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

```
{
    econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
    econio_gotoxy(b.x,b.y);
    white==1?econio_textcolor(COL_WHITE):econio_textcolor(COL_BLACK);
    econio_textbackground(COL_BLACK);
    printf("|");econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
}
```

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

5.1.1.2 d_bullets()

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

draws bullet count

```
{
    econio_textbackground(COL_BLACK);
    econio_gotoxy(35, 32);
    econio_textcolor(COL_WHITE);
    printf("BULLETS: 7/ %d",7-sdb_b);
    econio_textcolor(COL_BLACK);
}
```

5.1.1.3 d_init()

```
void d_init (
              Invaders * first,
              Player * p)
```

draws map first time

```
{
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
    mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
    d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
    econio_flush();
}
```

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

```
{
    econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("*(-_-)*");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("$._$_.$");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
}
```

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

```
{
    econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf(".I.");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("(_U_)");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
}
```

5.1.1.6 d_score()

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

draws score

```
{
    econio_textbackground(COL_BLACK);
    econio_gotoxy(20,32);
    econio_textcolor(COL_WHITE);
    printf("SCORE: %d",score);
    econio_textcolor(COL_BLACK);
}
```

5.1.1.7 mov_invx()

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

move invaders horizontally

```
                                    {
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        way==1?mov->x_start[0]++:mov->x_start[0]--;
        way==1?mov->x_start[1]++:mov->x_start[1]--;
        way==1?mov->x_end[0]++:mov->x_end[0]--;
        way==1?mov->x_end[1]++:mov->x_end[1]--;
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
}
```

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

```
                            {
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov->y[0]+=2;
        mov->y[1]+=2;
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
}
```

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

```
                                                                {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf("  __  __   ___ .   . ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__| [__   \\_/  ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount + 4, y+2);econio_textbackground(bcolor);printf(" |__ |  | ___]   |   ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
    econio_textbackground(bcolor);printf("                     ");
    econio_textbackground(b2color);
}
```

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

```
                                                            {
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
        econio_textbackground(bcolor);
        econio_textcolor(tcolor);printf(" .  .  __  .--.  __  ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+1);
        econio_textbackground(bcolor);printf(" |__| |__| |__/ |  \\ ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+2);
        econio_textbackground(bcolor);printf(" |  | |  | |  \\ |__/ ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
        econio_textbackground(bcolor);printf("                      ");
        econio_textbackground(b2color);
        econio_textcolor(COL_BLACK);
}
```

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

```
                                                           {
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y);
        econio_textbackground(bcolor);econio_textcolor(tcolor);printf(" .  .  ___  __  . .  . .  . ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+1);
        econio_textbackground(bcolor);printf(" |\\/| |___ |  \\ | |  | |\\/| ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+2);
        econio_textbackground(bcolor);printf(" |  | |___ |__/ | |__| |  | ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+3);
        econio_textbackground(bcolor);printf("                            ");
        econio_textbackground(b2color);
}
```

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

```
                                                            {
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y);
        econio_textbackground(bcolor);econio_textcolor(tcolor);printf(" __         ___ ___");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+1);
        econio_textbackground(bcolor);printf("|  | |  |  |   | ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+2);
        econio_textbackground(bcolor);printf("|__\\ |__| _|_  | ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+3);
        econio_textbackground(bcolor);printf("                  ");econio_textbackground(b2color);
        econio_textcolor(COL_BLACK);
}
```

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

```
                                                    {
        econio_textbackground(bcolor);
        econio_textcolor(tcolor);
        econio_gotoxy(8*tabcounter, y);
        printf("#######\\######\\  #####\\  ######\\\#######\\    ##\\###\\   ##\\##\\
    ##\\#######\\######\\ #######\\#####\\  #######\\\n");
        econio_gotoxy(8*tabcounter, y+1);
        printf("##-----/##/--##\\##/--##\\##/----/##/----/    ##|####\\  ##|##|
    ##|##/--##|##/--##|##/----/##/-##\\ ##/----/\n");
        econio_gotoxy(8*tabcounter, y+2);
        printf("#######\\#######/#######|##|     #####\\      ##|##/##\\ ##|##|   ##|#######|##|
    ##|#####\\  ######/ #######\\\n");
        econio_gotoxy(8*tabcounter, y+3);
        printf("\\----##|##/---/ ##/--##|##|     ##/--/      ##|##|\\##\\##|\\##\\ ##/ ##/--##|##|
    ##|##/--/  ##/-##\\ \\----##|\n");
        econio_gotoxy(8*tabcounter, y+4);
        printf("#######|##|     ##|  ##|\\######\\#######\\    ##|##| \\####| \\####/  ##|  ##|######/
    #######\\##| ##| #######|\n");
        econio_gotoxy(8*tabcounter, y+5);
        printf("\\______/\\_/     \\_/  \\_/ \\______/\\_______/    \\_/\\_/  \\___/  \\___/  \\_/
    \\_/\\______/ \\_______/\\_/ \\_/ \\_______/\n");
    }
```

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

```
                                                           {
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
        econio_textbackground(bcolor);
        econio_textcolor(tcolor);printf(" __ ____________    __ __");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+1);
        econio_textbackground(bcolor);printf("(_ |_  |  |  | |\\ |/__(_ ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+2);
        econio_textbackground(bcolor);printf("__)|__ |  | _|_| \\|\\_|__)");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
        econio_textbackground(bcolor);printf("                          ");
        econio_textbackground(b2color);
    }
```

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

```
                                                                     {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount, y);econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf(" __  ___  __  __  ___");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount, y+1);econio_textbackground(bcolor);printf("(__   |  [__][__)  | ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount, y+2);econio_textbackground(bcolor);printf(".__)  |  |  ||  \\  | ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
    econio_textbackground(bcolor);printf("                     ");
    econio_textbackground(b2color);
}
```

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.

```
                                                                {
    Invaders* mov = first;
    Invaders* mov2 = mov;
    while(mov != NULL){
        for(int i = 0; i < db; i++){
            if(mov->y[0] == b[i].y || mov->y[1] == b[i].y){
                if(b[i].x >= mov->x_start[0] && b[i].x <= mov->x_end[0]){
                    if(numb_inv(first) == 1){
                        *score += 5;
                        return 0;
                    }
                    if(mov == first){
                        d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
                        b[i].y = 0;
                        *score += 5;
                        return 2;
                    }
                    else if(mov->next == NULL){
                        *score += 5;
                        d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
                        mov2->next=NULL;
                        free(mov);
                        return 1;
                    }
                    else{
                        *score += 5;
                        d_invader(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,mov->x_start[0],mov->y[0],mov->x_start[1],mov->y[1]);
                        Invaders* mov3 = mov;
                        mov=mov->next;
                        mov2->next = mov;
                        free(mov3);
                        mov=mov2;
                        b[i].y = 0;
                    }
                }
            }
        }
        mov2=mov;
        mov=mov->next;
    }
    return 1;
}
```

