Chapter 1

Testing

1.1 Iterative Testing

I have been playtesting the program throughout the development process to find any bugs and fix them accordingly. However, a few issues have required additional measures.

1.1.1 Minimax

Since minimax is recursive algorithm, debugging it has proven a challenge. I have therefore configured the Python `logging` library to help collect information on the minimax tree for every function call.

base.py

```
    def print_stats(self, score, move):
        """
        Prints statistics after traversing tree.

        Args:
            score (int): Final score obtained after traversal.
            move (Move): Best move obtained after traversal.
        """
        if self._verbose is False:
            return

        self._stats['time_taken'] = round(1000 * (time.time() - self._stats['
    time_taken']), 3)
        self._stats['ms_per_node'] = round(self._stats['time_taken'] / self._stats
    ['nodes'], 3)

        # Prints stats across multiple lines
        if self._verbose is True:
            logger.info(f'\n\n'
                        f'{self.__str__()} Search Results:\n'
                        f'{printer.pformat(self._stats)}\n'
                        f'Best score:  {score}   Best move: {move}\n'
                        )

        # Prints stats in a compacted format
        elif self._verbose.lower() == 'compact':
            logger.info(self._stats)
```

```
26                logger.info(f'Best score: {score}   Best move: {move}')
```

Listing 1.1: BaseCPU Method for logging minimax statistics

1.1.2 Migrations

To correct errors made to the `games` table, since recreating it would mean deleting all existing games, I have opted to use migrations to fix bugs by editing the table schema, as shown in Section ??.

1.2 Unit Tests

1.2.1 Board Evaluator

To test every aspect of the evaluation function, I have set up some unit tests with custom positions using my editor screen. These positions are designed to test every aspect of the evaluation, along with some obviously imbalanced positions to test the overall accuracy of the evaluation function. All positions are set up to give an advantage to the blue player.

Evaluating	FEN string	Score	Passed
Material	`sc9/10/10/4paPa4/5Pa4/10/10/9Sa b`	124	✓
Position	`sc9/4nanana3/10/10/10/4NaNaNa3/10/9Sa b`	66	✓
Mobility	See footnote[1]	196	✓
King Safety	`sc4fa3pa/10/10/10/10/10/10/5FaPa2Sa b`	3	✓
Combined	See footnote[2]	437	✓

Table 1.1: Board evaluator test results

1.2.2 CPU

Similarly, to evaluate the strength of my CPU, I have setup some custom positions that I already know the best continuation of, and run each CPU engine on them to test if they can solve it.

Description	FEN string	Best Move	Passed
Mate in 1	`sc9/pafa8/Fa9/10/10/10/10/9Sa b`	Rotate J3 clockwise	✓
Mate in 1	`sc9/10/10/10/8faRb/8FaRb/10/9Sa b`	Move J3 to J2	✓
Mate in 3	`sc9/10/10/8Ra1/7FaRafa/8RaRa/9Ra/9Sa b`	Move J2 to I1...[3]	✓
Mate in 3	`sc5fcnc2/4Pa4Pc/3pb6/2Pc2ra1pb2/10/pb9/7Pa2 /2PdNaFa4Sa b`	Move E7 to F7...[4]	✓
Mate in 3	`sc2pdfc5/2Ra1ra5/pa3Pc4Rb/pb1rd1Pdra4/7Rd2/3 Ra1Pa4/4Fa5/3PdNaPa3Sa b`	Move J6 to J7...[5]	✓

[1] scpapa7/papapa7/papapa1Pa1Pa1Pa1/10/4Pa1Pa1Pa1/10/4Pa1Pa3/9Sa b
[2] scnc1fncpbpb3/pa9/pb1pc1rbpa3Pd/1Pc2Pd4Pc/2Pd1RaRb4/10/7Pa2/2PdNaFaNa3Sa b
[3] 2. Move J4 to J5 3. Move J3 to I2
[4] 2. Rotate anticlockwise H8 3. Move F7 to G7
[5] 2. Move E7 to F8 3. Move E4 to E5

Description	FEN string	Best Move	Passed
Win Material	`sc9/2fa7/5NaPb3/pa2ra1Pd4/3pcnd1pd2Rd/3pc6 /10/8FaSa b`	Move F6 to G7...[6]	✓
Escape Mate	`sc9/10/10/10/8faRb/8FaRb/10/9Sa b`	Move J3 to J2	✓

Table 1.2: Iterative Deepening CPU test results

1.2.3 Shadow Mapping

To test the shadow mapping algorithm, I have set up some occluding objects togehter with a light source. Since visuals are subjective, me and my client have deemed the following results to be adequate.

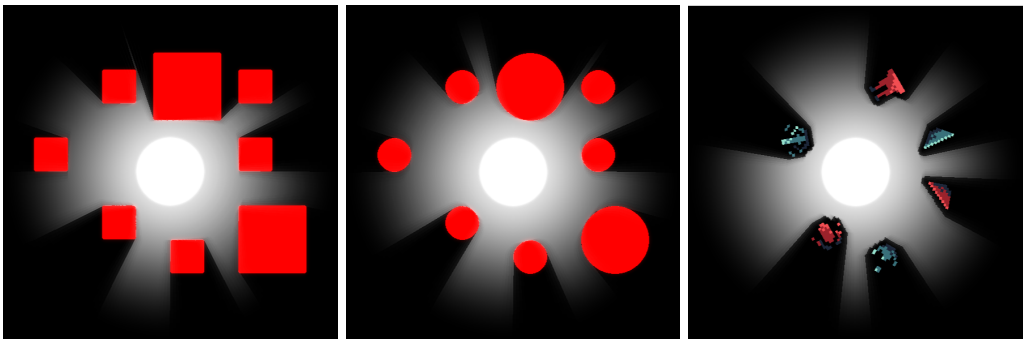

Figure 1.1: Shadow mapping algorithm test (softShadow=0.5, radius=0.5)

1.3 Final Tests

1.3.1 Objective 1

All laser chess game logic should be properly implemented.

