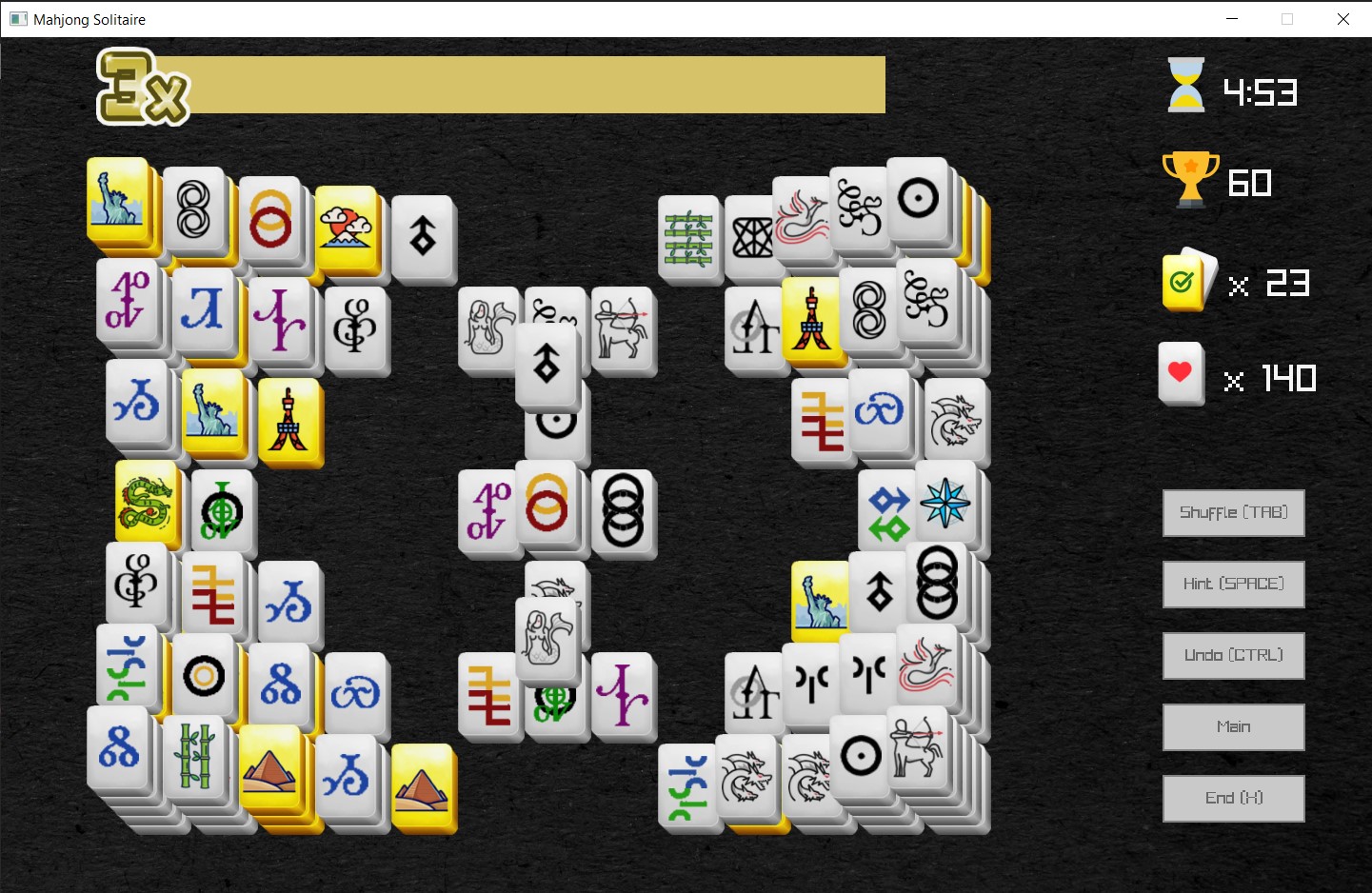
