Othello Master - Application Documentation

Roy Schestowitz March 10th, 2003

Contents

1 Callbacks

- *void load (int menuentry)*

  Description:

  the load game submenu callback function

  Inputs:

  the callback value

- *void save (int menuentry)*

  Description:

  the save game submenu callback function

  Inputs:

  the callback value

- *void determinism_callback (int menuentry)*

  Description:

  the determinism callback function

- *void edit_board_callback(int menuentry)*

Description:

board editing submenu callback function

Input:

the callback value

- *void customise_callback (int menuentry)*

Description:

customised computation submenu callback function

Input:

the callback value

- *void open_report (char *filename)*

Description:

opens the file where a game report would be appended

- *void open_log_file (char *filename)*

Description:

opens a log file displaying some initial information

Input:

the name of the log file to be created

- *void opening_library_callback (int menuentry)*

Description:

the opening moves library submenu callback function

Input:

the callback value

- *void report_callback (int menuentry)*

Description:

the report generation submenu callback function

Input:

the callback value

- *void log_file_callback (int menuentry)*

Description:

the log file submenu callback function

Input:

the callback value

- *void difficulty_description_callback (int menuentry)*

Description:

the difficulty description submenu callback function

Input:

the callback value

- *void difficulty_callback (int menuentry)*

  Description:

  the difficulty submenu callback function

  Input:

  the callback value

- *void gamemode_callback (int menuentry)*

  Description:

  the game mode submenu callback function

  Input:

  the callback value

- *void menu (int menuentry)*

  Description:

  the menu callback function

  Input:

  the callback value

- *void mouse (int button, int state, int x_val, int y_val)*

  Description:

  reacts to mouse events sensibly

- *void idlefun ( )*

  Description:

  carries out operations when OpenGL is idle

- *void automated_moves (void)*

  Description:

  gets the CPU to play a move when necessary

- *void display (void)*

  Description:

  the main display loop. Called from main() to do all the drawing to the frame buffer

- *void reshape (int w, int h)*

  Description:

  the reshape callback function

  Input:

  w and h which are the new width and height allocated to the frame by the window manager

- *void mouse_motion (int x, int y)*

  Description:

  a callback function that is invoked upon an event of a mouse move

- *void keyboard (unsigned char key, int x, int y)*

  Description:

  the keyboard callback function. Invoked when a key is pressed

2 Computation

- *board_map make_nth_best_move (board_map inputboard, int color, int priority)*

  Description:

  makes the Nth best move for color given N

  Input:

  the board that is dealt with, the color for which the move is carried out and the priority n where 1 is best choice

  Returns:

  the board after the move was carried out

- *int get_mobility_of_color (board_map inputboard, int color)*

  Description:

  gets the number of moves available

  Input:

  the board that is dealt with and the color for which the number is calculated

  Returns:

  the number of moves available

- *int evaluate_straight_lines_complete (board_map input_board, int color)*

  Description:

  calculate the value of the complete straight lines on the given board

  Input:

  the board that is dealt with and the color for which the value is calculated

  Returns:

  the value of the lines

- *int evaluate_diagonal_lines_complete (board_map input_board, int color)*

  Description:

  calculate the value of the complete diagonal lines on the given board

  Input:

  the board that is dealt with and the color for which the value is calculated

Returns:

the value of the lines

- *int evaluate_straight_lines_incomplete (board_map input_board, int color)*

  Description:

  calculate the value of the incomplete straight lines on the given board

  Input:

  the board that is dealt with and the color for which the value is calculated

  Returns:

  the value of the lines

- *int evaluate_diagonal_lines_incomplete (board_map input_board, int color)*

  Description:

  calculate the value of the incomplete diagonal lines on the given board

  Input:

  the board that is dealt with and the color for which the value is calculated

  Returns:

  the value of the lines

- *int evaluate_lines (board_map input_board, int color)*

  Description:

  calculates the value of the line occupancy for color gap in a given board

  Input:

  the board that is dealt with and the color for which the value is calculated

  Returns:

  the value required

- *int calculate_mobility_difference_of_board (board_map inputboard, int color)*

  Description:

  calculates the mobility value gap in a given board

  Input:

  the board that is dealt with and the color for which the advantage is calculated

  Returns:

  the mobility gap in board (positive if color's mobility is higher)

