FEH SDP

1.0

Generated by Doxygen 1.8.20

Chapter 1

README

FEH Software Design Project - Connect4

Pietro Lavezzo, Adam Exley, Lauren Pokonosky

Additional documentation can be found on the `offical website`

1.1 Compiling Code and Running Connect 4

As long as you have the `Proteus Simulator` installed, compilation is simple.

1. Download the code as a ∗∗.zip∗∗ file from the project `GitHub page`
2. Unzip this file (usually right click -> Extract All/ Unzip)
3. Open the resulting folder in Visual Studio
4. Run **mingw32-make** in the terminal
5. Using the Proteus Simulator, run the resulting **game** file

1.2 File Structure

Header files are used, `here's a reference on how they work.`

∗∗.h∗∗ files contain the *declarations* of classes and associated functions.

∗∗.cpp∗∗ files contain the *definitions* of all functions in the corresponding ∗.h∗ file.

main.cpp contains all the basic code for the project, as in any other simple program.

1.3 Classes and Functions Used

Brief descriptions provided; see documentation or code for more information.

Bold functions are public

1.3.1 AI

- **AI()** - Constructs AI object
- **setDifficulty()** - Sets AI difficulty
- **pickMove()** - AI makes a move based on the current game state
- easyMove() - A random column is selected as a move
- isValidMove() - Modified from board class
- inARow() - Counts number of playable chip combos in a row given a game state
- fromCenter() - Gives the average dispersion of a player's chips from the center

1.3.2 Board

- **Board()** - Constructs board object
- **Reset()** - Resets object to inital state
- **DrawBoard()** - Draws full board
- **DrawChips()** - Draws only chips of board
- **DrawBoardMenu()** - Draws menu buttons to left of board
- **isValidMove()** - Checks that specified move is valid
- **getCurrentColumn()** - Finds the column that the player selects
- **getAIMove()** - Imports AI move
- **dropChip()** - Animates the dropping of a chip
- occludeChip() - Covers over a dropping chip with an approximation of the board
- **updateGameState()** - Adds new chip to board
- pushGameState() - Exports internal game state to external array
- **checkWin()** - Checks for a win or tie

1.3.3 Game

- **Game()** - Constructs object, sets current player to 1
- **Reset()** - Resets current player
- **setSingleplayer()** - Sets if game is single or multiplayer
- **getCurrentPlayer()** - Returns current player ID
- **isPlayerTurn()** - Returns if it is a human's turn
- **switchPlayer()** - Switches current player

Includes the classless function:

- **waitForInput()** - Waits for and returns location of a touch on the screen

1.3.4 Image

- **Image()** - Constructs an Image object, sets size and options
- **Draw()** - Converts, optimizes, and plots an image from a hexadecimal color array
- lookupColor() - Finds the cloesest Proteus color to a hexadecimal color
- HorizLineOptimize() - Attempts to change small discrepencies in color in lines
- PlotImg() - Displays image on screen

1.3.5 Menu

- **Menu()** - Constructs Menu object and sets menu state and statistics
- **checkTouchLocation()** - Waits for a touch and execues the function corresponding to the touch location
- **showMain()** - Draws main menu
- showPlay() - Draws game selection menu
- showDifficulty() - Draws singleplayer difficulty selection screen
- showStats() - Draws statistics screen
- showInstructions() - Shows game instructions
- showCredits() - Shows game credits
- drawReturnExit() - Draws the return and exit buttons present on most menus
- **getDifficulty()** - Returns selected difficulty
- **getSingleplayer()** - Returns if singleplayer mode was selected
- **updateStats()** - Updates stored statistics
- **showWinLoss()** - Shows the win/loss/tie screen
- displayTA() - Draws a random TA for win/loss/tie screens
- **showExit()** - Shows the game exit screen forever

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AI Class Reference

Computer controlled AI that has easy and hard difficulties. Makes decisions to output a desired move.

```
#include <AI.h>
```

Public Member Functions

- AI ()
  Constructs AI object, setting difficulty to easy, ai_id to 2, and player_id to 1.
- void setDifficulty (bool diff)
  Sets difficulty variable of AI object.
- int pickMove (const int game_state_array[BOARD_ROWS][BOARD_COLUMNS])
  Picks a move given the current game state.

