**Change Log**

**Interface Change Log:**

(March 12, 2014) Added **Start Menu**- Allows the user to select game mode and load games

(March 12, 2014) Added **Play Form**- Supports legal movement of checker pieces

(March 13, 2014) Modified **Custom Mode**- No longer shares form with standard Mode

(March 13, 2014) Removed **Standard Mode**- Interface no longer visible, standard mode is now private that passes parameters to the **Form: Play**

(March 17, 2014) Removed **Game Timer**- The game play no longer has a time

**Variables Change Log:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Name** | **Declaration Location** | **Date Changed** | **Change Type** |
| GamesetupL | Global Integer | (March 12, 2014) | Added |
| GamesetupL | Global Integer | (March 12, 2014) | Added |
| GamesetupS | Global Integer | (March 12, 2014) | Added |
| GamesetupC | Global Integer | (March 12, 2014) | Added |
| M\_Badnu, | Local PictureBox | (March 12, 2014) | Added |
| C\_B1,C\_B2,C\_R1.C\_R1 | Local, Integer | (March 12, 2014) | Added |
| C\_B3,B\_B4,C\_R3,C\_R4 | Local PictureBox | (March 12, 2014) | Added |
| MoveCount | Local Integer | (March 17, 2014) | Added |
| M\_Infoarray | Local String Array(31) | (March 17, 2014) | Added |
| Gamesetup | Local Integer | (March 17, 2014) | Modified |
| C\_Seconds | Local Integer | (March 13, 2014) | Removed |
| C\_Minutes | Local Integer | (March 13, 2014) | Removed |

**Internal Implementation Change Log:**

(March 13, 2014) Added Logics to **Start** **Menu- Custom Mode, Standard Mode, Load Game,** also included a call to **End**\_G**ame** module

(March 17, 2014) Added Logics for Moving Pieces-**Jump**, **CrownKing**, **GetClick,** all of which are methods under the new Form: Play

(March 13, 2014) Added Private **Standard** **Mode**- Functionally equal to standard mode, no longer generates a graphical interface until Form: Play Loads

(March 12, 2014) Modified **Reset-** Reset functionally works the same, some state code removed to improve modularity

(March 12, 2014) Removed **Standard Mode**- Removed all internal coding involved with Standard Mode

(March 12, 2014) Removed Functions No longer Being Used- **Stoptime**(), **MakeInvis**(). MakeInvis still a part of the code, no calls being made at the moment

**Form: Start\_Menu (New)**

**Description (4.1)**: This succeeded the original form\_load (now named custom\_load). This is what the user initially sees when the game is loaded. It presents several options for gameplay, including: Start Game, Load Game, Custom Game and Standard Game. All of which are available to the user.

The form’s origin is largely due to the principles of modularity as this is a place that initializes one of the possible games modes and having them done in 1 location rather than making states inside the actual play module would drastically increase maintainability and reduce the possibility of information leaking through internal states.

**Interface Specification (4.2):**

StartGame()

Makes Custom and Standard Visible

Custom()

Enters Custom Setup Mode

Standard()

Enters Standard Game

Load()

Loads Saved Games

HighScore()

A list of the player’s achievements, not yet implemented

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Input | GamesetupC | Integer Array(32) | Global |
| Input | GamesetupS | Integer Array(32) | Global |
| Input | GamesetupL | Integer Array(32) | Global |

**Internal Implementation (4.5)-Selection Logic:**

Output: Gamesetup(32) Type: Integer Array(32)

|  |  |
| --- | --- |
| **Logic: Input** | **Output Action** |
| Custom | Initialize GameSetupC |
| Standard | Initialize GameSetupS |
| Load | Initialize GameSetupL |

**Internal Implementation (4.5)-Parameter setup:**

Output: Gamesetup(32) Type: Integer Array(32)

|  |  |
| --- | --- |
| **Logic: Input** | **Output Action** |
| 1 | Black |
| 2 | Red |
| 3 | Black King |
| 4 | Red King |

**Sub: Custommode\_load (New Name)**

**Description (4.1):** The Custom\_load sub is done as soon as the custom mode is loaded as a result, this sub becomes necessary for declaration of initial variables and drawing graphic pieces, as a result, the group will create the checker board array using this sub. In addition to drawing graphics, this sub also linked an array to the picture boxes in order to handle future action.

**Interface Specifications (4.2):**

New PictureBox()

Draw the Checkers board.

References: None

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variables** | **Variable Type** | **Declaration Location** |
| Output | C\_track | Integer | Local |
| Output | InitialX | Integer | Local |
| Output | C\_trackarray | PictureBox Array(31) | Global |
| Output | M\_custom | Boolean | Local |
| Output | M\_standard | Boolean | Local |
| Input | ErrorClick | PictureBox | Local |

**Internal Implementation (4.5):**

This code is meant to create picture boxes at given locations in a square matrix to represent the black checker boxes. The logic is to have a variable (i) increase by 50 each time and draw a new picture box there and every 4 I variables a J variable will be present to move to the next time. The implementation of the Logic can be found below

For I in range 0 to 7

For j in range 0 to 3

C\_trackarray(c\_track) = new PictureBox

C\_trackarray(c\_track).position = (I\*50 + InitialX,j\*50)

C\_track +=1

ErrorClick.sendtoback

Next

Next

**Sub: Game\_Timer (Revised)**

**Description (4.1):** This function counts the time elapsed since a game has started and stores them inside a seconds variable that the user will see.