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

```
                              {
    ///will get current linked list member, free its sub arrays, got to next member and free previous
member
    while(i != NULL){
        Invaders* buff = i;
        free(i->x_start);
        free(i->x_end);
        free(i->y);
        i = i->next;
        free(buff);
    }
}
```

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

```
                    {
    econio_clrscr();
    //setup phase
    int score = 0; // the score which to export
    int b_db = 0;
    int sdb_b = 0;
    bool run = true;
    Player* p = Setup_Player();//strange, but it is what it is
    Invaders* first = Setup_Inv(mode);
    int row_length[3] = {6,7,8};
    int cycle = 0;
    int way = 1;
    int test = 1;
    int x_ref = first->x_start[0];
    Bullets* b =(Bullets*)malloc(50*sizeof(Bullets)); //setup for future

    //first draw
    d_init(first, p);
    //game logic
    set_conio_terminal_mode();
    while(run){
        econio_rawmode();
        while(!kbhit()){
            //check if bullet has collision with invader
            if(b_db > 0)test = check_bullet(first,b,&score,b_db);
            if(test == 0){
                econio_normalmode();
                reset_terminal_mode();
                //before exit free the bullets :)
                free(b);
                free(p);
                freeInvaders(first);//yeah free the slaves!!!
                debugmalloc_dump();
                return score;
            }
            else if(test == 2){
                Invaders* mov = first;
                first = first->next;
                free(mov);
            }
            //checks if invader is in line with player
            if(first->y[0] == 28 || first->y[0] == 29){
                econio_normalmode();
                reset_terminal_mode();
                //before exit free the bullets :)
                free(b);
                free(p);
                freeInvaders(first);//yeah free the slaves!!!
                debugmalloc_dump();
                return score;
            }
            //check collisions
            //bullets draw and such
            if(cycle%3==0){
                for(int i = 0; i < b_db; i++){
                    if(b[i].y != 0){
                        d_bullet(b[i], 0);
                        b[i].y--;
                        d_bullet(b[i], 1);
                        econio_gotoxy(0, 30);
                        printf("%d",b[0].y);
                        econio_flush();
                    }
                    else {d_bullet(b[i], 0);}
                }
            }
            if(cycle%(1+numb_inv(first)) == 0 || cycle%20 == 0){
                //move invaders
                if(x_ref < 70 && way){
                    mov_invx(first, way);
                    x_ref++;
                }
                else{
                    if(way){mov_invy(first);}
                    way = 0;
                    if(x_ref >= 7*row_length[mode]){
                        mov_invx(first, way);
                        x_ref--;
                    }
                    else{
                        way = 1;
                        mov_invy(first);
                    }
                }
            }
            if(cycle%100 == 0){sdb_b = 0;}
            d_score(score);
            d_bullets(sdb_b);

            usleep(23000);
            cycle++;
            if(cycle == 101)cycle = 0;
        }
        econio_normalmode();
        //did it with switch too, but who cares
        //check keyboard press
        int key = getch();
        //move player to left
        if(key== 'a'){
            if(p->x_start[1] > 0){
    d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
                p->x_start[0]-=2;
                p->shoot_pos_x-=2;
                p->x_start[1]-=2;
    d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
            }
        }
            //move player to right
        else if(key == 'd'){
            if(p->x_start[1] < 70){
    d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
                p->x_start[0]+=2;
                p->shoot_pos_x+=2;
                p->x_start[1]+=2;
    d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
            }
        }
            //shoot bullet
        else if(key == KEY_SPACE && sdb_b < 7){
            sdb_b++;
            b[b_db].x=p->shoot_pos_x;
            b[b_db].y=p->shoot_pos_y;
            b_db++;
            if(b_db%49 == 0)b =(Bullets*)realloc(b,(b_db + 51)*sizeof(Bullets));
            d_bullet(b[b_db-1], 1);
        }
    }
    debugmalloc_dump();
    return score;
}
```

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

```
{
    int r;
    unsigned char c;
    if ((r = read(0, &c, sizeof(c))) < 0) {
        return r;
    } else {
        return c;
    }
}
```

5.2.1.5 kbhit()

```
int kbhit ()
```

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

```
{
    struct timeval tv = { 0L, 0L };
    fd_set fds;
    FD_ZERO(&fds);
    FD_SET(0, &fds);
    return select(1, &fds, NULL, NULL, &tv) > 0;
}
```

5.2.1.6 numb_inv()

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

counts how many invaders are there

```
                            {
    Invaders* mov = first;
    int count = 0;
    while(mov!=NULL){
        count++;
        mov=mov->next;
    }
    return count;
}
```

References Invaders::next.

5.2.1.7 reset_terminal_mode()

```
void reset_terminal_mode ()
```

resets terminal mode

```
{
    tcsetattr(0, TCSANOW, &orig_termios);
}
```

References orig_termios.

5.2.1.8 set_conio_terminal_mode()

```
void set_conio_terminal_mode ()
```

set terminal mode

```
{
    struct termios new_termios;

    // take two copies - one for now, one for later
    tcgetattr(0, &orig_termios);
    memcpy(&new_termios, &orig_termios, sizeof(new_termios));

    // register cleanup handler, and set the new terminal mode
    atexit(reset_terminal_mode);
    cfmakeraw(&new_termios);
    tcsetattr(0, TCSANOW, &new_termios);
}
```

