UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Change Report

Group 15 - HesHus

Adam Robinson

Tsveta Ivanova

April Jack

Oliver Goodwin

Luke Jackson

Luca Hammond

Chris Rolfe

Planning
After observing our updated requirements, interpreting how to modify our chosen project in order to meet them and distributing mark allocations evenly, we first constructed a Gantt Chart to identify how much time we had to work with, and what things needed to be finished by each week in order to stay on track to meet our requirements. We also started to use GitHub issues to keep track of individual implementation requirements for our game (e.g. adding a score function), and who would be responsible for implementing it. We continued to use Slack channels to communicate on a day to day basis.

Development
We largely used the same technologies to develop our project, those being LibGDX, IntelliJ and Tiled, as our chosen project used these and we were most familiar with these technologies - there would be no point in learning something new to develop the remainder of the game. The previous group used a docstring format for detailing the purpose of code snippets, and we adopted this method when developing our project.

Tracking Progress
To track progress, aside from referring to the Gantt chart deadlines and GitHub Issues, we set up a continuous integration environment using GitHub actions. This was useful as it automatically generated useful artefacts whenever we pushed a version of the project to GitHub. These included a coverage report indicating how much code our unit tests covered and an executable .jar of our project which we could quickly run separately from the rest of the program. We also conducted weekly scrums and wrote summaries for all of our group meetings, which also helped to track progress made by the group and each individual.

Reviewing Changes
To review changes we made to the product, we used automated testing with JUnit5. This allowed us to automatically test basic functionalities of the product without having to do it ourselves which saved us time in development. For more complicated tests, such as integration and end to end tests that covered a larger scope of the program, or tests concerning non-functional requirements, we used manual test cases. Our scums and meeting summaries also helped to surmise what we accomplished/needed to accomplish individually and as a group on a weekly basis, and we also updated the Gantt Chart regularly to reflect any changes that we made to the plan.

i. Requirements

Introduction:
On inspection to the introduction of the Requirements deliverable, we noted that the previous team had followed a requirements elicitation process almost identical to ours. The deliverable clearly stated the sequence starting from the brief, to the client interview into forming a SSON. The only difference in this approach to ours was the increased communication to gather these requirements, so we added the statement: “Throughout the project, we also liaise with the client to gain clarification on any requirements that seemed ambiguous.”. Our only other changes to the first section were to mention specifically where

we had followed the IEEE 29148 standard for certain parts of the process - using an example of when to use ‘shall’, ‘should’, and ‘may’.

Requirements Table

We searched for any requirements that were missing but necessary for Assessment 1 but all of these were included, hence our only change for this criteria was making UR-ACCESSIBLE more descriptive.

Following this, we added the requirements stated in the product brief for Assessment 2.

Changes Made	Justification
Renamed IDs containing numbers	It is bad practice to use numbers in IDs as they become less clear (e.g. FR-MENU2). These have been appropriately renamed
Added UR-STUDY-TASK, UR-MAP-LOCATIONS, UR-RECREATIONAL-ACTIVITY-TASKS, UR-PAUSE-MENU UR-ACTIVITY-COUNTER, UR-RECEIVE-FEEDBACK, UR-POINTS-SYSTEM, UR-POINTS-PENALTY, UR-ACHIEVEMENTS, UR-LEADERBOARD, FR-RANDOM-EVENT, FR-RESOLUTION, FR-NPCS, FR-POINTS-REWARD, FR-ACHIEVEMENTS, FR_POSTGAME_LEADERBOARD	These are the user requirements that were not present in the previous document but were required for Assessment 2. They state the requirements of the full product brief. Those that were not explicitly stated in the brief were gathered from the client meeting by asking follow up questions.
Edited UR-SOUND	The previous requirement failed to mention that the sound is controllable by the player
Added FR-DYNAMIC-TIME	Edited FR-TIME to specifically FR-ACTIVITY-TIME, which covers the passing of time when completing activities, and allows for this new functional requirement to cover how the time passes while playing the game in general
Removed FR-ENERGY2	This was the same as FR-SLEEP2, which was then changed to FR-SLEEP-ENERGY
Added FR-ANIMATIONS	The previous game already included avatar animations but did not include this in the requirements document.
Edited FR-GAME-PLAY4	For the updated brief, players can interact with three leisure activities instead of one
Added FR_INTRODUCTION	This is necessary because the user needs to