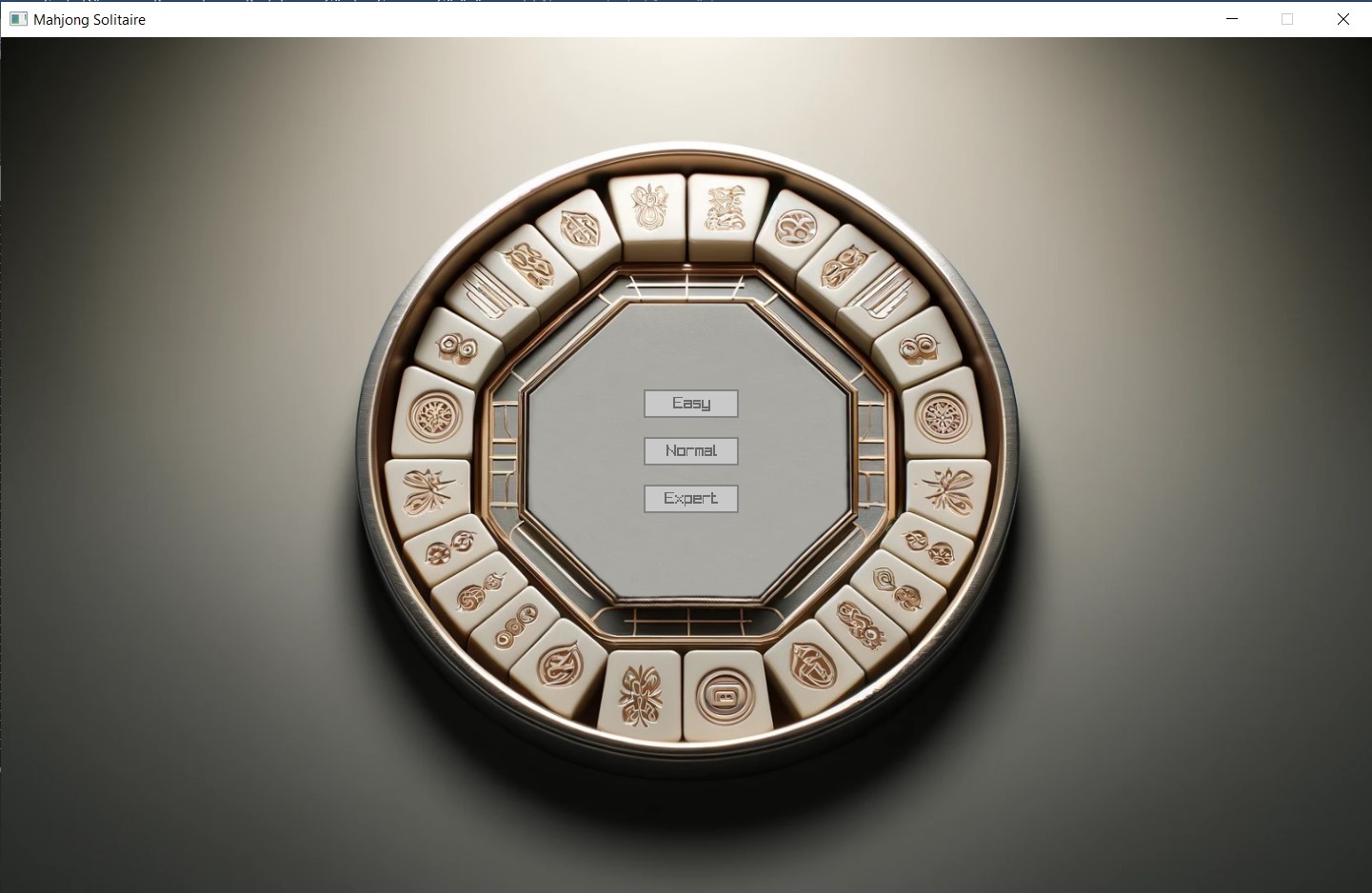
**MAHJONG SOLITAIRE**