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.

```
                                 {
    //0 easy, 1 is medium and 2 is hard
    //in easy half of the rows invaders will be, and it will move in medium speed
    //in medium there will be six in a row and it will move at medium speed
    //in hard there vill be more rows: not 7 but 11, they will move much faster
    int S_Width = 6;//number of white characters between 2 invader
    int row_length[3] = {6,7,8};
    int numb_row[3] = {4,5,7};
    //linked list setup
    Invaders* list = NULL;
    for(int j = 0; j < numb_row[mode]; j++){
        for(int i = 0; i < row_length[mode]; i++){
            //create one element
            Invaders* mov;
            mov =(Invaders*)malloc(sizeof(Invaders));
            mov->next = list;//connect to previous
            //allocate space to the arrays
            mov->x_start = (int*)malloc(2*sizeof(int));
            mov->x_end = (int*)malloc(2*sizeof(int));
            mov->y = (int*)malloc(2*sizeof(int));
            //first element of rows and the very first element restarts the x coords
            if(mov->next == NULL || i == 0){
                //starting coord
                mov->x_start[0] = 0;
                mov->x_start[1] = 0;
                //end of hitbox coords
                mov->x_end[0] = (mov->x_start[0]) + S_Width;
                mov->x_end[1] = (mov->x_start[1]) + S_Width;
            }
            else{
                //same just if its not the first element
                mov->x_start[0] = (list->x_end[0]) +2;
                mov->x_start[1] = (list->x_end[1]) +2;
                mov->x_end[0] = (mov->x_start[0]) + S_Width;
                mov->x_end[1] = (mov->x_start[1]) + S_Width;
            }
            //set y coords
            mov->y[0] = 3*j+1;
            mov->y[1] = 3*j+2;
            //next element
            list = mov;
        }
    }
    return list;
}
```

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

5.2.1.10 Setup_Player()

```
Player * Setup_Player ()
```

sets up the Player data structure

```
                           {
    Player* p = (Player*)malloc(sizeof(Player));

    p->x_start=(int*)malloc(2*sizeof(int));
    p->x_start[0] = 1;
    p->x_start[1] = 0;

    //where the bullet will start from
    p->shoot_pos_x = 2;
    p->shoot_pos_y = 27;

    //playing field will be 120*40, 120 width, 40 height
    p->y=(int*)malloc(2*sizeof(int));
    p->y[0] = 29;
    p->y[1] = 30;

    return p;
}
```

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.

```
          {
#if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
    SMALL_RECT WinSize = {0, 0, 135, 35};
    SMALL_RECT* WinWin = &WinSize;
    SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
    //plays in a loop music.wav file in windows, just do not have music.wav yet
    //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
#endif
    econio_clrscr();
    //draws menu items
    printSI(1,3,COL_WHITE, COL_BLACK);
    pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
    //pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
    pquit(6, 16, COL_WHITE, COL_BLACK, COL_WHITE);

    int last = 0;
    int before = 0;
   while(true){
        econio_rawmode();
        while(true){
                //reads key and do state magic, save previous state and update current state
                int ch = econio_getch();
                if(ch == KEY_UP && last != 0){before = last; last--;}
                else if(ch == KEY_DOWN && last != 1){before = last; last++;}
                else if(ch == KEY_ENTER) break;
                //draws new state of menu items or state machine
                switch(last){
                    case 0:
                        //start text is shiney, settings not shiney anymore
                        pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
                        pquit(6,16,COL_WHITE, COL_BLACK, COL_WHITE);
                        break;
                        /*
                         * was a good idea, just not in c, devistated am i, said Yoda as
                         * possible upgrade if client wants to get a new feture in like a settings button in production
                         * gonna leave it here as nobody gonna touch it and there is space for it
                         *
                    case 1:
                        if(before == 0){
                            //start not shiney, settings shiney
                            pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
                            pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
                        }
                        else {
                            //quit not shiney, settings shiney
                            pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
                            pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
                        }
                        break;
                        */
                    case 1:
                        //settings not shiney, quit shiney
                        pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
                        pquit(6, 16, COL_BLACK, COL_WHITE, COL_BLACK);
                        break;
                }
            }
                econio_normalmode();
                //choice is quit, do exit with screen to default
                if(last == 1){
                    econio_textbackground(COL_BLACK);
                    econio_clrscr();
                    econio_textcolor(COL_WHITE);
                    exit(0);
                }
                else {
                    //choice is start, do clear screen, draw dificulty elements
                    econio_textbackground(COL_BLACK);
                    econio_clrscr();
                    econio_textcolor(COL_WHITE);
                    peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
                    pmed(6, 13, COL_WHITE, COL_BLACK, COL_WHITE);
```

```
                        phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);

                        last = 0;

                        econio_rawmode();
                        while(true){
                            //read keys and save previous state, and update now state
                            int ch = econio_getch();
                            if(ch == KEY_UP && last != 0){before = last; last--;}
                            else if(ch == KEY_DOWN && last != 2){before = last; last++;}
                            else if(ch == KEY_ENTER) break;

                            //state machine goes brrrrrrrr
                            switch(last){
                                case 0:
                                    //easy text is shiney, medium is not
                                    peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
                                    pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
                                    break;
                                case 1:
                                    if(before == 0){
                                        //easy was shiney, not now anymore, medium shiney
                                        peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
                                        pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
                                    }
                                    else {
                                        //hard not shiney anymore, meium is shiney
                                        pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
                                        phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
                                    }
                                    break;
                                case 2:
                                    //medium not shiney, hard is shiney
                                    pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
                                    phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
                                    break;
                            }
                    }

                        econio_normalmode();
                    //before exit to who knows where, clean screen
                        econio_textbackground(COL_BLACK);
                    econio_clrscr();
                    econio_textcolor(COL_WHITE);

                   return last;
               }
        }
}
```

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

```
                            {
    //set window size for good windows users :), but i like linux better, just not the terminal
modifying
#if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
    SMALL_RECT WinSize = {0, 0, 70, 40};
    SMALL_RECT* WinWin = &WinSize;
    SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
#endif

    //highscore board setup
    FILE* fptr;
    char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
    int testDB = 0;int db = 0;
    printf("Give how many scores to show from the top, 0 to not show anything");
    printf("\nor negative number as it will be converted to a positive number\n");
    printf("number of players to show from top 1: ");
    testDB = scanf("%d",&db);
    while(testDB < 1 || db < 0 || db > 1000000){
        econio_clrscr();
        printf("number of players to show from top 1: ");
        testDB = scanf("%*c%d%*c",&db);
    }

    //checks if highscores directory exist and openable
    DIR* dir2;
    dir2 = opendir("../highscores");
    if(dir2){closedir(dir2);}
    //directory does not exist
    else if(ENOENT == errno){
        closedir(dir2);
        mkdir("../highscores", 0700);
        dir2 = opendir("../highscores");
        if(dir2){closedir(dir2);}
        //directory does not exist
        else if(ENOENT == errno){
            closedir(dir2);
            perror("Could not create highscores directory, please create it manually to the same
directory as main.c");
            exit(0);
        }
        //other problem
        else{
            closedir(dir2);
            perror("Error occured regarding the highscores directory");
            exit(0);
        }
    }
    //other problem
    else {
        closedir(dir2);
        perror("Error occured regarding the highscores directory");
        exit(0);
    }
    //state machine, would be better to put paths into a 2d char array, but who cares
    //its not like anybody gonna expand this sht
    switch(state){
        //show relevant highscore
        //dont wanna make a function to read file and store the data
        case 0:
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
        case 1:
            //show highscore in medium diff
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
        case 2:
            //high diff score
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
    }
}
```

