TEST REPORT

Designed by struct by_lightning{};

Group 6

Kuir Aguer, Brendan Duke, Jean Ferreira,

Zachariah Levine and Pranesh Satish

Description of the testing conducted and results for the modules in the Connect 4 game interface.

All of the expected outputs was the actual output expected as described by each method implementation. The description for each method implementation can be found in their respective Module Interface Specification and Module Internal Design.

Table of Contents

(syntax: functionName(parameters) : return type)

3.5 deleteStillToken(FallingToken *fallingToken) : void

3.6 drawFallingToken(FallingToken *token) : void

3.7 transitionSetupRender() : void

3.8 updateFallingToken(FallingToken *fallingToken, float dt) : void

3.9 displayBoard(void) : void

3.10 displaySetupTokens(void) : void

3.11 displayMainMenu(void) : void

3.12 mainMenuRender() : void

3.13 highlightToken(int row, int col) : void

4. linkedList.h

4.1 Template <T> class List

4.1.1 addToList(T *newitem, List<T> *list) : List<T> *

4.1.2 deleteFromList(T *toDeleteItem, List<T> *list) : List<T>*

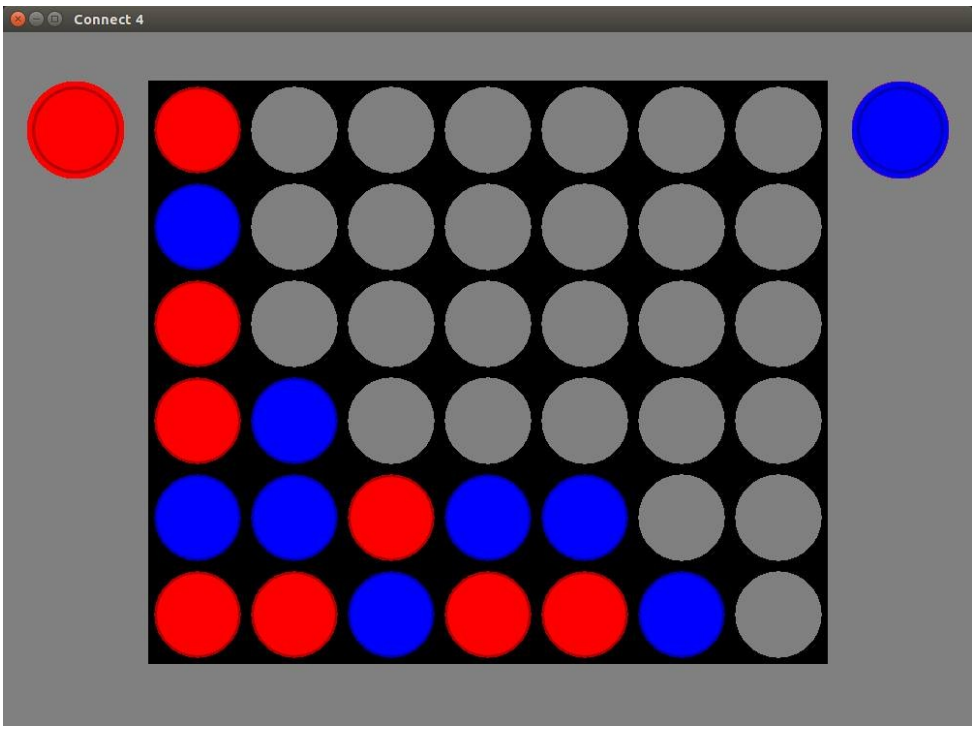
4.1.3 traverseList(void (*f)(T *item), List<T> *list) : void

5. sdl2_connect4.h

5.1 connect4() : int

Notes:

(i) To avoid repetition throughout this report, whenever the symbol, b, is seen as input for a function, it refers to a valid state of an object of type Board as depicted by the following image :



(ii) Any input with the format < *action* > implies the import is some action performed on client side, such as mouse clicks, and opening/closing a program.