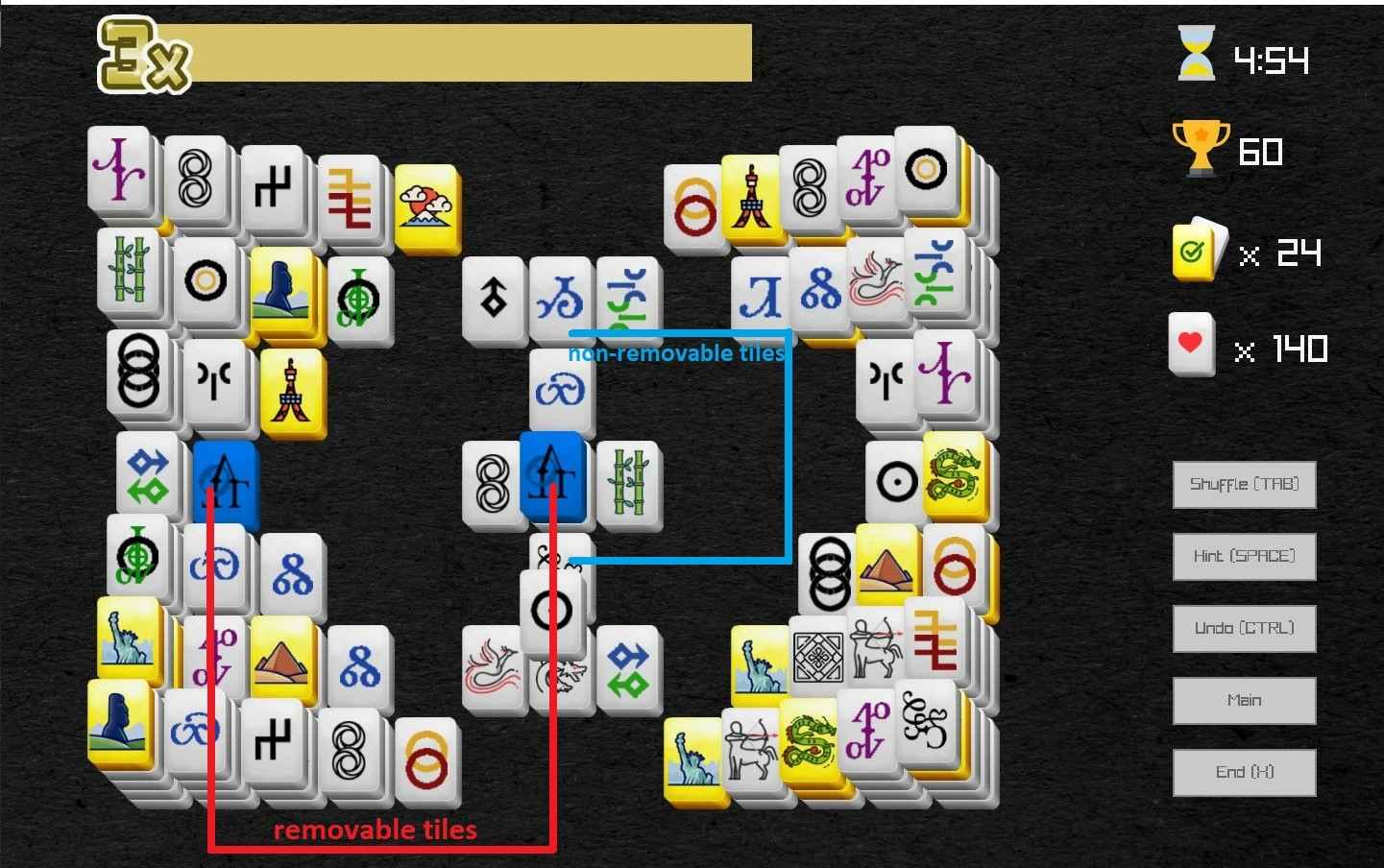


**SUMMARY OF THE PROJECT:**

The Mahjong Solitaire Game Project involves developing a digital version of the classic puzzle game Mahjong Solitaire. This project encompasses creating a game that allows players to match pairs of tiles to clear the board, with additional features like combo multipliers, a timer, and high-score tracking to enhance gameplay and user engagement.

