NHF (Asteroids)
Programozói dokumentáció

Tartalom

A program futtatásához szükséges

- SDL2 médiakönyvtár a megjelenítéshez
- A textúrákat tartalmazó képfájlok:
    - background.jpg
    - asteroid_big_0.png
    - asteroid_medium_0.png
    - asteroid_small_0.png
    - laser.png
    - logo.png
    - raketa.png
    - text0.png
    - text1.png
- A használt betűtípust tartalmazó fájlok:
    - ARCADE_N.TTF

Fájlok tartalma

basic_constructs.h/.c

Struktúrák

- `Size`
    - Egy méretet ad meg
    - Elemek:
        - Szélesség (`int w`)
        - Magasság (`int h`)
- `Point`
    - Egy pontot ad meg a képernyőn
    - Elemek:
        - x koordináta (`double x`)
        - y koordináta (`double y`)
- `Texture`
    - Egy textúrát tartalmaz
    - Elemek:
        - Egy betöltött textúrára mutató pointer (`SDL_Texture *texture`)
        - A betöltött képről levágandó részlet kezdőpozíciója (bal felső sarka) (`Point position`)
            - Megj.: Ha valamelyik tagja [-1] a pozíciónak, a teljes kép kerül betöltésre
        - A textúráról kivágandó részlet mérete (`Size size`)
        - A textúra részlet elforgatása (`double rotation`)

- `SDL_Texture_Array`
    - A betöltött textúrákra mutató pointereket tartalmazza
    - Megj.: A program végén megkönnyíti a betöltött textúrák felszabadítását
    - Elemek:
        - A betöltött textúrákra mutató pointereket tartalmazó dinamikus tömb (`SDL_Texture **items`)
        - A dinamikus tömb hossza (`int length`)

Enumerációk

- `enum event_codes`
    - Az event kezelésért felelős állapotgép állapotkódjai
    - Tagok:
        - `event_game_tick`
            - 10 milliszekundumonként generálja a main.c-ben található idozit függvény
            - Ez adja a program frissítési gyakoriságát
        - `event_quit`
            - Elindítja a kilépési szekvenciát
        - `event_main_to_diff_select`
            - A menüt a főmenü képernyőről átviszi a nehézségválasztó képernyőre
        - `event_diff_select_to_main`
            - A menüt a nehézségválasztó képernyőről átviszi a főmenü képernyőre
        - `event_main_to_leaderboard`
            - A menüt átviszi a főmenüből a dicsőséglistára
        - `event_leaderboard_to_main`
            - A menüt átviszi a dicsőséglistáról a főmenübe
        - `event_diff_select_easy/normal/hard`
            - Beállítja a kiválasztott nehézségi szintet
        - `event_start_game`
            - Elindítja a játékot a kiválasztott nehézségi szinten
        - `event_name_input_to_main`
            - Visszaviszi a menüt a névbekérő képernyőről a főmenübe
- `enum difficulty`
    - A játék nehézségi szintjeit tartalmazza
    - Tagok:
        - `easy`
        - `normal`
        - `hard`

Függvények

- `SDL_Texture_Array init_SDL_Texture_Array();`
  - Visszatér egy üres `SDL_Texture_Array` struktúrával
- `void append_item_to_SDL_Texture_Array(SDL_Texture_Array* texture_array, SDL_Texture* texture);`
  - Hozzáad egy `SDL_Texture` pointert egy `SDL_Texture_Array` tömbjének a végére, illetve megnöveli a lista hosszát tároló változót eggyel.
- `bool isInRange(int number, int lower_boundary_excluded, int upper_boundary_included);`
  - Megnézi, hogy egy egész szám bent van-e egy meghatározott intervallumban.
  - Az alsó határ inkluzív, a felső határ exkluzív
  - Visszatérési értéke igaz, vagy hamis
- `void free_items_in_SDL_Texture_Array(SDL_Texture_Array* texture_array);`
  - Felszabadítja az összes elemet egy `SDL_Texture_Array`-ben
  - A `texture_array` által mutatott pointert nem változtatja meg, csak felszabadítja a mutatott memóriaterületet!
- `double point_dist(Point a, Point b);`
  - Visszaadja 2 pont egymástól való abszolút távolságát
- `SDL_Texture* load_texture(SDL_Renderer* renderer, char* file_name, SDL_Texture_Array* texture_array);`
  - Betölt egy textúrát egy fájlból, majd a kapott pointert hozzáadja egy `SDL_Texture_Array`-hez
  - Visszatérési értéke a betöltött textúrára mutató pointer
- `void send_user_event(enum event_codes event_code);`
  - Törli az SDL Event queue-ból az összes `SDL_USEREVENT` típusú eventet, majd hozzáad egy `event_code` által definiált eventet.
- `void convert_score_to_string(int score, char* output);`
  - Egy egész számot 6 karakteres sztringgé konvertál, az alábbi formátumban: [182 -> "000182"]
  - Amennyiben a `score` értéke nagyobb, mint 999999, az output sztring értéke „999999” lesz.
  - Amennyiben a `score` értéke kisebb, mint 0, az output sztring értéke „000000” lesz.
  - Az output sztring hossza legalább 6 + 1 karakter kell, hogy legyen!