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

```
                                   {
    if(db == 0)return;
    //variables
    Data data[1] = {};
    int index = 0;
    const char* padding = "                                         ";//soooo long, idk why
    char buff[5] = "";
    //drawing begining
    econio_clrscr();
    econio_gotoxy(30,1);
    printf("++++++++++++++++++");
    econio_gotoxy(30,2);
    printf("|NAME     | SCORE|");
    econio_gotoxy(30,3);
    printf("++++++++++++++++++");

    int tName =0; int tSCR = 0;
    //magic is done here
    while(index < db){
        //checks if data was corrupted
        if((tName = fscanf(fptr,"%s", data->name)) == -1 || tName == 0||(tSCR = fscanf(fptr,
      "%d",&data->score)) == -1 || tSCR == 0){
            //checks if its end of file, if first if did not work
            if((tSCR = fscanf(fptr,"%d",&data->score))== -1){break;}
            econio_clrscr();
            printf("\nCorrupted file\n");
            exit(0);
        }
        //checks if earlier break is needed
        if((tName == EOF && tSCR == EOF) || (tName == 0 && tSCR == 0)){break;}

        sprintf(buff, "%d", data->score);//transform int to string aka char array

        //get padding
        int pad1Len = (TARGETLEN*3/5) - strlen(data->name);
        int pad2Len =(TARGETLEN*2/5)- strlen(buff);

        //draw board
        econio_gotoxy(30, 4+(2*index));
        printf("|%s%*.*s|%*.*s%d|", data->name, pad1Len, pad1Len,
    padding,pad2Len,pad2Len,padding,data->score);
        econio_gotoxy(30, 4+(2*index+1));
        printf("+++++++++++++++++");

        index++;//magic number
    }
    printf("\n");
    printf("\n\nPress ENTER to start the game\n");
    econio_rawmode();
    while(1){
        int key = econio_getch();
        if(key == KEY_ENTER){break;}
    }
    econio_normalmode();
}
```

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

```
                                            {
    econio_clrscr();
    econio_textcolor(COL_WHITE);
    //set file path with state
    FILE* fptr;
    char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
    fptr = fopen(paths[state], "r");
    if(fptr == NULL){perror("Scoresave: file opening error");}
    else{
        /*
         * for legacy code to just exist here or to return to something in case of shit
        switch(state){
            case 0:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
            case 1:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
            case 2:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
        }
        */

        //set up variables to work with
        char name[MAX_NAME + 2] = {};
        int index = 0;
        Data *p = calloc(3, sizeof(*p));//for safety and to pass new data allocate 3 times the space
        Data temp;
        bool in = false;
        int i_new;

        //checks if highscores directory exist and openable
        DIR* dir;
        dir = opendir("../highscores");
        if(dir){closedir(dir);}
        //checks if directory do not exist
        else if(ENOENT == errno){
            closedir(dir);
            mkdir("../highscores", 0700);
            dir = opendir("../highscores");
            if(dir){closedir(dir);}
            //checks if directory still not exist
            else if(ENOENT == errno){
                closedir(dir);
                free(p);
                debugmalloc_dump();
                perror("Could not create highscores directory, please create it manually to the same directory as main.c");
                exit(0);
            }
            //other problem
            else{
                closedir(dir);
                free(p);
                debugmalloc_dump();
                perror("Error occured regarding the highscores directory");
                exit(0);
            }
        }
        //other problem
        else {
            closedir(dir);
            free(p);
            debugmalloc_dump();
            perror("Error occured regarding the highscores directory");
            exit(0);
        }

        FILE* n = fopen("../highscores/n.txt", "w");
        if(n == NULL){perror("Scoresave: file opening error, at line 80");}
        else{
            //input name
            printf("!Your name can only be 12 character long!");
            printf("Your name: ");
            int testName = 0;
            while((testName = scanf("%s",name)) == 0 || testName == -1 || strlen(name) > 12){
                printf("!Your name can only be 12 character long!");
                printf("Your correct name: ");
            }
            //load to struct the saved data
            //yeah nested loops, its not good, but it is what it is, who else gonna touch this code except me anyway?
            //while(fscanf(fptr,"%s %d",p[index].name, &p[index].score) != EOF){


            int testStr = 0; int testINT = 0;
            while((testStr = fscanf(fptr,"%s",p[index].name)) != EOF && (testINT = fscanf(fptr,"%d", &p[index].score)) != EOF){
                //tests if readed data was corrupted
                if(testStr == 0 || testStr == -1 || testINT == 0 || testINT == -1){
                    perror("Corrupted file");
                    break;
                }
                /// if name is in list and current score is higher than old, change it, than sort current arrays to descending based on scores
                //check if name is in the list
                if(strcmp(name, p[index].name) == 0){
                    in = true;//is it in or not?
                    //check to update its score
                    if(score > p[index].score){
                        p[index].score = score;
                        if(index != 0){
                            //bubble sort
                            for(int i = 0; i <= index-1; i++){
                                for(int j = i+1; j <= index; j++){
                                    if(p[j].score >= p[i].score){
                                        temp = p[i];
                                        p[i] = p[j];
                                        p[j] = temp;
                                    }
                                }
                            }
                        }
                    }
                }
                //gets ready for next item, reallocating space
                index++;
                p = (Data*)realloc(p, (index+1)*sizeof(*p));
            }

            //if not in highscorer list then add it to the top of its score group
            bool found = false;
            if(!in){
                for(int i = 0; i < index; i++){
                    if(score >= p[i].score && !found){
                        i_new = i;
                        found = true;
                        break;
                    }
                }
            }
            //if nobody is in the list put him in
            if(index == 0)fprintf(n, "%s %d",name, score);
            //save list to file
            else {
                for(int i = 0; i < index; i++){
                    if(i == i_new) fprintf(n, "%s %d\n",name, score);
                    fprintf(n, "%s %d\n",p[i].name, p[i].score);
                }
            }
            //free buff and close file stream
            free(p);
            fclose(n);
            fclose(fptr);
            //check if any problem occured while freeing memory
            debugmalloc_dump();
            //delete old, rename new save file
            //yeah there are better ways to do it, but i dont want to touch this at this point
            remove(paths[state]);
            rename("../highscores/n.txt",paths[state]);
        }
    }
}
```

