Space Invaders

Generated by Doxygen 1.12.0

| | 1 |
|----------------------------------|---|
| 2 Data Structure Index | 3 |
| 2.1 Data Structures | 3 |
| 3 File Index | 5 |
| 3.1 File List | 5 |
| 4 Data Structure Documentation | 7 |
| 4.1 Bullets Struct Reference | 7 |
| 4.1.1 Detailed Description | 7 |
| 4.1.2 Field Documentation | 7 |
| 4.1.2.1 x | 7 |
| 4.1.2.2 y | 7 |
| 4.2 datas Struct Reference | 8 |
| 4.2.1 Detailed Description | 8 |
| 4.2.2 Field Documentation | 8 |
| 4.2.2.1 name | 8 |
| 4.2.2.2 score | 8 |
| 4.3 Invaders Struct Reference | 8 |
| 4.3.1 Detailed Description | 9 |
| 4.3.2 Field Documentation | 9 |
| 4.3.2.1 next | 9 |
| 4.3.2.2 x_end | 9 |
| 4.3.2.3 x_start | 9 |
| 4.3.2.4 y | 9 |
| 4.4 Player Struct Reference | 9 |
| 4.4.1 Detailed Description | 0 |
| 4.4.2 Field Documentation | 0 |
| 4.4.2.1 shoot_pos_x | 0 |
| 4.4.2.2 shoot_pos_y | 0 |
| 4.4.2.3 x_start | 0 |
| 4.4.2.4 y | 0 |
| 5 File Documentation 1 | 1 |
| 5.1 cfiles/draw.c File Reference | 1 |
| | 2 |
| | 2 |
| | 2 |
| | 2 |
| | 3 |
| | 3 |
| | 3 |
| | 4 |

| 5.1.1.8 mov_invy() | . 14 |
|--|------|
| 5.1.1.9 peasy() | . 14 |
| 5.1.1.10 phard() | . 15 |
| 5.1.1.11 pmed() | . 15 |
| 5.1.1.12 pquit() | . 15 |
| 5.1.1.13 printSI() | . 16 |
| 5.1.1.14 pset() | . 16 |
| 5.1.1.15 pstart() | . 16 |
| 5.2 cfiles/game.c File Reference | . 17 |
| 5.2.1 Function Documentation | . 18 |
| 5.2.1.1 check_bullet() | . 18 |
| 5.2.1.2 freeInvaders() | . 18 |
| 5.2.1.3 game() | . 19 |
| 5.2.1.4 getch() | . 21 |
| 5.2.1.5 kbhit() | . 21 |
| 5.2.1.6 numb_inv() | . 21 |
| 5.2.1.7 reset_terminal_mode() | . 21 |
| 5.2.1.8 set_conio_terminal_mode() | . 22 |
| 5.2.1.9 Setup_Inv() | . 22 |
| 5.2.1.10 Setup_Player() | . 23 |
| 5.2.2 Variable Documentation | . 23 |
| 5.2.2.1 orig_termios | . 23 |
| 5.3 cfiles/menu.c File Reference | . 23 |
| 5.3.1 Function Documentation | . 24 |
| 5.3.1.1 menu() | . 24 |
| 5.4 cfiles/score.c File Reference | . 25 |
| 5.4.1 Function Documentation | . 26 |
| 5.4.1.1 highscore() | . 26 |
| 5.4.1.2 printboard() | . 27 |
| 5.4.1.3 savewrite_score() | . 28 |
| 5.5 headers/data_struct.h File Reference | . 30 |
| 5.5.1 Macro Definition Documentation | . 31 |
| 5.5.1.1 KEY_SPACE | . 31 |
| 5.5.1.2 MAX_NAME | . 31 |
| 5.5.1.3 MAX_POINT_LENGTH | . 31 |
| 5.5.1.4 TARGETLEN | . 31 |
| 5.5.2 Typedef Documentation | . 31 |
| 5.5.2.1 Bullets | . 31 |
| 5.5.2.2 Data | . 31 |
| 5.5.2.3 Invaders | . 32 |
| 5.5.2.4 Player | . 32 |
| 5.6 data_struct.h | . 32 |

| 5.7 headers/draw.h File Reference | 33 |
|-------------------------------------|----|
| 5.7.1 Function Documentation | 34 |
| 5.7.1.1 d_bullet() | 34 |
| 5.7.1.2 d_bullets() | 34 |
| 5.7.1.3 d_init() | 35 |
| 5.7.1.4 d_invader() | 35 |
| 5.7.1.5 d_player() | 35 |
| 5.7.1.6 d_score() | 36 |
| 5.7.1.7 mov_invx() | 36 |
| 5.7.1.8 mov_invy() | 36 |
| 5.7.1.9 peasy() | 37 |
| 5.7.1.10 phard() | 37 |
| 5.7.1.11 pmed() | 37 |
| 5.7.1.12 pquit() | 38 |
| 5.7.1.13 printSI() | 38 |
| 5.7.1.14 pset() | 38 |
| 5.7.1.15 pstart() | 39 |
| 5.8 draw.h | 39 |
| 5.9 headers/game.h File Reference | 40 |
| 5.9.1 Function Documentation | 40 |
| 5.9.1.1 game() | 40 |
| 5.10 game.h | 42 |
| 5.11 headers/menu.h File Reference | 42 |
| 5.11.1 Function Documentation | 42 |
| 5.11.1.1 menu() | 42 |
| 5.12 menu.h | 44 |
| 5.13 headers/score.h File Reference | 44 |
| 5.13.1 Function Documentation | 44 |
| 5.13.1.1 highscore() | 44 |
| 5.13.1.2 printboard() | 46 |
| 5.13.1.3 savewrite_score() | 46 |
| 5.14 score.h | 48 |
| 5.15 main.c File Reference | 49 |
| 5.15.1 Function Documentation | 49 |
| 5.15.1.1 main() | 49 |
| 5.16 README.md File Reference | 49 |
| Index | 51 |

Chapter 1

README

This project uses econio!! Its licnese and readme file is located in the license folder, the header file is in the headers folder.

Sound folder is empty, as windows version do not work, which use the sounds.

Build description:

```
-linux build:
   build command: gcc -Wall -Werror -Iheaders cfiles/*.c main.c -o build/a
   necessary steps for the build:
        -step 1: if build directory is included, then skip step 2
        -step 2: create build directory in the same directory as the main.c file
        -step 3: run the build command in the same directory as the main.c file

-windows build(not working right now):

build command:
        x86_64-w64-mingw32-gcc -Wall -Werror -Iheaders cfiles/*.c main.c -o build/main64.exe -lwinmm
   necessary steps for the build:
        -step 1: if build directory is included, then skip step 2
        -step 2: create build directory in the same directory as the main.c file
        -step 3: run the build command in the same directory as the main.c file
```

2 README

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

| Bullets | | |
|----------|--|---|
| | Bullets data structure | 7 |
| datas | Data should be a filiabana and sandara | _ |
| Invaders | Data structure of highscore system | B |
| | Invaders data structure | 8 |
| Player | | |
| | Player data structure | 9 |

4 Data Structure Index

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

| main.c | | | | | | | | | | | | | | | | | | | | | | | | | 49 |
|-----------------|-----|----|-----|---|--|--|--|------|--|--|--|--|--|------|--|--|--|--|------|------|--|--|--|--|----|
| cfiles/draw.c . | | | | | | | | | | | | | | | | | | | | | | | | | 11 |
| cfiles/game.c | | | | | | | | | | | | | | | | | | | | | | | | | 17 |
| cfiles/menu.c | | | | | | | | | | | | | | | | | | | | | | | | | 23 |
| cfiles/score.c | | | | | | | | | | | | | | | | | | | | | | | | | 25 |
| headers/data_s | stı | ru | ct. | h | | | | | | | | | | | | | | | | | | | | | 30 |
| headers/draw.h | ı | | | | | | | | | | | | | | | | | | | | | | | | 33 |
| headers/game | .h | | | | | | | | | | | | | | | | | | | | | | | | 40 |
| headers/menu | .h | | | | | | | | | | | | | | | | | | | | | | | | 42 |
| headers/score. | h | | | | | | | | | | | | | | | | | | | | | | | | 44 |

6 File Index

Chapter 4

Data Structure Documentation

4.1 Bullets Struct Reference

Bullets data structure.

#include <data_struct.h>

Data Fields

- int x
- int y

4.1.1 Detailed Description

Bullets data structure.

array of bullets, where y will change as it goes up but x wont change x will compare its value if it is greater or equ or smaller or equ than invader x_start and x_end value to see if it hits it, also do this with y as invaders can run into bullets also the array will be dinamically allocated, as with every bullet shot the array will grow, and with each shot that go out to space or hits an invader the array maybe shrink

4.1.2 Field Documentation

4.1.2.1 x

int Bullets::x

4.1.2.2 y

int Bullets::y

The documentation for this struct was generated from the following file:

· headers/data_struct.h

4.2 datas Struct Reference

data structure of highscore system

```
#include <data_struct.h>
```

Data Fields

- char name [MAX_NAME+1]
- int score

4.2.1 Detailed Description

data structure of highscore system

4.2.2 Field Documentation

4.2.2.1 name

```
char datas::name[MAX_NAME+1]
```

4.2.2.2 score

```
int datas::score
```

The documentation for this struct was generated from the following file:

• headers/data_struct.h

4.3 Invaders Struct Reference

Invaders data structure.

```
#include <data_struct.h>
```

Data Fields

- int * x_start
- int * x_end
- int * y
- struct Invaders * next

4.3.1 Detailed Description

Invaders data structure.

linked list of invaders, x_start is an array of x coordinates, invader hitbox x starting coord x_end is an array of the x_start x end coords y is an array where y coords are of invaders (in documentation the star symbol will not show, in the character design there are stars still)

```
| (--) < y[0] = 0, x_start[0] = 3, x_end[0] = 9
| ... < -y[1] = 1, x_start[1] = 3, x_end[1] = 9
```

the related coords are in the same place in the arrays

head and body are 7 character long

4.3.2 Field Documentation

4.3.2.1 next

```
struct Invaders* Invaders::next
```

4.3.2.2 x_end

```
int* Invaders::x_end
```

4.3.2.3 x_start

```
int* Invaders::x_start
```

4.3.2.4 y

```
int* Invaders::y
```

The documentation for this struct was generated from the following file:

• headers/data_struct.h

4.4 Player Struct Reference

Player data structure.

```
#include <data_struct.h>
```

Data Fields

- int * x_start
- int shoot_pos_x
- int shoot_pos_y
- int * y

4.4.1 Detailed Description

Player data structure.

will store it in a struct cause why not, easier data storage for me to implement same way of thinking as with the Invaders, just dont need the full hitbox, as Invaders only kill the player if they reach its level(y coord) new design:

```
-.l. (- is not part of the player design) (U)
```

4.4.2 Field Documentation

4.4.2.1 shoot_pos_x

```
int Player::shoot_pos_x
```

4.4.2.2 shoot_pos_y

```
int Player::shoot_pos_y
```

4.4.2.3 x_start

```
int* Player::x_start
```

4.4.2.4 y

```
int* Player::y
```

The documentation for this struct was generated from the following file:

· headers/data_struct.h

Chapter 5

File Documentation

5.1 cfiles/draw.c File Reference

```
#include "data_struct.h"
#include <stdio.h>
#include "econio.h"
#include <stdbool.h>
#include <stdlib.h>
```

Functions

- void printSI (int tabcounter, int y, int tcolor, int bcolor)
 - draws SPACE INVADERS text starting y coords and tabcounts*8 x coord with tcolor as text color and bcolor as background color
- void pstart (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws START text starting y coords and tabcounts *8 x coord with tcolor as text color and bcolor as background color
- void pquit (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws quit text starting y coords and tabcounts*8 + 6 x coord with tcolor as text color and bcolor as background color
- void pset (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws settings text starting y coords and tabcounts *8 + 2 x coord with tcolor as text color and bcolor as background color
- · void peasy (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws easy text starting y coords and tabcounts*8 + 4 x coord with tcolor as text color and bcolor as background color
- void pmed (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws medium text starting y coords and tabcounts *8 + 1 x coord with tcolor as text color and bcolor as background color
- · void phard (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws hard text starting y coords and tabcounts*8 + 4 x coord with tcolor as text color and bcolor as background color
- void d invader (int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2)
 - draws invader starting y coords and x coord with tcolor as text color and bcolor as background color
- void d_player (int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2)
 - draws player starting y coords and x coord with tcolor as text color and bcolor as background color
- void d init (Invaders *first, Player *p)
 - draws map first time
- void d_score (int score)

draws score

void d_bullets (int sdb_b)

draws bullet count

void mov invx (Invaders *first, int way)

move invaders horizontally

void mov_invy (Invaders *first)

move invaders vertically

• void d_bullet (Bullets b, int white)

draws bullets

5.1.1 Function Documentation

5.1.1.1 d bullet()

```
void d_bullet (
              Bullets b,
               int white)
draws bullets
00181
          econio_textcolor(COL_BLACK); econio_textbackground(COL_BLACK);
00182
00183
          econio_gotoxy(b.x,b.y);
00184
          white==1?econio_textcolor(COL_WHITE):econio_textcolor(COL_BLACK);
00185
          econio_textbackground(COL_BLACK);
00186
          printf("|");econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
00187 }
```

References Bullets::x, and Bullets::y.

