Change Report

Cohort 1 Group 1:
Ileri Adegeye
Maida Ahmed
Holly Ainsworth
Anthony Colin Jones
Luca Gilligan
Alex Joyner
Minhaj Zaidi

Our Protocols

The project we took over does not satisfy the product brief; therefore we will have to change it as we go.

To manage and track changes to these already substantial documents we did as follows:

Kept the copies of the original PDFs in a shared folder on Google Drive for convenient access.

When a deliverable needed updating during development, any updates or amendments would be added in a tab of this document - one tab for each deliverable from Team 11.

The team members responsible for updating these tabs and tracking the changes were the same ones allocated to the corresponding deliverable previously - the members doing Requirements would understand how to extend and maintain the Requirements document, for example.

The members responsible for this had evaluated these deliverables when we selected which project to take over, and so could be assumed to understand how their deliverable(s) for team 11 specifically worked and fit together.

We used GitHub for version control as we updated and extended Team 11's code - separate repositories (1) for the game's code and (2) the source code for the website.

These repositories were forked copies of Team 11's original code.

Any further processes for updating specific deliverables are located in their specific section of this document.

Requirements

Document References

- Original Deliverable (Inherited): https://eng1-c1g1.github.io/assets/Req1.pdf
- Updated Deliverable (Current): https://eng1-c1g1.github.io/assets/Req1_updated.pdf

Overall, we maintained the core requirement specification from Team 11, as it accurately addressed the three different types of requirements (Functional, Non-functional and User requirements). It correctly covered the product brief and closely resembled our own requirements that we developed in the first assessment. However, we have now extended the requirements to support two new features - an Achievement System and a Leaderboard.

We have decided to keep the structure and logic of the inherited requirements from Team 11 because they are an excellent starting point for furthering the development of the maze game.

In addressing the feedback received in Assessment 1 about gameplay, Team 11 has successfully differentiated between Functional Requirements (i.e., FR_Timer) versus Non-functionality Requirements (i.e., NFR_Timer). This differentiation is an improvement upon that section of the report based on the feedback received in Assessment 1. Furthermore, the features of Assessment 2 have mostly already been covered by the requirements we inherited from Team 11, such as (FR_EVENT_AMOUNTS) and (FR_DEAN), a controllable "Dean" character that the player must avoid.

We have also added the Leaderboard and Achievements in order to satisfy the requirements of the product brief and enhance the competitiveness and replayability of our game. With the advent of a Leaderboard and Achievements, the user experience became more enjoyable and there was a competitive element to getting the highest score on a Leaderboard while playing.

- The Leaderboard's role is vital because it gives players the context they need to compare their performance to others, and highlights the fact that players win by getting the highest score, plus it boosts player engagement.
- Subsequently, an Achievement system really complements the addition of a leaderboard as it calls for rewards and challenges. It provides rewards for specific interactions (e.g. "collected all check-in codes in one run") encouraging players to explore different areas of the maze rather than just rushing towards the exit.

The following requirements were added to the updated deliverable to formally portray these new features in a tabular format:

- Added UR_LEADERBOARD: "The player shall be able to view a leaderboard of the top 5 scores to compare their performance." This meets the client's request for competitive features.

- Added FR_LEADERBOARD: "The system shall save the player's final score and sort and display the top 5 scores on the leaderboard."
- We linked this functional requirement to both UR_LEADERBOARD and the modified UR_DO_NOT_SAVE so the developers understood that saving this data was an exception to the "no saving" rule.
- Added UR_ACHIEVEMENTS: "The player can unlock achievements for specific feats (e.g., "collected all check-in codes in one run") and view them on the title screen, separate from the score."
- Added FR_ACHIEVEMENTS: "The system shall track achievements and save unlocked milestones to display on the title screen."

Modifications:

- The original idea was that no user data should be retained after the current game is over; this conflicted with the new requirement in the product brief, which needed an Achievement system.
- While keeping the restriction on saving the player's position, we changed UR_DO_NOT_SAVE to specifically let the score and achievement data to be saved.
- We also modified NFR_DO_NOT_SAVE: "The system shall save scores and achievements data immediately upon game end, while discarding all positional and inventory data". In order to conform to the new Leaderboard feature, we changed this criteria. The only data preserved or retained now is the user score and achievements.

Conclusion:

We have developed a Requirements Document based on the technical content of Team 11's Specifications, keeping only what was essential for compliance with the Product Brief for Assessment 2. The document captures every required feature and is well developed from a technical perspective. The contradiction between the new Leaderboard feature and the current 5-minute constraint was resolved by adjusting the requirements for UR, FR, and NFR with regard to data persistence. This rework satisfies engineering requirements.

Architecture

Document References

- Original Deliverable (Inherited): https://eng1-c1g1.github.io/assets/Arch1.pdf
- Updated Deliverable (Current): https://eng1-c1g1.github.io/assets/Arch1_updated.pdf

The original code had good architectural structure, therefore minimal changes were made to the core components of the game. The new Entity-Component diagram shows the new entities that have been added which includes: BullyBribe, Pi, LongBoi, Teleporter, TimeLoss, Puddle, Ankh and Bully. These features are added requirements in the brief for assessment two which is: an assortment of different positive, negative and hidden events. They utilise the core components that were originally in the code such as InteractableComponent, TransformComponent, PhysicsComponent and AnimationComponent.