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.

```
#ifndef data_struct_h
#define data_struct_h

//definition of space key
#define KEY_SPACE 32
///max length of name
#define MAX_NAME 12
///max length of collectable points
#define MAX_POINT_LENGTH 4
///target length of a board piece without the "|" tags(there are 3 of these tags)
#define TARGETLEN 15

///data structure of highscore system
typedef struct datas{
    char name[MAX_NAME + 1];
    int score;
}Data;

///@brief 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
 *
 */

typedef struct Bullets{
    int x;
    int y;
}Bullets;

///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
 *
 */


typedef struct Invaders{
    int* x_start;
    int* x_end;
    int* y;
    struct Invaders* next;
}Invaders;


///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_)
 *
 */

typedef struct Player{
    int* x_start;
    int shoot_pos_x;
    int shoot_pos_y;
    int* y;
}Player;

#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

```
                                   {
    econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
    econio_gotoxy(b.x,b.y);
    white==1?econio_textcolor(COL_WHITE):econio_textcolor(COL_BLACK);
    econio_textbackground(COL_BLACK);
    printf("|");econio_textcolor(COL_BLACK);econio_textbackground(COL_BLACK);
}
```

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

5.7.1.2 d_bullets()

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

draws bullet count

```
                          {
    econio_textbackground(COL_BLACK);
    econio_gotoxy(35, 32);
    econio_textcolor(COL_WHITE);
    printf("BULLETS: 7/ %d",7-sdb_b);
    econio_textcolor(COL_BLACK);
}
```

5.7.1.3 d_init()

```
void d_init (
            Invaders * first,
            Player * p)
```

draws map first time

```
                                         {
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0], mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
    d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
    econio_flush();
}
```

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

```
                                                                                          {
    econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("*(-_-)*");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("$._$_.$");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
}
```

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

```
                                                                                         {
    econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x,y);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf(".I.");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
    econio_gotoxy(x2,y2);econio_textcolor(tcolor);econio_textbackground(bcolor);
    printf("(_U_)");econio_textcolor(tcolor2);econio_textbackground(bcolor2);
}
```

5.7.1.6 d_score()

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

draws score

```
                        {
    econio_textbackground(COL_BLACK);
    econio_gotoxy(20,32);
    econio_textcolor(COL_WHITE);
    printf("SCORE: %d",score);
    econio_textcolor(COL_BLACK);
}
```

5.7.1.7 mov_invx()

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

move invaders horizontally

```
                                        {
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        way==1?mov->x_start[0]++:mov->x_start[0]--;
        way==1?mov->x_start[1]++:mov->x_start[1]--;
        way==1?mov->x_end[0]++:mov->x_end[0]--;
        way==1?mov->x_end[1]++:mov->x_end[1]--;
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
}
```

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

```
                                    {
    Invaders* mov = first;
    while(mov != NULL){
        d_invader(COL_BLACK, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov->y[0]+=2;
        mov->y[1]+=2;
        d_invader(COL_WHITE, COL_BLACK, COL_BLACK, COL_BLACK, mov->x_start[0], mov->y[0],
mov->x_start[1], mov->y[1]);
        mov=mov->next;
    }
}
```

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

```
                                                                {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf("  __  __   ___ .   . ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount + 4, y+1);econio_textbackground(bcolor);printf(" |__ |__| [__   \\_/  ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount + 4, y+2);econio_textbackground(bcolor);printf(" |__ |  | ___]   |   ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
    econio_textbackground(bcolor);printf("                     ");
    econio_textbackground(b2color);
}
```

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

```
                                                                 {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y);
    econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf(" .   .  __  .--.  __  ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+1);
    econio_textbackground(bcolor);printf(" |__| |__| |__/ |  \\ ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+2);
    econio_textbackground(bcolor);printf(" |  | |  | |  \\ |__/ ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 4, y+3);
    econio_textbackground(bcolor);printf("                       ");
    econio_textbackground(b2color);
    econio_textcolor(COL_BLACK);
}
```

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

```
                                                                {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y);
    econio_textbackground(bcolor);econio_textcolor(tcolor);printf(" .  .  ___  __  . .  . .  . ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+1);
    econio_textbackground(bcolor);printf(" |\\/| |___ |  \\ | |  | |\\/| ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+2);
    econio_textbackground(bcolor);printf(" |  | |___ |__/ | |__| |  | ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 1, y+3);
    econio_textbackground(bcolor);printf("                            ");
    econio_textbackground(b2color);
}
```

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

```
                                                                            {
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y);
        econio_textbackground(bcolor);econio_textcolor(tcolor);printf(" __      ___ ___");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+1);
        econio_textbackground(bcolor);printf("|  | |  |  |   | ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+2);
        econio_textbackground(bcolor);printf("|__\\ |__| _|_  | ");
        econio_textbackground(b2color);econio_gotoxy(8*tabcount + 2, y+3);
        econio_textbackground(bcolor);printf("                 ");econio_textbackground(b2color);
        econio_textcolor(COL_BLACK);
}
```

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

```
                                                                   {
        econio_textbackground(bcolor);
        econio_textcolor(tcolor);
        econio_gotoxy(8*tabcounter, y);
        printf("#######\\######\\  #####\\  ######\\#######\\    ##\\###\\   ##\\##\\
     ##\\#######\\######\\ #######\\#####\\  #######\\\n");
        econio_gotoxy(8*tabcounter, y+1);
        printf("##-----/##/--##\\##/--##\\##/----/##/----/    ##|####\\  ##|##|
     ##|##/--##|##/--##|##/----/##/-##\\ ##/----/\n");
        econio_gotoxy(8*tabcounter, y+2);
        printf("#######\\#######/#######|##|     #####\\      ##|##/##\\ ##|##|   ##|#######|##|
     ##|#####\\  ######/ #######\\\n");
        econio_gotoxy(8*tabcounter, y+3);
        printf("\\----##|##/---/ ##/--##|##|     ##/--/      ##|##|\\##\\##|\\##\\ ##/ ##/--##|##|
     ##|##/--/  ##/-##\\ \\----##|\n");
        econio_gotoxy(8*tabcounter, y+4);
        printf("#######|##|     ##|  ##|\\######\\#######\\    ##|##| \\####| \\####/  ##|  ##|######/
     #######\\##| ##| #######|\n");
        econio_gotoxy(8*tabcounter, y+5);
        printf("\\______/\\_/     \\_/  \\_/ \\_____/\\______/    \\_/\\_/  \\___/  \\___/  \\_/
     \\_/\\_____/ \\______/\\_/ \\_/ \\______/\n");
}
```

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