- *int calculate_score_difference_of_board (board_map inputboard, int color)*

  Description:

  calculates the score gap in a board

  Input:

  the board that is dealt with and the color for which the advantage is calculated

  Returns:

  the score gap in board (positive if color's score is higher)

- *int find_value_of_position (int i, int j, int color)*

  Description:

  finds the value of position i,j standing for A-H and 1-8.

  Input:

  the board coordinates ranging from 1-8 and the current color for randomisation reasons

  Returns:

  the value of the position inquired

- *int evaluate (int color, board_map input_board)*

  Description:

  evaluates the current position of the board from the point of view of color and assumes that color is currently up

  Input:

  the color for which the evaluation is carried out and the state of the board

  Returns:

  the evaluation value

- *board_map compute_move_for_color (board_map inputboard, int color)*

  Description:

  computes a complex move for a given colour

  Input:

  the colour of player whose turn it is, input board

  Returns:

  the board with the new stone put

- *int random_number (int N)*

  Description:

  returns a random integer between 0 and N

- *int zero_one_random (void)*

  Description:

  returns 0 or 1 randomly

- *int load_opening_library_file(char *filename)*

  Description:

  load an opening library of Othello Master from a file given a filename

  Entry:

  filename is a pointer to the characters of the file to open

  Returns:

  true if save operation was successful, false otherwise

- *void init_opening_library (void)*

Description:

initialises the opening hashtable library

- *char* find_board_state_ident (void)*

Description:

assigns the distinct identifier to a board state

- *void cpu_move (int color)*

Description:

makes a move on behalf of color using the CPU

3 Drawing

- *void drawString (void *font, float x, float y, char *str)*

Description:

draws a string in a given (x,y) position on the window

Entry:

- *font points to a given font defined by GLUT
- *str is the pointer to the string to be displayed
- x is the x-coordinate for the string to be drawn at
- y is the y-coordinate for the string to be drawn at

Exit:

the pointer at *str is unchanged as well as *font

- *void draw_scene (void)*

Description:

draws the scene in which the game takes place

4 Hashing

- *void error (char *message)*
- *void fatal_error (char *message)*
- *Index sec_hash (Key_Type key)*

Description:

secondary hash function

- *Table initialize_table (Table_size table_size)*

Description:

initialise a table of given size

- *int find_pos_for (Key_ Type key, Table H)*

  Description:

  find_Pos_For a key in a hash table. Uses linear probing

- *int find (Key_ Type key, Table H)*

  Description:

  see if key is in hash table

- *int getX (Key_ Type key, Table H)*

  Description:

  get X value of table entry

- *int getY (Key_ Type key, Table H)*

  Description:

  get Y value of table entry

- *Table insert (Key_ Type key, Table H, int the_X, int the_ Y)*

  Description:

  insert a key in a hash table

- *Table delete (Key_ Type key, Table H)*

  Description:

  delete a key from the hash table

- *Table rehash (Table H)*

  Description:

  rehashing function

- *void print_ table (Table H)*

  Description:

  prints out the given table

- *static Index hash (Key_ Type key, Table_ size H_ SIZE)*

  Description:

  load a game of Othello master from a file given a filename

  Entry:

  filename is a pointer to the characters of the file to open

  Returns:

  true if save operation was successful, false otherwise

5 Loaders

- *GLubyte * glmReadPPM (char *filename, int *width, int *height)*

  Description:

  loads a PPM file

  Entry:

  - *filename is a pointer to the string holding a filename to open
  - *width is a pointer to some address where image width will be stored
  - *height is a pointer to some address where image height will be stored