menu_items.h/.c

Struktúrák

- `Effect`
  - Egy `menu_item` struktúra effect tagját határozza meg
  - Elemek:
    - `enum effect_type effect_type`
      - Effekt típusa
    - `enum direction direction`
      - Effekt iránya
    - `int param0`
      - Első paraméter
    - `int param1`
      - Második paraméter
    - `int param2`
      - Harmadik paraméter
    - `int param3`
      - Negyedik paraméter
    - `void (*setNextEffect)(struct menu_item*)`
      - Az effekt végetérte után végrehajtandó függvényre mutató pointer
  - Megj.: A paraméterek jelentése a kiválasztott effekt típusától függ.
- `menu_item`
  - Egy menüben található objektumot (pl. gomb) definiáló struktúra
  - Elemek:
    - `Texture texture`
      - Alapállapotban használandó textúra
    - `Texture texture_hover`
      - A kurzor hover esetén használandó textúra
    - `Texture texture_pressed`
      - A menu_itemre kattintás közben használandó textúra
    - `Point pos`
      - A menu_item pozíciója (középpont)
    - `Size size`
      - A menu_item mérete
    - `bool clickable`
      - Kattintható? (true -> igen, false -> nem)
    - `void (*onClickEvent)(struct menu_item*)`
      - Kattintás esetén végrehajtandó függvényre mutató pointer
    - `void (*onHoverEvent)(struct menu_item*)`
      - Hover esetén végrehajtandó függvényre mutató pointer
    - `struct Submenu* submenu`
      - Melyik submenu-höz tartozik az adott menu_item
    - `Effect effect`
      - A menu_itemhez rendelt effekt

- `menu_text`
    - Egy menüben található szöveget, illetve annak attribútumait tartalmazó struktúra
    - Elemek:
        - `char* content`
            - A menu_text által tárolt szöveg
        - `SDL_Color color`
            - A szöveg színe
        - `TTF_Font* font`
            - A szöveg betűtípusa
        - `int alpha`
            - A szöveg áttetszősége (0 - 255)
        - `Point pos`
            - A szöveg pozíciója
        - `Size size`
            - A szöveg mérete
- `Submenu`
    - Egy menu_itemekre mutató pointereket tartalmazó dinamikus tömb (Almenü)
    - Elemek:
        - `menu_item **items`
            - A menu_item pointereket tartalmazó dinamikus tömb
        - `int length`
            - A dinamikus tömb hossza

Enumerációk

- `enum button_states`
    - A gombok lehetséges állapotait tartalmazza
    - Tagok:
        - `button_normal`
        - `button_hovered`
        - `button_pressed`
- `enum direction`
    - Az effekt típusok irányát meghatározó felsorolás
    - Tagok:
        - `dir_up`
        - `dir_down`
        - `dir_left`
        - `dir_right`
        - `dir_in`
        - `dir_out`