```
                                                               {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y);
    econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf(" __ ___________     __ __");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+1);
    econio_textbackground(bcolor);printf("(_ |_  |  |  | |\\ |/__(_ ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+2);
    econio_textbackground(bcolor);printf("__)|__ |  | _|_| \\|\\_|__)");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount + 6, y+3);
    econio_textbackground(bcolor);printf("                          ");
    econio_textbackground(b2color);
}
```

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

```
                                                                {
    econio_textbackground(b2color);econio_gotoxy(8*tabcount, y);econio_textbackground(bcolor);
    econio_textcolor(tcolor);printf(" __  ___  __  __  ___");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount, y+1);econio_textbackground(bcolor);printf("(__   |  [__][__)  | ");
    econio_textbackground(b2color);
    econio_gotoxy(8*tabcount, y+2);econio_textbackground(bcolor);printf(".__)  |  |  ||  \\  | ");
    econio_textbackground(b2color);econio_gotoxy(8*tabcount, y+3);
    econio_textbackground(bcolor);printf("                     ");
    econio_textbackground(b2color);
}
```

5.8 draw.h

Go to the documentation of this file.

```
//for menu

///draws SPACE INVADERS text starting y coords and tabcounts*8 x coord with tcolor as text color and
bcolor as background color
void printSI(int tabcounter, int y, int tcolor, int bcolor);
///draws START 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 settings text starting y coords and tabcounts*8 + 2 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 quit text starting y coords and tabcounts*8 + 6 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 easy text starting y coords and tabcounts*8 + 4 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 medium text starting y coords and tabcounts*8 + 1 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 hard text starting y coords and tabcounts*8 + 4 x coord with tcolor as text color and bcolor
as background color
void phard(int tabcount, int y, int tcolor, int bcolor, int b2color);


//______________________________________________________________________________________________________________
//for game

///draws invader starting y coords and 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 player 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 score
void d_score(int score);
///Invaders data struct
typedef struct Invaders Invaders;
///Player data struct
typedef struct Player Player;
///Bullets data struct
typedef struct Bullets Bullets;
///draws map first time
void d_init(Invaders* first, Player* p);
///draws bullets
void d_bullet(Bullets b, int white);
///move invaders horizontally
void mov_invx(Invaders* first, int way);
///move invaders vertically
void mov_invy(Invaders* first);
///draws bullet count
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

```
                    {
    econio_clrscr();
    //setup phase
    int score = 0; // the score which to export
    int b_db = 0;
    int sdb_b = 0;
    bool run = true;
    Player* p = Setup_Player();//strange, but it is what it is
    Invaders* first = Setup_Inv(mode);
    int row_length[3] = {6,7,8};
    int cycle = 0;
    int way = 1;
    int test = 1;
    int x_ref = first->x_start[0];
    Bullets* b =(Bullets*)malloc(50*sizeof(Bullets)); //setup for future

    //first draw
    d_init(first, p);
    //game logic
    set_conio_terminal_mode();
    while(run){
        econio_rawmode();
        while(!kbhit()){
            //check if bullet has collision with invader
            if(b_db > 0)test = check_bullet(first,b,&score,b_db);
            if(test == 0){
                econio_normalmode();
                reset_terminal_mode();
                //before exit free the bullets :)
                free(b);
                free(p);
                freeInvaders(first);//yeah free the slaves!!!
                debugmalloc_dump();
                return score;
            }
            else if(test == 2){
                Invaders* mov = first;
                first = first->next;
                free(mov);
            }
            //checks if invader is in line with player
            if(first->y[0] == 28 || first->y[0] == 29){
                econio_normalmode();
                reset_terminal_mode();
                //before exit free the bullets :)
                free(b);
                free(p);
                freeInvaders(first);//yeah free the slaves!!!
                debugmalloc_dump();
                return score;
            }
            //check collisions
            //bullets draw and such
            if(cycle%3==0){
                for(int i = 0; i < b_db; i++){
                    if(b[i].y != 0){
                        d_bullet(b[i], 0);
                        b[i].y--;
                        d_bullet(b[i], 1);
                        econio_gotoxy(0, 30);
                        printf("%d",b[0].y);
                        econio_flush();
                    }
                    else {d_bullet(b[i], 0);}
                }
            }
            if(cycle%(1+numb_inv(first)) == 0 || cycle%20 == 0){
                //move invaders
                if(x_ref < 70 && way){
                    mov_invx(first, way);
                    x_ref++;
                }
                else{
                    if(way){mov_invy(first);}
                    way = 0;
                    if(x_ref >= 7*row_length[mode]){
                        mov_invx(first, way);
                        x_ref--;
                    }
                    else{
                        way = 1;
                        mov_invy(first);
                    }
                }
            }
            if(cycle%100 == 0){sdb_b = 0;}
            d_score(score);
            d_bullets(sdb_b);

            usleep(23000);
            cycle++;
            if(cycle == 101)cycle = 0;
        }
        econio_normalmode();
        //did it with switch too, but who cares
        //check keyboard press
        int key = getch();
        //move player to left
        if(key== 'a'){
            if(p->x_start[1] > 0){
d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
                p->x_start[0]-=2;
                p->shoot_pos_x-=2;
                p->x_start[1]-=2;
d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
            }
        }
            //move player to right
        else if(key == 'd'){
            if(p->x_start[1] < 70){
d_player(COL_BLACK,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
                p->x_start[0]+=2;
                p->shoot_pos_x+=2;
                p->x_start[1]+=2;
d_player(COL_WHITE,COL_BLACK,COL_BLACK,COL_BLACK,p->x_start[0],p->y[0],p->x_start[1],p->y[1]);
```

```
                }
            }
                //shoot bullet
            else if(key == KEY_SPACE && sdb_b < 7){
                sdb_b++;
                b[b_db].x=p->shoot_pos_x;
                b[b_db].y=p->shoot_pos_y;
                b_db++;
                if(b_db%49 == 0)b =(Bullets*)realloc(b,(b_db + 51)*sizeof(Bullets));
                d_bullet(b[b_db-1], 1);
            }
        }
        debugmalloc_dump();
        return score;
}
```

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.

```
///game logic
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.