**INTRODUCTION:**

Mahjong Solitaire is a single-player puzzle game that utilizes a set of Mahjong tiles. The primary goal is to match and remove pairs of tiles from the game board, aiming to clear the board as quickly as possible. This digital adaptation aims to bring the traditional game into a virtual format that is accessible on personal computers.



***The tools:***

* ***Shuffle*:**This rearranges all the tiles so they are in different places, allowing new matching opportunities.
* ***Hint*:** Use this if you can't find any pairs. It shows the first match it finds, but it might not be the best option.
* ***Undo*:** This lets you reverse your last move. Use it if you've made a mistake or want to try a different move.

***There’s only two rules while selecting the tiles:***

1. There can't be any tiles on top of the one you're choosing.
2. The tile must be open on at least one side (either the right or the left).

**METHODOLOGY:**

During the development of our project, we deliberately extended the timeline to ensure that we didn't rush or face any unnecessary stress. We worked closely with all the members of our group, ensuring that everyone was on the same page. Together, we systematically updated each part of the project, maintaining a harmonious and collaborative atmosphere throughout the entire process.

It took us some time to understand the logic of the Mahjong Solitaire, but once we did, we managed to successfully write the necessary code and complete our game. While every function we created was important and served a purpose, there were a some functions that were particularly crucial to the game’s operation. These critical functions were key to making sure everything worked as it should.

*Here are a few examples of critical functions below:*

The **addBegin** function plays a crucial role in managing the history of player actions in Mahjong Solitaire, specifically for tracking tiles that have been matched and removed from the board.

1. **Memory Allocation**: At the start, the function attempts to allocate memory for a new **node**. This **node** will store information about the two most recently matched tiles. If the memory allocation fails, it prints an error message and exits the function to prevent further issues.
2. **Update Tile Status**: The function sets the **isExists** status of the two tiles involved in the most recent match (stored in **LastTwoClicked**) to **false**. This indicates that these tiles are no longer visible or interactive on the game board because they have been matched and are considered removed.
3. **Store Tile References**: It then stores references to these two tiles within the new node. This is done so that if an undo action is requested, the game knows which tiles to restore to the board.
4. **Save Combo Values**: The function also records the current combo multiplier with each tile in the node. This is important for recalculating the score correctly if these tiles are brought back onto the board via an undo action.
5. **Update Tile Existence in Array**: The function updates a global array (**isExist**), setting the existence value for both tiles to **0**, reinforcing that they are not active on the board anymore.
6. **Insert Node at List Start**: The new node is then inserted at the beginning of a linked list (**head**). This list tracks all the match actions in reverse order, making it easier to undo the most recent actions first.
7. **Update List Head**: Finally, the head of the linked list is updated to this new node, ensuring that the most recent action is always at the front of this history list.

**deleteBegin** function is designed to manage undoing a player's last move in the Mahjong Solitaire game. This function performs several crucial operations to reverse the effects of a match and restore the game state to its condition before the tiles were removed.

