PROCESS

The process for making and improving the game started with the creation of an online miro whiteboard where each element of the game could be displayed on its own note, and proposed ideas could be attached for discussion at meetings, alongside prioritising ideas that were essential compared to those to return to if we had time towards the end of the project. The white board can be seen at:

<https://miro.com/app/board/o9J_knYaSCE=/>

The miro whiteboard helped to identify tasks that could be initially worked on independently and those which required communication between different parts of the game via a messaging system. These helped inform which tasks each member of the group took on, with each person recording their progress and outstanding things to complete on a group Trello board which can be seen at:

<https://trello.com/b/gFyMKmaC/pommertalk>

Due to the fact the project would be worked on remotely, each member of the group initially took on a part of the game that could be worked on largely independently. These initial roles were as follows:

James – Implementing different alliance options and agent actions in response to them.

Gideon – The Graphic User Interface (GUI).

Guilherme – Messaging communication between the agents and with the GUI.

The group organised a call on discord every morning to discuss current progress, ask each other to test code changes and report what each person had to add to make sure each new addition could be implemented with the different parts of the game. Importing the original version of Pommerman into the IntelliJ IDE allowed for good version control using git. Each member of the group worked on their own branch, regularly merging code that had been confirmed by the group to be working into a master branch. This helped avoid situations where any code was accidentally added to the master that might cause a game breaking bug.

As well as internal play testing throughout the game’s development, it was also useful to get external opinions from other members of the IGGI group on the playtesting day. This helped identify a major issue in compiling the game to an executable jar file, as well as the initial difficulties in playing the game for the first time as a human player. This realisation led to pausing the game whilst human players and choosing their alliances and letting them continue when ready rather than being restricted to a pre-set countdown timer.

Graphical User Interface (GUI)

The GUI has two main functions:

1. To allow a human player to request alliances with other ai players. and to choose from the alliances received from ai opponents.
2. To allow a human observer to watch a game between four ai agents, seeing the different alliances formed at each stage between the different agents.

Early discussions informed that the GUI should use the same keyboard interface as the main game when playing as a human player. This involved the arrow keys to move and the space bar to set a bomb. Additionally, it was decided to use the blank space underneath the main game screen to display the alliances currently in play and to make selections. This new display was attached to a new Java panel.

The most sensible option was to use an array of Java toggle buttons as shown below, showing each possible alliance type with each of the selected player’s three opponents who the alliance could be made with. When selected both the rule and opponent are highlighted to display that this particular alliance is currently in play. When watching a game between four ai agents, an observer can select any of the four agents and the buttons will automatically update on the next game tick to show the alliances with each of their three respective opponents.

The information displayed to a human player was iteratively changed over the course of the game development to make the available options more intuitive and remove the frustration from unfair deaths. This involved some additions to the basic Pommerman game such as displaying the time to the next wall collapse, since players on the edge of the screen when a collapse occurs would be instantly killed. Each negotiation phase is now clearly colour coded and human players also have an unlimited amount of time to make their alliance choices with the game being paused, rather than being time limited like the ai opponents. In an ai game the alliances requested and chosen are displayed with 15 ticks of each negotiation phase remaining, making it clear to a human observer what choices have been made.

In the first stage of the alliance negotiation a human player can make up to 2 alliance requests with the three other players on any of the 5 possible alliances. In the second stage they can only choose from the alliances they have received from their opponents. This required changing the listener events of the buttons and the keyboard keys to move between them during each negotiation phase. Initially the available selections also flashed during the second negotiation phase, though this was removed when the game paused since the game ticks no longer updated allowing its inclusion. This might be something to reinstate in a future version of the game.

The GUI directly connects with the message system during the negotiation phases to create an array of all the alliances formed between all players. Keeping the data stored in this way allows the GUI to be updated with any player’s alliances, so that the human player is given the option to observe the other players’ tactics after they have died.

Video

A video of the game in action showing gameplay with a human player and an observer watching four ai agents playing can be seen at the following link:

<https://youtu.be/KPBOuO0dyK0>