```
            {
#if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
    SMALL_RECT WinSize = {0, 0, 135, 35};
    SMALL_RECT* WinWin = &WinSize;
    SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
    //plays in a loop music.wav file in windows, just do not have music.wav yet
    //PlaySound(TEXT("../sound/music.wav"),NULL, SND_FILENAME | SND_ASYNC | SND_LOOP);
#endif
    econio_clrscr();
    //draws menu items
    printSI(1,3,COL_WHITE, COL_BLACK);
    pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
    //pset(5,16,COL_WHITE, COL_BLACK, COL_WHITE); was good idea
    pquit(6, 16, COL_WHITE, COL_BLACK, COL_WHITE);

    int last = 0;
    int before = 0;
   while(true){
        econio_rawmode();
        while(true){
                //reads key and do state magic, save previous state and update current state
                int ch = econio_getch();
                if(ch == KEY_UP && last != 0){before = last; last--;}
                else if(ch == KEY_DOWN && last != 1){before = last; last++;}
                else if(ch == KEY_ENTER) break;
                //draws new state of menu items or state machine
                switch(last){
                    case 0:
                        //start text is shiney, settings not shiney anymore
                        pstart(6, 11, COL_BLACK, COL_WHITE, COL_BLACK);
                        pquit(6,16,COL_WHITE, COL_BLACK, COL_WHITE);
                        break;
                        /*
                         * was a good idea, just not in c, devistated am i, said Yoda as
                         * possible upgrade if client wants to get a new feture in like a settings button in production
                         * gonna leave it here as nobody gonna touch it and there is space for it
                         *
                    case 1:
                        if(before == 0){
                            //start not shiney, settings shiney
                            pstart(6, 11, COL_WHITE, COL_BLACK, COL_WHITE);
                            pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
                        }
                        else {
                            //quit not shiney, settings shiney
                            pset(5,16,COL_BLACK, COL_WHITE, COL_BLACK);
                            pquit(6, 21, COL_WHITE, COL_BLACK, COL_WHITE);
                        }
                        break;
                        */
                    case 1:
                        //settings not shiney, quit shiney
                        pstart(6,11,COL_WHITE, COL_BLACK, COL_WHITE);
                        pquit(6, 16, COL_BLACK, COL_WHITE, COL_BLACK);
                        break;
                }
            }
                econio_normalmode();
                //choice is quit, do exit with screen to default
                if(last == 1){
                    econio_textbackground(COL_BLACK);
                    econio_clrscr();
                    econio_textcolor(COL_WHITE);
                    exit(0);
                }
                else {
                    //choice is start, do clear screen, draw dificulty elements
                    econio_textbackground(COL_BLACK);
                    econio_clrscr();
                    econio_textcolor(COL_WHITE);
                    peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
                    pmed(6, 13, COL_WHITE, COL_BLACK, COL_WHITE);
                    phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);

                    last = 0;

                    econio_rawmode();
                    while(true){
                        //read keys and save previous state, and update now state
                        int ch = econio_getch();
                        if(ch == KEY_UP && last != 0){before = last; last--;}
                        else if(ch == KEY_DOWN && last != 2){before = last; last++;}
                        else if(ch == KEY_ENTER) break;

                        //state machine goes brrrrrrrr
                        switch(last){
                            case 0:
                                //easy text is shiney, medium is not
                                peasy(6, 7, COL_BLACK, COL_WHITE, COL_BLACK);
                                pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
                                break;
                            case 1:
                                if(before == 0){
                                    //easy was shiney, not now anymore, medium shiney
                                    peasy(6, 7, COL_WHITE, COL_BLACK, COL_WHITE);
                                    pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
                                }
                                else {
                                    //hard not shiney anymore, meium is shiney
                                    pmed(6,13,COL_BLACK, COL_WHITE, COL_BLACK);
                                    phard(6, 19, COL_WHITE, COL_BLACK, COL_WHITE);
                                }
                                break;
                            case 2:
                                //medium not shiney, hard is shiney
                                pmed(6,13,COL_WHITE, COL_BLACK, COL_WHITE);
                                phard(6, 19, COL_BLACK, COL_WHITE, COL_BLACK);
                                break;
                        }
                 }

                    econio_normalmode();
                 //before exit to who knows where, clean screen
                    econio_textbackground(COL_BLACK);
                 econio_clrscr();
                 econio_textcolor(COL_WHITE);

                return last;
             }
        }
}
```

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

5.12 menu.h

Go to the documentation of this file.

```
///Menu logic
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

```
                              {
    //set window size for good windows users :), but i like linux better, just not the terminal
modifying
#if defined(_WIN32) || defined(_WIN64) || defined(WIN32) || defined(WIN64)
    SMALL_RECT WinSize = {0, 0, 70, 40};
    SMALL_RECT* WinWin = &WinSize;
    SetConsoleWindowInfo(GetStdHandle(STD_OUTPUT_HANDLE), true, WinWin);
#endif

    //highscore board setup
    FILE* fptr;
    char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
    int testDB = 0;int db = 0;
    printf("Give how many scores to show from the top, 0 to not show anything");
    printf("\nor negative number as it will be converted to a positive number\n");
    printf("number of players to show from top 1: ");
    testDB = scanf("%d",&db);
    while(testDB < 1 || db < 0 || db > 1000000){
        econio_clrscr();
        printf("number of players to show from top 1: ");
        testDB = scanf("%*c%d%*c",&db);
    }

    //checks if highscores directory exist and openable
    DIR* dir2;
    dir2 = opendir("../highscores");
    if(dir2){closedir(dir2);}
    //directory does not exist
    else if(ENOENT == errno){
        closedir(dir2);
        mkdir("../highscores", 0700);
        dir2 = opendir("../highscores");
        if(dir2){closedir(dir2);}
        //directory does not exist
        else if(ENOENT == errno){
            closedir(dir2);
            perror("Could not create highscores directory, please create it manually to the same directory as main.c");
            exit(0);
        }
        //other problem
        else{
            closedir(dir2);
            perror("Error occured regarding the highscores directory");
            exit(0);
        }
    }
    //other problem
    else {
        closedir(dir2);
        perror("Error occured regarding the highscores directory");
        exit(0);
    }
    //state machine, would be better to put paths into a 2d char array, but who cares
    //its not like anybody gonna expand this sht
    switch(state){
        //show relevant highscore
        //dont wanna make a function to read file and store the data
        case 0:
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
        case 1:
            //show highscore in medium diff
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
        case 2:
            //high diff score
            fptr = fopen(paths[state], "r");
            if(fptr == NULL){
                fptr = fopen(paths[state], "w");
                if(fptr == NULL){
                    perror("File open error");
                    exit(0);
                }
            }
            printboard(fptr, db);
            break;
    }
}
```