5.1.1.2 d_bullets()

void d_bullets (

int sdb_b)

5.1.1.3 d_init()

```
void d_init (
              Invaders * first,
              Player * p)
draws map first time
00129
00130
          Invaders* mov = first;
         while (mov != NULL) {
00131
            d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
00132
     mov->x_start[1], mov->y[1]);
00133
             mov=mov->next;
00134
00135
         d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
00136
          econio_flush();
00137 }
```

References d_invader(), d_player(), Invaders::next, Invaders::x_start, Player::x_start, Invaders::y, and Player::y.

5.1.1.4 d_invader()

draws invader starting y coords and x coord with tcolor as text color and bcolor as background color

```
00111
00112 econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00113 econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
00114 printf("*(-_-)*");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00115 econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
00116 printf("$._$_.$");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00117 }
```

5.1.1.5 d_player()

draws player starting y coords and x coord with tcolor as text color and bcolor as background color

```
00120

00121 econio_textcolor(tcolor2);econio_textbackground(bcolor2);

00122 econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);

00123 printf(".I.");econio_textcolor(tcolor2);econio_textbackground(bcolor2);

00124 econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor2);

00125 printf("(_U_)");econio_textcolor(tcolor2);econio_textbackground(bcolor2);

00126 }
```

5.1.1.6 d_score()

5.1.1.7 mov_invx()

```
void mov_invx (
               Invaders * first,
               int way)
move invaders horizontally
00156
          Invaders* mov = first;
00157
         while (mov != NULL) {
00158
             d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
00159
     mov->x_start[1], mov->y[1]);
way==1?mov->x_start[0]++:mov->x_start[0]--;
00160
00161
              way==1?mov->x_start[1]++:mov->x_start[1]--;
00162
              way==1?mov->x_end[0]++:mov->x_end[0]--;
             way==1?mov->x_end[1]++:mov->x_end[1]--;
00163
              d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
00164
     mov->x_start[1], mov->y[1]);
             mov=mov->next;
00165
00166
00167 }
```

References d invader(), Invaders::next, Invaders::x end, Invaders::x start, and Invaders::y.

5.1.1.8 mov_invy()

```
void mov_invy (
             Invaders * first)
move invaders vertically
00170
00171
         Invaders* mov = first;
         while (mov != NULL) {
            d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
     mov->x_start[1], mov->y[1]);
      mov->y[0]+=2;
00174
             mov->y[1]+=2;
00175
             d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
00176
     mov->x_start[1], mov->y[1]);
00177
             mov=mov->next;
00178
00179 }
```

References d_invader(), Invaders::next, Invaders::x_start, and Invaders::y.

5.1.1.9 peasy()

draws easy text starting y coords and tabcounts *8 + 4 x coord with tcolor as text color and bcolor as background

```
color
00069
00070
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
00071
          econio_textcolor(tcolor);printf(" __ _ _ _ .
                                                                   . ");
          econio_textbackground(b2color);
00073
          econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__ | [__ \\_/ ");
00074
          econio_textbackground(b2color);
          \verb| econio_gotoxy(8*tabcount + 4, y+2); econio_textbackground(bcolor); printf(" | \_ | | | \_ |); \\
00075
00076
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
econio_textbackground(bcolor);printf(" ");
00077
00078
          econio_textbackground(b2color);
00079 }
```

5.1.1.10 phard()

```
void phard (
    int tabcount,
    int y,
    int tcolor,
    int bcolor,
    int b2color)
```

draws hard text starting y coords and tabcounts *8 + 4 x coord with tcolor as text color and bcolor as background

```
00096
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
00097
          econio_textbackground(bcolor);
          econio_textcolor(tcolor);printf(" . . __
00098
         00099
00100
00101
00102
          econio_textbackground(bcolor);printf(" | | | | \\ |__/
         econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
econio_textbackground(bcolor);printf(" ");
00103
00104
         econio_textbackground(b2color);
econio_textcolor(COL_BLACK);
00105
00106
00107 }
```

5.1.1.11 pmed()

draws medium text starting y coords and tabcounts *8 + 1 x coord with tcolor as text color and bcolor as background color

```
00082
              econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y);
econio_textbackground(bcolor);econio_textcolor(tcolor);printf("
00083
00084
00085
              econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+1);
              econio_textbackground(bcolor);printf(" |\\/| |__ | \\ | | | |\\/| ");
00086
              econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+2);
econio_textbackground(bcolor);printf(" | | |__ |__ / | |__ | ");
00087
00088
00089
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+3);
econio_textbackground(bcolor);printf(" ");
00090
00091
              econio textbackground(b2color);
00092 }
```

5.1.1.12 pquit()

draws quit text starting y coords and tabcounts *8 + 6 x coord with tcolor as text color and bcolor as background

```
color
00041
00042
           econio textbackground(b2color);econio gotoxy(8*tabcount + 2, y);
                                                                                               _ ___");
00043
          econio_textbackground(bcolor);econio_textcolor(tcolor);printf("
          00044
00045
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2
econio_textbackground(bcolor);printf("|_\\ |_| |_| ");
00046
00047
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+3);
econio_textbackground(bcolor);printf(" ");econio_textbackground(b2color);
00048
00049
00050
          econio_textcolor(COL_BLACK);
00051 }
```

5.1.1.13 printSI()

```
void printSI (
          int tabcounter,
          int y,
          int tcolor,
          int bcolor)
```

draws SPACE INVADERS text starting y coords and tabcounts *8 x coord with tcolor as text color and bcolor as background color

```
00010
                                                                {
00011
          econio_textbackground(bcolor);
00012
          econio_textcolor(tcolor);
00013
          econio_gotoxy(8*tabcounter, y);
     printf("#######\\#####\\ #####\\\\n");

##\\#######\\######\\#####\\\\n");

econio_gotoxy(8*tabcounter, y+1);

printf("##----/##/--##\\##/--##\\##/----/##/----/
00014
                                             #####\\#####\\
                                                                  ##\\###\\ ##\\##\\
00015
00016
                                                               ##|###\\ ##|##|
         ##/--##|##/--##|##/---/\n");
00017
         econio_gotoxy(8*tabcounter, y+2);
     00018
                                                  ####\\
                                                               ##|##/##\\ ##|##| ##|######|##|
     00019
                                                  ##/--/
                                                              ##|##|\\##\\##|\\##| ##/--##|##|
00020
00021
00022
          printf("#######|##|
                                 ##|
                                      ##|\\#####\\#####\\
                                                                ##|##| \\###| \\###/ ##| ##|####/
         ####\\##| ##| #####|\n");
00023
         econio_gotoxy(8*tabcounter, y+5);
                                      /\\_/\\<u></u>/\n");
                                                                  \\_/\\_/ \\__/ \\__/
00024
          printf("\\
00025 }
```

5.1.1.14 pset()

draws settings text starting y coords and tabcounts *8 + 2 x coord with tcolor as text color and bcolor as background

```
color
00054
00055
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
00056
        econio_textbackground(bcolor);
        econio_textcolor(tcolor);printf("
00057
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+1);
00058
        00060
        econio_textbackground(b2color);econio_gotoxy(8*tabcount
                                                        + 6, y+2);
00061
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
econio_textbackground(bcolor);printf(" ");
00062
00063
00064
        econio_textbackground(b2color);
00065 }
```

5.1.1.15 pstart()

```
void pstart (
          int tabcount,
          int y,
          int tcolor,
          int bcolor,
          int b2color)
```

draws START text starting y coords and tabcounts*8 x coord with tcolor as text color and bcolor as background

```
color
00028
00029
         econio_textbackground(b2color);econio_gotoxy(8*tabcount, y);econio_textbackground(bcolor);
         econio_textcolor(tcolor);printf(" __ __ __ __");
00030
00031
         econio_textbackground(b2color);
         \verb| econio_gotoxy(8*tabcount, y+1); econio_textbackground(bcolor); printf("(\_ | [_][_]) | | "); \\
00032
00033
         econio_textbackground(b2color);
00034
         econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
econio_textbackground(bcolor);printf(" ");
00035
00036
00037
         econio_textbackground(b2color);
00038 }
```

5.2 cfiles/game.c File Reference

```
#include "data_struct.h"
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "game.h"
#include "econio.h"
#include "draw.h"
#include "debugmalloc.h"
#include <string.h>
#include <sys/select.h>
#include <termios.h>
```

Functions

• void reset terminal mode ()

resets terminal mode

void set_conio_terminal_mode ()

set terminal mode

• int kbhit ()

check if a button is pressed(basically but a little bit complicated)

• int getch ()

gets the pressed button code

• Player * Setup_Player ()

sets up the Player data structure

Invaders * Setup_Inv (int mode)

Sets up the Invaders, makes the linked list, allocates memory to the linked list member and also allocates memory to the sub arrays and fills it up with data.

void freeInvaders (Invaders *i)

This function will free up the memory allocated to the Invaders and its sub allocated space.

int numb_inv (Invaders *first)

counts how many invaders are there

• int check_bullet (Invaders *first, Bullets *b, int *score, int db)

This function checks if the bullet hit an invader, if yes, then delete from the linked list.

• int game (int mode)

Main game logic function which will be accessed from main.c.

Variables

· struct termios orig_termios

5.2.1 Function Documentation

5.2.1.1 check bullet()

This function checks if the bullet hit an invader, if yes, then delete from the linked list.

```
00163
00164
           Invaders* mov = first;
00165
           Invaders* mov2 = mov;
00166
           while (mov != NULL) {
00167
                for(int i = 0; i < db; i++){
                    if(mov->y[0] == b[i].y || mov->y[1] == b[i].y) {
   if(b[i].x >= mov->x_start[0] && b[i].x <= mov->x_end[0]) {
      if(numb_inv(first) == 1) {
00168
00169
00170
00171
                                  *score += 5;
00172
                                  return 0;
00173
                              if (mov == first) {
00174
00175
      d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
00176
                                  b[i].y = 0;
*score += 5;
00177
00178
                                  return 2;
00179
00180
                             else if(mov->next == NULL) {
00181
                                  *score += 5;
00182
      d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
00183
                                  mov2->next=NULL;
00184
                                  free (mov);
00185
                                  return 1;
00186
00187
                             else{
00188
                                  *score += 5;
00189
      d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
00190
                                  Invaders* mov3 = mov;
00191
                                  mov=mov->next;
00192
                                  mov2 -> next = mov;
00193
                                  free(mov3);
00194
                                  mov=mov2;
00195
                                  b[i].y = 0;
00196
                         }
00197
00198
                    }
00199
00200
                mov2=mov;
00201
               mov=mov->next;
00202
00203
           return 1;
00204 }
```

References d_invader(), Invaders::next, numb_inv(), Bullets::x, Invaders::x_end, Invaders::x_start, Bullets::y, and Invaders::y.

5.2.1.2 freeInvaders()

This function will free up the memory allocated to the Invaders and its sub allocated space.

will get current linked list member, free its sub arrays, got to next member and free previous member

```
///will get current linked list member, free its sub arrays, got to next member and free previous
00137
     member
00138
         while(i != NULL){
00139
             Invaders* buff = i;
00140
              free(i->x_start);
00141
              free(i->x_end);
00142
              free(i->y);
              i = i -> next;
00143
00144
              free (buff);
00145
         }
00146 }
```

References Invaders::next, Invaders::x_end, Invaders::x_start, and Invaders::y.