4.1.1 Detailed Description

Computer controlled AI that has easy and hard difficulties. Makes decisions to output a desired move.

Easy difficulty is a mere random number generator. Hard difficulty does analysis on 1 future move using arbitrary scores for certain conditions.

Definition at line 23 of file AI.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AI()

```
AI::AI ( )
```

Constructs AI object, setting difficulty to easy, ai_id to 2, and player_id to 1.

Author

Adam Exley

Definition at line 7 of file AI.cpp.

4.1.3 Member Function Documentation

4.1.3.1 pickMove()

```
int AI::pickMove (
             const int game_state_array[BOARD_ROWS][BOARD_COLUMNS] )
```

Picks a move given the current game state.

Looks at possible moves and moves that could be played on those moves, and gives each move a score based on that. Picks the highest scoring move.

Returns

The column to make a move in

Author

Adam Exley

Strategies:

- Prefer the center of the board
- Try to get 4 in a row
- Try to get 3 in a row
- Try to get 2 in a row
- Don't let the opponent do any of the above

Definition at line 19 of file AI.cpp.

4.1.3.2 setDifficulty()

```
void AI::setDifficulty (
             bool diff )
```

Sets difficulty variable of AI object.

Parameters

diff	AI difficulty (0 = Easy, 1 = Hard)

Author

Adam Exley

Definition at line 13 of file AI.cpp.

The documentation for this class was generated from the following files:

- AI.h
- AI.cpp

4.2 Board Class Reference

Stores and dislays game state. Responsible for keeping track of, and updating, where chips are located.

```
#include <Board.h>
```

Public Member Functions

- Board (int game_state[BOARD_ROWS][BOARD_COLUMNS])

  Constructs an object of the Board class, setting local and global arrays.
- void Reset (int game_state[BOARD_ROWS][BOARD_COLUMNS])

  Resets all variables.
- void DrawBoard ()

  Draws entire game board with chips.
- void DrawChips ()

  Draws only the chips of the game. Skips redrawing board.
- void DrawBoardMenu ()

  Draws the options menu to the left of the gameboard.
- bool isValidMove ()

  Checks if making a move in current_column is valid by seeing if the top row of current_column is occupied.
- int getCurrentColumn ()

  Waits for a touch and returns the value of the column that was touched. Also stores this value in the current_column variable.
- void getAIMove (int column)

  Changes current_column. For use with AI.
- void dropChip (int current_player)

  Animates chip dropping.
- void updateGameState (int player, int game_state[BOARD_ROWS][BOARD_COLUMNS])

  Updates the game_state array to contain a new chip in a certain column. Uses the player paramater as the value in the array.
- int checkWin ()

  Determines if a player has won. Tests all possible orientations and positions of getting 4 in a row.

4.2.1 Detailed Description

Stores and dislays game state. Responsible for keeping track of, and updating, where chips are located.

Definition at line 19 of file Board.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Board()

```
Board::Board (
              int game_state[BOARD_ROWS][BOARD_COLUMNS] )
```

Constructs an object of the Board class, setting local and global arrays.

Initializes all elements of the board_state array to 0. Then pushes the contents of board_state to game_state.

Parameters

game_state	Array to push contents board_state into

Author

Adam Exley

Definition at line 7 of file Board.cpp.

4.2.3 Member Function Documentation

4.2.3.1 checkWin()

```
int Board::checkWin ( )
```

Determines if a player has won. Tests all possible orientations and positions of getting 4 in a row.

Returns

The ID of the winning player (1 or 2), 3 for a tie, otherwise returns 0.

Author

Adam Exley

Definition at line 336 of file Board.cpp.

4.2.3.2 DrawBoard()

```
void Board::DrawBoard ( )
```

Draws entire game board with chips.

Calls DrawChips() to draw game chips

Author

Adam Exley

Definition at line 39 of file Board.cpp.

4.2.3.3 DrawBoardMenu()

```
void Board::DrawBoardMenu ( )
```

Draws the options menu to the left of the gameboard.

Author

Lauren Pokonosky

Definition at line 97 of file Board.cpp.

4.2.3.4 DrawChips()

```
void Board::DrawChips ( )
```

Draws only the chips of the game. Skips redrawing board.

