UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Change Report
Cohort 2 - Group 16 (Skloch)

Group Members:

Charlotte MacDonald
Hollie Shackley
Luis Benito
Kaustav Das
Sam Hartley
Owen Gilmore

Summary of Changes

Most planning of changes to deliverables and code was done through informal team discussions both during meetings in practicals and over Discord. Changes that would impact more than one deliverable (for example, class changes that had large impacts on the code and the architecture deliverable) were discussed in depth before any work to implement them whereas changes that were less impactful, such as adding missing explanation to the planning document, were discussed less. After taking a look at each deliverable, the person in charge of it created a list of things that needed changing or adding and these were discussed amongst the team before the team member working on the deliverable began working their way through the lists.

Group 19 had been using Google Docs and Github previously so this was continued. By creating copies of their documents and cloning their repository, we could track the changes we made through version control. Changes were therefore made through these platforms so we could always review what was different using the change and commit histories. As we had created lists of what changes we wanted to make before starting, these were also used to keep track of what we were changing.

Some aspects of the changes were reviewed regularly. For example, the added plan for this assessment was reviewed with the team each week to determine progress so far and change the plan if necessary. Changes to the implementation and to the architecture document were regularly reviewed with the implementation team looking at architecture and the architecture team looking at implementation to ensure that the two deliverables were accurate to each other. The risk assessment continued to be reviewed each week.

Other parts of the changed deliverables that didn’t need reviewing during other processes were reviewed at the end of the project with each team member selecting at least one of the changed deliverables that they hadn’t worked on to review. This allowed us to find errors and ensure that deliverables were consistent with each other.

Due to the previous team’s website not being html based, we recreated it in the same theme. This makes it easier for us to host files and deliverables directly on the website, providing a bit more structure.

Architecture

Original Document: https://samh366.github.io/16-2-website/assets/deliverables/group1/Arch1.pdf
Updated Document: https://samh366.github.io/16-2-website/assets/deliverables/group2/Arch2.pdf

Although it was mentioned what tools Group 19 had used for their architectural diagrams, I added a statement about why those particular tools suited the project and the benefits of using them. They had also added the diagrams to the document without using PlantUML Gizmo [REPLACE] or providing the code so I went through and recreated them all with this tool so they will be easier to change in the future. On the website, there was no evidence of RDD, initial designs or interim versions as they did not document the initial process. To remedy this, I will put their current designs on the website as interim designs when I update them. When relating to the requirements, they did not clearly explain which classes meet the requirements, so I will expand the discussion on how the architecture fulfils user and functional requirements. The easiest way to ensure complete coverage of the requirements is by adding a table which I have done. The requirements will also have changed so I will update them to the new requirements as new features have been added. I will also leave in the discussion about what architectural design to go for but will add a statement about what Group 19 ended up going with.

The general Renderer and Trigger classes were removed and the game was restructured to function without these classes as this better divides up the responsibilities of classes. We decided it would be easier to call the already separate classes' render methods in the PlayScreen render method. The PlayScreen class is now responsible for checking for inputs, allowing the player to move and ensuring they remain inside the map and outside of objects. It also calls the map, player and HUD rendering classes to render the game as well as being in charge of telling the game state to pass time and notifying other classes of the game being over.

The Component diagram included the Renderer which now makes it obsolete as that has been removed. The diagram also used incorrect notation to show which components used which services and didn't seem to be consistent with how the project works. It also seemed unnecessary to nest InputHandler inside an Input component.

The input-update-render loop is now obsolete due to the removal of the Render class. Now, keyboard inputs are translated into actions using a map which maps the key pressed to an Action. Then this action will be added to pressed and held actions. The PlayScreen is then responsible for checking for any Actions in pressed and held actions. There is an Action class which contains all the possible actions that can be taken in the game. Methods are then called by the PlayScreen from other classes. For example, the Player class has a move() method which is called when a moving related Action is detected. As this is a Player movement, the CharacterRenderer can be used to render this change onto the screen.

The UML diagram of interaction between the MapManager, Trigger and GameState and the sequence diagram of it have both also become obsolete due to the removal of the Trigger class. It wasn't very clear from the architecture what the Trigger was for so we moved most of its functionality to other classes where it makes more sense to be. For instance, the Activity management was moved into the PlayScreen class. Before, the State seemed to be responsible for fetching information from the Trigger and updating itself if necessary. Now, the PlayScreen will tell the State when to update the time, day or anything else it stores.

The state diagram included the activities ReadInput, UpdateGameState and Render which shouldn't be included as they aren't states so I removed them. With the new requirements it was also necessary to add some more screens so I updated the diagram with the new screens and how to get between them.

Group 19 were missing a class diagram describing the game logic so we created one. However, it was quite big so it was broken down to show how different parts of the system work together. As the PlayScreen provides a lot of the functionality, one of the diagrams focussed on that class and the ones it interacted with, as well as how those classes interact with each other. Another was about the GameOverScreen and LeaderboardScreen, as these were the only other screens that interacted with unique classes. All screens interacted with HeslingtonHustleGame and SoundController so the final class diagram showed all the classes and how they interact with these.