5.2.1.3 game()

```
int game (
          int mode)
```

Main game logic function which will be accessed from main.c.

```
game logic
00209
00210
           econio_clrscr();
00211
           //setup phase
00212
           int score = 0; // the score which to export
00213
          int b_db = 0;
           int sdb_b = 0;
00214
          bool run = true;
00215
00216
          Player* p = Setup_Player();//strange, but it is what it is
00217
           Invaders* first = Setup_Inv(mode);
00218
          int row_length[3] = \{6,7,8\};
00219
          int cycle = 0;
          int way = 1;
int test = 1;
00220
00221
           int x_ref = first->x_start[0];
00222
00223
          Bullets* b =(Bullets*)malloc(50*sizeof(Bullets)); //setup for future
00224
00225
           //first draw
00226
          d_init(first, p);
00227
          //game logic
00228
          set_conio_terminal_mode();
00229
          while(run){
00230
               econio_rawmode();
00231
               while(!kbhit()){
                   //check if bullet has collision with invader
if(b_db > 0)test = check_bullet(first,b,&score,b_db);
00232
00233
00234
                    if(test == 0){
00235
                       econio_normalmode();
00236
                        reset_terminal_mode();
00237
                        //before exit free the bullets :)
00238
                        free(b);
00239
                        free(p):
00240
                        freeInvaders(first);//yeah free the slaves!!!
00241
                        debugmalloc_dump();
00242
                        return score;
00243
                   else if(test == 2) {
    Invaders* mov = first;
    first = first->next;
00244
00245
00246
00247
                        free (mov);
00248
00249
                    //checks if invader is in line with player
00250
                    if(first->y[0] == 28 || first->y[0] == 29){
00251
                        econio_normalmode();
00252
                        reset terminal mode();
00253
                        //before exit free the bullets :)
00254
00255
                        free(p);
00256
                        freeInvaders(first);//yeah free the slaves!!!
00257
                        debugmalloc_dump();
00258
                        return score:
00259
00260
                    //check collisions
```

```
//bullets draw and such
 00262
                                                          if(cycle%3==0){
 00263
                                                                       for(int i = 0; i < b_db; i++){
                                                                                 if(b[i].y != 0) {
 00264
00265
                                                                                              d_bullet(b[i], 0);
 00266
                                                                                              b[i].y--;
                                                                                              d_bullet(b[i], 1);
 00267
 00268
                                                                                               econio_gotoxy(0, 30);
                                                                                               printf("%d",b[0].y);
 00269
 00270
                                                                                               econio_flush();
 00271
00272
                                                                                  else {d bullet(b[i], 0);}
 00273
                                                                      }
 00274
 00275
                                                          if(cycle%(1+numb_inv(first)) == 0 || cycle%20 == 0){
 00276
                                                                      //move invaders
 00277
                                                                      if(x_ref < 70 && way) {
 00278
                                                                                 mov_invx(first, way);
                                                                                  x_ref++;
 00280
 00281
 00282
                                                                                   if(way) {mov_invy(first);}
00283
                                                                                  way = 0;
                                                                                  if(x_ref >= 7*row_length[mode]){
 00284
 00285
                                                                                              mov_invx(first, way);
                                                                                              x_ref--;
 00287
 00288
                                                                                  else{
00289
                                                                                               way = 1;
 00290
                                                                                              mov_invy(first);
 00291
 00292
                                                                      }
 00293
 00294
                                                         if(cycle%100 == 0){sdb_b = 0;}
 00295
                                                         d_score(score);
00296
                                                         d_bullets(sdb_b);
00297
                                                        usleep(23000);
 00299
                                                         cycle++;
 00300
                                                         if(cycle == 101)cycle = 0;
 00301
00302
                                            econio normalmode();
 00303
                                            //did it with switch too, but who cares
 00304
                                             //check keyboard press
                                            int key = getch();
 00305
 00306
                                             //move player to left
00307
                                             if (key== 'a') {
00308
                                                         if(p->x\_start[1] > 0){
00309
                   d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
 00310
                                                                     p->x_start[0]-=2;
 00311
                                                                      p->shoot_pos_x-=2;
 00312
                                                                     p->x_start[1]-=2;
00313
                    \texttt{d\_player}(\texttt{COL\_BLACK}, \texttt{COL\_BLACK}, \texttt{COL\_BLACK}, \texttt{P-} \times \texttt{start[0]}, \texttt{p-} \times \texttt{[0]}, \texttt{p-} \times \texttt{start[1]}, \texttt{p-} \times \texttt{[1]}, \texttt{p-} \times \texttt{[
 00314
 00315
 00316
                                                        //move player to right
 00317
                                            else if (key == 'd') {
00318
                                                        if(p->x_start[1] < 70){
00319
                   d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
 00320
                                                                    p->x_start[0]+=2;
 00321
                                                                      p->shoot_pos_x+=2;
00322
                                                                    p->x_start[1]+=2;
00323
                  d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
00324
00325
                                            }
 00326
                                                        //shoot bullet
 00327
                                             else if(key == KEY_SPACE && sdb_b < 7) {</pre>
 00328
                                                        sdb_b++;
 00329
                                                        b[b_db].x=p->shoot_pos_x;
 00330
                                                         b[b_db].y=p->shoot_pos_y;
 00331
                                                        b db++;
 00332
                                                         if (b_db%49 == 0)b = (Bullets*) realloc(b, (b_db + 51)*sizeof(Bullets));
 00333
                                                         d_bullet(b[b_db-1], 1);
 00334
 00335
00336
                                debugmalloc dump():
00337
                                return score;
00338 }
```

References check_bullet(), d_bullet(), d_init(), d_player(), d_score(), freeInvaders(), getch(), kbhit(), KEY_SPACE, mov_invx(), mov_invy(), Invaders::next, numb_inv(), reset_terminal_mode(), set_conio_terminal_mode(),

Setup_Inv(), Setup_Player(), Player::shoot_pos_x, Player::shoot_pos_y, Bullets::x, Invaders::x_start, Player::x_start, Bullets::y, Invaders::y, and Player::y.

5.2.1.4 getch()

```
int getch ()
gets the pressed button code
00052 {
00053
          int r;
00054
          unsigned char c;
          if ((r = read(0, &c, sizeof(c))) < 0) {</pre>
00055
00056
              return r;
00057
          } else {
00058
             return c;
          }
00059
```

5.2.1.5 kbhit()

00060 }

```
int kbhit ()
```

check if a button is pressed(basically but a little bit complicated)

5.2.1.6 numb_inv()

```
counts how many invaders are there
```

References Invaders::next.

5.2.1.7 reset_terminal_mode()

```
void reset_terminal_mode ()

resets terminal mode
00020 {
00021          tcsetattr(0, TCSANOW, &orig_termios);
00022 }
```

References orig_termios.

5.2.1.8 set_conio_terminal_mode()

```
void set conio terminal mode ()
set terminal mode
00028
          struct termios new_termios;
00029
00030
          // take two copies - one for now, one for later
          tcgetattr(0, &orig_termios);
memcpy(&new_termios, &orig_termios, sizeof(new_termios));
00031
00032
00033
00034
           // register cleanup handler, and set the new terminal mode
00035
          atexit(reset_terminal_mode);
00036
          cfmakeraw(&new_termios);
00037
          tcsetattr(0, TCSANOW, &new_termios);
00038 }
```

References orig_termios, and reset_terminal_mode().

5.2.1.9 Setup Inv()

Sets up the Invaders, makes the linked list, allocates memory to the linked list member and also allocates memory to the sub arrays and fills it up with data.

```
00088
             //O easy, 1 is medium and 2 is hard
00089
             //in easy half of the rows invaders will be, and it will move in medium speed
             //in medium there will be six in a row and it will move at medium speed //in hard there vill be more rows: not 7 but 11, they will move much faster
00090
00091
             int S_Width = 6;//number of white characters between 2 invader
00092
             int row_length[3] = \{6,7,8\};
00094
             int numb_row[3] = \{4,5,7\};
00095
             //linked list setup
            Invaders* list = NULL;
for(int j = 0; j < numb_row[mode]; j++) {
    for(int i = 0; i < row_length[mode]; i++) {</pre>
00096
00097
00098
00099
                      //create one element
00100
                       Invaders* mov;
00101
                       mov = (Invaders*) malloc(sizeof(Invaders));
00102
                       mov->next = list;//connect to previous
                       //allocate space to the arrays
mov->x_start = (int*)malloc(2*sizeof(int));
mov->x_end = (int*)malloc(2*sizeof(int));
00103
00104
00105
00106
                       mov \rightarrow y = (int*) malloc(2*sizeof(int));
00107
                        //first element of rows and the very first element restarts the x coords
00108
                        if(mov->next == NULL || i == 0){
00109
                            //starting coord
                            mov->x_start[0] = 0;
00110
                            mov \rightarrow x_start[1] = 0;
00111
00112
                             //end of hitbox coords
00113
                             mov \rightarrow x_end[0] = (mov \rightarrow x_start[0]) + S_Width;
                             mov \rightarrow x_end[1] = (mov \rightarrow x_start[1]) + S_width;
00114
00115
00116
                       elsef
                             //same just if its not the first element
00117
00118
                            mov \rightarrow x_start[0] = (list \rightarrow x_end[0]) +2;
00119
                             mov \rightarrow x_start[1] = (list \rightarrow x_end[1]) +2;
                            mov->x_end[0] = (mov->x_start[0]) + S_Width;
mov->x_end[1] = (mov->x_start[1]) + S_Width;
00120
00121
00122
                        //set y coords
00123
                       mov \rightarrow y[0] = 3*j+1;

mov \rightarrow y[1] = 3*j+2;
00125
00126
                        //next element
00127
                       list = mov;
00128
                  }
00129
00130
             return list;
00131 }
```

References Invaders::next, Invaders::x_end, Invaders::x_start, and Invaders::y.

5.2.1.10 Setup_Player()

```
Player * Setup_Player ()
sets up the Player data structure
            Player* p = (Player*) malloc(sizeof(Player));
00066
00067
00068
            p->x_start=(int*)malloc(2*sizeof(int));
           p->x_start[0] = 1;
p->x_start[1] = 0;
00069
00070
00071
00072
            //where the bullet will start from
            p->shoot_pos_x = 2;
p->shoot_pos_y = 27;
00073
00074
00075
00076
            //playing field will be 120*40, 120 width, 40 height
00077
            p->y=(int*)malloc(2*sizeof(int));
            p \rightarrow y (1110x) max

p \rightarrow y [0] = 29;

p \rightarrow y [1] = 30;
00078
00079
08000
            return p;
00081
00082 }
```

References Player::shoot_pos_x, Player::shoot_pos_y, Player::x_start, and Player::y.

