Space Invaders

Generated by Doxygen 1.12.0

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
```

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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:

-.I. (- 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                                   {
00182     econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
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)
```

draws bullet count

```
00147                           {
00148     econio_textbackground(COL_BLACK);
00149     econio_gotoxy(35, 32);
00150     econio_textcolor(COL_WHITE);
00151     printf("BULLETS: 7/ %d",7-sdb_b);
00152     econio_textcolor(COL_BLACK);
00153 }
```

5.1.1.3 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     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()

```
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

```
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()

```
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

```
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(bcolor);
00125     printf("(_U_)");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00126 }
```

5.1.1.6 d_score()

```
void d_score (
            int score)
```

draws score

```
00139                       {
00140     econio_textbackground(COL_BLACK);
00141     econio_gotoxy(20,32);
00142     econio_textcolor(COL_WHITE);
00143     printf("SCORE: %d",score);
00144     econio_textcolor(COL_BLACK);
00145 }
```

5.1.1.7 mov_invx()

```
void mov_invx (
            Invaders * first,
            int way)
```

move invaders horizontally

```
00156                                       {
00157     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],
      mov->x_start[1], mov->y[1]);
00160         way==1?mov->x_start[0]++:mov->x_start[0]--;
00161         way==1?mov->x_start[1]++:mov->x_start[1]--;
00162         way==1?mov->x_end[0]++:mov->x_end[0]--;
00163         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->x_start[1], mov->y[1]);
00165         mov=mov->next;
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;
00172     while(mov != NULL){
00173         d_invader(COL_BLACK, 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;
00176         d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
      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()

```
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

```
00069                                                                   {
00070     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
00071     econio_textcolor(tcolor);printf("  __  __   ___ .   . ");
00072     econio_textbackground(b2color);
00073     econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__| [__   \\_/  ");
00074     econio_textbackground(b2color);
00075     econio_gotoxy(8*tabcount + 4, y+2);econio_textbackground(bcolor);printf(" |__ |  | ___]   |   ");
00076     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
00077     econio_textbackground(bcolor);printf("                      ");
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 color

```
00095                                                          {
00096     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
00097     econio_textbackground(bcolor);
00098     econio_textcolor(tcolor);printf(" .  .  __  .--.  __  ");
00099     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+1);
00100     econio_textbackground(bcolor);printf(" |__| |__| |__/ |  \\ ");
00101     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+2);
00102     econio_textbackground(bcolor);printf(" |  | |  | |  \\ |__/ ");
00103     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
00104     econio_textbackground(bcolor);printf("                    ");
00105     econio_textbackground(b2color);
00106     econio_textcolor(COL_BLACK);
00107 }
```

5.1.1.11 pmed()

```
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

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

5.1.1.12 pquit()

```
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

```
00041                                                          {
00042     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y);
00043     econio_textbackground(bcolor);econio_textcolor(tcolor);printf(" __       ___ ___");
00044     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+1);
00045     econio_textbackground(bcolor);printf("|  | |  |  |    | ");
00046     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+2);
00047     econio_textbackground(bcolor);printf("|__\\| |__| _|_  | ");
00048     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+3);
00049     econio_textbackground(bcolor);printf("                ");econio_textbackground(b2color);
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);
00014     printf("#######\\######\\  #####\\  ######\\#######\\    ##\\###\\   ##\\##\\
      ##\\#######\\######\\ #######\\#####\\  #######\\\n");
00015     econio_gotoxy(8*tabcounter, y+1);
00016     printf("##-----/##/--##\\##/--##\\##/----/##/----/    ##|####\\  ##|##|
      ##|##/--##|##/--##|##/----/##/-##\\ ##/----/\n");
00017     econio_gotoxy(8*tabcounter, y+2);
00018     printf("#######\\#######/#######|##|     #####\\      ##|##/##\\ ##|##|   ##|#######|##|
      ##|#####\\  ######/ #######\\\n");
00019     econio_gotoxy(8*tabcounter, y+3);
00020     printf("\\----##|##/---/ ##/--##|##|     ##/--/      ##|##|\\##\\\##|\\##\\ ##/ ##/--##|##|
      ##|##/--/  ##/-##\\ \\----##|\n");
00021     econio_gotoxy(8*tabcounter, y+4);
00022     printf("#######|##|     ##|  ##|\\######\\#######\\    ##|##| \\####| \\####/  ##|  ##|######/
      #######\\##| ##| #######|\n");
00023     econio_gotoxy(8*tabcounter, y+5);
00024     printf("\\______/\\_/     \\_/  \\_/ \\_____/\\______/    \\_/\\_/  \\___/  \\___/  \\_/
      \\_/\\_____/ \\______/\\_/ \\_/ \\______/\n");
00025 }
```

