**3.4 Graphical User Interface Design And Implementation**

To create a fully-fledged Go Problem solver allowing the user to interact with life and death problems, a graphical user interface had to be designed and implemented to allow for communication between the user, GameEngine and artificial intelligence. The finished GUI was created using Java Swing *[reference here]* and the Graphics package *[reference here].* The user interface provides a wide range of functionalities through graphical board representation and the implementation of interactions such as placing stones and saving board states. Hence, the final design of the GUI successfully allows the user to create and play life and death problems as specified within the requirements.

**Initial Design**

During the preliminary design process of the GUI, a paper prototype was created as a basis for implementation. As can be seen in *Figure 1*, the paper prototype showed the original layout of the user view of the GUI which consisted of several elements. These elements included a representation of the board, featuring white and black stones where the human and AI would both play moves. Another component that can be seen are the radio buttons that were drawn out, which would allow the human and computer's stone colours to be chosen. Lastly, an undo button and a reset button were also included in the paper prototype. The undo button would remove the latest move on the board, whilst the reset button would adjust the board back to its original loaded state. The original design also included a toolbar featuring a single button, labelled "*Menu*". Pressing the menu button reveals the options seen in *Figure 2* for loading and saving the board, as well as saving a log which would be a series of played moves. Whilst, the paper prototype was a strong basis for the beginning of implementation it was missing a wide range of functionalities, such as bounds and player modes, which were included within the final implementation. Some redundant options within the original paper prototype's file menu were also removed during implementation, such as the save log.

*TBA: ADD FIGURE 1 (Paper Prototype 1 - User View)*

*TBA: ADD FIGURE 2 (Paper Prototype 2 - Menu Bar)*

**Visual Implementation**

The first step of implementing the GUI was the graphical board representation, which can be seen as the main visual feature of the problem solver. As described previously (see *Section 3.3*), the GameEngine uses a 2D array to represent the board within a Board object. This 2D board array had to be translated to a graphical representation to enable the GUI to communicate directly with the GameEngine and AI as well as the user. As the GUI was created using Java Swing [*reference here*], the Graphics package [*reference here*] was utilized to provide a visual of the board. To depict the board on screen, the current number of board lines (height/width) are taken from the Board object specified within the GameEngine. A series of brown rectangles are then drawn and filled in to represent the board using the number of specified board lines. These rectangles overlap to form the appearance of a grid and again using the line number provided, coordinates are drawn aligned to each of the rectangles. The user can control whether these coordinates are shown or not and they can be a useful aid during play. This described method of drawing out the board means that the GUI is flexible to representing any size of board specified by the user, as long as it is a square. Following on from the successful drawing of the board layout itself, the GameEngine's 2D array board representation is scanned and when a stone was found - being Integer 1 for black and Integer 2 for white - the appropriate counter is drawn on the board. The counters are painted in a top-left across and down fashion, comprising of a fully coloured circle with a given edge around them. Hence, a full board containing a grid, counters and available coordinates is drawn as seen in Figure 3 using the 2D Board object array provided to the GUI.

*TBA: ADD FIGURE 3 (GUI Board Example: Counters + Coordinates)*

Several other visual enhancements were made following the original drawing of the board, allowing for a higher level of feedback and a more user-friendly system. These included the use of transparent stones upon the board when the user's mouse hovers over an intersection, being where a stone can be placed upon the board. These transparent stones are colour-coded and are either black, white or red - which signifies an illegal move through utilization of the legal move checker, an important part of the GameEngine as described previously (see *Section 3.3*). The see-through stones are depicted in *Figure 4* and were also implemented using a 2D array, likewise to the current board representation. This 2D array refreshes itself each time a mouse move is detected. After this refresh, a stone is placed within the transparent stone's 2D array at the current location of the mouse. Checking whether the current stone location is legal or not occurs after the updating of the 2D array and once the check occurs, the appropriately coloured see-through stone is displayed.

*TBA: ADD FIGURE 4 (GUI Board Example: Transparent Stones)*

Yet another visual enhancement within the GUI can be seen in *Figure 5*, being the depiction of the artificial intelligence's search space as bounds. These bounds are depicted graphically and represent the board positions that the AI will consider moving to. The intersections that are available to the AI are coloured brown, whilst the rest of the board is greyed out. This graphical feedback allows the user to easily see the multiple positions which the computer will possibly move to and this feedback can be toggled on or off according to the user's preference. Originally the bounds took only the shape of a rectangle or square between two specified user points, however through a process of improvement the finished GUI allows the user to select a flexible set of bounds simply by clicking on the intersections to be included within the AI's search space. This change not only provided better visual feedback but was also more efficient for running the artificial intelligence whilst in-game.

