**SOFTWARE ENGINEERING G6046**

**APPENDIX A: SPRINT DOCUMENTATION TEMPLATE**

|  |  |
| --- | --- |
| 1. **Summary data** | |
| Team number | 11 |
| Sprint technical lead(s) | Anson Wong |
| Sprint start date | 20/04/2021, meet on Thursday |
| Sprint end date | 26/04/2021 |

*The technical lead may vary from one sprint to the next. This is down to how you collectively organise your team.*

|  |  |
| --- | --- |
| 1. **Individual key contributions** | |
| **Team member** | **Key contribution(s)** |
| Anson Wong | Programmer, Planner |
| Danny Newsom | Programmer |
| Tomasz Czarnecki | Programmer |
| Abdullah Al-Hiyarat | Programmer |
| Mohammad Jallad |  |
|  |  |

*This data should help you to agree your peer assessment at the end of the project. If there is a dispute over your peer assessment, the markers will refer to this section as evidence to support a final decision.*

|  |
| --- |
| 1. **User stories / task cards** |
| *Provide text descriptions of any user stories or task cards you have selected for this sprint. These should naturally emerge from the user requirements document and discussion on Canvas. If you produce task cards, they should show the relative priority of the task for this sprint.*  User stories   * When the suggestion is made, starting from the current player’s left, if they have one of the cards mentioned in the suggestion, then they must show one only of those cards to the current player (unseen by other players). If the player on the left is unable to show a card, then we move to the next player to the left until a player is able to show a card to the current player. Once that happens, or all players have been tried leading back to the current player, then the turn of the current player is ended.” * With the provision of the autonomous player agent feature, it should be possible for a full experience game to be enjoyed by just a single human player. * The game player agent should be able to play the game to the same extent that a human player would. The game player agent would roll moves, make suggestions and an accusation. The key point here is that the game player agent needs to be able to play the game, but it does not necessarily need to be any good at it, at least in the first instance. A game player could incorporate some simple rule for making in game decisions, or could feature relatively sophisticated Artificial Intelligence. But a game player could just operate on purely random decision making. That would still allow the game to be played. It is suggested that your make the game player agent perform random decision making initially, and then if time permits, look to increase the sophistication of the agent. The marking scheme places emphasis on the delivery of a working agent, and only a small amount of its relative game playing sophistication. Players may not retire from the game. * Some information needs to remain private between players – this is essential to the correct working of the game This will have an impact on the design of any GUI that you implement. Watson Games have no particular ideas as to how the GUI should look – they are looking for ideas from you. You can regard this project as producing a prototype for Watson Games to consider. As such, any reasonable approach to the GUI design that meets the needs of the game and its rules is acceptable. This will depend, to a large part, on how you decide to implement the game. * There are 6 cards representing the persons in the house. There are 6 cards representing the weapons. There are 9 cards representing the rooms shown on the playing board. In the physical version there are “detective notes cards” to enable players to record their investigations. * The electronic version should be for desktop machines, and ideally should be playable on both Mac and PCs. If this is difficult, then PC development should be preferred. * The game should be fun to play and have a colourful and intuitive interface that reflects the spirit and character of the original board game. * We are consulting with the National Clue! players society onnewideas for features. We are not yet in position to say exactly what they are, but the core idea is to add some extra game play value by having additional actions on some squares such as "roll again" or "have an extra accusation (corrected to Suggestion)". We will let you know more when we have the final information.   Task Cards   * Get the Game to work * Missing functionalities:   + Showing suggested cards from player (Tomasz)     - Display and click on a card to be suggested     - Move the suggested stuff   + Fix suggestion menu (Anson)   + Allow Suggestion to take room enum   + Black screen to hide the next player   + UI to show player got accusation wrong   + Game win screen   + Player eliminated screen   + Getting bonus tiles to work   + AI     - Design AI logic     - Implement AI to the game   + Main menu   + Improve the Visual elements   + Implement detective notebook |

|  |
| --- |
| 1. **Requirements analysis** |
| *For the user stories/task cards selected, set out what key functional, non-functional and domain requirements you have identified. Remember that functional and non-functional requirements can be further categorised as mandatory (“shall”) and desirable (“should”). You can use free text descriptions or tabular formats. Remember that domain requirements cannot be acted upon directly. They require domain expertise to refine them into meaningful functional and non-functional requirements. All requirements should be SMART (Specific, Measurable, Achievable, Realistic and Time-Bounded). The requirements analysis does not need to be exhaustive, but should focus on things that are important for this sprint. They should also form a basis for testing.* |