5.1.1.14 pset()

```
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

```
00054                                                      {
00055     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
00056     econio_textbackground(bcolor);
00057     econio_textcolor(tcolor);printf(" __ ___________      __ __");
00058     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+1);
00059     econio_textbackground(bcolor);printf("(_ |_  |  |  | |\\ |/__(_ ");
00060     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+2);
00061     econio_textbackground(bcolor);printf("__)|__ |  | _|_| \\|\\_|__)");
00062     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
00063     econio_textbackground(bcolor);printf("                        ");
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);
00030     econio_textcolor(tcolor);printf(" __  ___  __  __  ___");
00031     econio_textbackground(b2color);
00032     econio_gotoxy(8*tabcount, y+1);econio_textbackground(bcolor);printf("(__   |  [__][__)  | ");
00033     econio_textbackground(b2color);
00034     econio_gotoxy(8*tabcount, y+2);econio_textbackground(bcolor);printf(".__)  |  |  ||  \\  | ");
00035     econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
00036     econio_textbackground(bcolor);printf("                    ");
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 <unistd.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 accesed from main.c.

Variables

- struct termios orig_termios

5.2.1 Function Documentation

5.2.1.1 check_bullet()

```
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.

```
00163                                                                   {
00164     Invaders* mov = first;
00165     Invaders* mov2 = mov;
00166     while(mov != NULL){
00167         for(int i = 0; i < db; i++){
00168             if(mov->y[0] == b[i].y || mov->y[1] == b[i].y){
00169                 if(b[i].x >= mov->x_start[0] && b[i].x <= mov->x_end[0]){
00170                     if(numb_inv(first) == 1){
00171                         *score += 5;
00172                         return 0;
00173                     }
00174                     if(mov == first){
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;
00177                         *score += 5;
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()

```
void freeInvaders (
            Invaders * i)
```

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

```
00136                                {
00137     ///will get current linked list member, free its sub arrays, got to next member and free previous
      member
00138     while(i != NULL){
00139         Invaders* buff = i;
00140         free(i->x_start);
00141         free(i->x_end);
00142         free(i->y);
00143         i = i->next;
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 accesed 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;
00214     int sdb_b = 0;
00215     bool run = true;
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;
00220     int way = 1;
00221     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
00226     d_init(first, p);
00227     //game logic
00228     set_conio_terminal_mode();
00229     while(run){
00230         econio_rawmode();
00231         while(!kbhit()){
00232             //check if bullet has collision with invader
00233             if(b_db > 0)test = check_bullet(first,b,&score,b_db);
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             }
00244             else if(test == 2){
00245                 Invaders* mov = first;
00246                 first = first->next;
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                 free(b);
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){
00263                 for(int i = 0; i < b_db; i++){
00264                     if(b[i].y != 0){
00265                         d_bullet(b[i], 0);
00266                         b[i].y--;
00267                         d_bullet(b[i], 1);
00268                         econio_gotoxy(0, 30);
00269                         printf("%d",b[0].y);
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);
00279                     x_ref++;
00280                 }
00281                 else{
00282                     if(way){mov_invy(first);}
00283                     way = 0;
00284                     if(x_ref >= 7*row_length[mode]){
00285                         mov_invx(first, way);
00286                         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 
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();
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 
      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         }
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){
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_bullets(), 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;
00055     if ((r = read(0, &c, sizeof(c))) < 0) {
00056         return r;
00057     } else {
00058         return c;
00059     }
00060 }
```

