**Comprehensive Risk Analysis**

Risk analysis is an integral part of project planning that involves identifying potential hazards and taking them into account. For Shadow Walker, we have identified nine major risk categories, each of which is analyzed below with detailed explanations and corresponding mitigation strategies.

1 Cost Risk

Description:  
Cost risk is the possibility of unforeseen expenses that could disrupt the project budget. In this project, the initial decision to use free tools like VS Code and SFML minimizes costs. However, there is always a potential need for advanced tools or software licenses; however, as of today, it does not seem that the project will encounter this issue.

Potential Impacts:

* Budget Overruns: The need for upgraded tools or additional software licenses could lead to unplanned expenses.

Mitigation Strategies:

* Budget Monitoring: Regularly track expenses and maintain a detailed budget log.
* Research Alternatives: Continuously explore free or lower-cost alternatives that could meet advanced technical requirements.

**2 Schedule Risk**

**Description:**  
Schedule risk refers to possible delays that can happen due to unexpected problems, such as difficulties in debugging, team members having other responsibilities, or deadlines from other courses happening at the same time. Since the project must be completed in six weeks, even small delays in one part of the work could cause major setbacks.

**Potential Impacts:**

* **Missed Deadlines:** If important tasks, such as debugging or final testing, take longer than expected, the entire project schedule could be affected.
* **Lower Quality:** Rushing to meet deadlines may lead to less thorough testing or incomplete features, which can compromise the overall quality of the project.

**Mitigation Strategies:**

* **Careful Planning:** Create a schedule that includes extra time for unexpected delays.
* **Regular Check-ins:** Have weekly meetings to track progress, spot potential delays early, and adjust the schedule if needed.
* **Task Prioritization:** Work on the most important features first and follow a "minimum viable product" approach to make sure the main parts of the project are finished on time.

3 Performance Risk

Description:

Performance risk refers to the game's ability to run smoothly, particularly since it relies on real-time lighting and shadow effects. Problems can arise if the game is played on older computers or if complex graphics are not properly optimized.

Potential Impacts:

Lag and Slow Gameplay: If the game is not optimized, it might run slowly, making the experience frustrating for players.

System Crashes: If the game requires too much processing power, it may cause crashes or unexpected errors, especially when advanced effects are used.

Mitigation Strategies:

Early Performance Testing: Start testing performance early to find and fix any issues that slow down the game.

Efficient Optimization: Use SFML’s built-in functions and follow best practices to make the game run smoothly.

Hardware Compatibility Testing: Test the game on different computers to make sure it works well on a variety of systems.

4 Operational Risk

Description:

Operational risk includes challenges that might affect the daily progress of the project, such as accidentally losing data, differences in how team members set up their development environments, or availability issues among team members.

Potential Impacts:

Data Loss: Important files or code might be accidentally deleted, causing major delays.

Inconsistent Development Environments: If team members use different software settings, it can create problems when combining their work.

Work Delays: If one area of the project, such as code integration, gets delayed, it can slow down the entire development process.

Mitigation Strategies:

Version Control: Use a reliable system like GitHub to save different versions of the code and prevent data loss.

Standardized Setups: Make sure all team members use the same development environment and provide clear setup instructions.

Frequent Backups: Regularly save project files in a secure location to prevent data loss.

5 Technology Risk

Description:

Technology risk involves potential problems with the tools and software used for development. Since the team relies on SFML and some members are still learning its advanced features, there could be challenges.

Potential Impacts:

Slow Learning Process: Team members who are not familiar with SFML may take longer to complete tasks.

Integration Issues: Some advanced SFML features may be difficult to use correctly, leading to unexpected errors or compatibility problems.

Mitigation Strategies:

Training Sessions: Set aside time for team members to learn SFML through tutorials and practice.

Early Prototyping: Create small test versions of key features early to make sure they work before adding them to the main project.

Use Online Resources: Rely on forums, tutorials, and official documentation to quickly solve problems.

6 Communication Risk

Description:

Good communication is essential for teamwork. Communication risk occurs when tasks are not clearly assigned, messages are delayed, or important details are not documented.

Potential Impacts:

Confusion and Mistakes: Poor communication can lead to duplicated work or missed deadlines.

Slower Problem-Solving: If team members don’t communicate efficiently, it may take longer to fix issues.

Lack of Coordination: Without regular updates, team members might work separately, making the final project feel disconnected.

Mitigation Strategies:

Use Communication Tools: Platforms like Slack or Microsoft Teams can help keep everyone informed.

Regular Meetings: Schedule short weekly updates and detailed progress meetings to stay on track.

Shared Documentation: Keep a central document with notes from meetings, task assignments, and project guidelines.

7 Scope Creep Risk

Description:

Scope creep happens when a project grows beyond its original plan. In a creative project like Shadow Walker, it is tempting to add more features, improve animations, or expand levels, but this could make it difficult to complete the game on time.

Potential Impacts:

Project Delays: Adding extra features can push back the deadline for completing the core game.

Increased Complexity: A bigger project is harder to manage and test.

Overuse of Resources: Extra work requires more time and effort, which might overwhelm the team.

Mitigation Strategies:

Define Scope Clearly: Decide on the essential features at the start and make sure the team agrees.

Control Changes: Use a formal process to review and approve any new additions.

Focus on Core Features: Prioritize the most important parts of the game and save extra features for later updates.

8 Security Risk

Description:

Security risk involves problems that could affect the game’s stability and reliability. Even though Shadow Walker is a prototype, security issues could lead to crashes, memory leaks, or other failures.

Potential Impacts:

Game Instability: Errors or memory problems could cause crashes during gameplay.

Potential Exploits: If the game were expanded in the future, security flaws could allow unauthorized access or manipulation.

Mitigation Strategies:

Write Secure Code: Follow best practices to prevent vulnerabilities.

Review Code Regularly: Check the code often and use automated tools to find security issues.

Thorough Testing: Perform detailed tests, including stress tests and security checks, to ensure stability.

9 Skills Resource Risk

Description:

Skills resource risk refers to the experience level of the team, especially with new technologies like SFML and advanced C++. If some team members are less familiar with these, development could take longer.

Potential Impacts:

Slower Development: Inexperienced team members might need extra time to complete tasks.

More Debugging Time: Less experience can lead to more errors, which require extra time to fix.

Uneven Work Distribution: If some members struggle with certain tasks, others may need to take on extra work.

Mitigation Strategies:

Encourage Knowledge Sharing: Use pair programming and mentoring to help team members learn from each other.

Dedicated Training Time: Allocate time for learning SFML and advanced C++ techniques.

Use External Learning Resources: Take advantage of online courses, tutorials, and community forums to improve skills.

DIAGRAMS:

A screenshot of a graph

AI-generated content may be incorrect.

A yellow and green box

AI-generated content may be incorrect.

A screenshot of a graph

AI-generated content may be incorrect.