Risk Assessment

While developing our product to meet the requirements of our customer, there will be various risks that we will need to consider and mitigate their impact to achieve the desired specifications. We have created a comprehensive table documenting the possible complications that could occur during the development process or deployment of the final product and our proposed solutions to reduce the impact of such problems. We initially compiled these risks through a brainstorming session, before building on our initial ideas using relevant websites and papers.

We have categorised the potential risks into three types: Project, Product and Business risks. Project risks are risks which could impact the deadlines we have planned, or the resources we have decided to use. Product risks are risks which will affect the final product, either by resulting in requirements not being met or the gameplay not being as smooth as it could be. This can encompass all of the potential problems that may arise in the final version of the product, from problems during development to larger, more abstract problems about aspects of the game. Business risks are risks which affect the stakeholders who we are building the game for, resulting in potentially drastic changes in requirements. We believe this to be the best way to categorise our risks as they will all fall into one of these three categories.

We have also categorised the risks into three levels depending on their impact on the product and the customer - High, Moderate and Low.

High impact – A problem that would actively impact the customer experience on a large scale and seriously hamper the development or deployment of the product. Risks of this kind would render the final product unacceptable to the stakeholders.

Moderate impact – A problem that would result in disruptions to core gameplay to the extent that it would not meet the specifications given by the stakeholders. It would be an inconvenience to the customer but would result in a game which is still playable.

Low impact – A minor problem that would only result in minimal inconvenience to the customer. It may have a cosmetic effect to the game but all core features specified by the stakeholders would still be present and functional.

Using this system, we believe that we are able to categorise, in enough detail, all of the risks that could potentially occur and identify an accurate solution to mitigate each risk.

Furthermore, we decided to categorise the potential frequencies of the risks occurring in three ways; high, moderate and low frequency. This allows us to understand which risks we need to be the most aware of during development. As our project is reasonably small and non-critical, we believe that using three frequency categories will be sufficient.

We have agreed on a risk reviewing plan which involves re-assessing the likelihood and severity of risks at two-week intervals. We have assigned an ‘owner’ to each risk we identified; this person will perform the re-assessment for their given risks and report the status to the rest of the team. We decided on the owners of each risk by conducting a group meeting in which team members were assigned specific risks, and we ensured that ownership of risks was spread out evenly between team members.

Product Risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Impact** | **Frequency** | **Mitigation** | **Owner** |
| R1.1 | Variables having non-intuitive names | Moderate – misused variables could be assigned incorrect data, leading to incorrect properties in the game | Low | Write appropriate docstrings and always label variables appropriately. | Peter |
| R1.2 | Using discontinued libraries. | Moderate – Appropriate, up-to-date documentation may not be available and would lead to conflicts with other code. | Low | Avoid using discontinued libraries if possible, otherwise minimise use of them during development. | Jack |
| R1.3 | Assigning variables of different scope the same names | Moderate - a global variable could be affected when it shouldn’t be, leading to incorrect values in the game | Low | Ensure all team members are aware of the names of global variables and parameters so that these names are not repeated. | Max |
| R1.4 | A file which has data which needs to be read could be moved | High - if the program attempts to read a non-existent file from a location, it would cause the program to crash. | Low | Ensure file locations are updated when a file is moved, and use appropriate error-catching functions to minimise impact. | Peter |
| R1.5 | The program or major sections of code being deleted or becoming corrupt | High - this would prevent us from delivering any product to the customer | Moderate | Use version control systems such as Git to ensure that frequent backups are made. | Yuqing |
| R1.6 | Java may not support a GUI on mobile or computer | High - the game would be unplayable on the required systems | Low | Test early versions of the code regularly to ensure the game is playable. | James |

Project Risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Impact** | **Frequency** | **Mitigation** | **Owner** |
| R2.1 | The requirements of our product may change | High - if the code has been developed for one set of requirements then the entire game may need re-programming | Moderate | Ensure we receive constant and frequent feedback from customers when developing | William |
| R2.2 | Team members may leave the course or fall ill before the project is completed | Moderate - this could result in incomplete code which other team members may not know how to complete | Low | Ensure all code is properly documented with meaningful identifiers | Jack |
| R2.3 | Team members may not be able to code the required functions | Moderate - certain functions cannot be implemented if team members cannot implement them | Moderate | Ensure all team members have sufficient Java skills and practise before commencing | Tom |
| R2.4 | Certain requirements may be overlooked | Moderate - if requirements are not met then the stakeholder may reject the project | Moderate | Ensure all team members fully understand all requirements, and that these are central to the development of the game | William |

Business Risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Description** | **Impact** | **Frequency** | **Mitigation** | **Owner** |
| R3.1 | Java may become obsolete | High - if the code has been developed in Java then the entire program may need to be re-written | Low | Ensure we are aware of the advances in current technology and change language as soon as our chosen language appears to be obsolete. | Tom |
| R3.2 | Government policy may lead to our game not being legal | High - if it is not legal to produce certain parts of the game then it cannot be released | Low | Keep abreast with any potential changes to government policy, and ensure that plans for creating code or using images which do not breach government policy are in back-up. | Yuqing |
| R3.3 | A similar product may appear on the market | Moderate - this may result in the programming of a ‘copy’ game that no-one wants | Moderate | Monitor any potential competition and maintain communication with the stakeholder, anticipating major changes in requirements. | James |