5.2.1.5 kbhit()

```
int kbhit ()
```

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

```
00042 {
00043     struct timeval tv = { 0L, 0L };
00044     fd_set fds;
00045     FD_ZERO(&fds);
00046     FD_SET(0, &fds);
00047     return select(1, &fds, NULL, NULL, &tv) > 0;
00048 }
```

5.2.1.6 numb_inv()

```
int numb_inv (
            Invaders * first)
```

counts how many invaders are there

```
00150                                {
00151     Invaders* mov = first;
00152     int count = 0;
00153     while(mov!=NULL){
00154         count++;
00155         mov=mov->next;
00156     }
00157     return count;
00158 }
```

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

```
00027 {
00028     struct termios new_termios;
00029
00030     // take two copies - one for now, one for later
00031     tcgetattr(0, &orig_termios);
00032     memcpy(&new_termios, &orig_termios, sizeof(new_termios));
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()

```
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.

```
00087                                {
00088     //0 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
00090     //in medium there will be six in a row and it will move at medium speed
00091     //in hard there vill be more rows: not 7 but 11, they will move much faster
00092     int S_Width = 6;//number of white characters between 2 invader
00093     int row_length[3] = {6,7,8};
00094     int numb_row[3] = {4,5,7};
00095     //linked list setup
00096     Invaders* list = NULL;
00097     for(int j = 0; j < numb_row[mode]; j++){
00098         for(int i = 0; i < row_length[mode]; i++){
00099             //create one element
00100             Invaders* mov;
00101             mov =(Invaders*)malloc(sizeof(Invaders));
00102             mov->next = list;//connect to previous
00103             //allocate space to the arrays
00104             mov->x_start = (int*)malloc(2*sizeof(int));
00105             mov->x_end = (int*)malloc(2*sizeof(int));
00106             mov->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
00110                 mov->x_start[0] = 0;
00111                 mov->x_start[1] = 0;
00112                 //end of hitbox coords
00113                 mov->x_end[0] = (mov->x_start[0]) + S_Width;
00114                 mov->x_end[1] = (mov->x_start[1]) + S_Width;
00115             }
00116             else{
00117                 //same just if its not the first element
00118                 mov->x_start[0] = (list->x_end[0]) +2;
00119                 mov->x_start[1] = (list->x_end[1]) +2;
00120                 mov->x_end[0] = (mov->x_start[0]) + S_Width;
00121                 mov->x_end[1] = (mov->x_start[1]) + S_Width;
00122             }
00123             //set y coords
00124             mov->y[0] = 3*j+1;
00125             mov->y[1] = 3*j+2;
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

```
00065                       {
00066     Player* p = (Player*)malloc(sizeof(Player));
00067
00068     p->x_start=(int*)malloc(2*sizeof(int));
00069     p->x_start[0] = 1;
00070     p->x_start[1] = 0;
00071
00072     //where the bullet will start from
00073     p->shoot_pos_x = 2;
00074     p->shoot_pos_y = 27;
00075
00076     //playing field will be 120*40, 120 width, 40 height
00077     p->y=(int*)malloc(2*sizeof(int));
00078     p->y[0] = 29;
00079     p->y[1] = 30;
00080
00081     return p;
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)
00015     SMALL_RECT WinSize = {0, 0, 135, 35};
00016     SMALL_RECT* WinWin = &WinSize;
00017     SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00018     //plays in a loop music.wav file in windows, just do not have music.wav yet
00019     //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
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);
00025     //pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
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
00034                 int ch = econio_getch();
00035                 if(ch == KEY_UP && last != 0){before = last; last--;}
00036                 else if(ch == KEY_DOWN && last != 1){before = last; last++;}
00037                 else if(ch == KEY_ENTER) break;
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                         /*
00046                          * was a good idea, just not in c, devistated am i, said Yoda as
00047                          * 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                             //start not shiney, settings shiney
00053                             pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
00054                             pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00055                         }
00056                         else {
00057                             //quit not shiney, settings shiney
00058                             pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00059                             pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
00060                         }
00061                         break;
00062                         */
00063                     case 1:
00064                         //settings not shiney, quit shiney
00065                         pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
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                 else {
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){
00091                         //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++;}
00095                         else if(ch == KEY_ENTER) break;
00096
00097                         //state machine goes brrrrrrrr
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){
00106                                     //easy was shiney, not now anymore, medium shiney
00107                                     peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
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:
00117                                 //medium not shiney, hard is shiney
00118                                 pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00119                                 phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
00120                                 break;
00121                         }
00122                     }
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 <errno.h>
#include <ctype.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)
00232     SMALL_RECT WinSize = {0, 0, 70, 40};
00233     SMALL_RECT* WinWin = &WinSize;
00234     SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00235 #endif
00236
00237     //highscore board setup
00238     FILE* fptr;
00239     char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
00240     int testDB = 0;int db = 0;
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){
00246         econio_clrscr();
00247         printf("number of players to show from top 1: ");
00248         testDB = scanf("%*c%d%*c",&db);
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);
00258         mkdir("../highscores", 0700);
00259         dir2 = opendir("../highscores");
00260         if(dir2){closedir(dir2);}
00261         //directory does not exist
00262         else if(ENOENT == errno){
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{
00269             closedir(dir2);
00270             perror("Error occured regarding the highscores directory");
00271             exit(0);
00272         }
00273     }
00274     //other problem
00275     else {
00276         closedir(dir2);
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:
```