5.2.2 Variable Documentation

5.2.2.1 orig_termios

struct termios orig_termios

5.3 cfiles/menu.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "menu.h"
#include "econio.h"
#include <stdbool.h>
#include "draw.h"
```

Functions

• int menu ()

menu logic

5.3.1 Function Documentation

5.3.1.1 menu()

```
int menu ()
menu logic
```

```
Menu logic.
```

```
00013
00014 #if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
           SMALL_RECT WinSize = {0, 0, 135, 35};
00015
           SMALL_RECT* WinWin = &WinSize;
00017
           SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
           //plays in a loop music.wav file in windows, just do not have music.wav yet //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
00018
00019
00020 #endif
00021
           econio_clrscr();
00022
           //draws menu items
00023
           printSI(1,3,COL_WHITE, COL_BLACK);
00024
           pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
           //pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
pquit(6, 16, COL_WHITE, COL_BLACK, COL_WHITE);
00025
00026
00027
00028
           int last = 0;
00029
           int before = 0;
00030
          while(true) {
00031
                econio_rawmode();
00032
                while (true) {
00033
                         //reads key and do state magic, save previous state and update current state
                         int ch = econio_getch();
if(ch == KEY_UP && last != 0){before = last; last--;}
00034
00035
00036
                          else if(ch == KEY_DOWN && last != 1) {before = last; last++;}
                         else if(ch == KEY_ENTER) break;
00037
00038
                          //draws new state of menu items or state machine
00039
                         switch(last){
00040
                              case 0:
00041
                                   //start text is shiney, settings not shiney anymore
00042
                                   pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
00043
                                   pquit(6,16,COL_WHITE, COL_BLACK, COL_WHITE);
00044
                                   break;
00045
                                   /*
                                    * was a good idea, just not in c, devistated am i, said Yoda as
* possible upgrade if client wants to get a new feture in like a settings
00046
00047
      button in production
00048
                                    \star gonna leave it here as nobody gonna touch it and there is space for it
00049
00050
                              case 1:
00051
                                   if(before == 0){
                                       //start not shiney, settings shiney
pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
00052
00053
00054
                                        pset (5, 16, COL_BLACK, COL_WHITE, COL_BLACK);
00055
00056
                                       //quit not shiney, settings shiney pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00057
00058
00059
                                        pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
00060
                                   break;
00061
00062
00063
                              case 1:
                                  //settings not shiney, quit shiney
pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
00064
00065
00066
                                   pquit(6, 16, COL_BLACK, COL_WHITE, COL_BLACK);
00067
                                   break;
00068
                         }
00069
                     }
00070
                         econio normalmode();
00071
                          //choice is quit, do exit with screen to default
00072
                          if(last == 1){
00073
                              econio_textbackground(COL_BLACK);
00074
                              econio_clrscr();
00075
                              econio textcolor(COL WHITE);
00076
                              exit(0);
00077
00078
00079
                              //choice is start, do clear screen, draw dificulty elements
00080
                              econio_textbackground(COL_BLACK);
00081
                              econio_clrscr();
00082
                              econio_textcolor(COL_WHITE);
00083
                              peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
00084
                              pmed(6, 13, COL_WHITE, COL_BLACK, COL_WHITE);
```

```
00085
                             phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
00086
00087
                             last = 0:
00088
00089
                             econio rawmode();
00090
                             while (true) {
                                  //read keys and save previous state, and update now state
00092
                                  int ch = econio_getch();
00093
                                  if(ch == KEY_UP && last != 0) {before = last; last--;}
00094
                                  else if(ch == KEY_DOWN && last != 2) {before = last; last++;}
                                 else if (ch == KEY_ENTER) break;
00095
00096
00097
                                  //state machine goes brrrrrrrr
00098
                                  switch(last){
00099
                                      case 0:
                                          //easy text is shiney, medium is not
peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00100
00101
00102
00103
                                          break;
00104
                                      case 1:
00105
                                           if(before == 0){
00106
                                               //easy was shiney, not now anymore, medium shiney
                                               peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
00107
00108
                                               pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
00109
00110
                                           else {
00111
                                               //hard not shiney anymore, meium is shiney
00112
                                               pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
00113
                                               phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
00114
00115
                                          break:
00116
                                      case 2:
                                          //medium not shiney, hard is shiney
pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00117
00118
00119
                                          phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
00120
                                          break;
00121
                                 }
00123
00124
                             econio_normalmode();
00125
                         //before exit to who knows where, clean screen
00126
                            econio_textbackground(COL_BLACK);
00127
                         econio clrscr():
00128
                         econio_textcolor(COL_WHITE);
00129
00130
                         return last;
00131
00132
               }
00133 }
```

References peasy(), phard(), pmed(), pquit(), printSI(), and pstart().

5.4 cfiles/score.c File Reference

```
#include "data_struct.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include "econio.h"
#include "score.h"
#include <crype.h>
#include <dirent.h>
#include <sys/stat.h>
#include "debugmalloc.h"
```

Functions

• void printboard (FILE *fptr, int db)

prints specified highscore board

• void savewrite_score (int state, int score)

save score to related file if score the is higher than stored in the file connected to the name given by the player

void highscore (int state)

Main highscore logic.

5.4.1 Function Documentation

5.4.1.1 highscore()

```
void highscore (
          int state)
```

Main highscore logic.

get input to how many elements to write based on state, medium for printboard

```
00229
00230
           //set window size for good windows users :), but i like linux better, just not the terminal
      modifying
00231 #if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
          SMALL_RECT WinSize = {0, 0, 70, 40};
SMALL_RECT* WinWin = &WinSize;
00232
00233
00234
          SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00235 #endif
00236
00237
           //highscore board setup
00238
          FILE* fptr;
          char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
int testDB = 0;int db = 0;
00239
00240
00241
          printf("Give how many scores to show from the top, 0 to not show anything");
00242
          printf("\nor negative number as it will be converted to a positive number\n");
00243
          printf("number of players to show from top 1: ");
00244
           testDB = scanf("%d",&db);
00245
          while(testDB < 1 || db < 0 || db > 1000000) {
               econio_clrscr();
printf("number of players to show from top 1: ");
00246
00247
               testDB = scanf("%*c%d%*c",&db);
00248
00249
00250
00251
          //checks if highscores directory exist and openable
00252
          DIR* dir2;
00253
          dir2 = opendir("../highscores");
00254
          if (dir2) {closedir(dir2);}
00255
          //directory does not exist
00256
          else if(ENOENT == errno){
00257
              closedir(dir2);
              mkdir("../highscores", 0700);
dir2 = opendir("../highscores");
00258
00259
00260
               if (dir2) {closedir(dir2);}
00261
               //directory does not exist
               else if(ENOENT == errno){
00262
00263
                  closedir(dir2);
00264
                   perror("Could not create highscores directory, please create it manually to the same
      directory as main.c");
00265
                   exit(0):
00266
00267
               //other problem
00268
               else{
                   closedir(dir2);
perror("Error occured regarding the highscores directory");
00269
00270
00271
                   exit(0);
00272
00273
00274
           //other problem
00275
          else {
               closedir(dir2);
00276
00277
               perror("Error occured regarding the highscores directory");
00278
               exit(0);
00279
00280
           //state machine, would be better to put paths into a 2d char array, but who cares
00281
          //its not like anybody gonna expand this sht
00282
          switch(state){
00283
              //show relevant highscore
00284
               //dont wanna make a function to read file and store the data
00285
               case 0:
```

```
fptr = fopen(paths[state], "r");
                   if(fptr == NULL) {
    fptr = fopen(paths[state], "w");
00287
00288
00289
                       if(fptr == NULL){
                           perror("File open error");
00290
00291
                           exit(0);
00292
00293
                   printboard(fptr, db);
00294
00295
                   break;
00296
               case 1:
00297
                  //show highscore in medium diff
00298
                   fptr = fopen(paths[state], "r");
00299
                   if(fptr == NULL){
00300
                       fptr = fopen(paths[state], "w");
                       if(fptr == NULL) {
    perror("File open error");
00301
00302
00303
                           exit(0);
00304
00305
00306
                   printboard(fptr, db);
00307
                   break;
00308
               case 2:
                  //high diff score
00309
00310
                   fptr = fopen(paths[state], "r");
00311
                   if(fptr == NULL){
00312
                       fptr = fopen(paths[state], "w");
00313
                       if(fptr == NULL){
00314
                           perror("File open error");
00315
                           exit(0);
00316
00317
00318
                   printboard(fptr, db);
00319
                   break;
00320
          }
00321 }
```

References printboard().

5.4.1.2 printboard()

```
void printboard ( \label{eq:file} {\tt FILE} \, * \, fptr \text{,} int db)
```

prints specified highscore board

prints highscore board while getting the data from the save file, needs location and max amount to draw

```
if(db == 0)return;
00021
00022
           //variables
00023
          Data data[1] = {};
          int index = 0;
const char* padding = "
00024
00025
                                                                                ";//soooo long, idk why
00026
          char buff[5] = "";
00027
          //drawing begining
00028
          econio_clrscr();
          econio_gotoxy(30,1);
printf("++++++++++++++++");
00029
00030
          econio_gotoxy(30,2);
00031
00032
          printf("|NAME
                              | SCORE | ");
00033
           econio_gotoxy(30,3);
          printf("++++++++++++++");
00034
00035
00036
          int tName =0; int tSCR = 0;
00037
          //magic is done here
00038
          while(index < db) {</pre>
      "%d",&data->score)) == -1 || tSCR == 0){
    //checks if its end of file, if first if did not work
00039
00040
00041
00042
                   if((tSCR = fscanf(fptr, "%d", &data->score)) == -1) {break;}
00043
                   econio_clrscr();
00044
                   printf("\nCorrupted file\n");
00045
                   exit(0);
00046
00047
               //checks if earlier break is needed
00048
               if((tName == EOF && tSCR == EOF) || (tName == 0 && tSCR == 0)) {break;}
```

```
sprintf(buff, "%d", data->score);//transform int to string aka char array
00051
00052
                //get padding
                int padlLen = (TARGETLEN*3/5) - strlen(data->name);
int pad2Len = (TARGETLEN*2/5) - strlen(buff);
00053
00054
00055
00056
                 //draw board
00057
                econio_gotoxy(30, 4+(2*index));
00058
                printf("|\$s\$*.*s|\$*.*s\$d|", data->name, padlLen, padlLen,
padding,pad2Len,pad2Len,padding,data->score);
00059 econic gotoxy/30 4//0 ; ;
                econio_gotoxy(30, 4+(2*index+1));
printf("+++++++++++++");
00060
00061
00062
                index++;//magic number
00063
00064
           printf("\n");
            printf("\nPress ENTER to start the game\n");
00065
00066
            econio rawmode();
00067
            while(1){
00068
                int key = econio_getch();
                if(key == KEY_ENTER) {break;}
00069
00070
00071
            econio_normalmode();
00072 }
```

References datas::name, datas::score, and TARGETLEN.

5.4.1.3 savewrite_score()

```
void savewrite_score (
          int state,
          int score)
```

save score to related file if score the is higher than stored in the file connected to the name given by the player

write to file to the specified file based on state to save new data or update old score if name is in list and current score is higher than old, change it, than sort current arrays to descending based on scores

```
00075
00076
         econio clrscr();
00077
         econio_textcolor(COL_WHITE);
00078
         //set file path with state
00079
         FILE* fptr;
00080
         00081
         fptr = fopen(paths[state], "r");
00082
         if(fptr == NULL) {perror("Scoresave: file opening error");}
00083
         else{
00084
00085
              \ast for legacy code to just exist here or to return to something in case of shit
00086
             switch(state){
00087
                case 0:
00088
                     fptr = fopen("../highscores/l1.txt", "r+");
00089
                    break;
00090
                 case 1:
00091
                     fptr = fopen("../highscores/11.txt", "r+");
00092
                    break;
00093
                 case 2:
00094
                     fptr = fopen("../highscores/11.txt", "r+");
00095
                    break:
00096
00097
00098
00099
             //set up variables to work with
00100
             char name [MAX_NAME + 2] = \{\};
00101
             int index = 0;
00102
             Data *p = calloc(3, sizeof(*p)); //for safety and to pass new data allocate 3 times the space
00103
             Data temp;
00104
             bool in = false;
00105
             int i_new;
00106
             //checks if highscores directory exist and openable
00107
             DIR* dir;
00108
00109
             dir = opendir("../highscores");
00110
             if(dir){closedir(dir);}
00111
             //checks if directory do not exist
             else if(ENOENT == errno){
00112
00113
                closedir(dir);
00114
                mkdir("../highscores", 0700);
00115
                dir = opendir("../highscores");
```