We have only added one net component due to the extensiveness of the original code. This component is the BullyComponent and the purpose of it is to allow entities to be marked as a "bully" providing the parameters for bully behaviour. TimeLoss is an entity that only has an ID, acting as a trigger that interacts with the TimerComponent. When a TimeLoss entity is triggered it is detected and the effects are triggered, after this it is disabled.

Two new systems have been added to the behavioural diagram; PauseSystem and BullySystem. The purpose of the systems is to be responsible for the logic of a type of behaviour, overall executing the games logic. The systems are executed in a deliberate sequence. BullySystem is after PhysicsSystem so it can react to accurate collision data. It can also apply forces based on resolved physics. PauseSystem is placed last to override behaviour because it needs global visibility.

The BullySystem is responsible for making bully entities interfere with the player in a way that works with the physics of the game. It determines the bullies behaviour state, calculating movement and then applies the forces through the physics later. This feature was added as one of the negative events to fulfill UR_NEGATIVE_EVENTS and FR_NEGATIVE_EVENTS.

The PauseSystem checks whether time-based systems are allowed to update, checking the current game state and whether it is paused or unpaused. This means it freezes timers, player input and animation. It stops updates that depend on elapsed time. On the other hand, it allows exempt systems (e.g. menus or UI) to continue running. This feature was required to fulfill the UR_PAUSE requirement as well as related requirements like FR_PAUSE_MENU.

Method and Planning

Document References:

- Original Deliverable (Inherited): https://eng1-c1g1.github.io/assets/Plan1.pdf
- Updated Deliverable (Current): https://eng1-c1g1.github.io/assets/Plan1_updated.pdf

Overall, the planning and methodology required minimal modification because the original team's framework was already well-aligned with our project requirements.

Their structure functioned effectively as a skeleton, allowing us to preserve proven workflows while making targeted adjustments where our context differed. This continuity reduced overhead in re-planning and ensured methodological consistency, which is commonly recommended in iterative software development when inheriting an existing codebase and project structure.

Although our team adopted a more collaborative, team based working style, the underlying workflow closely resembled the original Gantt-chart driven structure. The primary change was time-based rather than procedural: milestone dates were updated, and implementation sprints commenced from the project outset. This adjustment reflects a front loaded development strategy, enabling earlier integration and testing without fundamentally altering dependencies. We have changed the gantt chart on the website to reflect this.

In the software engineering methodology section, the project selection phase was expanded to reflect our evaluation and change driven approach. Unlike the original brainstorming process, our method emphasised comparative assessment and feasibility analysis.

This change serves to demonstrate similarity in methodologies in effort and duration while improving decision traceability. Additionally, the inclusion of user and play-testing feedback formalises adaptability in the implementation plan, aligning with established agile principles that advocate iterative refinement based on critique.

Source control required only minor updates, as the existing system was retained. Since the infrastructure and workflow were already in use, no additional tooling or branching strategies were necessary. This decision prioritised continuity and risk reduction, which is consistent with best practices when maintaining or extending an inherited repository.

Team organisation remained largely unchanged due to its effectiveness and close alignment with our own practices. Only minor scheduling adjustments were made to meeting dates to accommodate availability, without impacting roles, responsibilities, or communication hierarchies.

For collaboration software, the documentation was updated to reflect our use of a single centralised group chat rather than a WhatsApp-based community. This change simplifies communication channels while preserving functional equivalence in coordination and responsiveness.

In the plan evolution section, small revisions were required because task progression and workload distribution closely mirrored the structure used in Assessment 2. This similarity indicates consistency between planned and executed work, reducing the need for justification.

Finally, the weekly plans section was removed due to redundancy and length constraints. Its content was already effectively represented through the Gantt charts, which gave a concise visualisation of timelines and dependencies. Removing this section improved document clarity without sacrificing informational completeness.

Risk Assessment and Mitigation

Document References:

- Original Document: https://eng1-c1g1.github.io/assets/Risk1.pdf
- Updated Document: https://eng1-c1g1.github.io/assets/Risk1_updated.pdf

Our Process:

When selecting a project to take over, we did include considerations for the quality of their documentation after considering how easy the implementation would be to extend - as although it would be easier to restructure their documentation than their entire codebase, it would still take time and effort that could be spent elsewhere in the project.

On reviewing their documentation, we found that their risk assessment and mitigation plans were:

- Systematic and measurable, identifying risks in a risk register - containing their severity, likelihood, what mitigation measures were needed and who was responsible for those mitigations.
- Realistic and appropriate, updating the risk register throughout development and when assigning new tasks to each member - allowing the team to account for unforeseen risks.
- Actionable, as the constant updates and systematic classification allowed team members to prioritise mitigating the most critical risks to their project.

As a result, we were satisfied with largely reusing Team 1's risk assessment documentation; the only element we needed to change was the team members responsible for each mitigation.

After an initial group brainstorming session, we allocated responsibilities to each member as described in the risk register of the updated requirements document, linked above.