* UI Controller: (Tomasz)
  + Shall
    - Suggestion
      * Only allow the player to select accusation when the player is in a room
      * The UI screen will have the room selected
      * The player cannot change the room
      * Pass the selection to the round manager
      * Wait for the round manager to return a card to be displayed
      * Have the ability to change their choices
      * Have the ability for the other players to select which card to display
    - Free suggestion
      * Same for suggestion, but the Player can change the room as well
    - Black Screen
      * A black screen to tell the next player to take control of the computer
* Round Manager: (Abdullah)
  + Shall
    - Suggestion
      * A method to accept the suggested list of cards
      * Loop through all the players via the Turn Manager
      * Get the first instance of the player that has that card
      * return which player has that card and that card it is
      * Return that results to the UI controller to be displayed
* AI (Anson)
  + Shall
    - Perform actions depending on the AI Behaviour flow chart
    - make educated guesses and takes risks in the accusation
* Detective notebook (Danny)
  + Shall
    - Allow players to cross out cards
    - Have the notebook saved to the player
  + Should
    - automatically cross out cards
* Visual Elements (Danny)
  + Shall
    - Image for all cards with names
* Documents
  + Use Doxygen

|  |
| --- |
| 1. **Design** |
| *Remember that you only need to do enough design to support the objectives of the sprint. For teams working with OO implementation languages (likely most of you), this would include a class diagram. You may find it useful to develop simple Application Programming Interfaces (APIs) for key classes. This will focus your attention on what each class needs to make available for other classes to use. It also supports good documentation practice and helps coders work together.*  AI Behaviour flow chart    AI UML |

|  |
| --- |
| 1. **Test plan and evidence of testing** |
| *You should consider:*   * *Unit/component level testing – typically achieved using automated test procedures such as Junit in Java. This level of testing demonstrates that individual classes are working as you intend.* * *System level testing – typically a human lead and documented test process that shows the prototype working as a whole entity.*   *Testing should show that the requirements you set out are being delivered on. They provide a means of showing that we have delivered what the user stores and task cards set out. Remember to identify a useful set of boundary test conditions.*  *Evidence of testing should demonstrate that the prototype achieved has been tested according to the test plan. If there are deficiencies, then these should be documented, as they will need further work in a subsequent sprint.*  Test Scenes:   * AI * Suggestion   **\*Please Refer to Testing Document** |

|  |
| --- |
| 1. **Summary of sprint** |
| *You should consider and discuss:*   * *Did you achieve your objectives for this sprint?* * *Is there a working prototype?* * *What went well, and what did not go well? If things did not go well, what have you learned and what will you do differently for the next sprint?* * *Is there any feedback from the customer?* |

Due to our lack of planning, and some of our teammates being inexperienced in Unity we have realised we are missing a lot of the functionalities to the game, and some functionalities like the UI only works locally but not in other scenes. There were also functionalities where we thought it works but doesn’t function as it should when doing some simple testing.

As a result of that, the team have resulted in a typical game industry practice, Crunch!

We did however manage to finish the game with 3 builds (Windows, Mac and WebGL) with Unity. It has all the elements completed from the task cards. The UI has been significantly improved, and the AI, though it is new, has been performing really well and has become a challenging player to face against.

We have showed a final product to Mr. Raffles and he is very pleased with the game and said that “it keeps in spirit of the original game” and the scene with 6 AI battling each other as “mesmerising”