```
if(dir){closedir(dir);}
                   //checks if directory still not exist
00117
00118
                   else if(ENOENT == errno){
00119
                        closedir(dir);
00120
                        free(p);
                       debugmalloc_dump();
00121
00122
                       perror("Could not create highscores directory, please create it manually to the same
      directory as main.c");
00123
                       exit(0);
00124
00125
                   //other problem
00126
                   else{
00127
                        closedir(dir);
00128
                        free(p);
00129
                        debugmalloc_dump();
00130
                       perror("Error occured regarding the highscores directory");
00131
                        exit(0):
00132
                   }
00133
00134
               //other problem
00135
               else {
00136
                  closedir(dir);
00137
                   free(p);
00138
                   debugmalloc dump();
00139
                   perror("Error occured regarding the highscores directory");
00140
                   exit(0);
00141
00142
              FILE* n = fopen("../highscores/n.txt", "w");
00143
               if(n == NULL) {perror("Scoresave: file opening error, at line 80");}
00144
00145
               else{
00146
                   //input name
00147
                   printf("!Your name can only be 12 character long!");
00148
                   printf("Your name: ");
                   int testName = 0;
00149
                   while((testName = scanf("%s", name)) == 0 || testName == -1 || strlen(name) > 12){
00150
                      printf("!Your name can only be 12 character long!");
printf("Your correct name: ");
00151
00152
00153
00154
                   ^{\prime} //load to struct the saved data
00155
                   //yeah nested loops, its not good, but it is what it is, who else gonna touch this code
      except me anyway?
00156
                   //while(fscanf(fptr, "%s %d", p[index].name, &p[index].score) != EOF){
00157
00158
00159
                   int testStr = 0; int testINT = 0;
00160
                   while((testStr = fscanf(fptr,"%s",p[index].name)) != EOF && (testINT = fscanf(fptr,"%d",
      &p[index].score)) != EOF){
00161
                        //tests if readed data was corrupted
00162
                        if (testStr == 0 || testStr == -1 || testINT == 0 || testINT == -1) {
                            perror("Corrupted file");
00163
00164
00165
00166
                        /// if name is in list and current score is higher than old, change it, than sort
      current arrays to descending based on scores
00167
                       //check if name is in the list
                        if(strcmp(name, p[index].name) == 0){
   in = true;//is it in or not?
00168
00169
00170
                            //check to update its score
00171
                            if(score > p[index].score) {
                                p[index].score = score;
if(index != 0){
00172
00173
00174
                                     //bubble sort
00175
                                     for(int i = 0; i <= index-1; i++){</pre>
00176
                                         for (int j = i+1; j \le index; j++) {
                                             if(p[j].score >= p[i].score) {
00177
                                                 temp = p[i];
p[i] = p[j];
00178
00179
                                                 p[j] = temp;
00180
00181
                                             }
00182
                                        }
00183
                                    }
00184
                                }
00185
                           }
00186
00187
                        //gets ready for next item, reallocating space
00188
00189
                        p = (Data*)realloc(p, (index+1)*sizeof(*p));
00190
                   }
00191
00192
                   //if not in highscorer list then add it to the top of its score group
00193
                   bool found = false;
00194
                   if(!in){
00195
                        for(int i = 0; i < index; i++) {</pre>
                           if(score >= p[i].score && !found) {
   i_new = i;
00196
00197
                                found = true;
00198
```

```
break;
00200
00201
00202
                       //if nobody is in the list put him in
if(index == 0)fprintf(n, "%s %d",name, score);
00203
00204
                       //save list to file
00206
                            for(int i = 0; i < index; i++) {
    if(i == i_new) fprintf(n, "%s %d\n", name, score);
    fprintf(n, "%s %d\n",p[i].name, p[i].score);</pre>
00207
00208
00209
00210
00211
00212
                       //free buff and close file stream
00213
                       free(p);
00214
                       fclose(n);
00215
                       fclose(fptr);
00216
                      //check if any problem occured while freeing memory
debugmalloc_dump();
00218
                       //delete old, rename new save file
00219
                       //yeah there are better ways to do it, but i dont want to touch this at this point
00220
                       remove(paths[state]);
                      rename("../highscores/n.txt",paths[state]);
00221
00222
00223
            }
00224 }
```

References MAX_NAME, and datas::score.

5.5 headers/data_struct.h File Reference

Data Structures

· struct datas

data structure of highscore system

struct Bullets

Bullets data structure.

struct Invaders

Invaders data structure.

• struct Player

Player data structure.

Macros

- #define KEY SPACE 32
- #define MAX_NAME 12

max length of name

• #define MAX POINT LENGTH 4

max length of collectable points

#define TARGETLEN 15

target length of a board piece without the "|" tags(there are 3 of these tags)

Typedefs

typedef struct datas Data

data structure of highscore system

typedef struct Bullets Bullets

Bullets data structure.

· typedef struct Invaders Invaders

Invaders data structure.

· typedef struct Player Player

Player data structure.

5.5.1 Macro Definition Documentation

5.5.1.1 KEY_SPACE

#define KEY_SPACE 32

5.5.1.2 MAX_NAME

#define MAX_NAME 12

max length of name

5.5.1.3 MAX_POINT_LENGTH

```
#define MAX_POINT_LENGTH 4
```

max length of collectable points

5.5.1.4 TARGETLEN

#define TARGETLEN 15

target length of a board piece without the "|" tags(there are 3 of these tags)

5.5.2 Typedef Documentation

5.5.2.1 Bullets

typedef struct Bullets Bullets

Bullets data structure.

Bullets data struct.

array of bullets, where y will change as it goes up but x wont change x will compare its value if it is greater or equ or smaller or equ than invader x_start and x_end value to see if it hits it, also do this with y as invaders can run into bullets also the array will be dinamically allocated, as with every bullet shot the array will grow, and with each shot that go out to space or hits an invader the array maybe shrink

5.5.2.2 Data

typedef struct datas Data

data structure of highscore system

5.5.2.3 Invaders

```
typedef struct Invaders Invaders
```

Invaders data structure.

Invaders data struct.

linked list of invaders, x_start is an array of x coordinates, invader hitbox x starting coord x_end is an array of the x_start x end coords y is an array where y coords are of invaders (in documentation the star symbol will not show, in the character design there are stars still)

```
|(--)| < y[0] = 0, x_start[0] = 3, x_end[0] = 9
| ... < -y[1] = 1, x_start[1] = 3, x_end[1] = 9
```

the related coords are in the same place in the arrays

head and body are 7 character long

5.5.2.4 Player

```
typedef struct Player Player
```

Player data structure.

Player data struct.

will store it in a struct cause why not, easier data storage for me to implement same way of thinking as with the Invaders, just dont need the full hitbox, as Invaders only kill the player if they reach its level(y coord) new design:

```
-.l. (- is not part of the player design) (U)
```

5.6 data_struct.h

Go to the documentation of this file.

```
00001 #ifndef data_struct_h
00002 #define data_struct_h
00003
00004 //definition of space key
00005 #define KEY_SPACE 32
00006 ///max length of name
00007 #define MAX_NAME 12
00008 ///max length of collectable points
00009 #define MAX_POINT_LENGTH 4
00010 ///target length of a board piece without the "|" tags(there are 3 of these tags)
00011 #define TARGETLEN 15
00012
00013 ///data structure of highscore system
00014 typedef struct datas{
       char name[MAX_NAME + 1];
00015
00016
         int score;
00017 }Data;
00018
00019 ///@brief Bullets data structure
00020 /**
00021 *
00022
      * array of bullets, where y will change as it goes up
00023 \star but \bar{x} wont change
00024 \star x will compare its value if it is greater or equ or smaller or equ than
00025 \star invader x_start and x_end value to see if it hits it, also do this with y as invaders can run into
     bullets
00026 \, * also the array will be dinamically allocated, as with every bullet shot
```

```
* the array will grow, and with each shot that go out to space or hits an invader
00028 \star the array maybe shrink
00029 *
00030 */
00031
00032 typedef struct Bullets{
           int x;
00034
            int y;
00035 }Bullets;
00036
00037 ///Invaders data structure
00038 /**
00039 *
00040 * linked list of invaders,
00041 \star x_start is an array of x coordinates, invader hitbox x starting coord
00042 \,\, * x_end is an array of the x_start x end coords 00043 \, * y is an array where y coords are of invaders
00044 * (in documentation the star symbol will not show, in the character design there are stars still)
00046 * | *(-_-)* <- y[0] = 0, x_start[0] = 3, x_end[0] = 9 00047 * | $._$_.$ <- y[1] = 1, x_start[1] = 3, x_end[1] = 9
00048 *
00049 \,\,\star\, the related coords are in the same place in the arrays
00050 *
00051 * head and body are 7 character long
00052
00053 */
00054
00055
00056 typedef struct Invaders{
00057
           int* x_start;
00058
           int* x_end;
           int* y;
00059
00060
          struct Invaders* next;
00061 }Invaders;
00062
00064 ///Player data structure
00065 /**
00066 *
00067 * will store it in a struct cause why not, easier data storage for me to implement
00068 \star same way of thinking as with the Invaders, just dont need the full hitbox, as Invaders only kill 00069 \star the player if they reach its level(y coord)
00070 * new design:
00071 *
00072 *-.I.
               (- is not part of the player design)
00073 *(_U_)
00074 *
00076
00077 typedef struct Player{
        int* x_start;
int shoot_pos_x;
00078
00079
08000
        int shoot_pos_y;
int* y;
00082 }Player;
00083
00084 #endif
```

5.7 headers/draw.h File Reference

Functions

- void printSI (int tabcounter, int y, int tcolor, int bcolor)
 - draws SPACE INVADERS text starting y coords and tabcounts *8 x coord with tcolor as text color and bcolor as background color
- void pstart (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws START text starting y coords and tabcounts*8 x coord with tcolor as text color and bcolor as background color
- void pset (int tabcount, int y, int tcolor, int bcolor, int b2color)
 - draws settings text starting y coords and tabcounts $*8 + 2 \times 10^{-2}$ x coord with tcolor as text color and bcolor as background color
- · void pquit (int tabcount, int y, int tcolor, int bcolor, int b2color)

draws quit text starting y coords and tabcounts*8 + 6 x coord with tcolor as text color and bcolor as background color

void peasy (int tabcount, int y, int tcolor, int bcolor, int b2color)

draws easy text starting y coords and tabcounts *8 + 4 x coord with tcolor as text color and bcolor as background color

void pmed (int tabcount, int y, int tcolor, int bcolor, int b2color)

void phard (int tabcount, int y, int tcolor, int bcolor, int b2color)

draws hard text starting y coords and tabcounts *8 + 4 x coord with tcolor as text color and bcolor as background color

• void d_invader (int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2)

draws invader starting y coords and x coord with tcolor as text color and bcolor as background color

void d_player (int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2)

draws player starting y coords and x coord with tcolor as text color and bcolor as background color

void d score (int score)

draws score

void d_init (Invaders *first, Player *p)

draws map first time

• void d_bullet (Bullets b, int white)

draws bullets

void mov invx (Invaders *first, int way)

move invaders horizontally

void mov_invy (Invaders *first)

move invaders vertically

void d_bullets (int sdb_b)

draws bullet count

5.7.1 Function Documentation

5.7.1.1 d bullet()

References Bullets::x, and Bullets::y.

5.7.1.2 d_bullets()

00153 }

5.7.1.3 d_init()

```
void d init (
                     Invaders * first,
                    Player * p)
draws map first time
00129
00130
              Invaders* mov = first;
00131
             while (mov != NULL) {
00132
                  d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
       mov->x_start[1], mov->y[1]);
00133
                   mov=mov->next;
00134
00135
              \texttt{d\_player}(\texttt{COL\_WHITE}, \texttt{COL\_BLACK}, \texttt{COL\_BLACK}, \texttt{COL\_BLACK}, \texttt{p-} \times \texttt{start[0]}, \texttt{p-} \times \texttt{[0]}, \texttt{p-} \times \texttt{start[1]}, \texttt{p-} \times \texttt{[1]}); \\
00136
              econio_flush();
00137 }
```

References d_invader(), d_player(), Invaders::next, Invaders::x_start, Player::x_start, Invaders::y, and Player::y.

5.7.1.4 d_invader()

draws invader starting y coords and x coord with tcolor as text color and bcolor as background color

```
00111
00112 econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00113 econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
00114 printf("*(-_-)*");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00115 econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
00116 printf("$._$_.$");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00117 }
```

5.7.1.5 d_player()

```
void d_player (
    int tcolor,
    int bcolor,
    int tcolor2,
    int bcolor2,
    int x,
    int y,
    int x2,
    int y2)
```

draws player starting y coords and x coord with tcolor as text color and bcolor as background color

5.7.1.6 d_score()

```
void d_score (
             int score)
draws score
00139
00140
         econio_textbackground(COL_BLACK);
         econio_gotoxy(20,32);
00141
00142
         econio_textcolor(COL_WHITE);
00143
         printf("SCORE: %d", score);
00144
         econio_textcolor(COL_BLACK);
00145 }
5.7.1.7 mov_invx()
void mov_invx (
              Invaders * first,
              int way)
move invaders horizontally
00156
                                           {
         Invaders* mov = first;
00158
         while (mov != NULL) {
00159
            d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
way==1?mov->x_start[0]++:mov->x_start[0]--;
way==1?mov->x_start[1]++:mov->x_start[1]--;
00161
00162
            way==1?mov->x_end[0]++:mov->x_end[0]--;
            way==1?mov->x_end[1]++:mov->x_end[1]--;
00164
             d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov=mov->next;
```

References d_invader(), Invaders::next, Invaders::x_end, Invaders::x_start, and Invaders::y.

5.7.1.8 mov_invy()

}

00166

00167 }