Author

Adam Exley

Definition at line 59 of file Board.cpp.

4.2.3.5 dropChip()

```
void Board::dropChip (
             int current_player )
```

Animates chip dropping.

Parameters

current_player	The current player ID, used to set chip color

Author

Pietro Lavezzo

Definition at line 186 of file Board.cpp.

4.2.3.6 getAIMove()

```
void Board::getAIMove (
             int column )
```

Changes current_column. For use with AI.

Parameters

column	Value to set current_column (and thus the move)

Author

Pietro Lavezzo

Definition at line 179 of file Board.cpp.

4.2.3.7 getCurrentColumn()

```
int Board::getCurrentColumn ( )
```

Waits for a touch and returns the value of the column that was touched. Also stores this value in the current_column variable.

Returns

A column value, or MAIN_MENU_CALL_VALUE or EXIT_CALL_VALUE

Author

Lauren Pokonosky

Definition at line 136 of file Board.cpp.

4.2.3.8 isValidMove()

```
bool Board::isValidMove ( )
```

Checks if making a move in current_column is valid by seeing if the top row of current_column is occupied.

Returns

True/false of if said move can be performed

Author

Pietro Lavezzo

Definition at line 119 of file Board.cpp.

4.2.3.9 Reset()

```
void Board::Reset (
             int game_state[BOARD_ROWS][BOARD_COLUMNS] )
```

Resets all variables.

Same as constructor.

Author

Adam Exley

Definition at line 23 of file Board.cpp.

4.2.3.10 updateGameState()

```
void Board::updateGameState (
             int player,
             int game_state[BOARD_ROWS][BOARD_COLUMNS] )
```

Updates the game_state array to contain a new chip in a certain column. Uses the player paramater as the value in the array.

Parameters

player	Integer value (1 or 2) to set the array cell as
game_state	Array to update

Author

Adam Exley

Definition at line 308 of file Board.cpp.

The documentation for this class was generated from the following files:

- Board.h
- Board.cpp

4.3 Game Class Reference

Controls flow of the game once in play.

```
#include <Game.h>
```

Public Member Functions

- Game ()

  Constructs, sets current player to 1.
- void Reset ()

  Resets current player to 1.
- void setSingleplayer (bool single)

  Sets singleplayer status.
- int getCurrentPlayer ()

  Gets current player.
- bool isPlayerTurn ()

  Returns if it is currently a huamn player's turn.
- void switchPlayer ()

  switches the current player

4.3.1 Detailed Description

Controls flow of the game once in play.

Keeps track of AI/Player turns.

Definition at line 14 of file Game.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Game()

```
Game::Game ( )
```

Constructs, sets current player to 1.

Author

Adam Exley

Definition at line 7 of file Game.cpp.

4.3.3 Member Function Documentation

4.3.3.1 getCurrentPlayer()

```
int Game::getCurrentPlayer ( )
```

Gets current player.

Returns

Current player ID

Author

Pietro Lavezzo

Definition at line 27 of file Game.cpp.

4.3.3.2 isPlayerTurn()

```
bool Game::isPlayerTurn ( )
```

Returns if it is currently a huamn player's turn.

Returns

True/False as to if it is a human's turn

Author

Pietro Lavezzo

Definition at line 33 of file Game.cpp.

4.3.3.3 Reset()

```
void Game::Reset ( )
```

Resets current player to 1.

Author

Pietro Lavezzo

Definition at line 14 of file Game.cpp.

4.3.3.4 setSingleplayer()

```
void Game::setSingleplayer (
            bool single )
```

Sets singleplayer status.

Author

Pietro Lavezzo

Definition at line 20 of file Game.cpp.

4.3.3.5 switchPlayer()

```
void Game::switchPlayer ( )
```

switches the current player

Author

Adam Exley

Definition at line 52 of file Game.cpp.

The documentation for this class was generated from the following files:

- Game.h
- Game.cpp

4.4 Image Class Reference

This class displays images exported from `https://www.piskelapp.com`.

`#include <Image.h>`

Public Member Functions

- Image (int w, int h)

  Default Image constructor. Enables all 22 LCD Colors for render and drawing.
- Image (int w, int h, const int enabled_colors[ ], int num_enabled, const int do_not_draw[ ], int num_no_draw)

  Image constructor. Enabled colors are specified.
- void Draw (const uint_fast32_t image_array[ ], int scale=1, bool optimize=true, int x_off=0, int y_off=0)

  Plots a given image with a certain scale, optimization, and coordinate offset.