	be told the context of the game and what they are meant to do
Removed NFR-TIMING1	This is more of a functional requirement than non-functional, and it is already included as FR-SLEEP-TIME
Traced back NFRs to URs	Some NFRs didn't trace back to a UR, so we found relevant user requirements to link them to

ii. Architecture

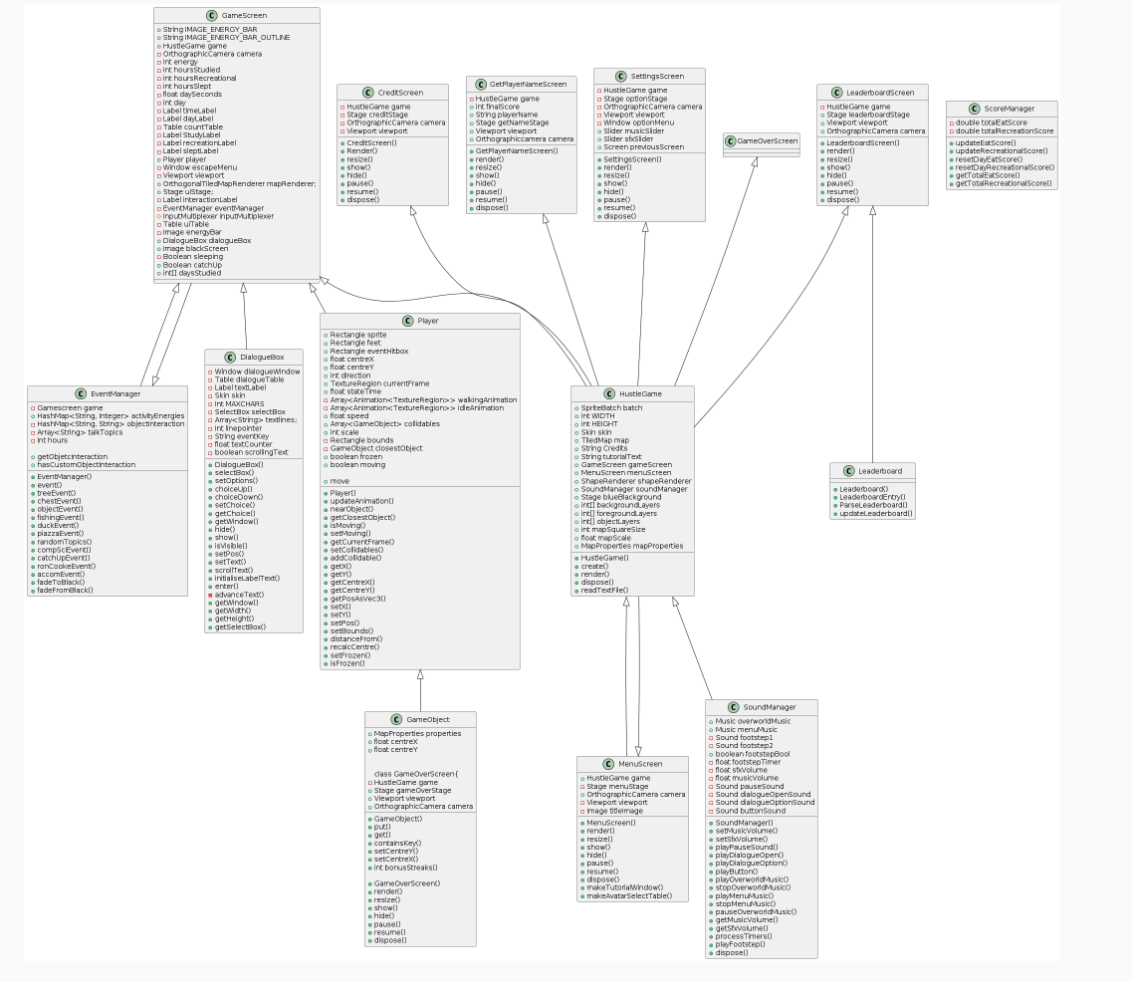
We initially decided to update all of the architecture diagrams team 16 had created. However, after closer inspection, we found that the class diagram was quite complex and didn't match up to the inherited code that well, making it hard to use and update. We therefore decided to create a new class diagram based on the original code, that we could then update with our new classes, functions and attributes. Because of this, we also didn't need to reproduce or update the package diagram, as it was implemented by team 16 due to how cluttered their initial class diagram was, a problem we did not have. We didn't have any problems with the

state diagram as only two new screens were required so we could easily update the current one.

