Project Organisation

- Team:
    - Michael Beal - mtb532@york.ac.uk
    - Lucas Solomon - lmgs504@york.ac.uk
    - Thomas Auburn - ta937@york.ac.uk
    - Ryan Early - re734@york.ac.uk
    - Inna Strogonova - is844@york.ac.uk
    - Daniel Pearce - dp1039@york.ac.uk
- Clients:
    - Dimitris Kolovos - dimitris.kolovos@york.ac.uk
    - Javier Camara - javier.camaramoreno@york.ac.uk
    - Nicholas Matragkas - nicholas.matragkas@york.ac.uk
- Stakeholders:
    - ENG1 Cohort
    - Clients
    - The University of York Communications Office
- Communication mechanisms and schedule:
    - Discord
    - Zoom
    - Github
    - Email
    - Google Drive
    - deadline: Aut/9/November 25

Resources

- Preferences of the customer:
    - Must be written in Java
    - Should be able to run on university machines
    - Can run on multiple operating systems
- Tools and infrastructure:
    - Java
    - Gradle build tools
    - Game engine
    - Image manipulation software
    - Personal computers
    - Internet research sources
    - Github/Git
    - IDE of choice
    - Google Drive

Justification of communication methods and tools we used

Discord

We chose to use Discord for general communication and text messaging as it has a lot of useful features such as voice/text communication as well as creating specific channels for different parts of the project. We chose it over other, similar services such as Slack because most of the team has more experience with Discord. Additionally, the features that Discord doesn't provide will likely go unused, unhelpful to the project.

Email

We are using email to communicate with lecturers to arrange customer meetings and to setup initial communication between our team members. In addition, messages that have been sent and received can be stored and searched through safely and easily.

Zoom

We are using Zoom as it allows us to have contact with the Lecturers (Customers) while in a Practical session. It also has the options to chat, share screen, documents and whiteboard - everything needed for team meetings.

Google Drive

Google Drive provides an easy way to collaboratively work on files as a team and be able to see changes live. It provides an option to create a safe backup of our files and documentation on edits and changes made over time. It also allows us to control the level of access to shared documents.

Github

Used to host the code for the game. This service stores files in Git repositories, allowing for version control and collaborative work. It also has an issue tracker that allows us to flag issues to be fixed and discussed among other features.

Game engine

The game engine we have decided to use is called Libgdx. We decided to develop our game using this engine because it provides an open source framework for the game to be built on, providing some of the important features that we require, as well as having multiple tutorials for its use online.

Gradle build tools

Build tools are used to automate the building of the software and manage dependencies(such as the game engine library), among other reasons. We decided to use Gradle because we will likely use libgdx which is designed for this build system. Originally, we decided on Maven because some of the members of our group had used it previously. However, their experience wasn’t extensive enough for

this to be considered and as a result, we decided to use Gradle, considering the game engine.

IDE of choice
The IDE that each team member uses doesn't matter much as long as it has modern tooling and supports our chosen build system, however IntelliJ or Eclipse have been recommended.

Team Organisation:

Team roles

- Michael Beal - Team leader and all deliverables except implementation
- Lucas Solomon - Architecture and website
- Thomas Auburn - editing deliverables and assisting with implementation
- Ryan Early - Risks and editing deliverables
- Inna Strogonova - Architecture, editing and adding to deliverables, website
- Daniel Pearce - Implementation

How the team works
We have organised our team by trying to work through one or two deliverables at a time as a group and discussing how best to complete the task based on the lecture slides for that deliverable combined with the descriptions of how marks are allocated. We used the “eng1 team assessment” pdf that describes specifically what must be included in each section of the deliverable in “3.3.1 Deliverables for Assessment 1” as well as specifying the number of pages required for each segment of the deliverable. We also have been reading over each other's writing as we are working on the same areas at the same time which allows us to be able to provide a peer review as well as allowing members of the team to easily ask questions to each other without confusion of the topic or explanation of what they are talking about. As we complete a deliverable we go back to the topic and ensure that any changes we decided are reflected in it so that it accurately displays our current intentions, this allows us to leave areas blank that we are not currently sure about how to complete or areas that we can only decide how to fill after completing a later task. This method is appropriate for our team and for the project as a whole because it clearly outlines what must be submitted for each section of each deliverable and allows us a clear understanding of what is expected. It also gives us an easier time deciding what we must include in each segment as a blueprint for what we are writing.

Critical path
https://enigmagroup29.github.io/ENG1-Project/documents/methods_planning/critical_path.png

Project Schedule

Why is this a sensible project schedule and describe what's going on

- Auber:
    - Website:
        - Design the website so it is easy to locate and access
        - Maintain and update
    - Requirements:
        - Conduct customer meetings
        - Give an Introduction
        - Complete Statement of requirements
            - Relevant environmental assumptions
            - Associated risks
            - Potential alternatives
        - Identify user requirements
        - Identify software requirements
            - Identify functional requirements
            - Identify non-functional requirements
    - Architecture:
        - Give a concrete representation
        - Give an abstract representation
        - Justify the representations
        - Relate the concrete representation to the requirements
    - Method selection and planning:
        - Complete Project organisation and Resources
        - Justify communication methods and tools used
        - Choose a development methodology
        - Complete Project Schedule
        - Provide weekly snapshots
    - Risk assessment and mitigation:
        - Give an Introduction
        - Justify the chosen format of presentation
        - Give a tabular representation
    - Implementation:
        - Do a research, gather information
        - Learn how to control Game engine
        - Decide upon the game design
        - Implement the game
        - Build AI
        - Extensibility
        - Identify unimplemented features
    - Testing:
        - Complete unit testing
        - Complete customer testing

Gantt chart

https://docs.google.com/spreadsheets/d/1Upd7DET_l2K07VoP0pFK6DxKQoOQjlUdwAGG8TZFjn4/edit?usp=sharing

How plans changed over the project

Our schedule for our implementation was shifted backwards a couple of weeks due to issues with uploading code to Github and difficulty communicating with members of the team which caused difficulty beginning the implementation process. However, once the implementation was started, it went smoothly, and quickly, so this was not a major issue.

During our customer meetings we asked several questions about the requirements and what exactly was needed in order to successfully complete the project. One of the areas that they informed us should be added was a Demo feature of the game where it would run a pre-scripted game to be demonstrated on repeat during open days for potential students to view.

We were originally going to include all diagrams on the documents themselves but we discovered that they would fill up too much of the page count of the deliverables. Due to this we asked our customer if we could include the diagrams outside of the deliverables themselves and link to them on the page which was accepted. For the rest of the schedule we remained on track, completing the main deliverables that were not dependent on the implementation on schedule.

Software engineering method

As a software development framework for our project we chose an agile method Scrum. Agile development focuses on delivering work in small, frequent increments, its approaches are easy to learn and sufficiently documented. It is suitable for small teams working in a flexible environment and, therefore, for our engineering project. In Scrum, development is divided into short ‘sprints’ preceded by sprint planning where the goal of a sprint is decided and followed by sprint reviews where the result of a sprint is inspected. The methodology also features daily scrums – short meetings where team members discuss what is done and what is yet to be done. There are several reasons why we opted for Scrum. To start with, Scrum acknowledges that the team does not know everything at the start of a project and will evolve through experience, which fits in with our team project as we continuously learn and research in order to complete relevant parts of the project. Scrum is also known to be simple, straightforward, and easy to implement. In addition, quick releases keep the team motivated throughout the development process.