4.4.1 Detailed Description

This class displays images exported from `https://www.piskelapp.com`.

Author

Adam Exley Includes functions to convert and then display complex images. Honestly it's just a party trick. A majority of the hexadecimal conversion code had been written before this project in order to try to display images, only to realize that the Proteus was unable (at the time) to reproduce more than 7 colors.

Definition at line 19 of file Image.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Image() [1/2]

```
Image::Image (
                int w,
                int h )
```

Default Image constructor. Enables all 22 LCD Colors for render and drawing.

Parameters

w	Image width in pixels
h	Image height in pixlels

Author

Adam Exley

Definition at line 6 of file Image.cpp.

4.4.2.2 Image() [2/2]

```
Image::Image (
            int w,
            int h,
            const int enabled_colors[],
            int num_enabled,
            const int do_not_draw[],
            int num_no_draw )
```

Image constructor. Enabled colors are specified.

Parameters

w	Image width in pixels
h	Image height in pixels
enabled_colors	Int array corresponding to enabled colors. Use defines in FEHLCDColors.h
num_enabled	Number of values in enabled_colors
do_not_draw	Int array that stores colors that should be rendered, but should not be drawn
num_no_draw	Number of values in do_not_draw

Author

Adam Exley

Definition at line 28 of file Image.cpp.

4.4.3 Member Function Documentation

4.4.3.1 Draw()

```
void Image::Draw (
            const uint_fast32_t image_array[],
            int scale = 1,
            bool optimize = true,
            int x_off = 0,
            int y_off = 0 )
```

Plots a given image with a certain scale, optimization, and coordinate offset.

Converts hexadecimal color array before optionally optimizing it and then displaying it on screen

Parameters

image_array	Source array of image to plot
scale	(Optional) Default 1. Factor to scale image by. Integers only.
optimize	(Optional) Default True. Whether to apply horizontal line optimization to the image.
x_off	(Optional) Default 0. X-coordinate offset of top left corner of image.
y_off	(Optional) Default 0. Y-coordinate offset of top left corner of image.

Author

Adam Exley

Definition at line 179 of file Image.cpp.

The documentation for this class was generated from the following files:

- Image.h
- Image.cpp

4.5 Menu Class Reference

Displays all menus for game and stores current menu state.

```
#include <Menu.h>
```

Public Member Functions

- Menu ()

  Constructs a Menu object. Sets initial values for stats/states.
- bool checkTouchLocation (int x, int y)

  Executes the menu function associated with a certain touch location.
- void showMain ()

  Displays main menu.
- bool getDifficulty ()
- bool getSingleplayer ()
- void updateStats (int winner, int turns)

  Updates the stats stored in the menu object.
- void showWinLoss (int winner)

  Shows win or loss screens with messages and a random TA.
- void showExit ()

  Displays an exit screen thanking the user for playing. Runs forever.

4.5.1 Detailed Description

Displays all menus for game and stores current menu state.

Also displays win/loss screens and statistics.

Definition at line 24 of file Menu.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Menu()

```
Menu::Menu ( )
```

Constructs a Menu object. Sets initial values for stats/states.

Author

Adam Exley

Definition at line 8 of file Menu.cpp.

4.5.3 Member Function Documentation

4.5.3.1 checkTouchLocation()

```
bool Menu::checkTouchLocation (
             int x,
             int y )
```

Executes the menu function associated with a certain touch location.

Parameters

x	X-location of touch
y	Y-location of touch

Returns

True/false corresponding as to if the menu selection should continue looping

Authors

Adam Exley, Lauren Pokonosky

Definition at line 16 of file Menu.cpp.

4.5.3.2 getDifficulty()

```
bool Menu::getDifficulty ( )
```

Returns

Value of difficulty variable

Author

Adam Exley

Definition at line 281 of file Menu.cpp.

4.5.3.3 getSingleplayer()

```
bool Menu::getSingleplayer ( )
```

Returns

Value of singleplayer variable

Author

Adam Exley

Definition at line 286 of file Menu.cpp.

