CMPE 492
SENIOR PROJECT 2

Project Name
Astroni

Project Supervisor
Ulaş GÜLEÇ

Project Team Members
Ata AYYILDIZ
Nilay ÖZKIR
Ömercan YALÇIN

Project Jury Members
Emin KUĞU
Gökçe Nur YILMAZ

Course Coordinator
Saiful Islam

Project Website
https://teduastroni.github.io

Table of Content

1. Introduction

1.1 Purpose

We, the development team as “Astroni” present this comprehensive Test Plan Report for our groundbreaking serious game in 2D world by using Unity Game Engine. The purpose of this document is to define the meticulous approach, methodologies, and strategies employed in the testing phase.

1.2 Scope

1.2.1 Testing Approach

This test plan thoroughly evaluates our 2D game, aiming for a perfect balance between fun and adventuring. It covers identifying features, testing methods, and analyzing risks.

1.2.2 Gameplay Mechanics

The scope includes adding needed mechanics according to games that are 2D top-down adventure pixel art game that are Inventory System, Skill Tree, portals that are gates of between levels, aimed at having fun and giving fundamental feel of exploring and feeling the concept of Astroni World. The game’s focus on 2D top-down adventure gameplay, combined with its pixel art style and narrative driven approach, sets the stages for a quality adventuring experience.

2. Defining Scope

2.1 Game Overview

Astroni is an immersive 2D Top-Down Adventure, narrative focused game with a style of pixel art. The game offers a wonderful experience on the world of Astroni and gives the story to manage the situation in the game. Levels are designed to engage players through stories, interact with NPCs and fight with the enemies. The game falls within the role playing and narrative based gaming genres and is targeted at individuals who are interested in exploring and feeling new worlds into the digital side of life.

Key Features: Role Playing Mechanics, Exploring new stories and world.

Target Audience: Individuals interested in feeling and exploring the games that gives them and looking for the styiled with the pixel art world.

Platform: Computer based game.

2.2 Understanding the Game

2.2.1 Story and Levels

In the land of Eldoria, the legend of the lost star known as Astroni has long been a source of hope and wonder for all races. Emelcuil, a young elf, is particularly intrigued by this legend. One fateful night, he has a mysterious dream that directs him towards several stars, each holding a clue to unraveling the mystery of Astroni's disappearance. These stars are linked to portals representing different regions of Eldoria, each offering a unique story related to Astroni. Guided by the stardust, Emelcuil embarks on his quest to find Astroni.

On his journey, Emelcuil encounters a wise sage who is familiar with the legend of Astroni. This sage becomes Emelcuil's guide, leading him through the various challenges he will face. Along the way, Emelcuil interacts with characters from different races, each interaction revealing more about the legend of Astroni. Through choice and relationship mechanics, players will uncover the deeper meaning behind the legend.

However, Emelcuil is not alone in his quest for Astroni. Other characters are also in pursuit of the lost star, leading to conflicts and challenges that Emelcuil must overcome. This game narrates Emelcuil's journey as he strives to reach Astroni and unlock its secrets.

The progression can be done by finishing the four levels of the story, Pastures of The Eldoria, King of the Mountains, Follow the Star, and Come Up Against the Dark.

2.2.2 Characters, Weapons, and Abilities

Other than our lovely main character Emelcuil, we can also see NPCs and several bosses in the levels. For the weapons side of the game, we can see bow and arrow usage and swords that we can use, also there are push and kick attacks according to side mechanics. For the abilities other than those, core mechanics attack, move and rotation of character, dashing are inside the game too.

2.3 Testing Needs Assessment

2.3.1 Number of Testers

We estimate that 10 to 12 people will test the skills, having fun, complexity of the game and other kinds of segments that need to build better games in the testing and development process. We are going to scale and measure the scope of target audience also in this section, who we are calling to play our games, who we really want to convince.

2.3.2 Estimated Testing Time

The testing phase is expected to span a minimum 2 weeks to ensure a thorough evaluation of all features, mechanics and contents that is offered by the game. This period includes iterative testing to address any identified issues.