1. **Check for Empty List**: Initially, the function checks if the linked list, which records the matches (or moves), is empty. If it is, the function triggers a visual shake effect to indicate that no undo is possible and then exits.
2. **Restore Tile State**: It retrieves the first node of the linked list, which represents the most recent match. The function sets the existence status of the two tiles involved in this match (**data1** and **data2**) to **true**, making them visible and interactive again on the game board.
3. **Reverse Scoring**: The function calculates and deducts the points that were added to the player's score when these tiles were matched. This ensures that the score reflects the undo by subtracting the points based on the combo multiplier and the point value of the tiles.
4. **Update Tile Existence and Color**: It updates the **isExist** array to reflect that these tiles are once again active on the board. Additionally, it resets the color of the tiles to white, the default state, to visually indicate that they are no longer part of a completed match.
5. **Adjust Tile Count**: The total count of remaining tiles in the game state is incremented by two to account for the addition of these two tiles back to the board.
6. **Manage Linked List**: The head of the linked list is moved to the next node, effectively removing the current node from the list since it no longer represents a valid match.
7. **Memory Management**: Finally, the function frees the memory allocated to the node that has just been removed, ensuring efficient memory usage and preventing leaks.

The **isClickable** function determines whether a specific tile can be clicked and matched with another tile. It checks the tile's surrounding conditions, ensuring no other tiles are blocking it from being matched. This mechanic is foundational to the gameplay, as it governs the player's ability to interact with the game board and make moves.

**getTopMostTile** identifies the tile at the position clicked by the player. It iterates through the stack of tiles from the topmost layer downwards, returning the first tile that matches the click location and is visible (i.e., not covered by other tiles). This function is integral for determining player interactions with the game board.

The **processClick** function is central to the gameplay mechanics of the Mahjong Solitaire game, handling what happens when a player clicks on a tile. Here's a breakdown of what each part of this function does:

**1.** **Reset Hint**: Initially, the function resets any existing hint information. This ensures that any visual indications provided as hints to the player are cleared when a new action is taken.

**2.** **Identify Clicked Tile**: It then determines which tile the player has clicked on using the **getTopMostTile** function, which checks the mouse position against the game tiles to find the topmost, clickable tile.

**3.** **Check Tile Clickability**: If the clicked tile (pointer) is found but is not clickable or if there are no matchable tiles left (gameState.matchable == 0), the game triggers a visual shake effect to indicate that the action is not valid.

**4.** **Tile Selection Logic**:

- *Sound Effect*: If the clicked tile is valid and different from the previously clicked tile (LastClicks->lastClicked), the game plays a sound to acknowledge the selection.

- *Color Change*: The function changes the color of the previously interacted tiles back to white, resetting their state visually. This includes the last clicked tile and the one before it.

- *Update Last Clicks*: It updates the tracking of the last two clicked tiles. The last clicked tile becomes the previously clicked, and the new clicked tile is set as the last clicked.

- *Highlight New Click*: The newly clicked tile is highlighted in red to indicate it is currently active or selected.

**5.** **Check for Removable Pair**:

- *Add to History*: If the last two clicked tiles form a removable pair (they can be removed according to the game rules), the function records this action in a linked list through the addBegin function. This is used for undoing actions if needed.

- *Update Score*: It calculates and adds points based on the match, updating the game’s score.

- *Reset Last Clicks*: Resets the information about the last interactions to prepare for the next player action.

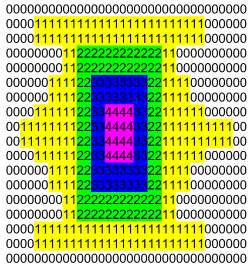
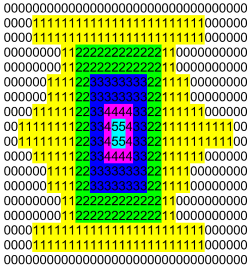
- *Update Matchable Tiles*: The function recalculates the tiles that can still be matched and updates the game state accordingly.

- *Update Remaining Tiles*: Reduces the count of remaining tiles by two since a pair has been matched and removed.

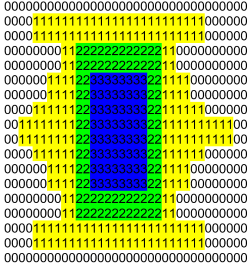
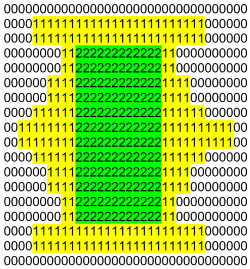
- *No More Matches*: If there are no tiles left that can be matched, another shake effect is triggered to inform the player.