4.5.3.4 showExit()

```
void Menu::showExit ( )
```

Displays an exit screen thanking the user for playing. Runs forever.

Author

Lauren Pokonosky

Definition at line 430 of file Menu.cpp.

4.5.3.5 showMain()

```
void Menu::showMain ( )
```

Displays main menu.

Author

Lauren Pokonosky

Definition at line 110 of file Menu.cpp.

4.5.3.6 showWinLoss()

```
void Menu::showWinLoss (
            int winner )
```

Shows win or loss screens with messages and a random TA.

Parameters

winner	Integer correspondng to winning player (1 or 2)

Author

Lauren Pokonosky

Definition at line 320 of file Menu.cpp.

4.5.3.7 updateStats()

```
void Menu::updateStats (
             int winner,
             int turns )
```

Updates the stats stored in the menu object.

Parameters

winner	Integer correspondng to winning player (1 or 2)

Author

Lauren Pokonosky

Definition at line 293 of file Menu.cpp.

The documentation for this class was generated from the following files:

- Menu.h
- Menu.cpp

Chapter 5

File Documentation

5.1 AI.cpp File Reference

Contains all code for the AI class. Declarations are in AI.h.

```
#include "FEHRandom.h"
#include "AI.h"
```

5.1.1 Detailed Description

Contains all code for the AI class. Declarations are in AI.h.

5.2 AI.h File Reference

Contains declarations of AI class functions.

```
#include "FEHLCD.h"
#include "FEHUtility.h"
#include "FEHRandom.h"
#include "FEHSD.h"
#include "config.h"
#include <climits>
#include <cmath>
```

Classes

- class AI

  Computer controlled AI that has easy and hard difficulties. Makes decisions to output a desired move.

5.2.1 Detailed Description

Contains declarations of AI class functions.

5.3 alex.c File Reference

Data for Alex's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define ALEX_FRAME_COUNT 1
- #define ALEX_FRAME_WIDTH 149
- #define ALEX_FRAME_HEIGHT 240

5.3.1 Detailed Description

Data for Alex's picture. Exported from piskel.com.

5.3.2 Macro Definition Documentation

5.3.2.1 ALEX_FRAME_COUNT

```
#define ALEX_FRAME_COUNT 1
```

Definition at line 7 of file alex.c.

5.3.2.2 ALEX_FRAME_HEIGHT

```
#define ALEX_FRAME_HEIGHT 240
```

Definition at line 9 of file alex.c.

5.3.2.3 ALEX_FRAME_WIDTH

```
#define ALEX_FRAME_WIDTH 149
```

Definition at line 8 of file alex.c.

5.4 bailey.c File Reference

Data for Bailey's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define BAILEY_FRAME_COUNT 1
- #define BAILEY_FRAME_WIDTH 167
- #define BAILEY_FRAME_HEIGHT 240

5.4.1 Detailed Description

Data for Bailey's picture. Exported from piskel.com.

5.4.2 Macro Definition Documentation

5.4.2.1 BAILEY_FRAME_COUNT

```
#define BAILEY_FRAME_COUNT 1
```

Definition at line 6 of file bailey.c.

5.4.2.2 BAILEY_FRAME_HEIGHT

```
#define BAILEY_FRAME_HEIGHT 240
```

Definition at line 8 of file bailey.c.

5.4.2.3 BAILEY_FRAME_WIDTH

```
#define BAILEY_FRAME_WIDTH 167
```

Definition at line 7 of file bailey.c.

5.5 Board.cpp File Reference

```
#include "Board.h"
```

5.5.1 Detailed Description

This contains all the definitions for the Board class. Declarations are in Board.h

5.6 Board.h File Reference

Contains the class definition and prototypes for the Board class.

```
#include "FEHLCD.h"
#include "FEHUtility.h"
#include "FEHRandom.h"
#include "config.h"
#include "Game.h"
#include <cmath>
```

Classes

- class Board

  Stores and dislays game state. Responsible for keeping track of, and updating, where chips are located.

Macros

- #define _USE_MATH_DEFINES

5.6.1 Detailed Description

Contains the class definition and prototypes for the Board class.

5.6.2 Macro Definition Documentation

5.6.2.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

Definition at line 11 of file Board.h.

