Introduction to the Risk Assessment

We have made a few changes to our risk assessment from the document that we inherited, as laid out below:

- We have decided to *score* each risk by two factors rather than just one. We will score the risks individually by **likelihood** and **impact** as we feel like categorising a risk by one factor may be quite vague (does a risk labelled *high* signify a risk that will have a large impact, or one which is likely, or both, or a combination?) and allows us to deal with it more appropriately (it may be more important for some risks to concentrate on mitigating the likelihood more than impact, or vice versa).
- We have *added* a new risk category entitled 'Product Risks'. This is explained in the risk categories below.
- We have *removed* a couple of the risks from the table:
  - *Risk 1* from the previous document, as there is little risk now with choosing a programming language; having used it for Assessment 2 we have programmed it before and had chance to examine the new game's code and programming structure, and choose the language accordingly.
  - *Risk 9* from the previous document as we have now used the relevant software tools for several weeks and agreed to use it again, with the experience required for further use (or we would have changed it)
- A few of the other risks have been marginally amended where appropriate.
- We have *added* a new attribute to each risk, detailing the individual(s) responsible for dealing with the risk. We have done this as we feel risks need to be attributed to individuals or they are not given necessary attention (*if everyone is responsible for a risk, nobody is actually accountable to it*).
- We have *added* more risks, justified in our *identification* explanation below.
- Some of the risks have been *moved* into different categories or *re-worded* (some are too detailed).

Risk Categories

- **Technology** - includes the programming language, programming software or hardware used.
- **People / Team** - deal with issues pertaining to the people working on the software project.
- **Tools** - detail information about other non-development tools the software uses (see *Methods, Tools*)
- **Requirements** - relate to the specification, elicitation and following of project requirements.
- **Process and Planning** - risks which arise from the team performance, planning and SCRUM.
- **Product** - these are risks which relate to the ("commercial") interest and success of our product.

Identification of our risks

To identify our risk, we studied numerous existing risk tables[1], strategies[2] and papers[3] which explained to us the types, and examples of risks that would occur throughout a software development project. We then eliminated and selected risks based on the type, likelihood and severity of risks pertaining to a small software project of this nature. For example, we did not base our risks on Boehm's top-ten list of risk items[4] as it is valid for a business software project and thus includes many risks on personnel, financial costs and contracting which are superfluous for our project.

Our risk management process is a continuous one and not one which is only checked at infrequent intervals. During team meetings at least once a week we check the list of risks and ensure that the relevant action is being taken. The most important part of managing our risks is that we discuss and implement mitigation regularly[5], to ensure risks do not become reality; and we detect when the risks are occurring so we can plan for this. For example, if the project becomes behind schedule it is important that progress and due dates are checked at frequent intervals so as soon as it seems the project may be running behind schedule, the relevant action can be taken to avoid it from happening / bring it back up to speed as soon as possible.

How are risks are scored

Risks are rated on a 1-4 scale for likelihood (where 1 is unlikely to happen and 4 is almost guaranteed), and 1-4 for severity (where 1 bears little impact and 4 bears massive impact). They are then combined by multiplying the scores together and ranked as such, in the table to the right.

IMPACT \ LIKELIHOOD	1	2	3	4
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

Bibliography

[1]H. Hoodat and H. Rashidi, "Classification and Analysis of Risks in Software Engineering", International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 3, no. 9, 2009.
[2]B. Cukic, "Software Risk Analysis and Software Risk Analysis and Management", West Virginia University, 2011. [Online]. Available: http://www.csee.wvu.edu/~cukic/CS430/Risk.pdf. [Accessed: 04- Feb- 2016].
[3]D. Verdon and G. McGraw, "Risk analysis in software design", IEEE Security and Privacy Magazine, vol. 2, no. 4, pp. 79-84, 2004.
[4]H. Schaefer, "Software Risk Management: A Calculated Gamble", Oredev, 2006. [Online]. Available: http://archive.oredev.org/download/18.5bd7fa0510edb4a8ce4800019064/1385353971845/Hans_Schaefer_-_Workshop_Risk_Based_Testing_3.pdf. [Accessed: 08- Feb- 2016].
[5]I. Sommerville, Software engineering. Harlow, England: Addison-Wesley, 2000.