2.3.3 Resources

Resources required for testing include personal computers that are sourced by us the developer team, testing environment that we can collect needed information easily and access to game design experts from the field.

3. Risk Analysis

3.1 Prioritization of Testing Efforts

In our RPG game development, we prioritize elements that significantly impact the player experience, such as user interface and gameplay mechanics. Visual appeal is paramount for us, as we aim to captivate users with the environments we craft. We value diverse user feedback to ensure our game's visual design resonates with our audience.

Moreover, we place a high testing priority on game mechanics. As these mechanics form the foundation for player interactions, we strive to strike a balance where they enhance gameplay without overwhelming users. During testing, we meticulously evaluate the ease of implementation of these mechanics, aiming for seamless user experiences.

3.2 Identification of High-Risk Areas

Game mechanics and balance: Since the game primarily makes use of RPG components, such as character skills, weaponry, and skill trees for growth, it's critical to make sure these mechanics are both fun and well-balanced. Players may become frustrated or lose interest in these areas if there is an imbalance or a lack of polish.

Narrative & Storytelling: Because the game is narrative-driven, problems with the character relationships, storyline coherence, or story flow could have a big effect on how immersed and happy the player is. Making sure the story aspects are consistent and interesting throughout the player's journey should be the main goal of testing.

Technical Stability: Maintaining technical stability is essential for complicated features like level portals, inventory systems, and NPC interactions. These sections may have bugs or glitches that interfere with gameplay or even make the game unplayable, which would be bad for the player experience.

User Interface and User Experience (UI/UX): Creating a seamless and intuitive gaming experience for gamers is largely dependent on the UI/UX design. Any problems with the user interface's legibility, responsiveness, or navigation could irritate players and reduce their enjoyment in general.

Performance Optimization: Since Unity Game Engine was used in the game's development, it is crucial to maximize performance on a variety of hardware setups. In especially on lower-end devices, failure to optimize performance could lead to problems like lag, stuttering, or crashes.

Play tester feedback and iterative improvements: Play tester feedback is essential for fine-tuning the game's quality as it may be integrated into subsequent improvements. There may be lost chances to improve the gaming experience if player feedback is ignored if important problems found during testing are ignored.

3.3 Response to High-Risk Scenarios

Test results and information must be shared directly with the development team to prevent high risk scenarios from occurring. For this reason, providing feedback is very important during the testing phase. Test results should be reviewed regularly in team meetings and integrated into development strategies. Additionally, testing priorities must be set and determined. In this way, the development team can prioritize high-risk areas. When high-risk problems are identified, effective solution strategies must be produced and implemented. In this way, the development team has more comprehensive information about the possibility of high-risk scenarios occurring.

4. Scheduling

4.1 Development Stages

The development process of Astroni typically consists of four main stages:

Pre-production Phase (Week 1-3):

Conceptualization: Brainstorming sessions to define the game concept, mechanics, and narrative.

Prototyping: Creating initial prototypes to test core gameplay mechanics and features.

Team Formation: Assembling the development team, including programmers, artists, and designers.

Initial Planning: Outlining the development roadmap, setting milestones, and allocating resources.

Production Phase (Week 4-15):

Environment Design: Creating pixel art assets for game environments, including landscapes, buildings, and props.

Character Design: Designing and animating characters, NPCs, and enemies.

Gameplay Mechanics: Implementing core gameplay mechanics such as movement, combat, and interaction systems.

Narrative Development: Writing and integrating the game's storyline, dialogues, and quest objectives.

Feature Implementation: Adding features like inventory systems, skill trees, and portals between levels.

Testing Iterations: Conducting iterative testing to identify and address bugs, balance issues, and gameplay improvements.

Polishing Phase (Week 16-20):

UI/UX Refinement: Improving user interface elements for better navigation, readability, and player feedback.

Performance Optimization: Optimizing game performance across different hardware configurations and platforms.