```
void mov_invy (
              Invaders * first)
move invaders vertically
00170
00171
          Invaders* mov = first;
         while (mov != NULL) {
00172
00173
             d_invader(COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
     mov->x_start[1], mov->y[1]);
00174 mov -> y[0] += 2;
00175
             mov -> y[1] += 2;
             d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
00176
     mov->x_start[1], mov->y[1]);
mov=mov->next;
00177
00178
00179 }
```

References d_invader(), Invaders::next, Invaders::x_start, and Invaders::y.

5.7.1.9 peasy()

draws easy text starting y coords and tabcounts*8 + 4 x coord with tcolor as text color and bcolor as background

```
00070
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
00071
          econio_textcolor(tcolor);printf(" __ _ _ . .");
00072
          econio_textbackground(b2color);
          econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__ | [__ \\_/ ");
00073
00074
          econio_textbackground(b2color);
00075
          econio_gotoxy(8*tabcount + 4, y+2);econio_textbackground(bcolor);printf(" |__ | | ___] | ");
          econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
econio_textbackground(bcolor);printf(" ");
00076
00077
00078
          econio_textbackground(b2color);
00079 }
```

5.7.1.10 phard()

draws hard text starting y coords and tabcounts*8 + 4 x coord with tcolor as text color and bcolor as background

```
color
00096
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
00097
             econio_textbackground(bcolor);
             econio_textbackground(bccolor);printf(" . . . _ . - . _ ");
econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+1);
econio_textbackground(bccolor);printf(" | _ | | _ | | _ / | \ ");
00098
00099
00100
             00102
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
econio_textbackground(bcolor);printf(" ");
00103
00104
00105
             econio_textbackground(b2color);
econio_textcolor(COL_BLACK);
00106
00107 }
```

5.7.1.11 pmed()

draws medium text starting y coords and tabcounts *8 + 1 x coord with tcolor as text color and bcolor as background

```
color
00082
              econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y);
00083
00084
             econio_textbackground(bcolor);econio_textcolor(tcolor);printf("
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+1);
econio_textbackground(bcolor);printf(" |\\/| | __ | \\ | | | \\/| ");
00085
00086
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+2);
econio_textbackground(bcolor);printf(" | | |__ |__ / | |__ | ");
00087
00088
             econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+3);
econio_textbackground(bcolor);printf(" ");
00089
00090
00091
             econio_textbackground(b2color);
00092 }
```

5.7.1.12 pquit()

draws quit text starting y coords and tabcounts*8 + 6 x coord with tcolor as text color and bcolor as background

```
color
00041
           econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y);
00042
                                                                                                  _ ___");
00043
           econio_textbackground(bcolor);econio_textcolor(tcolor);printf("
           econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+1);
00044
00045
           econio_textbackground(bcolor);printf("| | |
00046
           econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+2);
00047
           econio_textbackground(bcolor);printf("|__\\ |__| _|_
           econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+3);
econio_textbackground(bcolor);printf(" ");econio_textbackground(bcolor);
00048
                                                                          ");econio_textbackground(b2color);
00049
00050
           econio_textcolor(COL_BLACK);
00051 }
```

5.7.1.13 printSI()

draws SPACE INVADERS text starting y coords and tabcounts*8 x coord with tcolor as text color and bcolor as background color

```
00010
          econio_textbackground(bcolor);
00011
00012
          econio textcolor(tcolor);
     00013
                                                                    ##\\###\\ ##\\##\\
0015 econio_gotoxy(8*tabcounter, y+1);

printf("##----/##/--##\\##/----/\n");

0017 econio_gotoxy(8*tabcounter, y+2);
                                                                 ##|###\\ ##|##|
      00018
                                                    ####\\
                                                                 ##|##/##\\ ##|##| ##|######|##|
00019
          econio_gotoxy(8*tabcounter, y+3);
     printf("\\---##|##/--/ ##/-##|##|
##|##/-/ ##/-##\\\\---##|\n");
econio_gotoxy(8*tabcounter, y+4);
                                                    ##/--/
                                                                ##|##|\\##\\##|\\##| ##/--##|##|
00020
00021
          printf("#########
                                   ##| ##|\\#####\\#####\\
                                                                  ##|##| \\###| \\###/ ##| ##|#####/
00022
      #######\\##| ##| #####|\n");
          econio_gotoxy(8*tabcounter, y+5);
printf("\\____/\\_/ \
00023
                                                                     \\_/\\_/ \\__/ \\__/
00024
                                       _/ \\_/ \\.
00025 }
```

5.7.1.14 pset()

5.8 draw.h 39

draws settings text starting y coords and tabcounts *8 + 2 x coord with tcolor as text color and bcolor as background

```
color
00054
00055
         econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
00056
         econio_textbackground(bcolor);
00057
         econio_textcolor(tcolor);printf("
         00058
00059
00060
         econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+2);
         econio_textbackground(bcolor);printf("__)|__ | | _|_| \\|\\_|__)
00061
         econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
00062
00063
         econio_textbackground(bcolor);printf("
00064
         econio_textbackground(b2color);
00065 }
```

5.7.1.15 pstart()

```
void pstart (
          int tabcount,
          int y,
          int tcolor,
          int bcolor,
          int b2color)
```

draws START text starting y coords and tabcounts*8 x coord with tcolor as text color and bcolor as background

```
color
00028
         econio_textbackground(b2color);econio_gotoxy(8*tabcount, y);econio_textbackground(bcolor);
00029
         econio_textcolor(tcolor);printf(" __ __ __ __ __");
00030
         econio_textbackground(b2color);
00032
         econio_gotoxy(8*tabcount, y+1);econio_textbackground(bcolor);printf("(__ | [__][__) | ");
00033
         econio_textbackground(b2color);
00034
         econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
econio_textbackground(bcolor);printf(" ");
00035
00036
00037
         econio_textbackground(b2color);
00038 }
```

5.8 draw.h

Go to the documentation of this file.

```
00001 //for menu
00002
00003 ///draws SPACE INVADERS text starting y coords and tabcounts \star 8 x coord with toolor as text color and
      bcolor as background color
00004 void printSI(int tabcounter, int y, int tcolor, int bcolor);
00005 ///draws START text starting y coords and tabcounts \star 8 x coord with tcolor as text color and bcolor as
      background color
00006 void pstart(int tabcount, int y, int tcolor, int bcolor, int b2color);
00007 ///draws settings text starting y coords and tabcounts \star 8 + 2 x coord with tcolor as text color and
      bcolor as background color
00008 void pset(int tabcount, int y, int tcolor, int bcolor, int b2color);
00009 ///draws quit text starting y coords and tabcounts \star 8 + 6 x coord with toolor as text color and boolor
      as background color
00010 void pquit(int tabcount, int y, int tcolor, int bcolor, int b2color);
00011 ///draws easy text starting y coords and tabcounts \star 8 + 4 x coord with tcolor as text color and bcolor
      as background color
00012 void peasy(int tabcount, int y, int tcolor, int bcolor, int b2color);
00013 ///draws medium text starting y coords and tabcounts \star 8 + 1 x coord with tcolor as text color and
      bcolor as background color
00014 void pmed(int tabcount, int y, int tcolor, int bcolor, int b2color);
00015 ///draws hard text starting y coords and tabcounts \star 8 + 4 x coord with tcolor as text color and bcolor
      as background color
00016 void phard(int tabcount, int y, int tcolor, int bcolor, int b2color);
00017
00018
00019 //for game
00020
00021 //draws invader starting v coords and x coord with tcolor as text color and bcolor as background
00022 void d_invader(int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2);
```

```
00023 ///draws player starting y coords and x coord with tcolor as text color and bcolor as background color
00024 void d_player(int tcolor, int bcolor, int tcolor2, int bcolor2, int x, int y, int x2, int y2);
00025 ///draws score
00026 void d_score(int score);
00027 ///Invaders data struct
00028 typedef struct Invaders Invaders;
00029 ///Player data struct
00030 typedef struct Player Player;
00031 ///Bullets data struct
00032 typedef struct Bullets Bullets;
00033 //draws map first time
00034 void d_init(Invaders* first, Player* p);
00035 ///draws bullets
00036 void d_bullet(Bullets b, int white);
00037 ///move invaders horizontally
00038 void mov_invx(Invaders* first, int way);
00039 ///move invaders vertically
00040 void mov_invy(Invaders* first);
00041 //draws bullet count
00042 void d_bullets(int sdb_b);
```

5.9 headers/game.h File Reference

Functions

• int game (int mode)

game logic

5.9.1 Function Documentation

5.9.1.1 game()

int game (

```
int mode)
game logic
game logic
00210
           econio_clrscr();
00211
           //setup phase
          int score = 0; // the score which to export
int b_db = 0;
00212
00213
00214
           int sdb_b = 0;
00215
           bool run = true;
          Player* p = Setup_Player();//strange, but it is what it is
Invaders* first = Setup_Inv(mode);
00216
00217
00218
           int row_length[3] = \{6,7,8\};
00219
          int cycle = 0;
00220
          int way = 1;
           int test = 1;
00222
           int x_ref = first->x_start[0];
00223
          Bullets* b =(Bullets*)malloc(50*sizeof(Bullets)); //setup for future
00224
00225
          //first draw
          d_init(first, p);
00226
00227
          //game logic
00228
           set_conio_terminal_mode();
00229
           while(run){
00230
               econio_rawmode();
               while(!kbhit()){
00231
                   //check if bullet has collision with invader
00232
                    if(b_db > 0)test = check_bullet(first,b,&score,b_db);
if(test == 0){
00234
00235
                        econio_normalmode();
00236
                        reset_terminal_mode();
                        //{\tt before} exit free the bullets :)
00237
00238
                        free(b);
00239
                        free(p);
                        freeInvaders(first);//yeah free the slaves!!!
```

```
00241
                      debugmalloc_dump();
00242
                      return score;
00243
                  else if(test == 2){
00244
                      Invaders* mov = first;
first = first->next;
00245
00246
                      free(mov);
00248
00249
                   //checks if invader is in line with player
00250
                  if(first->y[0] == 28 || first->y[0] == 29){
                      econio_normalmode();
00251
00252
                      reset terminal mode();
00253
                       //before exit free the bullets :)
00254
00255
                      free(p);
00256
                       freeInvaders(first);//yeah free the slaves!!!
00257
                      debugmalloc_dump();
00258
                      return score;
00259
00260
                   //check collisions
00261
                   //bullets draw and such
00262
                   if(cycle%3==0){
                       for(int i = 0; i < b_db; i++){</pre>
00263
                          if(b[i].y != 0) {
    d_bullet(b[i], 0);
00264
00265
00266
                               b[i].y--;
00267
                               d_bullet(b[i], 1);
00268
                               econio_gotoxy(0, 30);
00269
                               printf("%d",b[0].y);
                               econio_flush();
00270
00271
00272
                           else {d_bullet(b[i], 0);}
00273
00274
00275
                   if(cycle%(1+numb_inv(first)) == 0 || cycle%20 == 0){
00276
                       //move invaders
                      if(x_ref < 70 && way){</pre>
00277
00278
                          mov_invx(first, way);
00279
                           x_ref++;
00280
00281
                       else{
00282
                           if (way) {mov_invy(first);}
00283
                           wav = 0:
                           if(x_ref >= 7*row_length[mode]){
00284
00285
                              mov_invx(first, way);
00286
                               x_ref--;
00287
00288
                           else{
00289
                               wav = 1:
00290
                               mov_invy(first);
00291
00292
00293
00294
                  if(cycle%100 == 0){sdb_b = 0;}
00295
                  d_score(score);
00296
                  d_bullets(sdb_b);
00297
00298
                  usleep(23000);
00299
                  cycle++;
00300
                  if(cycle == 101)cycle = 0;
00301
00302
              econio_normalmode();
00303
              //did it with switch too, but who cares
00304
              //check keyboard press
00305
              int key = getch();
              //move player to left
if(key== 'a'){
00306
00307
                  if(p->x_start[1] > 0){
00308
00309
      00310
                      p->x_start[0]-=2;
00311
                      p->shoot_pos_x-=2;
00312
                      p->x_start[1]-=2;
00313
      d player (COL WHITE, COL BLACK, COL BLACK, COL BLACK, p->x start[0], p->y[0], p->x start[1], p->y[1]);
00314
00315
              }
              //move player to right
else if(key == 'd'){
00316
00317
                  if(p->x_start[1] < 70){
00318
00319
     d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
00320
                      p->x_start[0]+=2;
00321
                      p->shoot_pos_x+=2;
00322
                      p->x_start[1]+=2;
00323
      d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
```

```
}
00325
00326
                  //shoot bullet
00327
              else if(key == KEY_SPACE && sdb_b < 7) {</pre>
00328
                  sdb_b++;
00329
                  b[b_db].x=p->shoot_pos_x;
00330
                  b[b_db].y=p->shoot_pos_y;
00331
00332
                   if(b_db%49 == 0)b =(Bullets*)realloc(b,(b_db + 51)*sizeof(Bullets));
00333
                  d_bullet(b[b_db-1], 1);
00334
              }
00335
00336
          debugmalloc_dump();
00337
          return score;
00338 }
```

References check_bullet(), d_bullet(), d_init(), d_player(), d_score(), freeInvaders(), getch(), kbhit(), KEY_SPACE, mov_invx(), mov_invy(), Invaders::next, numb_inv(), reset_terminal_mode(), set_conio_terminal_mode(), Setup_Inv(), Setup_Player(), Player::shoot_pos_x, Player::shoot_pos_y, Bullets::x, Invaders::x_start, Player::x_start, Bullets::y, Invaders::y, and Player::y.

5.10 game.h

Go to the documentation of this file.