- `enum effect_type`
    - A használható effekt típusok
    - Tagok:
        - **`no_effect`**
            - Nincs effekt, a többi paraméter értéke nem számít
        - **`move_vertical`**
            - Vertikális mozgatás
                - Lehetséges irányok:
                    - **`dir_up`** -> felfele
                    - **`dir_down`** -> lefele
                - Paraméterezés:
                    - **`param0`**:
                        - A távolság, amit szeretnénk, ha az objektum megtenne
                    - **`param1`**:
                        - A távolság megtételének sebessége
                    - A többi paramétert nem használja.
        - **`move_horizontal`**
            - Vízszintes mozgatás
                - Lehetséges irányok:
                    - **`dir_left`** -> jobbra
                    - **`dir_right`** -> balra
                - Paraméterezés:
                    - **`param0`**:
                        - A távolság, amit szeretnénk, ha az objektum megtenne
                    - **`param1`**:
                        - A távolság megtételének sebessége
                    - A többi paramétert nem használja.
        - **`blink`**
            - Egy objektum villogtatása
                - Lehetséges irányok:
                    - **`dir_in`** -> megjelenik
                    - **`dir_out`** -> eltűnik
                - Paraméterezés:
                    - **`param0`**:
                        - A minimum alfa (0 - 255), ameddig szeretnénk, hogy az objektum eltűnjön
                    - **`param1`**:
                        - A minimum alfa, ameddig szeretnénk, hogy az objektum megjelenjen
                    - **`param2`**:
                        - Az eltűnés, illetve megjelenés sebessége

        - **param3**:
            - Pillanatnyi alfa érték
    - Megj.: Az effekt váltogatja a **dir_in**, illetve a **dir_out** irányokat a maximum, illetve a minimum elérése esetén
    - Megj.: Az effekt ismétlődik, vagyis soha nem ér véget, tehát az effekt vége függvény soha nem lesz meghívva

Függvények

- `Submenu init_Submenu();`
    - Visszatér egy üres Submenu struktúrával
- `void append_menu_item_to_Submenu(Submenu* submenu, menu_item* menu_item);`
    - Hozzáad egy menu_itemet egy almenühöz
- `void render_Submenu(SDL_Renderer* renderer, Submenu* submenu);`
    - Egy almenü összes elemére alkalmazza a rajtuk lévő effektet, majd továbbadja őket a renderernek
- `bool is_hovered(menu_item* item, Point cursor_pos);`
    - Megnézi, hogy a kurzor egy adott menu_item-en helyezkedik-e el
- `void menu_item_rajzol(SDL_Renderer* renderer, menu_item* item, enum button_states button_state);`
    - Átad egy menu_item-et a renderernek
- `void execute_effect_on_menu_item(menu_item* item);`
    - Alkalmazza egy menu_itemre a hozzárendelt effektet
- `void apply_Effect_to_Submenu(Submenu* submenu, Effect effect);`
    - Hozzárendel egy effektet egy almenü összes objektumához
- `void set_title_effect_to_blink(menu_item* item);`
    - Egy megadott menu_itemhez hozzárendel egy villogás effektet
- `void start_game(menu_item* item);`
    - Leveszi egy menu_itemről a rajta lévő effektet, majd elindítja a játékot
- `void back_from_diff_screen(menu_item* item);`
    - A menüt visszaviszi a nehézségválasztó képernyőről a főmenübe
- `void render_text(SDL_Renderer* renderer, menu_text* item);`
    - Kirajzol egy szöveget

file_handler.h/.c

Struktúrák

- `file_entry`
    - Egy nevet, illetve a névhez tartozó pontszámot tartalmazó struktúra
    - Elemek:
        - `char name[PLAYER_NAME_MAX_LENGTH + 1]`
            - Név, hosszát a `PLAYER_NAME_MAX_LENGTH` határozza meg
        - `int score`
            - Pontszám

- `Array_file_entry`
    - Egy file_entrykből álló dinamikus tömböt, illetve annak hosszát tartalmazó struktúra
    - Elemek:
        - `struct file_entry* obj`
            - A dinamikus tömb
        - `int length`
            - A tömb hossza

Függvények

- `void save_scores_in_file(char* filename, Array_file_entry* array);`
    - Megnyit egy fájlt, majd beleírja a az összes file_entryt egy megadott Array_file_entry-ből (A fájl eredeti tartalma törlődik)
- `void destroy_Array_file_entry(Array_file_entry* array);`
    - Felaszabadít egy Array_file_entryt, amennyiben a struktúrában található tömbre mutató pointer nem NULL
- `Array_file_entry read_file_contents(char* filename);`
    - Lértehoz egy új Array_file_entryt, majd beolvassa a megadott fájlban eltárolt adatokat.
- `void sort_Array_file_entry(Array_file_entry* array);`
    - Pontszám szerinti csökkenő sorrendbe rendezi egy megadott Array_file_entry tartalmát
- `void file_entry_cserel(Array_file_entry* array, int index0, int index1);`
    - Egy Array_file_entry tömbjén belül két objektum helyét megcseréli
- `void beszur_elem_majd_rendez(Array_file_entry* array, file_entry entry);`
    - Hozzáad egy új elemet egy Array_file_entryhez, majd pontszám szerinti csökkenő sorrendbe rendezi a tömböt. Ezután a tömb hosszát limitálja 10 file_entryre, levágva a maradékot (a végéről).
- `void append_file_entry_to_Array(Array_file_entry* array, file_entry item);`
    - Hozzáad egy új file_entryt egy Array_file_entry tömbjéhez