```
00286             fptr = fopen(paths[state], "r");
00287             if(fptr == NULL){
00288                 fptr = fopen(paths[state], "w");
00289                 if(fptr == NULL){
00290                     perror("File open error");
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");
00299             if(fptr == NULL){
00300                 fptr = fopen(paths[state], "w");
00301                 if(fptr == NULL){
00302                     perror("File open error");
00303                     exit(0);
00304                 }
00305             }
00306             printboard(fptr, db);
00307             break;
00308         case 2:
00309             //high diff score
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 (
            FILE * fptr,
            int db)
```

prints specified highscore board

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

```
00020                                   {
00021     if(db == 0)return;
00022     //variables
00023     Data data[1] = {};
00024     int index = 0;
00025     const char* padding = "                                                  ";//soooo long, idk why
00026     char buff[5] = "";
00027     //drawing begining
00028     econio_clrscr();
00029     econio_gotoxy(30,1);
00030     printf("++++++++++++++++++");
00031     econio_gotoxy(30,2);
00032     printf("|NAME     | SCORE|");
00033     econio_gotoxy(30,3);
00034     printf("++++++++++++++++++");
00035
00036     int tName =0; int tSCR = 0;
00037     //magic is done here
00038     while(index < db){
00039         //checks if data was corrupted
00040         if((tName = fscanf(fptr,"%s", data->name)) == -1 || tName == 0||(tSCR = fscanf(fptr,
      "%d",&data->score)) == -1 || tSCR == 0){
00041             //checks if its end of file, if first if did not work
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;}
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
00057         econio_gotoxy(30, 4+(2*index));
00058         printf("|%s%*.*s|%*.*s%d|", data->name, pad1Len, pad1Len,
      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("\n\nPress ENTER to start the game\n");
00066     econio_rawmode();
00067     while(1){
00068         int key = econio_getch();
00069         if(key == KEY_ENTER){break;}
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     char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
00081     fptr = fopen(paths[state], "r");
00082     if(fptr == NULL){perror("Scoresave: file opening error");}
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/l1.txt", "r+");
00089                 break;
00090             case 1:
00091                 fptr = fopen("../highscores/l1.txt", "r+");
00092                 break;
00093             case 2:
00094                 fptr = fopen("../highscores/l1.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
00107         //checks if highscores directory exist and openable
00108         DIR* dir;
00109         dir = opendir("../highscores");
00110         if(dir){closedir(dir);}
00111         //checks if directory do not exist
00112         else if(ENOENT == errno){
00113             closedir(dir);
00114             mkdir("../highscores", 0700);
00115             dir = opendir("../highscores");
```