```
00001 ///game logic
00002 int game(int mode);
```

5.11 headers/menu.h File Reference

Functions

• int menu ()

Menu logic.

5.11.1 Function Documentation

5.11.1.1 menu()

```
int menu ()
```

Menu logic.

```
Menu logic.
```

```
00014 #if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
            SMALL_RECT WinSize = {0, 0, 135, 35};
SMALL_RECT* WinWin = &WinSize;
00015
00016
00017
            SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
            //plays in a loop music.wav file in windows, just do not have music.wav yet //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
00018
00019
00020 #endif
00021
            econio_clrscr();
00022
            //draws menu items
            printSI(1,3,COL_WHITE, COL_BLACK);
00023
            pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
//pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
00024
00025
00026
            pquit(6, 16, COL_WHITE, COL_BLACK, COL_WHITE);
00027
00028
            int last = 0;
00029
            int before = 0;
00030
           while(true) {
00031
                 econio_rawmode();
```

```
00032
                while(true) {
00033
                         //reads key and do state magic, save previous state and update current state
                          int ch = econio_getch();
if(ch == KEY_UP && last != 0){before = last; last--;}
else if(ch == KEY_DOWN && last != 1){before = last; last++;}
else if(ch == KEY_ENTER) break;
00034
00035
00036
00037
                          //draws new state of menu items or state machine
00039
                          switch(last){
00040
                              case 0:
                                    //start text is shiney, settings not shiney anymore
pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
pquit(6,16,COL_WHITE, COL_BLACK, COL_WHITE);
00041
00042
00043
00044
                                    break;
00045
00046
                                     \star was a good idea, just not in c, devistated am i, said Yoda as
00047
                                     \star possible upgrade if client wants to get a new feture in like a settings
      button in production
00048
                                     * gonna leave it here as nobody gonna touch it and there is space for it
00049
00050
                               case 1:
00051
                                    if(before == 0){
00052
                                         //\mathrm{start} not shiney, settings shiney
                                         pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
00053
00054
                                         pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00055
00056
                                    else {
00057
                                         //quit not shiney, settings shiney
                                         pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00058
00059
                                         pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
00060
00061
                                    break:
00062
00063
                               case 1:
                                    //settings not shiney, quit shiney
pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
pquit(6, 16, COL_BLACK, COL_WHITE, COL_BLACK);
00064
00065
00066
00067
                                    break;
00068
                          }
00069
                     }
00070
                          econio_normalmode();
00071
                           //choice is quit, do exit with screen to default
00072
                          if(last == 1){
00073
                               \verb|econio_textbackground(COL_BLACK)|;
00074
                               econio_clrscr();
00075
                               econio_textcolor(COL_WHITE);
00076
                               exit(0);
00077
00078
                          else {
00079
                               //choice is start, do clear screen, draw dificulty elements
                               econio_textbackground(COL_BLACK);
00080
00081
                               econio_clrscr();
00082
                               econio_textcolor(COL_WHITE);
00083
                               peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
                               pmed(6, 13, COL_WHITE, COL_BLACK, COL_WHITE);
phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
00084
00085
00086
00087
00088
00089
                               econio_rawmode();
00090
                               while (true) {
00091
                                    //read keys and save previous state, and update now state
00092
                                    int ch = econio_getch();
if(ch == KEY_UP && last != 0){before = last; last--;}
00093
00094
                                    else if(ch == KEY_DOWN && last != 2) {before = last; last++;}
00095
                                    else if(ch == KEY_ENTER) break;
00096
00097
                                    //state machine goes brrrrrrr
00098
                                    switch(last){
00099
                                        case 0:
00100
                                              //easy text is shiney, medium is not
00101
                                              peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
00102
                                              pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00103
                                             break;
00104
                                         case 1:
00105
                                              if(before == 0){
                                                  //easy was shiney, not now anymore, medium shiney
00106
00107
                                                  peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
00108
                                                  pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
00109
00110
                                              else {
                                                  //hard not shiney anymore, meium is shiney pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
00111
00112
00113
                                                  phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
00114
                                             break;
00115
00116
                                         case 2:
00117
                                             //medium not shinev, hard is shinev
```

```
pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00119
                                       phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
00120
00121
                               }
00122
00123
00124
                          econio_normalmode();
00125
                       //before exit to who knows where, clean screen
00126
                          econio_textbackground(COL_BLACK);
00127
                       econio_clrscr();
                       econio_textcolor(COL_WHITE);
00128
00129
00130
                      return last;
00131
00132
              }
00133 }
```

References peasy(), phard(), pmed(), pquit(), printSI(), and pstart().

5.12 menu.h

Go to the documentation of this file.

```
00001 ///Menu logic 00002 int menu();
```

5.13 headers/score.h File Reference

Functions

- void printboard (FILE *fptr, int db)
 - prints highscore board while getting the data from the save file, needs location and max amount to draw
- void savewrite_score (int state, int score)
 - write to file to the specified file based on state to save new data or update old score
- void highscore (int state)

get input to how many elements to write based on state, medium for printboard

5.13.1 Function Documentation

5.13.1.1 highscore()

```
void highscore (
          int state)
```

get input to how many elements to write based on state, medium for printboard

get input to how many elements to write based on state, medium for printboard

```
00229
00230
           //set window size for good windows users :), but i like linux better, just not the terminal
      modifying
00231 #if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
         SMALL_RECT WinSize = {0, 0, 70, 40};
SMALL_RECT* WinWin = &WinSize;
00232
00233
          SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00234
00235 #endif
00236
00237
          //highscore board setup
00238
         FILE* fptr;
          char paths[][50] = {"../highscores/11.txt","../highscores/12.txt","../highscores/13.txt"};
00239
00240
          int testDB = 0;int db = 0;
00241
          printf("Give how many scores to show from the top, 0 to not show anything");
```

```
printf("\nor negative number as it will be converted to a positive number\n");
00243
           printf("number of players to show from top 1: ");
00244
           testDB = scanf("%d",&db);
          while(testDB < 1 || db < 0 || db > 1000000) {
00245
00246
               econio_clrscr();
00247
               printf("number of players to show from top 1: ");
               testDB = scanf("%*c%d%*c",&db);
00249
00250
00251
           //checks if highscores directory exist and openable
          DIR* dir2;
dir2 = opendir("../highscores");
00252
00253
00254
           if (dir2) {closedir(dir2);}
00255
           //directory does not exist
00256
          else if(ENOENT == errno){
00257
               closedir(dir2);
              mkdir("../highscores", 0700);
dir2 = opendir("../highscores");
00258
00259
               if(dir2) {closedir(dir2);}
00260
00261
               //directory does not exist
               else if(ENOENT == errno){
00262
00263
                  closedir(dir2);
00264
                   perror ("Could not create highscores directory, please create it manually to the same
      directory as main.c");
00265
                   exit(0);
00266
00267
               //other problem
00268
               else{
                   closedir(dir2);
perror("Error occured regarding the highscores directory");
00269
00270
00271
                   exit(0):
00272
               }
00273
00274
           //other problem
00275
           else {
              closedir(dir2);
00276
00277
               perror("Error occured regarding the highscores directory");
00278
               exit(0);
00279
00280
           //state machine, would be better to put paths into a 2d char array, but who cares
00281
           //its not like anybody gonna expand this sht
          switch(state){
00282
              //show relevant highscore
00283
00284
               //dont wanna make a function to read file and store the data
               case 0:
00285
00286
                   fptr = fopen(paths[state], "r");
                   if(fptr == NULL){
   fptr = fopen(paths[state], "w");
   if(fptr == NULL){
00287
00288
00289
                           perror("File open error");
00290
00291
                            exit(0);
00292
00293
00294
                   printboard(fptr, db);
00295
                   break;
00296
               case 1:
00297
                  //show highscore in medium diff
00298
                   fptr = fopen(paths[state], "r");
                   if(fptr == NULL) {
    fptr = fopen(paths[state], "w");
00299
00300
00301
                        if(fptr == NULL){
                           perror("File open error");
00302
00303
                            exit(0);
00304
00305
00306
                   printboard(fptr, db);
00307
                   break;
00308
               case 2:
00309
                  //high diff score
                   fptr = fopen(paths[state], "r");
00310
                   if(fptr == NULL) {
    fptr = fopen(paths[state], "w");
00311
00312
                        if(fptr == NULL) {
    perror("File open error");
00313
00314
00315
                            exit(0);
00316
00317
00318
                   printboard(fptr, db);
00319
                   break;
00320
          }
00321 }
```

References printboard().

5.13.1.2 printboard()

```
void printboard ( \label{eq:file} {\tt FILE} \, * \, {\tt fptr,} int {\tt db})
```

prints highscore board while getting the data from the save file, needs location and max amount to draw

prints highscore board while getting the data from the save file, needs location and max amount to draw

```
00021
           if(db == 0)return;
00022
           //variables
00023
           Data data[1] = {};
           int index = 0;
const char* padding = "
00024
00025
                                                                                    ";//soooo long, idk why
           char buff[5] = "";
00026
00027
           //drawing begining
00028
           econio_clrscr();
00029
           econio_gotoxy(30,1);
00030
           printf("++++++++
00031
           econio_gotoxy(30,2);
00032
           printf("|NAME
                              | SCORE | ");
           econio_gotoxy(30,3);
printf("+++++++++++++++");
00033
00034
00035
00036
           int tName =0; int tSCR = 0;
00037
           //magic is done here
00038
           while(index < db){</pre>
     //checks if data was corrupted
if((tName = fscanf(fptr, "%s", data->name)) == -1 || tName == 0||(tSCR = fscanf(fptr, "%d", &data->score)) == -1 || tSCR == 0){
00039
00040
                  //checks if its end of file, if first if did not work
00041
00042
                    if((tSCR = fscanf(fptr, "%d", &data->score)) == -1) {break;}
                    econio_clrscr();
printf("\nCorrupted file\n");
00043
00044
00045
                    exit(0);
00046
00047
               //checks if earlier break is needed
00048
               if((tName == EOF && tSCR == EOF) || (tName == 0 && tSCR == 0)) {break;}
00049
00050
               sprintf(buff, "%d", data->score);//transform int to string aka char array
00051
00052
               //get padding
00053
               int pad1Len = (TARGETLEN*3/5) - strlen(data->name);
00054
                int pad2Len = (TARGETLEN*2/5) - strlen(buff);
00055
00056
               //draw board
               cconio_gotoxy(30, 4+(2*index));
printf("|%s%*.*s|%*.*s%d|", data->name, padlLen, padlLen,
00057
00058
      padding,pad2Len,pad2Len,padding,data->score);
00059
               econio_gotoxy(30, 4+(2*index+1));
00060
               printf("+++++++++++++++");
00061
00062
               index++;//magic number
00063
00064
           printf("\n");
00065
           printf("\nnPress ENTER to start the game\n");
00066
           econio_rawmode();
00067
           while(1){
               int key = econio_getch();
if(key == KEY_ENTER) {break;}
00068
00069
00070
00071
           econio_normalmode();
00072 }
```

References datas::name, datas::score, and TARGETLEN.

5.13.1.3 savewrite_score()

