5. Risk Assessment and Mitigation

Our team followed a systematic management process divided into 4 stages: Risk Identification, Risk Analysis, Risk Planning, and Risk Monitoring. This ensured we had an effective and secure plan.

1. Risk Identification: we categorised risks into three primary areas: project risks, product risks and business risks. Project risks affect the project timeline or resources. The product risks affects the quality of the game. Business risks affect the team developing the software. By anticipating potential setbacks in each of these areas, we ensured a holistic view of the risks that could disrupt the project.
2. Risk Analysis: each risk was assigned a likelihood and severity. This helped us realise which needed immediate attention and also eliminate the ones with very low likelihood and severity since they don't really affect the project. For example, timeline slippage was identified as a high-likelihood and moderate-severity risk, which made it a top priority for mitigation efforts.
3. Risk Planning: we created the risk register, where we came up with avoidance strategies for the different risks and also mitigation strategies. We also identified team members responsible for addressing specific risks depending on the roles they play in the project. We allocated at least two people per section to ensure we have a low bus factor.
4. Risk Monitoring: We implemented a continuous risk monitoring process to identify any changes in risk status. For instance, in case of engine limitations (like Java malfunctioning), we resolved the issue by reinstalling Java. This proactive monitoring ensured that risks were addressed before they escalated. We also worked closely during our team meetings and practicals,reporting what is going on in each section of the project to avoid redundancy. In addition, we kept all the files shared on google drive or github so everyone has access to them and they don't get lost.

<u>Risk Register Format:</u>

Our risk register is a tabular document that includes the following columns for each risk:

- ID: a unique number assigned to each risk for tracking purposes.
- Type: the category of risk (Project, Product, or Business).
- Description: a brief explanation of the risk.
- Likelihood: an estimate of how likely is the risk to occur (low, moderate or high).
- Severity: an estimate of the impact it would have on the project (low, moderate or high).
- Mitigation: steps to take to reduce or manage the risk.
- Owner: the person or persons in charge of mitigating the risk depending on roles.

<u>Risks explained</u>

1. Computer breaks - might cause problems if work is not is not saved or cannot be accessed. Most work is saved online with github being used for code commits.
2. Timeline slippage - certain parts of the project may be worked on too heavily, leaving other areas to be rushed or incomplete.
3. Requirements not met / with errors - there is a chance requirements may not be met in allocated time, or they are incorrectly made.
4. Bugs in code - Code may be inefficient with unforeseen problems, errors or bugs.

5. Having engine limitations - Ensure the engine can deal with our needs and can be run with few issues on our systems.
6. Having unoptimised code - Code may not be well optimised, may cause issues such as memory leaks and cause the game to be too slow.
7. Teammates getting sick - Teammates getting sick may slow down progress on their assigned roles, cause problems with team cohesion etc.
8. Teammates disagreeing - Could lead to issues in communication and mismatch between goals and deliverables.
9. Miscommunication between teammates - Can lead to further mismatch between team members and work may overlap leading to wasting time which can cause further issues and risks.
10. Competition between teams - As teams are working towards the same goals competition may take place if cross team discussions are had, may lead to ideas being given to other teams or competition leading to loss of focus on own team goals.

Risk Register

ID	Type	Description	Likelihood	Severity	Mitigation	Owner
R1	Project	A computer from the team breaks/stops working.	M	M	Use a computer from the university labs/get the computer fixed. Save work often.	Entire team.
R2	Project	Timeline slippage, time is wasted.	H	M	Try to plan ahead on timings, ask for assistance	Leader
R3	Project	Requirements are not met or with errors.	L	H	Make sure working to strengths. Check requirements after added.	Leader
R4	Product	Getting bugs in the code.	H	M	Check the code and figure where the bug is.	Joel, Euan
R5	Product	Having engine limitations. For example: a computer cannot	H	M	Try using another computer. Or look	Entire team

		run a program because of its system.			what options you have for your system.	
R6	Product	Having unoptimised code.	H	M	Check code in pull requests, ensure code is working without memory leaks or similar.	Joel, Euan
R7	Business	Getting ill. For example: catching a cold or getting an injury.	H	L	Try to work as long as you are capable. Delegating your work to a teammate, ensure work is available online.	Entire team
R8	Business	Teammates may have disagreements.	M	M	Ensure people feel listened to, try to find solutions between members. Ask for help from lecturers.	Entire team
R9	Business	Having miscommunications between the teammates	L	M	Sit down and listen carefully to each others opinions, have a third person as a moderator.	Entire team
R10	Business	Competition between teams	M	H	Try to keep work within team, do not discuss	Entire team

					methods with other teams, try to ensure work is needed not done in competition.	