*TBA: ADD FIGURE 5 (GUI Board Example: Flexible Bounds)*

**Player Modes**

Following the implementation of the board representation itself, as well as several other visual features, a major design change from the paper prototype took place. The design decision was made to implement two separate user playing modes, being "*Creation Mode*" and "*Competitive Mode*." The availability of user modes allows for easy distinguishing between the two main features of the Go problem solver - being allowing a player to create a problem or playing out a problem. The creation of a problem provides the user with various options whilst playing out a problem allows the user to choose between playing against the AI or another human. The separation of the two modes also rids the GUI of the possibility of having overlap of certain features between modes, which could create numerous errors - for example, deletion of stones is only available in creation mode. Each mode also has their own dedicated menu upon the toolbar, named "*Problem Creation*" and "*Competitive Play*."

When the GUI is initially started, it is opened in problem creation mode and displays a blank 9x9 Go board - ready to begin letting the user to design life and death problems. The problem creation menu in the toolbar, as seen in *Figure 6*, allows access to various features during the creation of problems within the GUI. These include options during the placement of stones, such as using only a specific colour of stone (white/black) or being able to delete stones by pressing on them. Problem creation mode also allows the user to alter the AI's objective and bounds. The objective is specified in a pop up box, as displayed in *Figure 7*, whilst the bounds can be altered by the user selecting bound selection and then specifying their bounds by selecting specific intersections. This collection of features easily allows for problems to be made quickly and can then be saved by the "*Save Problem*" option in the file menu, ready to be played in competitive mode.

*TBA: ADD FIGURE 6 (Problem Creation Menu)*

*TBA: ADD FIGURE 7 (Objective Pop Up Box)*

Competitive mode meanwhilst is intentionally selected by the user after the initial opening of the GUI. This mode allows for the play through of a problem in several approaches, including human versus human, human versus AI (either Minimax or AlphaBeta) and AI versus AI. Upon selection of the "*Competitive Mode*" button a pop up box appears allowing for player choice as seen in *Figure 8*. If the bounds and objective are not specified before this selection, the GUI will prompt the user to choose them before allowing competitive mode to be entered. Once a game is in play, the problem creation menu is greyed out and the competitive play menu is available to use. This menu includes such features as swapping player colours, forcing the AI to move and also allowing the user to change the AI type currently being used within play. An example of an AI move can be seen in *Figure 9* and *Figure 10*, where labels on the side of the board specify the AI type and its move selection which allow the user to easily learn from its position selection, as seen in *Figure 11*. \*WRITE ABOUT HEURISTIC CHOOSER HERE\* Hence the options available in competitive mode allow for a flexible and user friendly environment whilst playing through a problem with the user interface.

*TBA: ADD FIGURE 8 (Player Choice Menu)*

*TBA: ADD FIGURE 9/10/11 (AI Plays A Move)*

*TBA: ADD FIGURE 12 (Heuristic Chooser)*

Several features of the GUI are included in both creation and competitive modes. These features include the "*Show Coordinates*" button which displays the coordinate numbers along all sides of the board, allowing easy stone placement. The user can also load or save problems at any point as well as use debug features. Debug features include displaying the log, being the history of all actions within the GUI. Also, the usage of the help menu is available in both modes which has options including the display of keyboard shortcuts. For example making the AI move against itself in AI versus AI mode can be quickly done using *ALT + I* and there are several other keyboard shortcuts which enable even quicker use of the GUI. Other useful functionalities provided in both modes include the undo and reset button, allowing for the quick backing up of mistakes or reset to the original board, which is practical at all times for the user.

**Finalised Graphical User Interface**

Through the evidence given prior, it is clear that a user interface was completed successfully allowing for easy interaction during problem creation and competitive play. The entire finished view of the GUI can be seen in *Figure 12*. The final implementation of the interface allows for the quick making and saving of problems before the user is able to play them in a variety of ways and against several different entities, including the AI and other human players. The GUI also has a wide range of useful features, such as making the AI's functions including search space and heuristics available to the user, as well as having a clean look and layout.

*TBA: ADD FIGURE 13 (Full GUI View)*