1. board.h

1.1 board_create() : Board

Input: none
Expected Output: A new board object that is empty

1.2 board_destroy(Board b) : Board
Input: A board object

Expected Output: The input board object should be freed from memory, nothing should be returned

1.3 board_checkCell(Board b, int row, int col) : Token

Input: b, 1, 1
Expected Output: BLUE

Input: b, 1, 6
Expected Output: EMPTY

Input: b, 0, 0
Expected Output: RED

1.4 board_dropToken(Board b, Token token, int col) : int

Input: b, RED, 3

Expected Output: 0 (successful token drop)

Input: b, BLUE, 0

Expected Output: -1 (column is full, cannot drop token)

1.5 board_dropPosition(Board b, int col) : int

Input: b, 5
Expected Output: 1

Input: b, 1
Expected Output: 3

Input: b, 0
Expected Output: -1 (full column, so row is out of bounds)

1.6 board_empty(Board b) : void

Input: b
Expected Output: An empty board, with no tokens in play

2. gameLogic.h

2.1 handleMainMenuMouseClick(int x, int y) : MenuState

Input: 450, 580
Expected Output: Open the Setup game mode

Input: 480, 750
Expected Output: Closes the game window

Input: 251, 30
Expected Output: Nothing, remain in the Main Menu

2.2 checkBoardStatus(Board b) : BoardStatus

Input: b
Expected Output: IN_PROGRESS

Input: b (where red token has 4 in a row in the setup mode)
Expected Output: INVALID_BOARD

Input: b (where red token has 4 in a row in 1 player mode)
Expected Output: RED_WON

3. gaphics.h

3.1 init() : bool

Input: < starting the program >
Expected Output: true

Input: < starting the program

Expected Output: false (if error(s) occurred when starting the program such as: incorrect installation of SDL, and invalid window creation.)

3.2 loadMedia() : bool

Input: < starting the program >

Expected Output: true (all media file successfully found in given path)

Input: < starting the program >

Expected Output: false (one or more media files failed to load)

3.3 close_sdl() : void

Input: **<** Closing the game >

Expected Output: Sets all loaded media to NULL and program window closes

3.4 dropToken(Board b, Token tokenColour, int col) : bool

Input: b, RED, 3

` **Expected Output:** true

Input: b, BLUE, 0

Expected Output: false (column already full)

3.5 deleteStillToken(FallingToken *fallingToken) : void

Input: fallingToken->isFalling == true

Expected Output: Change noting

Input: fallingToken->isFalling == false

Expected Output: Removes the token from the structure

3.6 drawFallingToken(FallingToken *fallingToken) : void

Input: fallingToken->token == RED

Expected Output: Render (display) a red token in the appropriate column and row

3.7 transitionSetupRender() : void

Input: < Staring the setup mode gameplay >
Expected Output: All media required for the setup game mode is rendered onto the screen

3.8 updateFallingToken(FallingToken *fallingToken, float dt) : void

Input: fallingToken with the following properties:
fallingToken->isFalling == true
fallingToken->y == a
fallingToken->v == b
dt == c
Expected Output: using the values a, b, and c, calculate the new velocity and height of the falling token.

3.9 displayBoard(void) : void

Input: < start any game mode >
Expected Output: The game board is rendered on the screen

3.10 displaySetupTokens(void) : void

Input: < Starting setup game mode >
Expected Output: Renders any setup tokens needed in the setup game board

3.11 displayMainMenu(void) : void

Input: <Starting program or returning to the main menu from another screen >
Expected Output: The main menu is displayed

3.12 mainMenuRender() : void

Input: <Starting program or returning to the main menu from another screen >
Expected Output: Renders images of the main menu

3.13 highlightToken(int row, int col) : void

Input: 2,1
Expected Output: Token located in row 3, column 1 is highlighted

4. linkedList.h

4.1 Template <T> class List

This list is responsible for keeping track of all moving tokens in the game

To simplify testing we will assume the list is represented as:

{ obj0, obj1, obj2, obj3 }

4.1.1 addToList(T *newitem, List<T> *list) : List<T> *

Input: BLUE, { RED, RED, BLUE }
Expected Output: { BLUE, RED, RED, BLUE }

4.1.2 deleteFromList(T *toDeleteItem, List<T> *list) : List<T>*

Input: RED, { RED, BLUE, BLUE, RED }
Expected Output: { BLUE, BLUE, RED }

Input: BLUE, { }
Expected Output: NULL

4.1.3 traverseList(void (*f)(T *item), List<T> *list) : void

Input: free(), { BLUE, BLUE, RED }
Expected Output: { } (The list is traversed, and each element is set to NULL)

5. sdl2_connect4.h

5.1 connect4() : int

Input: <Starting the program> (This is the main function that starts the program)
Expected Output: 0 (When game closes successfully)