This function is merely a small part of the game experience and as it is not related with any other modules, it would be illogical to count it as such.

**Interface Specification (4.2) (New):**

The time works internally and will not display a time at this deliverable

**Variables Used (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Output | C\_Minutes | Integer | Local |
| Output | C\_Seconds | Integer | Local |

**Internal Implementation (4.5):**

Implemented using the Tick operation provided in VB, the time runs every interval and this setting the internal to 1second updating the seconds local variable will provide a simple but functional timer. Updating the Minutes counter and resetting seconds is done at 60 seconds intervals.

Seconds +=1

If Seconds = 60

Seconds = 0

Minutes +=1

End If

**Sub: Reset (Revised)**

**Description (4.1):** This sub is meant to reset everything inside the checkers game to the state when the program was first opened. This is done with a menu strip drop down option.

This is its own module instead of using every other module is because the other modules is dependent of global variable states which allows this sub to not be dependent on every other sub which improves coupling.

**Interface Specification (4.2):**

M\_reset\_Click()

The interface for this sub is a menu strip drop down and click. The click will promptly remove everything on the game board and return it to the state similar to when the program just started.

**Variables (4.5) (2 Variables No Longer being used here):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Output | Trackarray | Picture box array (31) | Local |

**Internal Implementation (4.5) :**

Reset is now a simpler module and its responsibilities are to reset all the checker pieces when clicked in custom mode. This is implemented with the pseudo code below:

For I in range of trackarray

Trackarray(I).image = nothing

Next I

**Function: Stop\_Time (Removed)**

**Description (4.1):** Completely internal operation that resets the time variables involved in the counting process.

**Variables:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Output | C\_Minutes | Integer | Local |
| Output | C\_Seconds | Integer | Local |

**Internal Implementation:**

The implementation was just about setting c\_minutes and c\_seconds equal to 0.

Return C\_minutes = 0

Return C\_seconds = 0

**Sub: Standard\_Mode (Revised)**

**Description (4.1)**: Standard Mode has the information required to setup the game in the standard checkers mode. The standard game is trigger by a user click and updates the checker board with appropriate pieces.

The reasoning behind this module is to have it tie in, later, with the save functionality in the game. Both of these requires a logic from an array of integers for piece setting. Though not implemented in this specific deliverable, the additional parts will be implemented later

**Interface Specification (4.2) (New):**

This sub is no longer a part of the interface, now it is completely private and only updates the game setup for other modules

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Input | M\_custom | Integer Array(32) | Local |
| Input | M\_standard | Integer Array(32 | Local |

**Internal Implementation (4.5):**

Output: Gamesetup(32) Type: Integer Array(32)

|  |  |
| --- | --- |
| **Logic: Input** | **Output Action** |
| 1 | Black |
| 2 | Red |
| 3 | Black King |
| 4 | Red King |

**Form: Play (New)**

**Description (4.1)**: This is the form where the entire playing experience is done, it is an individual form and forms a module by itself. This is the largest module that there is in this project and has many methods all specifically pertaining to setting up the board and logic for moving/jumping the pieces.

The group decided that a logical convergence for all previous calculations would lead here for an actual playing experience. Due to the fact that this module takes setups and contains move logic, it supports the modularity principles. This module minimal logic from other modules and has very high cohesion, making it an ideal choice for a well-defined component of this design.

**Interface Specification (4.2):**

Gamesetup()

Sets up the game as dictate by the user previously, either through loading, custom setup or standard setup

Movepiece()

Allows the user to make legal moves, including sideways moving, jumping and reverse moving for kings

CrownKing()

If the user moves a piece to the opposing end, the piece is immediately crowned king and gains the movability of a king

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Input | GamesetupC | Integer Array(32) | Global |
| Input | GamesetupS | Integer Array(32) | Global |
| Input | GamesetupL | Integer Array(32) | Global |
| Input | M\_Badnum | Picturebox | Local |
| Internal | C\_B1 | Integer | Local |
| Internal | C\_B2 | Integer | Local |
| Internal | C\_R1 | Integer | Local |
| Internal | C\_R2 | Integer | Local |
| Internal | C\_B3 | Picturebox | Local |
| Internal | C\_B4 | Picturebox | Local |
| Internal | C\_R3 | Picturebox | Local |
| Internal | C\_R4 | Picturebox | Local |
| Internal | BlackT | Boolean | Local |
| Internal | RedT | Boolean | Local |
| Internal | MoveCount | Integer | Local |
| Output | M\_Infoarray | String Array (31) | Local |