  Exit:

  width and height are pointers to image dimensions

- *int save_game_to_filename (char *filename)*

  Description:

  saves a game of Othello master on a file

  Entry:

  filename point to filename string

  Returns:

  true if save operation was successful, false otherwise

- *int save_game (int slot_number)*

  Description:

  saves a game of Othello master on a slot

  Entry:

  slot number

  Returns:

  true if save operation was successful, false otherwise

- *int load_game (int slot_number)*

  Description:

  load a game of Othello master from a file given a slot number

  Entry:

  slot number

  Returns:

  true if save operation was successful, false otherwise

- *int load_game_from_filename (char *filename)*

  Description:

  load a game of Othello master from a file given a filename

  Entry:

  filename is a pointer to the characters of the file to open

  Returns:

  true if save operation was successful, false otherwise

6 Misc

- *int reducible (int i, int j, int color, board_map board)*

  Description:

  determines whether a move/placement is legal for a given colour/side in an i,j coordinate on the board

  Inputs:

  color of the player whose turn it is; the i and j board coordinates which are virtually the X,Y position of the board at which the stone is to be put (progressing left to right, top to bottom); the board to be dealt with

  Returns:

  boolean indicating is the placement is legal or not

- *void calculatescore (void)*

  Description:

  calculates the current score for red and black

- *board_map reduce (int i, int j, board_map board)*

  Description:

  a function used to do all the Othello-wise reductions and substitutions in colours of the objects.

  Inputs:

  the coordinates of the last stone put, according to which the correct reductions can be carried out; the board to be dealt with

  Returns:

  the new layout of the board

- *void calculate_mobility (void)*

  Description:

  calculates mobility of both sides

- *void check_deadlock (void)*

  Description:

  checks if a deadlock has occurred in which case the game has reached an end or turn passed (if mobility of current side is 0)

- *void finishoff (void)*

  Description:

  called when the game has reached an end and declares the result

- *void file_draw_ascii_board (board_map board, FILE * file)*

  Description:

  puts an ASCII representation of the board on the log file

Inputs:

the input board and the file pointer

- *void close_log_file (void)*

  Description:

  closes the log file when a game is finished and displays the game score

- *void add_to_report (void)*

  Description:

  adds the given game to a report file

7 Omcore

- *void adjust_look_at_center (void)*

  Description:

  sets the camera to point to the X,Y,Z origin

- *void draw_objects (void)*

  Description:

  draws the stoned that are dynamically changing their layout

- *void initboard (void)*

  Description:

  initialises the board when a new game begins

- *void initmenu (void)*

  Description:

  sets up the menu and takes care of the entries and their callback value

- *void init (void)*

  Description:

  initializes the program. This function is called once at bootstrap

- *void draw_ascii_board (board_map board)*

  Description:

  draws an ASCII representation of the board

  Inputs:

  the input board

- *void annotate_board (void)*

  Description:

  annotates the overhead main view

- *void annotate_help (void)*

  Description:

  annotates the help screen

- *void annotate_difficulty (void)*

  Description:

  displays the description of the algorithm used

- *void annotate (void)*

  Description:

  adds text on top of the scene depending on which view is enabled

- *void set_name_of_player (char *name)*

  Description:

  called when the name of the player is entered and to record that name

  Inputs:

  the pointer to the characters representing that name

- *void display_debugging_instructions(void)*

  Description:

  Prints some brief debugging instructions

- *void display_help (void)*

  Description:

  displays the command line option to the user and quits

- *void process_command_line (int argc, char *argv[])*

  Description:

  a function to process the command line arguments inputs - the arguments and the number of arguments

- *void set_difficulty(char *diff)*

  Description:

  sets up the difficulty of player 2

- *void set_up_stat_mode_difficulty(char *diff)*

  Description:

  sets up the difficulty of player 1 when gathering statistics

- *int main (int argc, char **argv)*

  Description:

  the main function. called when the program is started.

Inputs:

the arguments and the number of arguments from the command line

- *void quit_game (void)*

Description:

the main exit procedure. Close any connection and files here.