Deliverable: Method selection and planning

Group number and name: Group 10 - Uptown Func()

Group member names: Freddie Cooper, Brandon Stahl, Matthew Tsang, Ryuchie Tjhuanda, Joel Weston, Ben Williams

a)
For our project, we have chosen to use the Scrum software engineering method. Scrum is an agile method, often used in collaborative projects. It is the most appropriate method for us to use, as it requires the project to be broken down into sections which are to be achieved in a 1-2 week ‘sprint’. Alongside this, due to its iterative nature we would be able to adapt to evolving requirements or changes in team member availability.

To begin with, we will decide on all of the parts of the project that we need to complete, including all the features that the game will have. Then we will meet twice a week and decide in these sessions what the focus or focuses will be for our next sprint. After this, we will go away and complete the work we have assigned whilst having text message conversations daily to brief others who are working on the same or a related focus. At the next meeting we can review progress of the last week’s sprint and plan out what needs to be done in the next sprint. This reflection will allow us to gather feedback on what can be improved over the next sprint, and this will result in a more streamlined process as we progress through the project.

This is suitable for our group situation, since we are not working daily in a centralised location it is not feasible to meet up and have a daily debrief as is usually the case with the Scrum method. However, we have adapted this to work for us by substituting this with messaging over text daily. We will be meeting twice a week, and this will allow us to have the important discussions over what needs to be done and feedback on what has been done. This is a crucial stage of the Scrum method as it allows the group to adapt to any changes and ensure it is working coherently.

When implementing the project we used the code editor Visual Studio Code. When we were researching which code development tool to use, we had some key features that we would need it to have. These were: compatible with Java, syntax highlighting, built-in compiler and execution environment, real-time error checking and integration with collaboration tools. These features would allow us to be more efficient with our programming, eliminating any time looking for a small syntax error or searching for instances of a certain variable for example. We found a few tools that were appropriate, they were Visual Studio Code, NetBeans, IntelliJ IDEA and Eclipse.

All of these tools met our requirements, so then it was more important to pick one that the members of the group that would be programming preferred. Eclipse was considered to have a poor UI, as it was too cluttered, and it is also said to be harder than the others to set up. NetBeans was a good tool, however it had poor community support compared to the other tools we were looking into. IntelliJ IDEA was a highly liked development tool with our members, however we went with Visual Studio Code since most of our members had experience using it before.

In Visual Studio Code (VS Code) we would have access to all of the features we required, and, given our experience with this platform before, it was the best choice for us to develop in.

For our project we used GitHub to aid collaboration. Before we researched different collaborative tools to use, we came up with a list of the most important features for our project. These were: being able to work concurrently, having access to past and present versions of the code, version control integration and being compatible with our development tool VS Code. The collaboration tools that we found that had all these features were: GitHub, GitLab and BitBucket.

It was important for us to pick a tool that we understood how to use as this would support the basis for successful teamwork between the programmers. All of the three were considered to be very good at all the metrics we required. However since we were unfamiliar with them all we decided that the best one to go for would be the one with the best community support and documentation. This would be vital, as it would speed up the processes of merging work together towards the end of the project.

We found that GitHub was the best tool in terms of community support, and decided to use it. It also had the upshot of being compatible with VS Code, which would mean we could further streamline the process of implementing our project. Using GitHub also links in well with the Scrum method which we are using. Since GitHub supports collaboration well, through the use of merging and branching, we can work concurrently on the same focus of a sprint and easily see how the project is progressing to supplement the daily text messages.

b)

For our team’s organisation, we decided it would be best to split into two teams. One team would be programming, and the other writing up documentation. We found that this was the best method for us after our first three or four meetings, as it leant itself well to our different skill sets. We had some people more familiar with using VS Code and GitHub, so the logical move was for those people to focus on the implementation side of the project. The others were happy to complete all documentation, however there would still be a strong link between the programming team as some of the documentation requires knowledge of the code.

We decided to have two meetings per week, this then allows us to have conversations as a group to make any big decisions on the project. It also allowed the programming and documentation teams to meet face-to-face and discuss where they are at and what the next steps will be before meeting again. Between meetings we will have text message chats to ask any questions and keep other members of our respective teams updated on any progress that may affect their roles.

