Risk Assessment and Mitigation

Risk Elicitation Process

There are various methods that big companies use to identify potential risks. One of those methods is analysing past projects and what kind of risks occurred that were not identified beforehand. That way, companies can maintain their own risk register. Unfortunately, we did not have any past projects nor a risk register before, thus we had to identify another method that could possibly yield the same results. Therefore, we arrived at the conclusion that brainstorming might be a suitable option.

Firstly, all of the team members were present during the brainstorming session. We followed the basic rules of brainstorming such as that “quantity is more important than quality”, “build on the ideas put forward by others” and “every idea has equal worth”. Secondly, we presented our ideas for potential risks in turn. Again, all of the ideas presented during the session were equally good at this stage, we did not intend to filter them. Moreover, since it is very hard to determine the length of a brainstorming session in advance, we decided to end the session when no team member had any new ideas.

The next stage of the risk elicitation process was narrowing down the list we created, evaluating those risks based on their impact and the possibility of occurrence, and creating a mitigation plan for each of them.

Risk Format

This section of the documentation will report on the possible risks that could affect the development and success of the game. The format used for this risk register is a slightly modified version of the risk identification, analysis and planning described in *‘Software Engineering’* by Sommerville [1], and has been chosen because the literature is well respected and the format is not overly complicated for the type of product being developed.

In the book [1], the format described uses many different types of risk, however, we felt that so many classifications we not necessary for the small number of risks identified in our project. Our format categorises the risks into three types of risk, Product, Project and Business. In general, product risks present issues that could affect the final game, project risks affect the implementation of the product, and business risks affect the management and finances of the product once released to the public.

The register also includes an analysis of the likelihood and severity of each risk. These measures have been added because they allow the team to identify which risks need to most attention and the most effort put into their prevention or mitigation. The likelihoods are separated into five levels; Very Low, Low, Moderate, High and Very High, which represent specific probability ranges as described in Table.1 below.

Likelihood	Very Low	Low	Moderate	High	Very High
Probabilities	< 10%	10-25%	25-50%	50-75%	> 75%

Table 1. A table linking the likelihood categories with the probability ranges they represent. These ranges follow the suggested levels stated by Sommerville, pp.599 [1].

Additionally, the severity ratings describe the scale of the risk’s impact and how badly it affects the Project, Product or Business. Unlike the likelihood values, these ratings are only separated into three categories, High, Moderate and low, because the software is reasonably low risk, so there is no need for higher severity ratings. High severity risks will cause significant delays to the project and a large impact on the final product. Moderate risks will cause some delays to the project and Low severity risks have little impact on the product as a whole and the effects can be easily mitigated.

Furthermore, the second table in the register (Table 3), contains a description of the steps needed to be taken to reduce the impact of the risk if it were to happen or reduce the chance of it happening. However, there is a larger focus on the mitigation strategies instead of prevention strategies because there will always be a chance that the risks will occur, even if every step has been taken to prevent it.

The second table in the register also states the owner of each risk. The owner of a risk is responsible for monitoring it and re-evaluating the likelihood and severity measures at a later stage in the development process. The team felt this was an important feature to add to the register because it ensures the risks are not forgotten and the responsibility for the risks is shared throughout the team.

Risk Register

ID	Type	Description	Likelihood	Severity
1	Product	The customer changes the requirements during the development of the product, resulting in all or part of the current implementation to be changed.	Moderate	High
2	Project	A group member responsible for implementing a system feature becomes unable to work and no one else has access to their implementation.	Low	High
3	Business	Legal action taken by users who experience eye strain from playing the game.	Very Low	High
4	Business	Legal action taken by users who experience Carpal tunnel from playing the game.	Very Low	High
5	Business	Risk of causing Epilepsy with flashing images on the screen.	Very Low	High
6	Project	Fault with the GitHub servers meaning we lose source code.	Very Low	High
7	Project	The team's GitHub repository experiences a security breach and the code is lost.	Very Low	High
8	Business	Threat of competition for other similar games so lack of player base.	Very High	Moderate
9	Product	The customer changes the requirements during the development of the product, causing additional features to be added to the game.	Very High	Moderate
10	Project	One of the group members becomes unable to work on the development of the product during the implementation phase.	High	Moderate
11	Project	A requirement that was not decided in the planning phases is later discovered and then the customers opinion is required.	High	Moderate
12	Project	Lack of experience using C# and Unity for team members results in missing deadlines and/or poor code quality.	High	Moderate
13	Project	The time allocated to implement the product is unrealistic and so the game does not get finished on time.	High	Moderate
14	Product	Not all of the requirements are met.	Moderate	Moderate
15	Product	The product does not meet the performance requirements, i.e. the game runs below 30fps on the University of York computers.	Very Low	Moderate
16	Product	The game takes too long from start to end, so players lose interest.	Low	Low

17	Business	Unity requires payment if the product is profitable at more than £100k in revenue.	Very Low	Low

Table 2. The first section of the register, providing a list of possible risks, along with the type, likelihood and severity of each.

ID	Mitigation	Owner
1	The code will be written in a manageable way, using classes and functions. This means that if code has to be changed, it has less chance of affecting other areas of the software.	Haydn
2	Make sure that everyone has access to the most recent version of the source code by using GitHub repositories.	Tom
3	Include a reminder to take a break from playing the game.	Steve
4	Include a reminder to take a break from playing the game.	Kate
5	Make sure that the development team does not include any flashing colours or animations in the game. Test the graphical user interface to reduce the risk of visual bugs, such as flickering.	James
6	Organise the team so that at least one member has an up to date version of the source code stored locally.	Haydn
7	Make sure that the GitHub accounts have strong passwords and ensure at least one member of the team has an up to date version of the source code stored locally.	Andrius
8	To mitigate this risk, the customer could run advertisements to encourage people to play the game and ensure the game is interesting and engaging.	Steve
9	If this occurs, the team will assign more team members to the development of the new features.	Tom
10	Ensure that all team members are familiar with the code, so that if this occurs another team member can continue working on the code without wasting a lot of time understanding the implementation.	Kate
11	To reduce the delays caused by this, the development team should have a reliable to contact the customer whenever an issue arises.	James
12	If a team member is struggling with C# or Unity, then a more experienced team member will help. Also, the use of online tutorials and documentation should be used if needed.	Andrius
13	If there is not enough time to finish the implementation on time, then the features with the highest priority *(see requirements)* should be implemented first. Also, regular meetings should take place during the implementation phase to review progress.	Kate
14	Regularly revisit requirements as well as keep in touch with customer and show progress to see if requirements change and ensure requirements are met to a satisfactory standard	Tom
15	Make sure that the code is maintainable and easy to modify so that if changes need to be implemented to improve the game’s performance, then there will be a minimal risk of introducing new bugs.	Steve

16	Include an option within the game’s settings to add a time limit to a player’s turn.	James
17	If the game earns £100K or more in revenue then the group must obtain a pro license for Unity	Andrius

Table 3. The second section of the register, providing a description of the mitigation strategies and ownership of each risk.

Bibliography:

[1] I. Sommerville, “Risk Management,” in *Software Engineering Ninth Edition*. Addison Wesley, 2010, pp. 595-602.