_2d_object.h/.c

Struktúrák

- `ColliderCircle`
  - Egy ütközés érzékelő kör struktúrája
  - Elemek:
    - `double offset_abs`
      - Középpontjának abszolút távolsága az őt tartalmazó _2d_object középpontjától
    - `double offset_rotation`
      - Melyik irányba van értelmezve az offset_abs
    - `Point current_pos`
      - A ColliderCircle pillanatnyi pozíciója (ezt az értéket általában függvények kezelik)
    - `int radius`
      - A ColliderCircle sugara
- `Array_ColliderCircle`
  - Egy ütközés érzékelő köröket tartalmazó dinamikus tömböt tartalmazó struktúra
  - Elemek:
    - `ColliderCircle* array`
      - Dinamikus tömb, melynek minden tagja egy ColliderCircle
    - `int length`
      - A dinamikus tömb hossza
- `_2d_object`
  - Egy 2d objektum struktúrája
  - Elemek:
    - `int type`
      - Az objektum típusa (`enum object_type`)
    - `Texture texture`
      - Az objektum textúrája
    - `Point pos`
      - Az objektum pozíciója (középpont)
    - `Size size`
      - Az objektum mérete
    - `double speed_abs`
      - Az objektum abszolút sebessége
    - `double speed_ang`
      - Az objektum forgásának szögsebessége
    - `double rotation`
      - Az objektum pillanatnyi elforgatása
    - `Array_ColliderCircle collider_circles`
      - Az objektumhoz rendelt ütközésérzékelő köröket tartalmazó tömb

- `Ll_2d_object`
    - Egy _2d_objecteket tartalmazó láncolt lista nodeja
    - Megj.: Az objektumok tárolására azért a láncolt lista a legjobb választás, mert a listához gyakran kell hozzáadnunk elemet, illetve törölni belőle.
    - Elemek:
        - `_2d_object obj`
            - Maga az objektum
        - `struct Ll_2d_object* next`
            - A listában a következő objektumra mutató pointer

Enumerációk

- `enum object_type`
    - Az objektumok típuskódjait tartalmazó felsorolás
    - Tagok:
        - `obj_player`
        - `obj_laser`
        - `obj_asteroid_big`
        - `obj_asteroid_medium`
        - `obj_asteroid_small`

Függvények

- `void rajzol_2d_object(SDL_Renderer* renderer, _2d_object* object);`
    - Átad egy objektumot a renderernek
- `Ll_2d_object* dispose_Ll_2d_object_node(Ll_2d_object* node);`
    - Felszabadít egy elemet egy láncolt listában. Visszatérési értéke a lista első eleme
- `Ll_2d_object* delete_item_from_Ll(Ll_2d_object* list, Ll_2d_object* node_to_delete);`
    - Töröl egy elemet egy láncolt listából, melyet fel is szabadít. Visszatérési értéke a lista első eleme.
- `Ll_2d_object* find_last_node(Ll_2d_object* list);`
    - Visszatér egy láncolt lista utolsó elemére mutató pointerrel
- `Ll_2d_object* append_2d_object_to_Ll(Ll_2d_object* list, _2d_object new_obj);`
    - Hozzáad egy új listaelemet egy láncolt lista végére. Visszatérési értéke a lista első eleme. A függvény üres listán is működik.
- `void destroy_Ll_2d_object(Ll_2d_object* list);`
    - Felszabadítja az összes elemet egy láncolt listában
- `void render_remaining_lives(SDL_Renderer* render, _2d_object player, int max_lives, int current_lives);`
    - Kirajzolja a képernyőre a játékos fennmaradó életeinek számát
- `void adjust_collision_boxes(_2d_object* object);`
    - Beállítja a z ütközésérzékelő körök abszolút helyzetét
- `bool check_collision_between_2_obj(_2d_object* obj_a, _2d_object* obj_b);`
    - Két objektum között ellenőrzi az ütközést (true, ha ütköznek; false, ha nem)