```
00116             if(dir){closedir(dir);}
00117             //checks if directory still not exist
00118             else if(ENOENT == errno){
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);
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 
00143         FILE* n = fopen("../highscores/n.txt", "w");
00144         if(n == NULL){perror("Scoresave: file opening error, at line 80");}
00145         else{
00146             //input name
00147             printf("!Your name can only be 12 character long!");
00148             printf("Your name: ");
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!");
00152                 printf("Your correct name: ");
00153             }
00154             //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){
00163                     perror("Corrupted file");
00164                     break;
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
00168                 if(strcmp(name, p[index].name) == 0){
00169                     in = true;//is it in or not?
00170                     //check to update its score
00171                     if(score > p[index].score){
00172                         p[index].score = score;
00173                         if(index != 0){
00174                             //bubble sort
00175                             for(int i = 0; i <= index-1; i++){
00176                                 for(int j = i+1; j <= index; j++){
00177                                     if(p[j].score >= p[i].score){
00178                                         temp = p[i];
00179                                         p[i] = p[j];
00180                                         p[j] = temp;
00181                                     }
00182                                 }
00183                             }
00184                         }
00185                     }
00186                 }
00187                 //gets ready for next item, reallocating space
00188                 index++;
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++){
00196                     if(score >= p[i].score && !found){
00197                         i_new = i;
00198                         found = true;
```

```
00199                       break;
00200                   }
00201               }
00202           }
00203           //if nobody is in the list put him in
00204           if(index == 0)fprintf(n, "%s %d",name, score);
00205           //save list to file
00206           else {
00207               for(int i = 0; i < index; i++){
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);
00216           //check if any problem occured while freeing memory
00217           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]);
00221           rename("../highscores/n.txt",paths[state]);
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:

-.I. (- 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{
00015     char name[MAX_NAME + 1];
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  * but x wont change
00024  * x will compare its value if it is greater or equ or smaller or equ than
00025  * 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
```

```
00027  *  the array will grow, and with each shot that go out to space or hits an invader
00028  *  the array maybe shrink
00029  *
00030  */
00031
00032 typedef struct Bullets{
00033     int x;
00034     int y;
00035 }Bullets;
00036
00037 ///Invaders data structure
00038 /**
00039  *
00040  * linked list of invaders,
00041  * 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)
00045  *

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  * 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;
00059     int* y;
00060     struct Invaders* next;
00061 }Invaders;
00062
00063
00064 ///Player data structure
00065 /**
00066  *
00067  * will store it in a struct cause why not, easier data storage for me to implement
00068  * same way of thinking as with the Invaders, just dont need the full hitbox, as Invaders only kill
00069  * 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  *
00075  */
00076
00077 typedef struct Player{
00078     int* x_start;
00079     int shoot_pos_x;
00080     int shoot_pos_y;
00081     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 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)

  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_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()

```
void d_bullet (
            Bullets b,
            int white)
```

draws bullets

```
00181                                   {
00182     econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
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.

5.7.1.2 d_bullets()

```
void d_bullets (
            int sdb_b)
```

draws bullet count

```
00147                           {
00148     econio_textbackground(COL_BLACK);
00149     econio_gotoxy(35, 32);
00150     econio_textcolor(COL_WHITE);
00151     printf("BULLETS: 7/ %d",7-sdb_b);
00152     econio_textcolor(COL_BLACK);
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     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.

5.7.1.4 d_invader()

```
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

```
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

```
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(bcolor);
00125     printf("(_U_)");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
00126 }
```

5.7.1.6 d_score()

```
void d_score (
              int score)
```

draws score

```
00139                       {
00140     econio_textbackground(COL_BLACK);
00141     econio_gotoxy(20,32);
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                                       {
00157     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],
      mov->x_start[1], mov->y[1]);
00160         way==1?mov->x_start[0]++:mov->x_start[0]--;
00161         way==1?mov->x_start[1]++:mov->x_start[1]--;
00162         way==1?mov->x_end[0]++:mov->x_end[0]--;
00163         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->x_start[1], mov->y[1]);
00165         mov=mov->next;
00166     }
00167 }
```