Updated State Diagram

The updated state diagram includes two new screens, GetPlayerNameScreen and LeaderboardScreen. Both are quite self-explanatory and only appear once the game has finished. The game automatically goes into GetPlayerNameScreen, where the user is prompted to enter their name, they then have the option of viewing the leaderboard on LeaderboardScreen by pressing the ‘Leaderboard’ button

New Class Diagram



The new class diagram includes all inherited classes from the first iteration, whilst removing some of the unused classes and entities described in the original. We added new classes and attributes that were to be used to satisfy the new requirements, such as the LeaderboardScreen and Leaderboard classes which are used to implement the leaderboard shown at the end of the game.

iii. Method selection and planning

For method selection and planning, we kept the same methodologies as the previous iteration. This was to aim to reduce changes to the workflow of the group as the methodology worked well previously. This choice resulted in some changes from the previous team's planning. Like the previous team, we used meeting summaries to keep track of weekly meetings and responsibilities, which were then represented in a visual form using a gantt chart. However we did not make use of the requirement and tasks table as we felt the information from these tables was well represented in our meeting summaries and gantt chart.

iv. Risk Assessment and Mitigation

Updates to the Introduction:
There was very little change required for the introduction - explaining and justifying the team's approach to risk management. This was due to the whole process being very similar to ours in Assessment 1, although since reading their introduction, we adopted their use of the risk matrix (a more meaningful way to consider likelihood and severity), where the measure for likelihood was multiplied by the severity, and the result is categorised into red, yellow and green- describing the significance of the risk, with red being most significant and green being least significant. We also continued with using the extra columns of ‘Impact Description’ and ‘Reassessment Date’ as they provide more information and clarity. As a team, we thought this process was explained well, with appropriate citations, and describes the risk management process in good depth. All aspects of the risk register were justified sufficiently using the prior explanation of the process. Our only change was to the use of google forms for reporting new/changed risks throughout the project. We felt like our previous approach of setting up weekly/biweekly meetings, along with a dedicated Slack text channel, worked well for monitoring risks - so we didn't want to add any extra complication to the process. We appreciated the anonymity to the google forms however our group dynamic made this feel like it wasn't required.

Updates to the Risk Register:

Risk ID	Added/ Edited	Description/Justification

18	Added	This describes the unfamiliarity of any technology in the new codebase due to the situation of building from another team's code. It was initially a priority level of 9, but as our team started to understand the codebase more, it was downgraded to a priority level of 6.
19	Added	This describes a lack of understanding the new code due to bad comments/documentation since we were building from another team's code. It was initially a priority level of 16, but as our team started to understand the codebase more, it was downgraded to a priority level of 8.
20	Added	This risk addresses legal issues brought about from inheriting other peoples code- of which we are not clear about any licenced assets used. This began as a priority level of 15 then decreased to 10 as we discovered any licences required.
14	Edited	Here, we elaborated on the impact description and mitigation strategy due to the testing section of Assessment 2 requiring more attention - also increasing the probability level.
16	Edited	We added some extra ideas to the mitigation section: defining an environment for testing + continuous integration. This is because we found that there were more ways to mitigate the risk of this happening.
17	Edited	We added more methods to avoid this risk (merging more frequently and including more decomposition to allow for shorter branches).

Additionally, we updated some general likelihood and severity values (e.g. 2, 4, 9) as time passed, along with fixing some incorrect ID references. We considered the rest of the risk register and found it to include everything useful for the project, particularly the separation between risks 8 and 9, as there is a significantly different impact between losing 1 or 2 members.