Sea of Geese

Risk Assessment and Mitigation

To begin any project, software related or otherwise, it's vital we consider the risks involved, the risk format I have chosen is founded upon the four basic principles outlined by Ian Sommerville [1]: Risk Identification, Analysis, Decomposition and Reduction, or in this case mitigation, we have also included sections not originally defined, Type and Damage Extent.

- ID – Identification – An Identification number for future reference.
- Risk – Identification – A description describing the risk itself.
- Type – Project, Product or Business, to segment our problems.
- Likelihood – Analysis – The probability of a risk occurring.
- Impact – Analysis – The damage caused by a risk occurring.
- Damage Extent – The overall

We also used the three types of risk explained by Sommerville [1] in page 596: Project risks, which affects team members, the schedule, or resources of the project and could lead to issues in development; Product risks, which are issues involving the quality or performance of the product or the tools used to develop the product; and Business risks, which are risks that affect the company and their ability to sell the finished product.

Impact is recorded for each risk as one of five options: Minor, trivial, moderate, major, and severe, with minor being the least impact and Severe being the most. Likelihood is also recorded as one of five options: highly unlikely, unlikely, moderate, likely, and highly likely.

We used the same threat level from [1] page 315-316: Intolerable, ALARP and acceptable, with ALARP being "As low as reasonably possible" Meaning that it is either so unlikely or not serious at all and so it's seemed acceptable but only just, this acts as a good middle ground. By using only three different severity options, it's much clearer to see which risks we absolutely must avoid, which can be ignored and those we should keep an eye on but not waste too much time on.

ID	Risk	Type	Likelihood	Impact	Damage Extent	Mitigation
1	A group member becomes unavailable or unable to complete their share of the work.	Project	Highly Likely	Severe	Intolerable	This can be mitigated by assigning sections to multiple people so there is still someone to finish it, and managing our time so that there is time left before the deadline to complete a missing section if necessary.
2	Important documentation is lost due to data corruption / lost USB stick / etc...	Product	Unlikely	Harmful	ALARP	Store all data in multiple places, including Google Drive, so that there's a backup in case a copy is lost.
3	Requirements change during development, meaning that some work already done is now obsolete and other functions need implementing instead.	Product	Likely	Severe	ALARP	Keep in contact with the customer throughout development so we can check in with them that what we are doing still matches the requirements.
4	Underestimated project leads to issues completing everything within the given time before the deadline.	Project	Moderate	Harmful	Acceptable	Divide up work between people to be done with time to spare, so that we have time before the deadline to add anything missing later.
5	Game not as popular as expected, leading to more work for little reward.	Business	Moderate	Harmful	Intolerable	Get requirements from customers so that we know in advance what sort of things they are looking for in a game, so that the game lives up to their expectations.

6	Expected optional task to be simpler, but could not learn it in required time, wasting groups time.	Project	Moderate	Trivial	Acceptable	Depending on how our team decides to plan, the risk changes, by deciding early on that it will likely be impossible we can avoid it.
7	We're unable to implement one of the core requirements for our game.	Product	Unlikely	Severe	Intolerable	We assign several groups, 2 or more people working on any core feature, that makes debugging easier and could drastically save time.
8	Insufficient planning for a task, with several people working on it, they may imagine it different ways, creating problems with ill-fitting code.	Product	Moderate	Harmful	ALARP	Detailed planning (Well defined task), This is a blanket risk, covering many possible problems including diagrams, plan revision and communication also support mitigation here.
9	Someone tries to force push using git, overwriting project history and deleting past work	Project	Unlikely	Severe	ALARP	No one force pushes, it shouldn't be necessary and especially when someone is inexperienced.
10	Two people pull and work on the same document at the same time, one pushing after another and not realising the document was updated	Project	Moderate	Moderate	ALARP	Communication, with a quick update in chat if you decide to work on any piece of code, or if you have a meeting at the start of the day and state you will be working on it, then you can coordinate when each person does what.

11	Inadequate advertising	Business	Moderate	Trivial	Acceptable	This could lead to no one selecting it as we move forward, leaving it undeveloped if we wanted to move back to it.
12	Unpopular design choices or content (e.g. Controversial story or unpopular mechanics)	Business	Highly Unlikely	Severe	ALARP	Overview of these decisions within the game by full group.
13	Unknown software problem causes none of the code to function	Product	Unlikely	Harmful	Intolerable	Possibly no counter assuming no one in the group can find the problem, to avoid this before it has become we must use modularisation, testing each module as we progress.

Bibliography

[1] Ian Sommerville, “Software Engineering 9th Edition”, Pages: 313, 315-316, 596. March 2010.