```
void savewrite_score (
    int state,
    int score)
```

write to file to the specified file based on state to save new data or update old score

write to file to the specified file based on state to save new data or update old score if name is in list and current score is higher than old, change it, than sort current arrays to descending based on scores

```
00075
00076
          econio clrscr();
00077
          econio_textcolor(COL_WHITE);
00078
          //set file path with state
          FILE* fptr;
00079
          char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
00080
00081
          fptr = fopen(paths[state], "r");
          if(fptr == NULL) {perror("Scoresave: file opening error");}
00082
00083
          else{
00084
00085
               * for legacy code to just exist here or to return to something in case of shit
00086
              switch(state){
00087
                  case 0:
00088
                       fptr = fopen("../highscores/11.txt", "r+");
00089
                       break;
00090
                   case 1:
00091
                       fptr = fopen("../highscores/11.txt", "r+");
00092
                      break:
00093
                  case 2:
00094
                      fptr = fopen("../highscores/11.txt", "r+");
00095
                       break;
00096
00097
              */
00098
              //set up variables to work with
00099
00100
              char name [MAX_NAME + 2] = {};
              int index = 0;
00101
00102
              Data *p = calloc(3, sizeof(*p)); //for safety and to pass new data allocate 3 times the space
00103
              Data temp;
00104
              bool in = false;
00105
              int i new;
00106
               //checks if highscores directory exist and openable
              DIR* dir;
00108
00109
              dir = opendir("../highscores");
00110
               if(dir){closedir(dir);}
              //checks if directory do not exist
00111
              else if (ENOENT == errno) {
00112
00113
                  closedir(dir);
00114
                  mkdir("../highscores", 0700);
00115
                  dir = opendir("../highscores");
00116
                   if(dir){closedir(dir);}
                  //checks if directory still not exist
else if(ENOENT == errno) {
00117
00118
00119
                      closedir(dir);
00120
                       free(p);
00121
                       debugmalloc_dump();
00122
                       perror("Could not create highscores directory, please create it manually to the same
     directory as main.c");
00123
                       exit(0);
00124
00125
                   //other problem
00126
                  else{
00127
                       closedir(dir);
00128
                       free(p);
                       debugmalloc_dump();
00129
00130
                       perror ("Error occured regarding the highscores directory");
00131
                       exit(0);
00132
                   }
00133
00134
              //other problem
00135
              else {
00136
                  closedir(dir);
00137
                  free(p);
00138
                  debugmalloc_dump();
00139
                  perror("Error occured regarding the highscores directory");
00140
                  exit(0);
00141
              }
00142
00143
              FILE* n = fopen("../highscores/n.txt", "w");
00144
                     = NULL) {perror("Scoresave: file opening error, at line 80");}
00145
00146
                  //input name
                  printf("!Your name can only be 12 character long!");
printf("Your name: ");
00147
00148
00149
                   int testName = 0;
00150
                  while((testName = scanf("%s", name)) == 0 || testName == -1 || strlen(name) > 12){
00151
                       printf("!Your name can only be 12 character long!");
                      printf("Your correct name: ");
00152
00153
                   ^{\prime}//load to struct the saved data
00154
                  //yeah nested loops, its not good, but it is what it is, who else gonna touch this code
00155
      except me anyway?
00156
                  //while(fscanf(fptr,"%s %d",p[index].name, &p[index].score) != EOF){
00157
```

```
00158
                    int testStr = 0; int testINT = 0;
while((testStr = fscanf(fptr,"%s",p[index].name)) != EOF && (testINT = fscanf(fptr,"%d",
00159
00160
      &p[index].score)) != EOF) {
00161
                         //tests if readed data was corrupted
00162
                         if(testStr == 0 || testStr == -1 || testINT == 0 || testINT == -1){
                             perror("Corrupted file");
00163
00164
00165
                         /// if name is in list and current score is higher than old, change it, than sort
00166
      current arrays to descending based on scores
00167
                        //check if name is in the list
                         if(strcmp(name, p[index].name) == 0){
   in = true;//is it in or not?
00168
00169
00170
                             //check to update its score
00171
                             if(score > p[index].score) {
                                  p[index].score = score;
if(index != 0){
00172
00173
00174
                                      //bubble sort
00175
                                       for(int i = 0; i <= index-1; i++) {</pre>
00176
                                           for(int j = i+1; j <= index; j++) {</pre>
00177
                                               if(p[j].score >= p[i].score){
                                                    temp = p[i];
p[i] = p[j];
00178
00179
00180
                                                    p[j] = temp;
00181
                                               }
00182
                                           }
00183
                                      }
00184
                                  }
00185
                             }
00186
00187
                         //gets ready for next item, reallocating space
00188
00189
                         p = (Data*) realloc(p, (index+1)*sizeof(*p));
00190
                    }
00191
00192
                    //if not in highscorer list then add it to the top of its score group
                    bool found = false;
00193
00194
                    if(!in){
00195
                         for(int i = 0; i < index; i++) {</pre>
00196
                             if(score >= p[i].score && !found) {
   i_new = i;
00197
                                  found = true;
00198
00199
                                  break;
00200
                             }
00201
                         }
00202
                    //if nobody is in the list put him in
if(index == 0)fprintf(n, "%s %d",name, score);
00203
00204
00205
                    //save list to file
00206
                    else {
00207
                         for(int i = 0; i < index; i++) {</pre>
00208
                             if(i == i_new) fprintf(n, "%s %d\n", name, score);
00209
                             fprintf(n, "%s %d\n",p[i].name, p[i].score);
00210
00211
00212
                    //free buff and close file stream
00213
                    free(p);
00214
                    fclose(n);
00215
                    fclose(fptr);
                    //check if any problem occured while freeing memory debugmalloc_dump();
00216
00217
00218
                    //delete old, rename new save file
00219
                    //yeah there are better ways to do it, but i dont want to touch this at this point
00220
                    remove(paths[state]);
00221
                    rename("../highscores/n.txt",paths[state]);
00222
               }
00223
           }
00224 }
```

References MAX NAME, and datas::score.

5.14 score.h

Go to the documentation of this file.

```
00001
00002
00003 ///prints highscore board while getting the data from the save file, needs location and max amount to draw
00004 void printboard(FILE* fptr, int db);
```

5.15 main.c File Reference 49

```
00005
00006 ///write to file to the specified file based on state to save new data or update old score
00007 void savewrite_score(int state, int score);
00008
00009 ///get input to how many elements to write based on state, medium for printboard
00010 void highscore(int state);
```

5.15 main.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "menu.h"
#include "score.h"
#include "game.h"
```

Functions

• int main ()

5.15.1 Function Documentation

5.15.1.1 main()

```
int main ()
00007
         //it is looped to be played again while(1){
00008
00009
          int state = menu();
00010
00011
              highscore(state);
00012
              int score = game(state);
              savewrite_score(state, score);
00013
00014
          }
00015 }
```

References game(), highscore(), menu(), and savewrite_score().

5.16 README.md File Reference

Index

```
Bullets. 7
                                                              d_score, 13
     data_struct.h, 31
                                                              mov_invx, 13
     x, 7
                                                              mov_invy, 14
                                                              peasy, 14
    y, 7
                                                              phard, 14
cfiles/draw.c, 11
                                                              pmed, 15
cfiles/game.c, 17
                                                              pquit, 15
cfiles/menu.c, 23
                                                              printSI, 15
cfiles/score.c, 25
                                                              pset, 16
check bullet
                                                              pstart, 16
     game.c, 18
                                                         draw.h
                                                              d bullet, 34
d bullet
                                                              d_bullets, 34
     draw.c, 12
                                                              d init, 34
    draw.h, 34
                                                              d invader, 35
d bullets
                                                              d_player, 35
    draw.c, 12
                                                              d_score, 35
     draw.h, 34
                                                              mov_invx, 36
d init
                                                              mov_invy, 36
     draw.c, 12
                                                              peasy, 36
    draw.h, 34
                                                              phard, 37
d invader
                                                              pmed, 37
     draw.c, 12
                                                              pquit, 37
     draw.h, 35
                                                              printSI, 38
d player
                                                              pset, 38
     draw.c, 13
                                                              pstart, 39
    draw.h, 35
d score
                                                         freeInvaders
     draw.c, 13
                                                              game.c, 18
     draw.h, 35
Data
                                                         game
     data_struct.h, 31
                                                              game.c, 19
data_struct.h
                                                              game.h, 40
     Bullets, 31
                                                         game.c
                                                              check bullet, 18
     Data, 31
     Invaders, 31
                                                              freeInvaders, 18
     KEY_SPACE, 31
                                                              game, 19
     MAX NAME, 31
                                                              getch, 21
     MAX_POINT_LENGTH, 31
                                                              kbhit, 21
     Player, 32
                                                              numb_inv, 21
     TARGETLEN, 31
                                                              orig_termios, 23
datas, 8
                                                              reset_terminal_mode, 21
     name, 8
                                                              set_conio_terminal_mode, 21
    score, 8
                                                              Setup_Inv, 22
draw.c
                                                              Setup_Player, 22
     d bullet, 12
                                                         game.h
     d bullets, 12
                                                              game, 40
    d init, 12
                                                         getch
     d_invader, 12
                                                              game.c, 21
     d_player, 13
```

52 INDEX

| headers/data_struct.h, 30, 32 | data_struct.h, 32 |
|--|--|
| headers/draw.h, 33, 39 | shoot_pos_x, 10 |
| | _, _ |
| headers/game.h, 40, 42 | shoot_pos_y, 10 |
| headers/menu.h, 42, 44 | x_start, 10 |
| headers/score.h, 44, 48 | y, 10 |
| highscore | pmed |
| _ | • |
| score.c, 26 | draw.c, 15 |
| score.h, 44 | draw.h, 37 |
| | pquit |
| Invaders, 8 | |
| | draw.c, 15 |
| data_struct.h, 31 | draw.h, 37 |
| next, 9 | printboard |
| x_end, 9 | score.c, 27 |
| x_start, 9 | |
| | score.h, 45 |
| y, <mark>9</mark> | printSI |
| | draw.c, 15 |
| kbhit | draw.h, 38 |
| game.c, 21 | |
| KEY SPACE | pset |
| - | draw.c, 16 |
| data_struct.h, 31 | draw.h, 38 |
| | |
| main | pstart |
| main.c, 49 | draw.c, 16 |
| | draw.h, 39 |
| main.c, 49 | • |
| main, 49 | README, 1 |
| MAX NAME | |
| data_struct.h, 31 | README.md, 49 |
| | reset_terminal_mode |
| MAX_POINT_LENGTH | game.c, 21 |
| data_struct.h, 31 | ga |
| menu | agyourite goore |
| | |
| monu o 24 | savewrite_score |
| menu.c, 24 | score.c, 28 |
| menu.c, 24 menu.h, 42 | score.c, 28 |
| • | score.c, 28 score.h, 46 |
| menu.h, 42 menu.c | score.c, 28 score.h, 46 score |
| menu.h, 42 menu.c menu, 24 | score.c, 28 score.h, 46 |
| menu.h, 42 menu.c | score.c, 28 score.h, 46 score |
| menu.h, 42 menu.c menu, 24 | score.c, 28 score.h, 46 score datas, 8 score.c |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_Inv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 TARGETLEN |
| menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 draw.h, 36 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 draw.h, 36 phard | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 TARGETLEN data_struct.h, 31 |
| menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 draw.h, 36 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 TARGETLEN data_struct.h, 31 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 draw.h, 36 phard draw.c, 14 | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 TARGETLEN data_struct.h, 31 |
| menu.h, 42 menu.c menu, 24 menu.h menu, 42 mov_invx draw.c, 13 draw.h, 36 mov_invy draw.c, 14 draw.h, 36 name datas, 8 next Invaders, 9 numb_inv game.c, 21 orig_termios game.c, 23 peasy draw.c, 14 draw.h, 36 phard | score.c, 28 score.h, 46 score datas, 8 score.c highscore, 26 printboard, 27 savewrite_score, 28 score.h highscore, 44 printboard, 45 savewrite_score, 46 set_conio_terminal_mode game.c, 21 Setup_lnv game.c, 22 Setup_Player game.c, 22 shoot_pos_x Player, 10 shoot_pos_y Player, 10 TARGETLEN data_struct.h, 31 |

INDEX 53

```
Invaders, 9
x_start
Invaders, 9
Player, 10

y
Bullets, 7
Invaders, 9
Player, 10
```