User Experience, Performance, and Fuzz-Testing Subplan

A Documentation of Testing Details and Strategies for a Structured Approach

ENGR302: Engineering Project Management 2

Created 22 September 2024

1.0 User Experience (UX) Testing

This type of testing evaluates how users interact with the game, focusing on ease of use, engagement, and overall satisfaction.

1.1 UX Testing Methodology

- **Define objectives:** Understand what UX elements need to be tested, such as the game's interface, usability, navigation, learning curve, teamwork dynamics, and how well Agile concepts are communicated.
- **Create a UX testing group:** Gather users (plausibly the development team) who will represent the target audience to assume gameplay to check Agile concepts.
- **Test in a realistic environment:** Have players play the game in the environment it is intended for (i.e., online multiplayer).
- **Capture feedback:** Use direct observation and post-play discussions to track issues related to gameplay, difficulty level, learning of Agile concepts, and teamwork effectiveness.
- **Analyse results:** Look for patterns in user frustration, confusion, or enthusiasm. Pay attention to how intuitive the game's Agile-related mechanics are and how well they support collaboration.
- **Report:** Document user-flow, engagement, pain points, and suggestions for improvement, including UI/UX design, game mechanics, and how Agile processes are presented.

1.2 UX Report Structure

Introduction: Overview of the game and UX testing goals.

Methodology: How testing was conducted (participants, scenarios).

Findings: Summarise player feedback, usability issues, and their learning experience.

Recommendations: Suggestions for improvements, including game mechanics, UI, and player guidance.

2.0 Performance Testing

Performance testing ensures that the game runs smoothly under various conditions, focusing on frame rates, load times, network latency, and memory usage.

2.1 Performance Testing Methodology

- **Set performance goals:** Define metrics like acceptable frame rates (e.g., 30+ FPS), load times, and responsiveness during multiplayer sessions.
- **Identify OS requirements:** Test the game on various OS setups (e.g., Windows, MacOS, Linux).
- **Analyse network performance:** Test the game with multiple players and check how it performs with different numbers of users in a session. Test for multiplayer latency, synchronisation issues, and responsiveness in online sessions.
- **Test game stress points:** Focus on parts of the game where performance might degrade, such as high-action sequences, multiplayer interactions, and complex levels.
- **Monitor performance:** Use profiling tools like Unity Profiler to track CPU, GPU, and memory usage. Record how well the game handles frame drops, lag, and load times under various conditions.
- **Report:** Document how the game performs in each test scenario and note any performance bottlenecks or crashes.

2.2 Performance Report Structure

Introduction: Overview of the game and performance testing goals.

Methodology: How testing was conducted (OS setups, number of players, game configurations tested).

Metrics recorded: FPS, memory usage, latency, etc.

Findings: Identify any performance issues, like lag, slow load times, or crashes.

Recommendations: Offer solutions to address performance bottlenecks, such as optimising assets, improving networking code, or reducing resource usage.

3.0 UX and Performance Insights

Since the game incorporates teamwork and Agile processes, a cross-reference on how performance impacts the user experience will be conducted. By testing and documenting both UX and performance thoroughly, a well-rounded evaluation of the game's quality and areas for improvement can be captured.

3.1 UX and Performance Cross-Reference

a. Teamwork Dynamics vs. Game Complexity

- **Insight:** Assess how increasing game complexity (e.g., more tasks, time pressure, or challenges) impacts teamwork and collaboration.
    - Do complex game mechanics promote or hinder teamwork?
    - How well do players cooperate when Agile sprints involve more complex tasks?
- **Example:** Players might struggle with teamwork during complex levels. If so, see if this is due to a lack of understanding of Agile collaboration processes or separate aspects like overwhelming game mechanics.

b. Multiplayer Responsiveness vs. Collaboration

- **Insight:** Understand how the game's performance in multiplayer mode (e.g., latency, synchronisation) affects real-time collaboration and teamwork.
    - Are players experiencing difficulties with Agile task completion or communication due to multiplayer lag or network issues?
    - Does poor synchronisation affect Agile processes within the game?
- **Example:** If players fail to complete objectives due to delays in communication or action syncing, this may indicate performance issues, which in turn will harm Agile-related collaboration mechanics.

c. Learning vs. Game Progression

- **Insight:** Examine how players' ability to learn Agile concepts correlates with the game's progression system (e.g., levels, objectives).
    - Are players gaining a deeper understanding of Agile as the game progresses?
    - Do later levels introduce Agile concepts in a way that feels intuitive based on previous levels?
- **Example:** If players fail to understand more advanced or different Agile concepts in later levels, it might mean the game progression is not aligned with the learning curve. Check if earlier levels build enough of a foundation for more difficult tasks.