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

5.7.1.8 mov_invy()

```
void mov_invy (
              Invaders * first)
```

move invaders vertically

```
00170                               {
00171     Invaders* mov = first;
00172     while(mov != NULL){
00173         d_invader(COL_BLACK, 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;
00176         d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
      mov->x_start[1], mov->y[1]);
00177         mov=mov->next;
00178     }
00179 }
```

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

5.7.1.9 peasy()

```
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

```
00069                                                       {
00070     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
00071     econio_textcolor(tcolor);printf("  __  __   ___ .   . ");
00072     econio_textbackground(b2color);
00073     econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__| [__   \\_/  ");
00074     econio_textbackground(b2color);
00075     econio_gotoxy(8*tabcount + 4, y+2);econio_textbackground(bcolor);printf(" |__ |  | ___]   |   ");
00076     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
00077     econio_textbackground(bcolor);printf("                    ");
00078     econio_textbackground(b2color);
00079 }
```

5.7.1.10 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 color

```
00095                                                        {
00096     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
00097     econio_textbackground(bcolor);
00098     econio_textcolor(tcolor);printf(" .  .  __  .--.  __  ");
00099     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+1);
00100     econio_textbackground(bcolor);printf(" |__| |__| |__/ |  \\ ");
00101     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+2);
00102     econio_textbackground(bcolor);printf(" |  | |  | |  \\ |__/ ");
00103     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
00104     econio_textbackground(bcolor);printf("                      ");
00105     econio_textbackground(b2color);
00106     econio_textcolor(COL_BLACK);
00107 }
```

5.7.1.11 pmed()

```
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

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

5.7.1.12 pquit()

```
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

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

5.7.1.13 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);
00014     printf("#######\\######\\  #####\\  ######\\#######\\    ##\\###\\   ##\\##\\
      ##\\#######\\######\\ #######\\#####\\  #######\\\n");
00015     econio_gotoxy(8*tabcounter, y+1);
00016     printf("##-----/##/--##\\##/--##\\##/----/##/----/    ##|####\\  ##|##|
      ##|##/--##|##/--##|##/----/##/-##\\ ##/----/\n");
00017     econio_gotoxy(8*tabcounter, y+2);
00018     printf("#######\\#######/#######|##|     #####\\      ##|##/##\\ ##|##|   ##|########|##|
      ##|#####\\  ######/ #######\\\n");
00019     econio_gotoxy(8*tabcounter, y+3);
00020     printf("\\----##|##/---/ ##/--##|##|     ##/--/      ##|##|\\##\\\##|\\##\\ ##/ ##/--##|##|
      ##|##/--/  ##/-##\\ \\----##|\n");
00021     econio_gotoxy(8*tabcounter, y+4);
00022     printf("#######|##|     ##|  ##|\\######\\#######\\    ##|##| \\####| \\####/  ##|  ##|######/
      ########\\##| ##| #######|\n");
00023     econio_gotoxy(8*tabcounter, y+5);
00024     printf("\\______/\\_/     \\_/  \\_/ \\_____/\\______/    \\_/\\_/  \\___/  \\___/  \\_/
      \\_/\\_____/ \\______/\\_/ \\_/ \\______/\n");
00025 }
```