5.7 bridgette.c File Reference

Data for Bridgette's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define BRIDGETTE_FRAME_COUNT 1
- #define BRIDGETTE_FRAME_WIDTH 209
- #define BRIDGETTE_FRAME_HEIGHT 240

5.7.1 Detailed Description

Data for Bridgette's picture. Exported from piskel.com.

5.7.2 Macro Definition Documentation

5.7.2.1 BRIDGETTE_FRAME_COUNT

```
#define BRIDGETTE_FRAME_COUNT 1
```

Definition at line 6 of file bridgette.c.

5.7.2.2 BRIDGETTE_FRAME_HEIGHT

```
#define BRIDGETTE_FRAME_HEIGHT 240
```

Definition at line 8 of file bridgette.c.

5.7.2.3 BRIDGETTE_FRAME_WIDTH

```
#define BRIDGETTE_FRAME_WIDTH 209
```

Definition at line 7 of file bridgette.c.

5.8 config.h File Reference

Contains global constants that can be modified to change program behavior.

Macros

- #define BUTTON_COLOR SCARLET
- #define TEXT_COLOR WHITE
- #define BOARD_COLOR BLUE
- #define BOARD_BACKGROUND_COLOR BLACK
- #define PLAYER_1_COLOR RED
- #define PLAYER_2_COLOR YELLOW
- #define SQUARE_SIDE 40
- #define HOLE_RADIUS 15
- #define BOARD_ROWS 6
- #define BOARD_COLUMNS 7
- #define G 9.8
- #define MAIN_MENU_CALL_VALUE 10
- #define EXIT_CALL_VALUE 11
- #define CENTER_WEIGHT 10
- #define ONE_PLAYABLE 10
- #define TWO_IN_A_ROW 30
- #define THREE_IN_A_ROW 85
- #define FOUR_IN_A_ROW 100000
- #define DISABLE_VALUE 4000
- #define PER_SIDE 6

    Optimization Constant.

5.8.1 Detailed Description

Contains global constants that can be modified to change program behavior.

5.8.2 Macro Definition Documentation

5.8.2.1 BOARD_BACKGROUND_COLOR

```
#define BOARD_BACKGROUND_COLOR BLACK
```

Definition at line 12 of file config.h.

5.8.2.2 BOARD_COLOR

```
#define BOARD_COLOR BLUE
```

Definition at line 11 of file config.h.

5.8.2.3 BOARD_COLUMNS

```
#define BOARD_COLUMNS 7
```

Definition at line 21 of file config.h.

5.8.2.4 BOARD_ROWS

```
#define BOARD_ROWS 6
```

Definition at line 20 of file config.h.

5.8.2.5 BUTTON_COLOR

```
#define BUTTON_COLOR SCARLET
```

Definition at line 7 of file config.h.

5.8.2.6 CENTER_WEIGHT

```
#define CENTER_WEIGHT 10
```

Definition at line 30 of file config.h.

5.8.2.7 DISABLE_VALUE

```
#define DISABLE_VALUE 4000
```

Definition at line 38 of file config.h.

5.8.2.8 EXIT_CALL_VALUE

```
#define EXIT_CALL_VALUE 11
```

Definition at line 27 of file config.h.

5.8.2.9 FOUR_IN_A_ROW

```
#define FOUR_IN_A_ROW 100000
```

Definition at line 34 of file config.h.

5.8.2.10 G

```
#define G 9.8
```

Definition at line 22 of file config.h.

5.8.2.11 HOLE_RADIUS

```
#define HOLE_RADIUS 15
```

Definition at line 19 of file config.h.

5.8.2.12 MAIN_MENU_CALL_VALUE

```
#define MAIN_MENU_CALL_VALUE 10
```

Definition at line 26 of file config.h.

5.8.2.13 ONE_PLAYABLE

```
#define ONE_PLAYABLE 10
```

Definition at line 31 of file config.h.

5.8.2.14 PER_SIDE

```
#define PER_SIDE 6
```

Optimization Constant.

The number of pixels required padding a region of pixels in order to replace the region with the surrounding color.

Definition at line 44 of file config.h.

5.8.2.15 PLAYER_1_COLOR

```
#define PLAYER_1_COLOR RED
```

Definition at line 13 of file config.h.