		Likelihood and Impact				
ID	Description of Risk	Likelihood	Impact	Risk Score	Mitigation	Risk Owner
Technology and Development Risks						
1	Implementation of the specification is more difficult than expected.	2	4	8	There is a plethora of help resources online, in the implementation program and on the code library. Consult this if in need of help.	HT
2	Code, software or libraries used contain bugs.	2	4	8	In order to reduce the time wasted on finding bugs, the code should be kept as simple and clean as possible, with good documentation of code, use existing libraries where possible as well as keep the code modularised.	HT
3	The architecture is too complex to be implemented	1	4	4	In order to prevent this from happening, the architecture should be designed so that it follows the standard format, so that all team members are familiar with it.	HT, TP
4	Lack of skill developing the project	2	3	6	Team members carrying out implementation to share knowledge to help each other and consult the large amount of online help.	HT, RE
5	Code is created which cannot be understood or edited at a later date	3	2	6	Implementation team members must keep their code well documented **during** implementation, not as a post-process. This also means multiple team members can work on the same thing.	HT, RE
People Risks						
6	Unplanned absence of one or more team members in the intermediary project stage.	4	3	12	The team's policy is that all current work being used is uploaded regularly (or hosted on) the Google Drive to ensure one member does not have exclusive resource access.	AW
7	Disagreements between multiple team members which may delay work or the project's completion.	3	2	3	In order to avoid any delays, if there is a disagreement, it should be brought to the attention of all team members and a decision should be made democratically.	AW
8	A team member leaves permanently.	4	2	8	The team member's work should be redistributed evenly taking members' skills and workload into account.	AW

9	Team members may not follow the desired process or not complete their work properly.	2	4	8	Team members should follow the agreed process and try to do the work assigned to them. Team Members should ask the chair for help if necessary - who is also responsible for monitoring work.	AW
Tools Risks						
10	Important updates are released for the support tools used whilst in development.	2	2	4	Unless updates are absolutely necessary, all tools should be kept on the same version until the project is finished. If an update is mandatory, team members will have to help each other adjust to any major changes in the software, or seek online help.	HT
11	Support tools fail to function in any way, or do not work as well as anticipated.	2	2	4	If the performance of a particular tool has come to the attention of most of the team, a replacement tool will have to be found as soon as possible.	HT, RE
Requirements						
12	Changing of requirements results in the team requiring to modify the project. (Feature creep)	4	3	12	This is a guaranteed risk based on Assessment 4's content and one which will cause impact. It is highly important therefore that code is kept clean, well documented and easy to modify.	AD, AW
13	Requirements incomplete, ambiguous or untestable.	3	3	9	If a requirement has come to the attention of most of the team due to being ambiguous and unclear, the client should be consulted to either remove it or change it to be more precise.	AW
Estimation Risks						
14	Inaccurate or insufficient planning of the of tools required to complete the project.	2	2	4	Ensure that tools used are well-researched and discussed with the team to ensure they are the correct ones.	SD, AW
15	Inaccurate or insufficient planning of the project schedule, deadlines and punctuality.	2	3	6	Sufficient time should be given for each piece of work when planning and milestones. Deadlines should not be underestimated and team members should agree with the time allocated for each piece of work.	AW
Process Risks						

16	Development of the project without an appropriate, defined process, no good plan and management.	1	3	3	A suitable, appropriate development process should be mutually agreed upon by all team members.	AW, SD, HT
17	Lack of interest during the swap process	2	3	6	The game's code should be kept clean and clear, making it easy to modify. The website will also be kept up-to-date, attractive, and with advertisement materials on.	AW, HT