An Achievement class had to be created to fulfil the updated product brief. This stores all the data about the achievements and streaks the user can complete and tracks their progress towards them. An NPC class was created to better differentiate between the Player and NPCs, as both use the same renderer (CharacterRenderer). An AvatarSelectScreen, CreditScreen, GameOverScreen, LeaderboardScreen, OptionsScreen, PauseMenu and TutorialScreen were all created due to changing requirements and separating functionality better. Before, there was no separate class for the game over screen, there was just a pop up in the PlayScreen. Having a separate screen is much clearer.

Method Selection and Planning

Original Document: https://samh366.github.io/16-2-website/assets/deliverables/group1/Plan1.pdf

Updated Document: https://samh366.github.io/16-2-website/assets/deliverables/group2/Plan2.pdf

Most of the changes made to section a) of the method selection and planning document were around justifying the continuation of the tools used by team 19. There were also some tools that team 19 used that were missing from this write-up (PlantUML, Google Drive, Github and IntelliJ) so the justification for choosing these originally and continuing with their use was added. Write-up around Kanban boards was removed as we hadn't used Kanban boards in assessment 1 and felt that our team organisation worked well without them. It was felt that it was better to not disrupt the way we had been working and to continue with current practices as much as possible as we didn't want to waste time getting used to new tools or methods.

The original document was also lacking in discussion of software engineering methods. We decided to continue with the same methods as assessment 1 for the same reason as sticking to the same tools. A discussion of agile and spiral approaches was added.

In the original section b), team 19 had discussed how they used scrum to organise their team. We had also taken inspiration from scrum but had used it differently within our meetings so this was changed. The original document also referred to the project as "large". Due to the small time frame, small team size and small scope of the game, we did not agree with this viewpoint and so removed these instances.

Further discussion about our meetings during practicals was added as this was our main face-to-face communication and discussion of extra regular meetings was removed as we did not have these meetings. Team 19 had not included a breakdown of what marks and deliverables were assigned to each team member. Our assessment 2 breakdown of this was included along with discussion of how the principle of equitable work allocation was used to achieve this. Part b) was also missing discussion of leadership and risk management roles, upper management and stakeholder communication and decision making and conflict management so these were added.

Team 19 had discussed a review system where each piece of code and each write-up was reviewed by another team member and then discussed. Although we had some informal reviews and all work was looked over by at least one other team member at some point, we didn't have a formal system like this and it was felt that starting this would take too much time from the small timeframe so this was removed. The conclusion from part b) was also removed as this was just repetition of points that had already been made.

The main change to part c) was the move away from Kanban boards as this had not been a tool we had used previously and it was felt that our current workflow was working well and shouldn't be disrupted. The old Gantt charts were removed from the deliverable as these would be better placed on the website. Team 19 hadn't included a work breakdown structure or tasks and deliverables tables so these were added and completed for assessment 2. The plan evolution was extended to discuss the changes in our plan throughout assessment 2 and our assessment 2 Gantt charts were added to the website.

A references section was also added.

Risk assessment and mitigation
Original Document: https://samh366.github.io/16-2-website/assets/deliverables/group1/Risk1.pdf
Updated Document: https://samh366.github.io/16-2-website/assets/deliverables/group2/Risk2.pdf

The Risk changes being made to Group 19's project are major as the entire assessment down to the formatting was changed, as their risks and mitigation strategies are catered towards their team. We omitted most of their risk assessment and replaced it with our usual 17 major risks that are more likely to happen to our team, judging by our group dynamic. We did incorporate some of the aspects and mitigation strategies mentioned by Group 19 in their assessment and risks related to the code and it's quality/functionality (i.e risk ID: 18) but even for the risks incorporated from Group 19's assessment we had to adjust the Impact Level, Probability level and Priority level to that of the competency of our team.

The reason we decided to change the formatting was because Agile's risk assessment method was easier to navigate not just for our team members, but also the general workforce as many of the industries rely on those methodologies.

For example, our numerical ID, instead of their name based ID is way more intuitive as it can be stored and referenced more easily. For example, if multiple variations of a similar subcategory of risks are needed to be added, a serial number would be cleaner and way more intuitive than relying on finding a unique identifier for each new unique variant of broad risks. For example, we had 3 different variations of "Communication failure" and 4 variations of "Team member shortage (temporary/permanent)" each with their unique causes and mitigation strategies. On a big enough time frame it would be infinitely scalable to include as many risks as possible to maximise risk mitigation. Though changes were made using the feedbacks for both our previous project and Group 19's project, for Risk 16 we plan on using continuous integration and also defining clearly an environment for testing the game. And in addition to that, to change the mitigation strategy of Risk 17 by merging more frequently and not letting branches go on for too long on their own.