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

```
                                        {
    if(db == 0)return;
    //variables
    Data data[1] = {};
    int index = 0;
    const char* padding = "                                        ";//soooo long, idk why
    char buff[5] = "";
    //drawing begining
    econio_clrscr();
    econio_gotoxy(30,1);
    printf("+++++++++++++++++");
    econio_gotoxy(30,2);
    printf("|NAME      | SCORE|");
    econio_gotoxy(30,3);
    printf("+++++++++++++++++");

    int tName =0; int tSCR = 0;
    //magic is done here
    while(index < db){
        //checks if data was corrupted
        if((tName = fscanf(fptr,"%s", data->name)) == -1 || tName == 0||(tSCR = fscanf(fptr, "%d",&data->score)) == -1 || tSCR == 0){
            //checks if its end of file, if first if did not work
            if((tSCR = fscanf(fptr,"%d",&data->score))== -1){break;}
            econio_clrscr();
            printf("\nCorrupted file\n");
            exit(0);
        }
        //checks if earlier break is needed
        if((tName == EOF && tSCR == EOF) || (tName == 0 && tSCR == 0)){break;}

        sprintf(buff, "%d", data->score);//transform int to string aka char array

        //get padding
        int pad1Len = (TARGETLEN*3/5) - strlen(data->name);
        int pad2Len =(TARGETLEN*2/5)- strlen(buff);

        //draw board
        econio_gotoxy(30, 4+(2*index));
        printf("|%s%*.*s|%*.*s%d|", data->name, pad1Len, pad1Len, padding,pad2Len,pad2Len,padding,data->score);
        econio_gotoxy(30, 4+(2*index+1));
        printf("+++++++++++++++++");

        index++;//magic number
    }
    printf("\n");
    printf("\n\nPress ENTER to start the game\n");
    econio_rawmode();
    while(1){
        int key = econio_getch();
        if(key == KEY_ENTER){break;}
    }
    econio_normalmode();
}
```

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

```
                                                {
    econio_clrscr();
    econio_textcolor(COL_WHITE);
    //set file path with state
    FILE* fptr;
    char paths[][50] = {"../highscores/l1.txt","../highscores/l2.txt","../highscores/l3.txt"};
    fptr = fopen(paths[state], "r");
    if(fptr == NULL){perror("Scoresave: file opening error");}
    else{
        /*
         * for legacy code to just exist here or to return to something in case of shit
        switch(state){
            case 0:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
            case 1:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
            case 2:
                fptr = fopen("../highscores/l1.txt", "r+");
                break;
        }
        */

        //set up variables to work with
        char name[MAX_NAME + 2] = {};
        int index = 0;
        Data *p = calloc(3, sizeof(*p));//for safety and to pass new data allocate 3 times the space
        Data temp;
        bool in = false;
        int i_new;

        //checks if highscores directory exist and openable
        DIR* dir;
        dir = opendir("../highscores");
        if(dir){closedir(dir);}
        //checks if directory do not exist
        else if(ENOENT == errno){
            closedir(dir);
            mkdir("../highscores", 0700);
            dir = opendir("../highscores");
            if(dir){closedir(dir);}
            //checks if directory still not exist
            else if(ENOENT == errno){
                closedir(dir);
                free(p);
                debugmalloc_dump();
                perror("Could not create highscores directory, please create it manually to the same
      directory as main.c");
                exit(0);
            }
            //other problem
            else{
                closedir(dir);
                free(p);
                debugmalloc_dump();
                perror("Error occured regarding the highscores directory");
                exit(0);
            }
        }
        //other problem
        else {
            closedir(dir);
            free(p);
            debugmalloc_dump();
            perror("Error occured regarding the highscores directory");
            exit(0);
        }

        FILE* n = fopen("../highscores/n.txt", "w");
        if(n == NULL){perror("Scoresave: file opening error, at line 80");}
        else{
            //input name
            printf("!Your name can only be 12 character long!");
            printf("Your name: ");
            int testName = 0;
            while((testName = scanf("%s",name)) == 0 || testName == -1 || strlen(name) > 12){
                printf("!Your name can only be 12 character long!");
                printf("Your correct name: ");
            }
            //load to struct the saved data
            //yeah nested loops, its not good, but it is what it is, who else gonna touch this code
      except me anyway?
            //while(fscanf(fptr,"%s %d",p[index].name, &p[index].score) != EOF){

```

```

            int testStr = 0; int testINT = 0;
            while((testStr = fscanf(fptr,"%s",p[index].name)) != EOF && (testINT = fscanf(fptr,"%d", &p[index].score)) != EOF){
                //tests if readed data was corrupted
                if(testStr == 0 || testStr == -1 || testINT == 0 || testINT == -1){
                    perror("Corrupted file");
                    break;
                }
                /// if name is in list and current score is higher than old, change it, than sort current arrays to descending based on scores
                //check if name is in the list
                if(strcmp(name, p[index].name) == 0){
                    in = true;//is it in or not?
                    //check to update its score
                    if(score > p[index].score){
                        p[index].score = score;
                        if(index != 0){
                            //bubble sort
                            for(int i = 0; i <= index-1; i++){
                                for(int j = i+1; j <= index; j++){
                                    if(p[j].score >= p[i].score){
                                        temp = p[i];
                                        p[i] = p[j];
                                        p[j] = temp;
                                    }
                                }
                            }
                        }
                    }
                }
                //gets ready for next item, reallocating space
                index++;
                p = (Data*)realloc(p, (index+1)*sizeof(*p));
            }

            //if not in highscorer list then add it to the top of its score group
            bool found = false;
            if(!in){
                for(int i = 0; i < index; i++){
                    if(score >= p[i].score && !found){
                        i_new = i;
                        found = true;
                        break;
                    }
                }
            }
            //if nobody is in the list put him in
            if(index == 0)fprintf(n, "%s %d",name, score);
            //save list to file
            else {
                for(int i = 0; i < index; i++){
                    if(i == i_new) fprintf(n, "%s %d\n",name, score);
                    fprintf(n, "%s %d\n",p[i].name, p[i].score);
                }
            }
            //free buff and close file stream
            free(p);
            fclose(n);
            fclose(fptr);
            //check if any problem occured while freeing memory
            debugmalloc_dump();
            //delete old, rename new save file
            //yeah there are better ways to do it, but i dont want to touch this at this point
            remove(paths[state]);
            rename("../highscores/n.txt",paths[state]);
        }
    }
}
```

References MAX_NAME, and datas::score.

5.14 score.h

Go to the documentation of this file.

```


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

```

///write to file to the specified file based on state to save new data or update old score
void savewrite_score(int state, int score);

///get input to how many elements to write based on state, medium for printboard
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 ()
          {
    //it is looped to be played again
    while(1){
        int state = menu();
        highscore(state);
        int score = game(state);
        savewrite_score(state, score);
    }
}
```

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

5.16 README.md File Reference

Index