Bug Fixing: Addressing any remaining bugs or issues identified during testing.

Playtesting: Organizing playtesting sessions with external testers to gather feedback and validate gameplay experience.

Finalizing Assets: Completing remaining art assets, animations, sound effects, and music compositions.

Post-production Phase (Week 21-33):

Final Testing: Conducting comprehensive testing to ensure the game meets quality standards and is ready for release.

Marketing and Promotion: Creating marketing materials, trailers, and press releases to generate buzz and anticipation for the game.

Distribution: Preparing the game for distribution on digital platforms such as Steam, itch.io, or the developer's website.

Launch: Releasing the game to the public, monitoring player feedback, and providing support for any issues or inquiries.

Post-launch Support: Continuing to address player feedback, release updates, and patches to improve the game post-launch.

4.2 Test Timeline

Phase	**Timeline**	**Milestones**	**Key Deliverables**
Test Planning	Week 1	Test Plan Finalized	Review Test Plan Test Strategy Development Test Environment Setup
Test Preparation	Week 2 - 3	Test Cases Designed and Reviewed	Test Case Design Test Data Preparation Test Plan Review
Test Execution	Week 4 - 7	Testing Phase Completed	Functional Testing Regression Testing Performance Testing Compatibility Testing User Experience Testing
Test Reporting & Analysis	Week 8 - 9	Test Results Analyzed	Test Execution Documentation Defect Triage Root Cause Analysis Test Metrics Analysis
Test Closure	Week 10	Testing Completed and Analysis Submitted	Test Summary Report Lessons Learned Handover Post-Mortem

Review Test Plan: Review and finalize the comprehensive Test Plan Report, ensuring alignment with project goals and objectives.

Test Strategy Development: Develop a detailed test strategy outlining testing methodologies, tools, and resources required for each testing phase.

Test Environment Setup: Set up the necessary testing environments, including hardware and software configurations, to facilitate testing activities.

Test Case Design: Create test cases covering all aspects of the game, including gameplay mechanics, UI/UX, performance, and compatibility.

Test Data Preparation: Gather or generate test data required for executing test cases, including sample characters, items, and environments.

Test Plan Review: Conduct a peer review of the test plan and test cases to ensure completeness, accuracy, and relevance.

Functional Testing: Execute functional tests to verify that all game features and mechanics are working as intended.

Regression Testing: Conduct regression tests to ensure that new changes or fixes do not introduce unintended side effects or regressions.

Performance Testing: Evaluate the game's performance under various conditions, including stress testing to assess its stability and responsiveness.

Compatibility Testing: Test the game on different platforms, devices, and configurations to ensure compatibility and optimal performance.

User Experience Testing: Gather feedback from users through playtesting sessions to assess the overall user experience and identify areas for improvement.

Test Execution Documentation: Document test execution results, including test logs, defect reports, and any observations or recommendations.

Defect Triage: Prioritize and triage reported defects based on severity, impact, and urgency for resolution.

Root Cause Analysis: Conduct root cause analysis for critical defects to identify underlying issues and prevent recurrence in future releases.

Test Metrics Analysis: Analyze test metrics such as test coverage, defect density, and test execution trends to assess the overall quality of the game.

Test Summary Report: Prepare a comprehensive test summary report summarizing the testing activities, results, and key findings.

Lessons Learned: Conduct a lesson learned session to identify strengths, weaknesses, and areas for improvement in the testing process.

Handover: Hand over the finalized test artifacts, including test plans, test cases, and defect reports, to the development team for further action.

Post-Mortem: Conduct a post-mortem meeting to review the overall testing process, identify successes and challenges, and make recommendations for future projects.

5. Test Approach

5.1 Testing Types and Methodologies

Functional Testing:

- Verify the core gameplay mechanics, such as movement, game systems,npcs and mobs.
- Validate that the story concepts are accurately conveyed and integrated with the gameplay.
- Ensure the seamless functioning of interactions (movement,mechanics, etc.).
- Test the user interface (UI) for intuitive navigation and clarity.