We also changed the owner column of the register to mention the managing body's role/title (i.e Project Manager, Product Manager, Team leader etc) instead of the individual team member's name directly in the table to make it easier for the client to recognise the responsibilities are not arbitrarily given out and follows a strict structure that relies on the competency of the individuals taking that role/title. We added a separate table to mention who the said managing bodies are (the individual behind the risk manager role/Team leader role), but in case a team member permanently drops out, their role can be given to another member and updated easily in the separate table instead of editing numerous rows of the main risk register.

The reassessment dates for each risk is also mentioned with three major categories (weekly, biweekly and monthly) with some exceptions as risks can come in various different level of impact, priority level and nature, and to standardise a reassessment date for all the risks would be unresourceful and a waste of time and effort and **not** having regular checkups/reassessment for the risks would be a liability. For example:

- Risks such as Risk ID: 9 (2+ team members permanently drop out) would **not** need regular weekly checkups, since we would be aware when it happens without thorough examination, due to us being a small team of 6 members. Though, this may not be the case in a larger company with thousands of employees. *(As you can see, our risk assessment is catered to our group)*

- Risks such as Risk ID: 2 (Scope Creep) cannot remain neglected after a single checkup, as it can be a recurring problem if left unregulated, since despite having a low priority level, it has a high impact level.

Requirements

Original Document: https://samh366.github.io/16-2-website/assets/deliverables/group1/Req1.pdf

Updated Document: https://samh366.github.io/16-2-website/assets/deliverables/group2/Req2.pdf

1. Choosing more suitable ID names

We first checked that all IDs had appropriate names, which some did not. For example UR_DUCK and UR_BOOK are quite vague names, and could be interpreted in numerous ways, e.g. UR_DUCK could imply the player's in-game character can duck down, UR_BOOK could imply the player can book an event of sorts - when they're actual description is for the player to feed the ducks and read a book. So these ID's were updated to UR_FEED_DUCKS and UR_READ_BOOK (and UR_SPORT became UR_PLAY_SPORT) to more accurately reflect their descriptions. Furthermore their descriptions were updated to specify which in-game objects must be interacted with in order to complete the given activity, thereby making it clear to the development team which activities belong to which object.

NFR_START_CONDITION and NFR_SCOR_CALC were incorrectly named with the NFR prefix, when they were located in the FR table and are actually FR, so they were appropriately renamed to FR_START_CONDITION and FR_SCOR_CALC.

2. Establishing distinct separation of UR and FR.

We noticed that some of the UR requirements were too specific and not high level enough. For example the user requirements for the activities were: UR_BOOK, UR_SPORT, UR_DUCK, UR_STUDY, UR_SLEEP, giving a requirement for each of the specific activities the player can do. We decided to change this to a single UR requirement of UR_INTERACT and then have FR requirements for each of the specific interactions/activities (the word interact was chosen over activity as the player doesn't actually carry out the activity but rather interacts with an object to confirm that they took part in a given activity). By doing so we have separated our high level UR requirements from our specific functional requirements.

Another example of a lack of separation between FR and UR was the requirement was UR_PAUSE. UR_PAUSE: "The system will allow the user to pause the game. It will be shown with a large pause symbol displayed over the map." was turned into UR_PAUSE: "The user can pause the game" and FR_PAUSE : "When the game pauses, a large pause symbol is displayed over the map". Thereby making the requirement more appropriately documented, by splitting it into a high level UR and a specific FR.

3. Adding missing requirements and Corrections

The UR table forgot to mention the requirements of the in-game map/world and the map was only mentioned briefly in the UR_MOVEMENT description. So to comply with the client's request for a map representative of the university campus (as discovered in the interview) we decided to add the UR_MAP requirement. Furthermore we also decided to add UR_SOUND, UR_AVATAR and UR_ACCESSIBLE, as in our initial interview the client said it would be nice for the game to have sound effects and happy music, but it wasn't a must have.

Also the requirement of NFR_COMPATABILITY was too vague as it does not specify which type of OS the game should run on, so its description was updated to include the mention of Microsoft Windows 10.

FR_SUCCESS and FR_FAILURE were replaced with FR_SCORING, as the client stated that he doesn't want a player to win or lose the game, but rather achieve the highest score possible.

Many FR requirements were also added as some key requirements weren't mentioned, these requirements include: FR_DOUBLE_STUDY, FR_TIME, FR_ENERGY, FR_INTERACT_DISTANCE etc.

4. Updating the tables with the additional requirements

From the updated brief and the second interview with the client, we needed to add the requirements of UR_ADDITIONAL_MAP, UR_MAP_OVERVIEW, UR_LEADERBOARD, UR_STREAKS, UR_CREDITS to comply with the new needs of the client (discovered in an additional interview) and updated brief. We then added the appropriate FR requirements to complement these additional user requirements, which includes FR_LEADERBOARD, FR_STREAKS, FR_STREAKS_GYM_BRO, FR_STREAK_LOTD, FR_STREAK_JOGGER, FR_ACCESSIBLE, FR_MAP_OVERVIEW_CONTROLS, FR_MAP_OVERVIEW_MOVEMENT, FR_CREDITS