5.7.1.14 pset()

```
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

```
00054                                                                    {
00055     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
00056     econio_textbackground(bcolor);
00057     econio_textcolor(tcolor);printf(" __ ___________     __ __");
00058     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+1);
00059     econio_textbackground(bcolor);printf("(_ |_  |  |  | |\\ |/__(_ ");
00060     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+2);
00061     econio_textbackground(bcolor);printf("__)|__ |  | _|_| \\|\\_|__)");
00062     econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
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                                                                    {
00029     econio_textbackground(b2color);econio_gotoxy(8*tabcount, y);econio_textbackground(bcolor);
00030     econio_textcolor(tcolor);printf(" __  ___  __  __  ___");
00031     econio_textbackground(b2color);
00032     econio_gotoxy(8*tabcount, y+1);econio_textbackground(bcolor);printf("(__   |  [__][__)  | ");
00033     econio_textbackground(b2color);
00034     econio_gotoxy(8*tabcount, y+2);econio_textbackground(bcolor);printf(".__)  |  |  ||  \\  | ");
00035     econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
00036     econio_textbackground(bcolor);printf("                    ");
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*8 x coord with tcolor 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*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*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*8 + 6 x coord with tcolor as text color and bcolor
      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*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*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*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 y coords and x coord with tcolor as text color and bcolor as background
      color
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

```
00209                   {
00210     econio_clrscr();
00211     //setup phase
00212     int score = 0; // the score which to export
00213     int b_db = 0;
00214     int sdb_b = 0;
00215     bool run = true;
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;
00220     int way = 1;
00221     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
00226     d_init(first, p);
00227     //game logic
00228     set_conio_terminal_mode();
00229     while(run){
00230         econio_rawmode();
00231         while(!kbhit()){
00232             //check if bullet has collision with invader
00233             if(b_db > 0)test = check_bullet(first,b,&score,b_db);
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             }
00244             else if(test == 2){
00245                 Invaders* mov = first;
00246                 first = first->next;
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                 free(b);
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){
00263                 for(int i = 0; i < b_db; i++){
00264                     if(b[i].y != 0){
00265                         d_bullet(b[i], 0);
00266                         b[i].y--;
00267                         d_bullet(b[i], 1);
00268                         econio_gotoxy(0, 30);
00269                         printf("%d",b[0].y);
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);
00279                     x_ref++;
00280                 }
00281                 else{
00282                     if(way){mov_invy(first);}
00283                     way = 0;
00284                     if(x_ref >= 7*row_length[mode]){
00285                         mov_invx(first, way);
00286                         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 
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();
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 
      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         }
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){
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_bullets(), 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.

```
00013         {
00014 #if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
00015     SMALL_RECT WinSize = {0, 0, 135, 35};
00016     SMALL_RECT* WinWin = &WinSize;
00017     SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00018     //plays in a loop music.wav file in windows, just do not have music.wav yet
00019     //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
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);
00025     //pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
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
00034                 int ch = econio_getch();
00035                 if(ch == KEY_UP && last != 0){before = last; last--;}
00036                 else if(ch == KEY_DOWN && last != 1){before = last; last++;}
00037                 else if(ch == KEY_ENTER) break;
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                         /*
00046                          * was a good idea, just not in c, devistated am i, said Yoda as
00047                          * 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                             //start not shiney, settings shiney
00053                             pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
00054                             pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00055                         }
00056                         else {
00057                             //quit not shiney, settings shiney
00058                             pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
00059                             pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
00060                         }
00061                         break;
00062                         */
00063                     case 1:
00064                         //settings not shiney, quit shiney
00065                         pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
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                 else {
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){
00091                         //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++;}
00095                         else if(ch == KEY_ENTER) break;
00096
00097                         //state machine goes brrrrrrrr
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){
00106                                     //easy was shiney, not now anymore, medium shiney
00107                                     peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
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:
00117                                 //medium not shiney, hard is shiney
```

```
00118                                   pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
00119                                   phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
00120                                   break;
00121                           }
00122                   }
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().

5.12 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)
00232     SMALL_RECT WinSize = {0, 0, 70, 40};
00233     SMALL_RECT* WinWin = &WinSize;
00234     SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
00235 #endif
00236
00237     //highscore board setup
00238     FILE* fptr;
00239     char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
00240     int testDB = 0;int db = 0;
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){
00246         econio_clrscr();
00247         printf("number of players to show from top 1: ");
00248         testDB = scanf("%*c%d%*c",&db);
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);
00258         mkdir("../highscores", 0700);
00259         dir2 = opendir("../highscores");
00260         if(dir2){closedir(dir2);}
00261         //directory does not exist
00262         else if(ENOENT == errno){
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{
00269             closedir(dir2);
00270             perror("Error occured regarding the highscores directory");
00271             exit(0);
00272         }
00273     }
00274     //other problem
00275     else {
00276         closedir(dir2);
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:
00286             fptr = fopen(paths[state], "r");
00287             if(fptr == NULL){
00288                 fptr = fopen(paths[state], "w");
00289                 if(fptr == NULL){
00290                     perror("File open error");
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");
00299             if(fptr == NULL){
00300                 fptr = fopen(paths[state], "w");
00301                 if(fptr == NULL){
00302                     perror("File open error");
00303                     exit(0);
00304                 }
00305             }
00306             printboard(fptr, db);
00307             break;
00308         case 2:
00309             //high diff score
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().