Smoke Testing:

- Confirm that the game can start and shut down without issues.
- Check that the main menu, settings, and early levels are accessible.
- Verify basic interactions and game progression.

Performance Testing:

Assess the game's performance on various hardware configurations to ensure smooth operation and optimal frame rates.

Compatibility Testing:

Validate compatibility with different operating systems to guarantee a seamless experience across various platforms.

Accessibility Testing:

Ensure the game is accessible to players with alternative input methods and visual enhancements.

Playtesting:

Conduct iterative playtesting sessions with a diverse group of users to gather feedback on gameplay, engagement and overall experience.

5.2 Test Case Creation and Management

Test Case Creation:

- Gather requirements from game design documents and specifications.
- Identify key game features and functionalities to be tested.
- Decompose features into testable scenarios and create detailed test cases.
- Define expected outcomes and pass/fail criteria for each test case.
- Prioritize test cases based on risk and importance.

Test Suite Organization:

- Group test cases logically based on features or functionalities.
- Create separate test suites for different testing types (functional, smoke, regression).
- Utilize a test case management tool for organization and tracking.

Test Case Management:

- Track test case execution and results throughout testing phases.
- Update test cases as the game evolves.
- Maintain test case documentation for future reference and regression testing.

5.3 Utilization of Testing Tools

Compatibility Testing Tools:

Leverage compatibility testing tools to validate game compatibility across various hardware and software configurations.

Playtesting Tools:

Employ playtesting tools to record and analyze player behavior, gather feedback, and identify areas for improvement

6. Defect Management

6.1 Bug Reporting

A standardized bug reporting process is essential for effective defect management:

Bug Identification:
Testers will identify and document any encountered bugs or discrepancies during testing sessions.

Bug Reporting Form:
Testers will utilize a bug reporting form to provide detailed information about each bug, including steps to reproduce, severity level, and any relevant screenshots or videos.

Bug Repository:
Reported bugs will be stored in a centralized bug repository, such as Bugzilla for easy tracking and management.

6.2 Bug Resolution Process

A streamlined bug resolution process ensures timely and effective bug fixes:

Bug Prioritization:
Reported bugs will be prioritized based on their severity and impact on the game's functionality and overall experience.

Bug Assignment:
High-priority bugs will be assigned to developers for immediate attention and resolution.

Bug Fixing:
Developers will implement bug fixes and thoroughly test the affected areas to ensure regression issues are prevented.

Regression Testing:
Regression testing will be conducted to verify that bug fixes have not introduced any new issues.

Bug Closure:
Resolved bugs will be marked as closed in the bug repository and communicated to testers.

Communication:
Developers and testers will maintain open communication throughout the bug resolution process to ensure timely updates and issue resolution.

7. Conclusion

7.1 Summary

This Test Plan Report outlines a comprehensive approach to testing Astroni project, ensuring that the game meets the highest standards of quality and delivers an engaging, educational, and enjoyable experience for players. The testing strategy encompasses a variety of methodologies, tools, and processes to thoroughly evaluate all aspects of the game, from core gameplay mechanics to accessibility considerations. By adhering to this rigorous testing plan, we aim to deliver an exceptional game that seamlessly blends entertainment.

8. Appendix

A. References

- "Software Testing Fundamentals: A Comprehensive Guide for Beginners" by A.G. Stefanescu and I. Budimac
- "Testing Computer Software" by Cem Kaner, James Bach, Pettichord, and Elizabeth Hunt
- "Game Testing: A Practical Guide" by Ian Myre

B. Glossary of Terms

- **Functional Testing:** Verifies that software functions as intended and meets all requirements and specifications.
- **Smoke Testing:** A quick test to ensure that the basic functions of the software are working before more comprehensive testing is performed.
- **Regression Testing:** Verifies that changes made to the software have not introduced any new bugs.
- **Performance Testing:** Assesses the performance of the software under various workloads.