**6.** **Return**: Finally, the function concludes its operations and returns, waiting for the next player interaction.

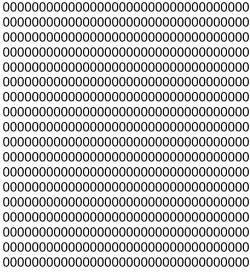
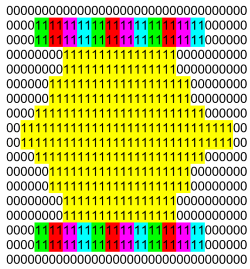
This function is a critical part of managing game interactions, ensuring that each player move is processed accurately and effectively, updating the game state, and providing feedback through visual and audio cues.



**THE LOGIC OF READING MAP TEXT FILES**



**TILES ARE CALCULATED IN QUARTERS AS SHOWN BELOW**



**RESULTS & CONCLUSION:**

The successful parts of the project are that there are no bugs in gameplay and that it has a scoreboard added to achieve a competitive and immersive design. On the other hand, the parts of the project that can be improved are that the number of maps available can be increased and tiles with different point values can be added.

Since our project is still open for development, if we want to develop it in the future, the elapsed time can also be taken into account instead of using only the score multiplier in the score calculations.

**SUGGESTIONS:**

Suggestions for the development of the project are that the drawings could be made more original and different color schemes could be used. If the project is to be taken to a more professional level, the game could be made available for online players. In this way, the game could become more competitive.

**RESOURCES:**

https://www.freepik.com/icon/dragon\_14236098#fromView=search&page=1&position=10&uuid=9aed1b64-472d-46f6-9f96-353cb73a5d9c

https://www.freepik.com/icon/greek\_14236070#fromView=resource\_detail&position=11

https://www.freepik.com/icon/bird\_14236080#fromView=resource\_detail&position=8

https://www.freepik.com/icon/centaurs\_14236101#fromView=resource\_detail&position=1

https://www.freepik.com/icon/bamboo\_2602999#fromView=resource\_detail&position=22

https://www.freepik.com/icon/dragon\_2602996

https://www.freepik.com/icon/mount-fuji\_410207#fromView=resource\_detail&position=32

https://www.freepik.com/icon/tokyo-tower\_410223#fromView=resource\_detail&position=4

https://www.freepik.com/icon/world\_14041592#fromView=resource\_detail&position=16

https://www.freepik.com/icon/world\_14041645#fromView=resource\_detail&position=6

https://www.freepik.com/icon/world\_14041586#fromView=resource\_detail&position=17

https://www.freepik.com/icon/compass\_2536643

https://www.freepik.com/icon/triangle\_9068259#fromView=search&page=1&position=18&uuid=8af37f02-e6a2-4330-bcc9-c100f9e52b23

https://www.freepik.com/icon/tile\_8347287#fromView=search&page=1&position=80&uuid=3a51745a-711e-48d5-904b-40183a6dbf41

https://www.freepik.com/icon/obwarzanek-krakowski\_5840792#fromView=resource\_detail&position=16

https://www.freepik.com/icon/game\_7960378#fromView=search&page=1&position=27&uuid=b26f35f6-704a-44c8-8881-2c0b87ddc66c

https://www.freepik.com/icon/checked\_7492346#fromView=search&page=1&position=64&uuid=820e5990-c5b9-4f75-9028-cb5794b1cba9

https://www.freepik.com/icon/hourglass\_15135175#fromView=search&page=1&position=2&uuid=e0d55c62-b80a-4676-a120-d107d455e7cb

https://www.freepik.com/icon/heart\_9131541#fromView=search&page=1&position=23&uuid=9c33e0de-7e4d-4cf4-85cf-b3d88e14147c

All rights of these designs are belong to the author and free to use.

Other designs are made by ourselves in Adobe Photoshop CS6.