first meeting after selecting the project, once everyone had had the chance to evaluate it, we talked as a group about what we needed to implement and where we thought there were any issues with the previous team's approach or implementation. After identifying the changes we had to make and features to add, these were added to the project's Trello board. Tasks were then assigned as a group since we all had a good understanding of where each others' strengths lie.

The previous team had been using the Scrum framework for the Agile development approach [1]. Our group is familiar with the principles of Agile, however we felt that the Kanban framework would be more suitable for this assessment as opposed to Scrum. With only a slight adjustment period, we were able to continue with the project, assuming the roles within Kanban that we are comfortable with and most suited to. We used the tool Trello, which is very useful for implementing Kanban, to precisely track the changes that were being made, when, and by whom. Trello tasks can be assigned to specific users meaning the whole team could see who was working on what at any one time so that potentially conflicting changes weren't made to the project at the same time.

All completed work and changes were logged in Trello and GitHub was used to share the work that had been implemented. While the IEEE Standard for Configuration Management in Systems and Software Engineering [2] provided useful guidance for how to manage changes within a project such as this, we found that the in depth documentation was at odds with part of the Agile manifesto "working software over comprehensive documentation" [3]. For this reason, we decided not to use the IEEE Standard as it would result in a significant documentation overhead, choosing to rely instead upon the methods previously mentioned.

Section B.i

Justification of Changes

Testing

The largest changes to our testing is that we have not used JUnit to create unit tests as much of the newly implemented functionality is dependent on things only accessible at runtime, such as textures and user inputs for the minigame, or the ability to access the node map for the patrols AI pathing. Most of the complex logic to do with things such as range and attacking was fully tested in the previous assessment and was merely extended for this assessment. This means that we have manually tested a lot of the functionality implemented in this assessment.

We have extended our testing to include a lot more manual testing. The last assessment testing focussed mostly on UI Tests - testing for the user interface reacting and working as the user expects, such as testing that all the buttons do what they should, for example testing that the sound mute button works. For assessment 3, we have done more general manual tests, such as end to end tests, which test more of the main functionality of the game, since there were few UI elements added to the game, aside from the minigame. End to end testing is more relevant at this stage than previously as the whole game is now fully functional, unlike in assessment 2. This means our testing is more comprehensive and exploratory as we are not limited to currently-implemented functionality - all aspects of the program should function fully and as the requirements (user, functional and non functional) demand.

All of the members of our team have performed exploratory testing. The goal of this testing is to attempt to break the program, we run and play the game, purposefully performing actions we believe the game may bug or fail under, such as pressing incorrect buttons, attempting to move patrols instead of fire trucks etc. We will have recorded the statement coverage of all of our attempts and if it is over 80% for all of our attempts and we have had no crashes or issues then we will consider this testing a success.

Statement coverage is not the only method we have used to decide that our code is fully tested, as even if all code is covered, if we have not tested against the requirements and we do not have a full traceability matrix, our code may still not be complete and working.

We have tested the functional requirements that were not already fully tested in assessment 2 to make sure that the program now completes all of the original requirements set in assessment 1 that were not necessary for assessment 2, such as the minigame based requirements.

Testing Results

Our end to end tests, which can be found at https://gm-martin.github.io/assessment3/EndToEndTests.pdf, have been designed to specifically test all of the functional requirements not tested in the previous assessment. All of these functional requirements and the tests that test them can be found in our traceability matrix https://gm-martin.github.io/assessment3/TraceabilityMatrix.pdf. As can be seen from this we have designed end to end tests for all functional requirements and all of these tests have been passed.

We have also completed a series of UI tests to test that the UI for the minigame functioned (https://gm-martin.github.io/assessment3/UITests.pdf) that were also successfully passed. And all members of the team completed exploratory testing, highlighting a few issues that were then solved and retested, details of which can be found at (https://gm-martin.github.io/assessment3/ExploratoryTesting.pdf)

Section B.ii

Methods and plans

We have continued to use the Agile development approach as it is well suited to our team's style of working, however we have chosen to switch from Scrum to the Kanban framework. This is because, while the specific roles assigned in Scrum are useful, the shorter time scale of assessment 3 compared to 1 or 2 will require all team members to be more flexible with the roles they take on so that problems can be dealt with quickly and efficiently when they arise. In addition, Kanban allows for a continuous flow of work so when a problem or bug is found, the team member can immediately start work on it as opposed to waiting for the end of the sprint, which is key given the shorter timeline for this part of the project.

Given this change in methodology, we have changed some of the tools we have used accordingly. Trello has been used to implement the Kanban methodology as it has boards, cards and lists that represent the cards in Kanban very well. It allows team members to assign themselves or other tasks that must be completed, and means everyone can stay up to date on the progression of the entire project. It has replaced the role GitKrakenGlo previously played as it more suited to the Kanban framework. Discord has been used primarily for holding meetings, as well as communication, as it is a reliable way to conduct video or voice calls at times when some or all members cannot be present at a physical meeting. We have now been using WhatsApp as opposed to Facebook Messenger as our primary form of communication as this method is more practical for some team members and has many of the same features. We have also changed our testing methodology so now no longer use the unit testing tools from the previous assessment (Mockito or JUnit) for our testing.

Unlike the previous assessment, much of which took place over the Christmas holiday, we are able to meet together more frequently, facilitating better collaboration between the entire group at once and allowing for easier tracking of what and how much work each team member is doing. Outside of meetings in person, we have held regular Discord meetings with sub-groups or the entire team to make sure all members stay up to date on the work that is being done, and to enable easier collaboration between team members working on overlapping parts of the project.

Given we have finished two assessments, the group has a strong idea of each team member's strengths and so we have been able to assign roles accordingly: James, Ryan, Tanay worked primarily on implementation, Georgina and Adam worked on documentation, Kheng has worked on asset creation, and Adam has worked on testing. These roles are guidelines for what each team member will work on, however they are flexible and it is expected that everyone will contribute to parts of the project outside of their assigned role. Ryan has also taken a leadership role, which includes uploading and assigning tasks on Trello at the start of assessment 3, however all group members will be adding tasks/subtasks when needed as the assessment continues.

We have also used Gantt chart made during assessment 2 to guide the schedule of this assessment so we can be sure all components will be completed in time for the deadline. In addition, we are creating a new Gantt chart using PlantUML [4] (a tool for the creation of UML and some non-UML diagrams) to schedule the tasks that must be completed for assessment 4 [5].

References

[1] Assessment 2 - Method and Planning [Online] Available:]https://gm-martin.github.io/assessment2/#method-planning [Accessed 16 Feb. 2020]
[2] IEEE Standard for Configuration Management in Systems and Software Engineering. [Online] Available: https://ieeexplore.ieee.org/servlet/opac?punumber=6170933 [Accessed 16 Feb. 2020]
[3] Manifesto for Agile Software Development [Online] Available: https://agilemanifesto.org/ [Accessed 16 Feb. 2020]
[4] PlantUML [Online] Available: https://plantuml.com/ [Accessed 16 Feb. 2020]
[5] Gantt Chart for Assessment 4 [Online] Available: https://gm-martin.github.io/assessment3/Assessment4GanttChart.pdf [Accessed 16 Feb. 2020]