- `Array_ColliderCircle init_Array_ColliderCircle();`
    - Inicializál egy új dinamikusan foglalt tömböt, melybe ütközésérzékelő körök kerülnek
- `void append_collider_to_array(Array_ColliderCircle* array, ColliderCircle circle);`
    - Egy CollisionCircle-ből álló tömbhöz hozzáad egy új elemet

objects.h/.c

Függvények

- `bool check_if_outside(_2d_object obj);`
    - Megnézi, hogy egy objektum elhagyta-e a képernyőt
- `void generate_random_pos_outside_boundaries(Point* pos, double* rotation, int radius);`
    - Generál egy új random pozíciót a képernyőn kívül, figyelembe véve egy objektum sugarát, illetve generál egy, a generált pozícióhoz képest a képernyő irányába mutató mozgási irányt.
- `Ll_2d_object* spawn_asteroid(Ll_2d_object* list, enum object_type object_type, SDL_Texture* texture, Point starting_point, bool starting_point_is_random);`
    - Generál egy új aszteroidát, majd hozzáadja egy objektum láncolt listához
    - Visszatérési értéke a lista első elemére mutató pointer
    - Ha a **`starting_point_is_random`** változó true, a függvény a képernyőn kívülre generálja az aszteroidát
    - Ha a **`starting_point_is_random`** paraméter false, a függvény figyelembe veszi a megadott **`starting_point`** paramétert, és oda generálja az aszteroidát
        - Az aszteroida mozgásiránya ez esetben random
- `bool check_if_outside_player(_2d_object* player);`
    - Megnézi, hogy a játékos a képernyőn kívülre került-e, és áthelyezi a képernyő másik oldalára, amennyiben igen
- `Ll_2d_object* generate_laser(Ll_2d_object* list, _2d_object* player, SDL_Texture* texture);`
    - Generál egy lézert, majd hozzáadja egy objektumokból álló láncolt listához
    - Visszatérési értéke a lista első elemére mutató pointer
    - Megj.: Azért szükséges a lista elejére mutató pointer visszaadása, mert lehetséges, hogy az első elem megváltozik, így a lista első elemére mutató pointer is más lesz.

menu_item_functions.h/.c

Függvények

- `void check_button_state(SDL_Renderer* renderer, menu_item* item);`
    - Megnézi egy gomb állapotát, majd ez alapján kirajzolja, illetve végrehajtja az állapothoz hozzárendelt függvényt (hover esetén az onHover, kattintás esetén az onClick függvény)
- A fájlban található többi függvény az adott gombokhoz tartozó onClick event. Ezek mindegyike a gombhoz tartozó állapotot helyezi el az SDL Event queue-ba.

game_view.h/.c

Függvények

- `int GameView(SDL_Renderer* renderer, enum difficulty diff, Submenu background_sub);`
    - Ez a függvény tartalmazza magát a játékteret, illetve ebben játszódik maga a játék.
    - A játék végetérte után a függvény felszabadítja maga után a lefoglalt memóriaterületeket, illetve a textúrákat.
    - Paraméterei:
        - `SDL_Renderer*`, az objektumok kirajzolásához
        - `enum difficulty` a nehézségi szint beállításához
        - `Submenu` a háttér átadásához
    - Visszatérési érték:
        - A játékos által megszerzett pontszám.

main.c

Ez a fájl tartalmazza a játék menüit, illetve az azt működtető állapotgépet.

Függvények

- `void sdl_init(const char *felirat, int szeles, int magas, SDL_Window **pwindow, SDL_Renderer **prenderer)`
    - Létrehoz egy új ablakot, melybe az SDL2 képes rajzolni.
- `Uint32 idozit(Uint32 ms, void *param)`
    - Időzítő függvény, megadott időközönként hozzáad egy **`event_game_tick`** kódú **`SDL_USEREVENT`**-et az event queue-hoz.
    - A programban 10 milliszekundumonként kerül meghívásra.

Fájlkezelés

- A „leaderboard.txt” fájlba mindig a legjobb 10 játékos adatai kerülnek elmentésre.
- Ha a fájl nem létezik, a program egy új fájlt hoz létre.
- Ha a fájl több elemet tartalmaz, mint 10, a leaderboard képernyőn csak a top 10 eredménye látható.
- Ha a fájlban formátumhiba van, a program a hiba pontjáig található elemeket megtartja, a többit törli.