METHOD SELECTION AND PLANNING

GROUP 5 - BITCRUSHED BOB

Maryam Mathews
Joseph Hinde
Jacob Mace
Will Aston
Zathia Jacquesson-Ahmad
Bulganchimeg Munkhjargal
Evan Weston

Software Engineering Methods:

Outline and Justification of Methods:

The team adopted an Agile software development methodology, specifically implementing the Scrum framework. Agile was chosen because it supports iterative development, continuous feedback, and adaptive planning - all of which align with the project's need for flexibility and ongoing refinement. The project involved multiple interdependent components so a rigid, plan-driven method would have limited our ability to work efficiently.

Scrum provided a structured yet flexible framework. We divided development into weekly sprints, each beginning with a sprint review of the previous week, and ending with sprint planning for the following week. This cycle enabled:

- Incremental development: delivering features in small, modular units to facilitate maintainability and future expansion
- Frequent feedback loops: improving quality and alignment with customer expectations
- Cross-functional collaboration: allowing team members to integrate and test their contributions continuously

Development and Collaboration Tools:

To support our Agile methodology, the team used several tools to streamline collaboration, task management, and development:

GitHub - Version Control:

- Required a version control system for saving previous versions, tracking changes and facilitating collaborative development, chose GitHub because:
    - Allowed team members to collaborate safely and maintain a history of all changes
    - Supports distributed development, branching, and merging workflows.
    - Pull requests enabled code review and traceability, aligning with Scrum's collaborative ethos.
- Alternatives: GitLab was considered, however GitHub was preferred for it's integration with other tools and familiarity among team members.

Trello - Task Management:

- Required a task management tool to keep track of the status of tasks, chose Trello because:
    - Provided a clear, visual overview of sprint progress through Kanban boards.
    - Tasks could be easily assigned, prioritised, and moved through workflow stages, reinforcing Scrum transparency
- Alternatives: Jira was considered, we deemed it overly complex for our needs.

Visual Studio Code - Code Editing and Debugging:

- Justification: Lightweight IDE with integrated Git support and extensions for collaborative development, aligning with Agile's emphasis on developer productivity
- Alternatives: IntelliJ was considered but required more configuration and was less consistent across all team member's systems.

Team Organisation:

The team approached team organisation by dividing the project into two main groups - technical and non-technical - and then further subdividing tasks within each group. This structure was decided by conducting the 'Design Team Alliance' exercise, proposed by the ORSC, which encourages explicit discussion at the outset regarding team organisation and the division of responsibilities. During this process we all concluded that we wanted the project's work to be evenly and explicitly divided, and that the task assignments should cater to each individuals' strengths.

This approach to team organisation was effective for this project as it aligned closely with agile principles: promoting autonomy, transparency, and iterative collaboration. Since we clearly divided the work, each member was able to work independently on their respective components, which greatly helped workflow efficiency as people were able to work whenever was convenient for them. In order to maintain consistency, we conducted two weekly meetings which ensured that each member's work was cleanly integrated into the project and that work was being completed on schedule.

Plan:

When it came to planning our work on a weekly basis we used Gantt charts to make sure everything stayed on track. Utilising this approach for planning allows us to easily identify dependencies as with most Gantt charts the tasks to the right are dependent on completion of the tasks to the left. These gantt charts were used to break down our keys weekly tasks into bite sized chunks that everyone related could work on.

When the team was drafting a rough plan, we decided we would plan until a week before the submission date to leave a sort of "Grace period" to mitigate any of the risks that do unfortunately come to fruition. This decision allowed us to adapt to any situation that could happen during the development of the game.

Visual representations of the plan can be found on the team's website (https://escape-from-uni.github.io) in the form of Gantt charts.

Week 1: Team building and Method selection:

Key tasks:

- Research possible java game libraries
- Understand the Assessment requirements
- Assign roles within the team

Priorities:

- Evaluate each members skills

Dependencies:

- None

Week 2: Interview and Requirements development

Key tasks:

- Create interview questions

- Create a comprehensive set of requirements taken from the client
- Assign functional and non-functional requirements of the system

Priorities:

- The whole team must be knowledgeable about the requirements by the end of week

Dependencies:

- Interview performed earlier in the week
- The transcript created from the interview

Week 3: Initial designs and Risk Assessment

Key tasks:

- Brainstorm a design and plan the gameplay loop
- Create a Risk Register covering all pitfalls during the project

Priorities:

- Have an initial game loop and design the players experience

Dependencies:

- Design dependent on the requirements given by the client
- Risk assessment dependent on the team choosing roles and tools for the project

Week 4: Architecture and Implementation

Key tasks:

- Start base implementation of generic movement and controls
- Create a plan for the architecture and class systems of the game

Priorities:

- By the end of the week have a moving sprite on the screen
- Have a design of what the structure of the game will look like

Dependencies:

- Both dependent on having the base design of what the game should do

Week 5: Implementation

Key tasks:

- Implement a tile map system to allow a graphical map
- Implement object interaction for the events in the game
- Implement timer
- Start finalising documentation

Priorities:

- More implementation

Dependencies:

- Simple game implementation must be complete before this

Week 6: Documentation Review and Submission

Key tasks:

- Finish and review documentation
- Give the game some final polishing

Priorities:

- Submission by end of week

Dependencies:

- All documentation must be finished
- Game must be functional with its mechanics

Evolution of the Plan:

Initially, the plan for the project was to explicitly divide work between two independent groups, with each group working largely autonomously.

This initial plan was significantly revised midway through the project in response to one member of the technical team becoming ill and being unable to complete their assigned work. This disruption required the teams to switch from an independent working structure to adopt a more collaborative approach, with members from the non-technical team helping to ease the workload of the technical team.

This shift demonstrated the importance of flexibility in a project plan to adapt to unforeseen circumstances, like illness. The team was able to adapt the plan efficiently in response to this change as this circumstance was acknowledged as a potential risk, and damages were mitigated by ensuring all tasks had a high bus factor - enabling our ability to meet the set deadlines to not be heavily affected.