5.8.2.16 PLAYER_2_COLOR

```
#define PLAYER_2_COLOR YELLOW
```

Definition at line 14 of file config.h.

5.8.2.17 SQUARE_SIDE

```
#define SQUARE_SIDE 40
```

Definition at line 18 of file config.h.

5.8.2.18 TEXT_COLOR

```
#define TEXT_COLOR WHITE
```

Definition at line 8 of file config.h.

5.8.2.19 THREE_IN_A_ROW

```
#define THREE_IN_A_ROW 85
```

Definition at line 33 of file config.h.

5.8.2.20 TWO_IN_A_ROW

```
#define TWO_IN_A_ROW 30
```

Definition at line 32 of file config.h.

5.9 Game.cpp File Reference

Contains definitions of functions in use throughout the entire game and the game class.

```
#include "Game.h"
```

Functions

- void waitForInput (int &x, int &y)

  Makes program idle until a touch is detected.
- void waitForInput (int ∗x, int ∗y)

  Makes program idle until a touch is detected.

5.9.1 Detailed Description

Contains definitions of functions in use throughout the entire game and the game class.

5.9.2 Function Documentation

5.9.2.1 waitForInput() [1/2]

```
void waitForInput (
             int & x,
             int & y )
```

Makes program idle until a touch is detected.

Parameters

x	variable to store x coord of touch
y	variable to store y coord of touch

Author

Adam Exley

Definition at line 65 of file Game.cpp.

5.9.2.2 waitForInput() [2/2]

```
void waitForInput (
             int * x,
             int * y )
```

Makes program idle until a touch is detected.

Parameters

x	address to store x coord of touch
y	address to store y coord of touch

Author

Adam Exley

Definition at line 72 of file Game.cpp.

5.10 Game.h File Reference

Contains declarations of functions in use throughout the entire game and the game class.

```
#include "FEHLCD.h"
#include "FEHUtility.h"
#include "FEHRandom.h"
#include "FEHSD.h"
#include "config.h"
```

Classes

- class Game

  Controls flow of the game once in play.

Functions

- void waitForInput (int ∗x, int ∗y)

  Makes program idle until a touch is detected.
- void waitForInput (int &x, int &y)

  Makes program idle until a touch is detected.

5.10.1 Detailed Description

Contains declarations of functions in use throughout the entire game and the game class.

5.10.2 Function Documentation

5.10.2.1 waitForInput() [1/2]

```
void waitForInput (
            int & x,
            int & y )
```

Makes program idle until a touch is detected.

Parameters

x	variable to store x coord of touch
y	variable to store y coord of touch

Author

Adam Exley

Definition at line 65 of file Game.cpp.

5.10.2.2 waitForInput() `[2/2]`

```
void waitForInput (
             int * x,
             int * y )
```

Makes program idle until a touch is detected.

Parameters

x	address to store x coord of touch
y	address to store y coord of touch

Author

Adam Exley

Definition at line 72 of file Game.cpp.

5.11 Image.cpp File Reference

Contains Image class defintions.

```
#include "Image.h"
```

5.11.1 Detailed Description

Contains Image class defintions.

5.12 Image.h File Reference

Contains declarations of Image class functions.

```
#include <stdint.h>
#include "FEHLCD.h"
#include "config.h"
#include <cstdlib>
#include <cstdio>
#include <cmath>
```

Classes

- class Image

  This class displays images exported from `https://www.piskelapp.com`.

5.12.1 Detailed Description

Contains declarations of Image class functions.

5.13 jamie.c File Reference

Data for Jamie's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define JAMIE_FRAME_COUNT 1
- #define JAMIE_FRAME_WIDTH 176
- #define JAMIE_FRAME_HEIGHT 240

5.13.1 Detailed Description

Data for Jamie's picture. Exported from piskel.com.

5.13.2 Macro Definition Documentation

5.13.2.1 JAMIE_FRAME_COUNT

```
#define JAMIE_FRAME_COUNT 1
```

Definition at line 6 of file jamie.c.

5.13.2.2 JAMIE_FRAME_HEIGHT

```
#define JAMIE_FRAME_HEIGHT 240
```

Definition at line 8 of file jamie.c.

5.13.2.3 JAMIE_FRAME_WIDTH

```
#define JAMIE_FRAME_WIDTH 176
```

Definition at line 7 of file jamie.c.