**Internal Implementation (4.5)-Board Setup**

Similarity to previous custom mode, this section is going to require a game board, as setup by the following pseudo code:

For I in range 0 to 7

For j in range 0 to 3

C\_trackarray(c\_track) = new PictureBox

C\_trackarray(c\_track).position = (I\*50 + InitialX,j\*50)

C\_track +=1

ErrorClick.sendtoback

Next

Next

**Internal Implementation (4.5)-Move Logic**

Output: {C\_B1, C\_B2, C\_R1, C\_R2} Types: Integers

|  |  |  |
| --- | --- | --- |
| **Logic: Row Number** | **Output Action: C\_R1,C\_R2** | **Output Action: C\_B1,C\_B2** |
| Odd | 3,4 | 4,5 |
| Even | 4,5 | 3,4 |

**Internal Implementation (4.5)-Detect Piece Logic**

Updates: MoveCount Types: Integers

|  |  |  |
| --- | --- | --- |
| **Logic: Click** | **Update: Movecount (Current Value: 0)** | **Update: Movecount (Current Value: 1)** |
| Object: Empty | 0 | 1 |
| Object: Not Empty | 1 | 0 |

**Internal Implementation (4.5)-Jump Logic**

This logic follows the move piece logic, it simply checks if the moves will land on the pieces of another piece or not. If a jump can be made, it removes the occupied space from possible moves and adds the jumped location

Output: {C\_B1, C\_B2, C\_R1, C\_R2} Types: Integers

|  |  |  |
| --- | --- | --- |
| **Logic: Next Piece** | **Update: Same Color** | **Update: Different Color** |
| Top Right Corner: Empty | C\_B1, C\_B2, C\_R1, C\_R2 | N/A |
| Top Right Corner: Not Empty | C\_B1, C\_B2, C\_R1, 0 | C\_B1, C\_B2, C\_R1, C\_R2+4 |
| Top Left Corner: Empty | C\_B1, C\_B2, C\_R1, C\_R2 | N/A |
| Top Left Corner: Not Empty | C\_B1, C\_B2, 0 C\_R2 | C\_B1, C\_B2, C\_R1+4, C\_R2 |
| Bottom Left Corner: Empty | C\_B1, C\_B2, C\_R1, C\_R2 | N/A |
| Bottom Left Corner: Not Empty | 0, C\_B2, C\_R1, C\_R2 | C\_B1+4, C\_B2, C\_R1, C\_R2 |
| Bottom Right Corner: Empty | C\_B1, C\_B2, C\_R1, C\_R2 | N/A |
| Bottom Right Corner: Not Empty | C\_B1, 0, C\_R1, C\_R2 | C\_B1, C\_B2+4, C\_R1, C\_R2 |

**Internal Implementation (4.5)-King Logic**

Outputs: {King, No Change} Types: Picturebox

|  |  |
| --- | --- |
| **Piece Location/ Color** | **Output** |
| 28-31/ Red | Red King |
| 28-31/ Black | No Change |
| 0-3/ Red | No Change |
| 0-3/ Black | Black King |
| Else | No Change |

**Sub: Save\_Game (New)**

**Description (4.1)**: This is a drop down menu option in the play game form that allows the user to save the current progress. Once the save is complete, the user is free to continue playing or close the game and load their progress back next time.

The save functionality is put into one module as it is necessary to maintain modularity, this area of code links to an form object and is well defined by itself.

**Interface Specification (4.2):**

Save\_Game()

Following the click of this button on the checkers interface, the game will be saved

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Output | Gamesetup | Integer Array(32) | Local |

**Internal Implementation (4.5):**

The implementation of this module is a simple write to file operation as modelled by the following pseudo code:

Dim writer as print writer

Write(../SavedGame.txt)

Writer(gamesetup)

**Sub: Load\_Game (New)**

**Description (4.1)**: This is a button in the start menu that allows the user to load their previously saved games, if no game was previous saved, the interface returns an error message notifying the user.

This sub was decomposed into a module by itself as it is an object on a form and contains a well-defined area of code.

**Interface Specification (4.2):**

Load\_Game()

Following the click of this button, the previously saved game will be loaded or an error message will be shown indicating that no saved game exists

**Variables (4.5):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable Use** | **Variable Name** | **Variable Type** | **Declaration Location** |
| Input | GamesetupC | Integer Array(32) | Global |
| Input | GamesetupS | Integer Array(32) | Global |
| Input | GamesetupL | Integer Array(32) | Global |

**Internal Implementation (4.5):**

The implementation of this module is a simple read file operation as modelled by the following pseudo code:

Dim reader as print writer

read(../SavedGame.txt)

if reader.empty = true

msgbox (“No Saved Game”)

end if

gamesetupC(0 to 32) = 0

gamesetupS(0 to 32) = 0

gamesetupL(0 to 32) = reader