To ensure that we have a high bus factor, we have implemented many methods to avoid any potential risks of the project relying on one or two people. In our respective teams, we will be keeping each other updated so that if need be another member can pick up someone else’s work. As well as this, the documentation is all stored on a cloud service which will allow all members of the team to have access to it and be able to edit it. Similarly for the implementation team, the code will all be accessible via GitHub which all team members will have access to. By having these fallbacks implemented, we are mitigating the risk of losing progress by losing a team member. It will also be important to assign major sections of the project to multiple people, as assigning them to just one person leaves us vulnerable to failure by having a low bus factor.

In meetings, we will make notes on what has been spoken about and what we are aiming to have achieved come the following meeting. This will allow us to track our progress throughout the project, so we can ensure we are on schedule as per the plan made. As well as this, taking meeting notes means that any members that could not attend are able to catch up and know what is expected of them for the next meeting.

c)

We have broken our project down into three main stages: planning, design and implementation. In the first stage, we will plan our project through various forms of research and elicit our client's requirements. Once the planning is complete and we understand the requirements we must meet, we will move on to designing the project. This will involve designing the game's backend architecture and frontend appearance. Finally, we will be able to implement both the backend and frontend, and integrate these to create our game. This breakdown can be seen in Plan figure A, whilst the starting dates, finishing dates, and the dependencies can be seen in Plan figure B and Plan figure C. Plan figures A, B and C can be found on our website using the names mentioned.

By Plan figure A, in the planning stage of our project (taking place between 26/09 and 16/10) we will research existing maze games. We expect this part of the planning stage to take 7 days, and it will be completed by three to four members of our team. This will allow us to find out what makes a successful maze game, and we can then use this information to shape our questions to the client. As can be seen in Plan figure B, the plan for the client meeting is dependent on our research of existing projects, so it is a very high priority that this is completed before 10/10. In our client meeting (occurring on 10/10), we will then use these planned questions to elicit the requirements for the project. Using this meeting's notes, we can then write up these requirements, which will then allow us to begin the design stage. This means that writing up the requirements is crucial before we can progress to the next stage. We will assign two members to writing up the requirements, and expect this to be completed between 11/10 and 16/10. The other research parts of the project will be researching development and collaboration tools and researching Java game development libraries. The client meeting should not depend on either of these, however they will also need to be completed by the start of the design stage. Since they are not difficult research topics, this is planned to be done alongside the research into existing maze games and it will be completed by the other three or four members of the team.

Once the planning stage is complete, as can be seen in Plan figure B, we can begin the design stage. This will be designing the backend workings and the frontend components of the project. We have scheduled this to begin on 16/10 when we will be having a meeting, and this has a very high priority as our moving on to the next stage is dependent on the design being completed by 22/10. The backend workings will include player input handling, game state updates and interactions between game objects and the frontend components are the sprite, the walls, collisions between these two and different events we will implement. These design stages will run concurrently,

As Plan figure C shows, once these two design sections are complete, we can begin implementing the backend and frontend. This will be done using the collaboration and development tools and the Java game development library that we researched during the planning phase, so it is important that these sections are completed by the start of the project. We have planned for the design of these sections to run concurrently and to be completed by 4/11. Each section will be divided up between the four members of the programming team. It will be important that each of these four understands both parts of the code so that implementation runs smoothly. Lastly, we can integrate the frontend and backend so that we have our fully functioning game. This is scheduled to take place between

4/11 and 7/11, it has highest priority since it is the point at which the game will be coming together. It will be completed by the four members of the programming team.
We expect some of these deadlines to be slightly altered when going through the project, potentially being completed quicker or slower depending on circumstances. Since almost all of our stages are dependent on one before it, this will cause delays that may affect the quality of the project. For this reason, we have made it so that all tasks have more than enough time to be completed with the possibility of errors occurring and being resolved.

Throughout the course of this project, we have deviated from the above plan occasionally, in the design phase mainly. Some tasks have been finished earlier than expected, this allowed us to move into the next stage quicker, giving us more time for potentially more difficult tasks later on. Meanwhile, other tasks took longer than expected, most of these are to do with implementing the game, we can afford these short delays due to allowing for this in our plan and completing previous tasks quicker than expected. However for the most part, the overall pace of the project aligned with the plan. Our meeting summaries can be found named Meeting notes [date] on our website.