5.14 jane.c File Reference

Data for Jane's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define JANE_FRAME_COUNT 1
- #define JANE_FRAME_WIDTH 199
- #define JANE_FRAME_HEIGHT 240

5.14.1 Detailed Description

Data for Jane's picture. Exported from piskel.com.

5.14.2 Macro Definition Documentation

5.14.2.1 JANE_FRAME_COUNT

```
#define JANE_FRAME_COUNT 1
```

Definition at line 6 of file jane.c.

5.14.2.2 JANE_FRAME_HEIGHT

```
#define JANE_FRAME_HEIGHT 240
```

Definition at line 8 of file jane.c.

5.14.2.3 JANE_FRAME_WIDTH

```
#define JANE_FRAME_WIDTH 199
```

Definition at line 7 of file jane.c.

5.15 logo.c File Reference

Data for Connect 4 logo. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define LOGO_FRAME_COUNT 1
- #define LOGO_FRAME_WIDTH 160
- #define LOGO_FRAME_HEIGHT 51

5.15.1 Detailed Description

Data for Connect 4 logo. Exported from piskel.com.

5.15.2 Macro Definition Documentation

5.15.2.1 LOGO_FRAME_COUNT

```
#define LOGO_FRAME_COUNT 1
```

Definition at line 6 of file logo.c.

5.15.2.2 LOGO_FRAME_HEIGHT

```
#define LOGO_FRAME_HEIGHT 51
```

Definition at line 8 of file logo.c.

5.15.2.3 LOGO_FRAME_WIDTH

```
#define LOGO_FRAME_WIDTH 160
```

Definition at line 7 of file logo.c.

5.16 main.cpp File Reference

```
#include "FEHLCD.h"
#include "FEHUtility.h"
#include "FEHRandom.h"
#include "FEHSD.h"
#include "AI.h"
#include "Board.h"
#include "Menu.h"
#include "Game.h"
#include "image.h"
#include "logo.c"
```

Functions

- int main ()

  The main loop for the game.

5.16.1 Function Documentation

5.16.1.1 main()

```
int main ( )
```

The main loop for the game.

Authors

Adam Exley, Lauren Pokonosky, Pietro Lavezzo Displays game logo before initiating menu. Menu runs until a game mode is selected Board is then drawn and game is played before a win/loss/tie screen is shown. Game is then reset to be played again.

Shared game state array. Is updated by the Board class.

Definition at line 32 of file main.cpp.

5.17 Menu.cpp File Reference

Contains all the definitions for the Menu class. Declarations are in Menu.h.

```
#include "Menu.h"
```

5.17.1 Detailed Description

Contains all the definitions for the Menu class. Declarations are in Menu.h.

5.18 Menu.h File Reference

Contains the class definition and prototypes for the Menu class.

```
#include "FEHLCD.h"
#include "FEHUtility.h"
#include "FEHRandom.h"
#include "config.h"
#include "Game.h"
#include "image.h"
#include "bridgette.c"
#include "bailey.c"
#include "alex.c"
#include "jamie.c"
#include "jane.c"
#include "paul.c"
```

Classes

- class Menu

  Displays all menus for game and stores current menu state.

5.18.1 Detailed Description

Contains the class definition and prototypes for the Menu class.

5.19 paul.c File Reference

Data for Paul's picture. Exported from piskel.com.

```
#include <stdint.h>
```

Macros

- #define PAUL_FRAME_COUNT 1
- #define PAUL_FRAME_WIDTH 240
- #define PAUL_FRAME_HEIGHT 240

5.19.1 Detailed Description

Data for Paul's picture. Exported from piskel.com.

5.19.2 Macro Definition Documentation

5.19.2.1 PAUL_FRAME_COUNT

```
#define PAUL_FRAME_COUNT 1
```

Definition at line 6 of file paul.c.

5.19.2.2 PAUL_FRAME_HEIGHT

```
#define PAUL_FRAME_HEIGHT 240
```

Definition at line 8 of file paul.c.

5.19.2.3 PAUL_FRAME_WIDTH

```
#define PAUL_FRAME_WIDTH 240
```

Definition at line 7 of file paul.c.

5.20 README.md File Reference

Index