5.13.1.2 printboard()

```
void printboard (
            FILE * fptr,
            int 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

```
00020                                   {
00021     if(db == 0)return;
00022     //variables
00023     Data data[1] = {};
00024     int index = 0;
00025     const char* padding = "                                                  ";//soooo long, idk why
00026     char buff[5] = "";
00027     //drawing begining
00028     econio_clrscr();
00029     econio_gotoxy(30,1);
00030     printf("++++++++++++++++++");
00031     econio_gotoxy(30,2);
00032     printf("|NAME     | SCORE|");
00033     econio_gotoxy(30,3);
00034     printf("++++++++++++++++++");
00035
00036     int tName =0; int tSCR = 0;
00037     //magic is done here
00038     while(index < db){
00039         //checks if data was corrupted
00040         if((tName = fscanf(fptr,"%s", data->name)) == -1 || tName == 0||(tSCR = fscanf(fptr,
      "%d",&data->score)) == -1 || tSCR == 0){
00041             //checks if its end of file, if first if did not work
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;}
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
00057         econio_gotoxy(30, 4+(2*index));
00058         printf("|%s%*.*s|%*.*s%d|", data->name, pad1Len, pad1Len,
      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("\n\nPress ENTER to start the game\n");
00066     econio_rawmode();
00067     while(1){
00068         int key = econio_getch();
00069         if(key == KEY_ENTER){break;}
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
00079     FILE* fptr;
00080     char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
00081     fptr = fopen(paths[state], "r");
00082     if(fptr == NULL){perror("Scoresave: file opening error");}
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/l1.txt", "r+");
00089                 break;
00090             case 1:
00091                 fptr = fopen("../highscores/l1.txt", "r+");
00092                 break;
00093             case 2:
00094                 fptr = fopen("../highscores/l1.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 
00107         //checks if highscores directory exist and openable
00108         DIR* dir;
00109         dir = opendir("../highscores");
00110         if(dir){closedir(dir);}
00111         //checks if directory do not exist
00112         else if(ENOENT == errno){
00113             closedir(dir);
00114             mkdir("../highscores", 0700);
00115             dir = opendir("../highscores");
00116             if(dir){closedir(dir);}
00117             //checks if directory still not exist
00118             else if(ENOENT == errno){
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);
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 
00143         FILE* n = fopen("../highscores/n.txt", "w");
00144         if(n == NULL){perror("Scoresave: file opening error, at line 80");}
00145         else{
00146             //input name
00147             printf("!Your name can only be 12 character long!");
00148             printf("Your name: ");
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!");
00152                 printf("Your correct name: ");
00153             }
00154             //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){
00163                     perror("Corrupted file");
00164                     break;
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
00168                 if(strcmp(name, p[index].name) == 0){
00169                     in = true;//is it in or not?
00170                     //check to update its score
00171                     if(score > p[index].score){
00172                         p[index].score = score;
00173                         if(index != 0){
00174                             //bubble sort
00175                             for(int i = 0; i <= index-1; i++){
00176                                 for(int j = i+1; j <= index; j++){
00177                                     if(p[j].score >= p[i].score){
00178                                         temp = p[i];
00179                                         p[i] = p[j];
00180                                         p[j] = temp;
00181                                     }
00182                                 }
00183                             }
00184                         }
00185                     }
00186                 }
00187                 //gets ready for next item, reallocating space
00188                 index++;
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++){
00196                     if(score >= p[i].score && !found){
00197                         i_new = i;
00198                         found = true;
00199                         break;
00200                     }
00201                 }
00202             }
00203             //if nobody is in the list put him in
00204             if(index == 0)fprintf(n, "%s %d",name, score);
00205             //save list to file
00206             else {
00207                 for(int i = 0; i < index; i++){
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);
00216             //check if any problem occured while freeing memory
00217             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]);
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);
```

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

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

5.16 README.md File Reference

Index