No.	Input	Output	Passed
6	Position piece with non-reflecting side facing laser	Piece is destroyed	✓
1	Laser fires on pyramid	Pyramid reflects laser by 90°	✓
2	Laser fires on scarab	Scarab reflects laser by 90°	✓
3	Laser fires on anubis	Anubis absorbs laser	✓
4	Move piece as blue player	Active colour switches to red player	✓
5	Move piece or rotate piece	Laser fires	✓
6	Fire laser onto pharoah	Pharoah is destroyed and opposite colour wins	✓
8	Repeat same position three times	Game displays game over screen	✓

[6] 2. `Move E4 to E3` 3. `Move H5 to D4` 4. `Move J4 to J3`

1.3.2 Objective 2

Game should process user input correctly.

No.	Input	Output	Passed
8	Click piece	Overlay appears showing selected piece	✓
9	Click piece and click outside board	Overlay disappears showing deselected piece	✓
10	Click piece and click adjacent square	Piece moves to clicked square	✓
11	Click and hold piece and release over adjacent square	Piece moves to adjacent square	✓
12	Click piece and press rotate clockwise button	Piece rotates clockwise	✓
12	Click piece and press rotate anticlockwise button	Piece rotates anticlockwise	✓

1.3.3 Objective 3

Save or load game options should be implemented.

No.	Input	Output	Passed
13	Click on copy FEN string button in editor or browser screen	Position formatted as FEN string copied to clipboard	✓
14	Click browser button	Program shows list of past games to be scrolled through	✓
1	Select time as sorting criterion	Browser updates to show most recent games played	✓
1	Select descending as ordering criterion	Browser updates to show oldest games played	✓
1	Click on previous game and click delete button	Selected game is deleted and dissapears	✓
1	Click on previous game and click review button	Game is displayed in review screen	✓
12	Enter review screen	Program displays list of past moves, winner and move number	✓
11	Click on previous button in review screen	Board undoes move	✓
11	Click on next move button in review screen	Board applies move	✓

1.3.4 Objective 4

Other board game requirements should be implemented.

No.	Input	Output	Passed
13	Click on draw button	Game ends and shows draw result	✓
14	Click on resign button	Game ends and shows win result for opponent	✓
15	Click on timer button to enable timer	Timers appear on the left and decrement every second	✓
16	Pause the game	Timer stops decrementing	✓
17	Allow timer to run to zero	Game ends and shows win result for opponent	✓

1.3.5 Objective 5

Game settings and config should be customisable.

No.	Input	Output	Passed
17	Click on CPU button	Opponent moves are played by minimax CPU	✓
17	Click on timer button to disable timer	Timer is not shown	✓
17	Click on timer duration and input new number	Timer starts with inputted duration	✓
21	Press starting colour button to set starting colour to red	Starting move is played by the red player	✓
22	Enter valid FEN string into text input	Board preview updates and game starts with inputted board layout	✓
23	Enter invalid FEN string into text input	Error message appears	✓
23	In editor screen, select piece and click on square	Piece placed on square	✓
12	Click piece and press rotate clockwise button	Piece rotates clockwise	✓
12	Click piece and press rotate anticlockwise button	Piece rotates anticlockwise	✓
23	Click empty button	All pieces dissapear (except sphinxes)	✓
23	Click reset button	Board resets to initial layout	✓
23	Click confirm button	Switches to config screen with edited FEN string and board preview	✓
23	Click return button	Switches to config screen with changes discarded	✓
17	In settings screen, change primary board colour to blue	Alternating board squares appear blue	✓
17	Change secondary board colour to red	Board squares alternate blue and red	✓
18	Change display mode to fullscreen	Application window enlarges to fill entire screen	✓

No.	Input	Output	Passed
18	Slide volume thumb to right and left of slider	Music increases and dereases in volume	✓
20	Toggle particle and shader switches to off position	Particles and shaders are disabled after program restart	✓

1.3.6 Objective 6

Game UI should improve player experience.

No.	Input	Output	Passed
18	Click on piece	Highlight overlay is rendered on selected square	✓
18	Click on piece	Circular overlays are rendered on surrounding unoccupied squares	✓
18	Fire laser on piece	Audio cue plays and piece is visibly destroyed	✓
18	Fire laser on piece	Piece appears on opponent's piece display	✓
18	Play a game	Status text updates to display the active player's colour or CPU status	✓
18	Hover over board, hold the mouse button, click on text input widget	Mouse cursor switches between arrow, open hand, closed hand and I-beam icons	✓

1.3.7 Objective 7

GUI design should be functional and display concise information.

No.	Input	Output	Passed
27	Click the play button on the main menu screen	Program switches to the config screen	✓
27	Click the browser button on the main menu screen	Program switches to the browser screen	✓
27	Click the settings button on the main menu screen	Program switches to the settings screen	✓
28	Click main menu button	Program switches to the menu screen	✓
29	Click help button	Help overlay appears	✓
29	Click quit button	Program quits	✓
31	Drag program window	Program continues running	✓
30	Resize program window	GUI Widgets resize continuously	✓

1.4 Videos

Link to video demonstrating final tests:
Link to video demonstrating unit tests: