Hazard Analysis
SFWRENG 4G06

Team 9, dice_devs
John Popovici
Nigel Moses
Naishan Guo
Hemraj Bhatt
Isaac Giles

Table 1: Revision History

Date	**Developer(s)**	**Change**
2024-10-23	Hemraj	Added content to sections 1, 2, and 3
2024-10-23	Isaac Giles	Added content to sections 5 and 6
2024-10-25	Isaac Giles	Updated reflection section
2024-10-25	John Popovici	Fixed LaTeX compiling error and table format
...	...	...

Contents

[You are free to modify this template. —SS]

1 Introduction

Game system design is often perceived as straightforward because users interact primarily with the front end, unaware of the complexities that lie behind the scenes in the back end. In reality, it consists of multiple different components working together in order to create a seamless experience for the user.

As digital gaming continues to evolve, understanding these various components and their interplay is essential. Thus it is crucial to examine the potential challenges and requirements that may emerge within these components to improve the overall system.

[You can include your definition of what a hazard is here. Hemraj note: added vague intro and moved defination to its own section —SS]

2 Scope and Purpose of Hazard Analysis

[You should say what **loss** could be incurred because of the hazards. —SS]

The purpose of this document is to assess the potential hazards associated with the system under development. The ultimate goal is to implement strategies that either eliminate these hazards or reduce them to an acceptable level. To achieve this, the Failure Modes and Effects Analysis (FMEA) method was employed, which aided in systematically identifying and prioritizing hazards. A thorough analysis was conducted on various aspects of the system, including requirements, design, and code implementation.

The scope of this document is to identify possible hazards within the software components of the game system, including the game mechanics, user interface, and multiplayer functionalities. It aims to analyze the effects and causes of potential failures such as performance degradation and outright system failure. Through this, mitigation strategies, safety and security requirements for users were established. Importantly, the scope does not include any hardware components as the system is purely software based and any hardware hazards are not within the control of the developers.

3 Hazard Definition

Latex	**Definition**
System Hazard	A condition that could foreseeably cause or contribute to the system going down or loss of performance.
Risk	A measure that indicates the likelihood of a system hazard.

Table 2: Definitions of System Hazard and Risk

A hazard, in the context of this system, is defined as any property, software, or component that leads to reduced performance or complete system failure.

4 System Boundaries and Components

[Dividing the system into components will help you brainstorm the hazards. You shouldn't do a full design of the components, just get a feel for the major ones. For projects that involve hardware, the components will typically include each individual piece of hardware. If your software will have a database, or an important library, these are also potential components. —SS]

5 Critical Assumptions

- **Godot Stability**: The Godot game engine is assumed to be stable and function correctly.

6 Failure Mode and Effect Analysis

6.1 Hazards Considered Out of Scope

1. **Hardware-Specific Failures**
    - Issues related to hardware malfunctions such as GPU or CPU overheating, RAM failures, or hard drive corruption.
    - *Rationale*: The FMEA table is focused on software development within the Godot game engine, and hardware reliability is typically managed by the user's computer environment.
2. **Operating System Crashes or Instability**
    - Operating system crashes, updates, or security vulnerabilities that interrupt game sessions.
    - *Rationale*: These are dependent on the player's system and not directly related to the game's software development or behavior.
3. **Network Infrastructure Failures**
    - Failures due to external network outages, router malfunctions, or ISP-level disruptions.
    - *Rationale*: These are beyond the game's control and depend on the player's internet setup or service provider.
4. **Third-Party Library Bugs or Vulnerabilities**
    - Bugs or security vulnerabilities in third-party libraries or plugins used within the Godot engine.
    - *Rationale*: While the game relies on third-party tools, the FMEA focuses on bugs and issues within the game code itself, not third-party dependencies.
5. **Player Misuse or Exploits**
    - Players intentionally trying to exploit the game, cheat, or use unauthorized modifications.

- *Rationale*: Handling intentional misuse or hacking is outside the game's core development, and managing these issues requires external anti-cheat measures or monitoring.

6. **Data Privacy and Security Breaches**

   - Unauthorized data access, or privacy breaches.
   - *Rationale*: The FMEA focuses on in-game functionalities like dice rolls, AI, and scoring. Data privacy concerns involve external security practices and infrastructure, which are beyond the game's core software behavior.

7. **Non-Game Software Interference**

   - Interference from other software running on the user's machine, like antivirus programs or system background tasks.
   - *Rationale*: These external software influences are outside the game's scope of control and would be handled by system administrators or users.

6.2 FMEA Table

Text FEMA table allows for us to quantify values related to possible failures and document each failure's severity, likelihood, and detectability. These values can help guide our countermeasures and priorities in risk mitigation.

Table 3: FEMA Table

Function	**Failures**	**Unacceptable Event**	**Severity of Failure (0-10)**	**Cause of Failure**	**Likelihood of Occurrence (0-10)**	**Recommended Action**	**Likelihood of Failure Detection (0-10)**
Dice Roll Simulation	Physics misbehaves	Unrealistic dice behavior	8	Physics engine glitch	6	Refine physics settings; improve collision detection	4
Score Calculation	Incorrect scoring	Inaccurate score computation	9	Logic error in scoring algorithm	4	Unit test scoring algorithms thoroughly	6
Player vs. Computer AI	Poor AI decisions	Computer opponent is too easy/unpredictable	3	Sub-optimal AI strategy	6	Refine AI strategy based on probability analysis	9
Multiplayer Functionality	Connection loss	Player disconnects mid-game	7	Network instability	5	Implement reconnect feature; improve connection stability	5
Dice Rendering	Dice not visible	Players cannot see the dice clearly	3	Rendering glitch	3	Reduce 3D model poly counts for best rendering reliability; ensure camera angles cover dice	3

Function	**Failures**	**Unacceptable Event**	**Severity of Failure (0-10)**	**Cause of Failure**	**Likelihood of Occurrence (0-10)**	**Recommended Action**	**Likelihood of Failure Detection (0-10)**
User Interface	Missing or confusing UI	Players are confused by the interface	7	Inadequate UI design	5	Conduct user testing; iterate on UI design	8
Camera Control	Unclear view of the board	Players can't properly view game elements	6	Inadequate camera angle logic	5	Allow manual camera adjustment; improve auto camera control	6
Audio Feedback	Missing or incorrect sounds	No sound feedback for player actions	4	Sound trigger event missed	5	Ensure audio events are linked to game actions with low latency	6
General Game Stability	Unexpected crashes during gameplay	Game session terminates abruptly	9	Memory leaks, unhandled exceptions, or rendering overload	4	Conduct stress tests; improve error handling and resource management	9
Scoreboard Display	Incorrect scores or missing player data on the scoreboard	Confusion over game results	6	Display update not synchronized with scoring logic	3	Ensure that scoreboard updates are triggered accurately, add validation	4

7 Safety and Security Requirements

[Newly discovered requirements. These should also be added to the SRS. (A rationale design process how and why to fake it.) —SS]

8 Roadmap

[Which safety requirements will be implemented as part of the capstone timeline? Which requirements will be implemented in the future? —SS]

Appendix — Reflection

[Not required for CAS 741 —SS]

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

   - Team members communicated often and effectively. - Isaac G.
   - Code changes were handled well without merge conflicts. - Isaac G.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   - With many midterms for everyone finding time to meet and work was a challenge. We were still able to compare shedules and find meeting times that we could all make, and everyone communicated well to let team members know when they would have time to get to their part(s) of the deliverable done. - Isaac G.

3. Which of your listed risks had your team thought of before this deliverable, and which did you think of while doing this deliverable? For the latter ones (ones you thought of while doing the Hazard Analysis), how did they come about?

4. Other than the risk of physical harm (some projects may not have any appreciable risks of this form), list at least 2 other types of risk in software products. Why are they important to consider?