A programozás alapjai. Nagy házi feladat programozói dokumentáció.

Készítette: Tóth Péter

Laborcsoport: R4L

Neptun kód: FLEYWS

Budapest, 2020. december 5.

Tartalom

1. Kígyó játék

A program egy kígyó játék egy- és kétszemélyes megvalósítása. A játék a Python 3.7.5-höz tartozó Pygame 1.9.6. grafikus moduljának segítségével készült el. A menüvezérelt programon belül a játékos ki tudja választani, hogy egy- vagy kétjátékos üzemmódot szeretne választani, majd a játékos vagy játékosok neveinek bekérése után elindul a játék maga. A játék lehetőséget ad a pontok számolására és grafikus visszatekintésére is. A játék futtatásához szükség van a megfelelő Python csomagra, illetve a PyGame telepítésére, amelyre vonatkozó információk a PyGame modul hivatalos oldalán találhatóak: https://www.pygame.org/wiki/GettingStarted

1.1. Kígyó játék logikája

A kígyó játék három fő elemből, a játékost reprezentáló kígyóból, a kígyónak ételül szolgáló almából és magából a pályából áll. Mind a három osztály a ***„Snake_functions.py”*** programban található. Az egyes játékelemhez tartozó adatokat egy-egy osztály tartalmazza.

A pálya valójában egy 25x25 db egység méretű cellából épül fel. A játékosok egy 4 cella hosszúságú kígyóval kezdenek, amely a pályán véletlenszerűen elhelyezkedő étkek elfogyasztásának hatására egy egységgel nőnek meg. A program a „kígyó” osztályban található listában tárolja a kígyó testéül szolgáló cellák x és y koordinátáit. A kígyó „mozgása” során valójában nem történik más csupán ezen listaelemek koordinátái változnak a kígyó haladási irányának megfelelően, mégpedig úgy, hogy a kígyó testét tartalmazó lista elé a haladási iránynak megfelelően beszúr egy eggyel különböző cella koordinátát, míg a lista utolsó elemét törli. Ha a kígyó az x-tengelyen jobbra halad, akkor a program a listaelemek x koordinátáját növeli eggyel. Balra haladás esetén pedig értelemszerűen az x-koordináták eggyel csökkennek. Az y-koordináta hasonlóan változik. Ha a kígyó a képernyőn lefelé halad a listaelemek y-koordinátái eggyel nőnek, ellenkező esetben eggyel csökkennek a képfrissítések alkalmával. A játékos kígyója abban az esetben nő, ha a kígyó fejének és a pályán véletlenszerűen elhelyezkedő alma koordinátája megegyezik. Ekkor a kígyó pozíciójának utolsó eleme nem kerül törlésre, a kígyó „hossza” eggyel nő.

A kígyó osztálynak ezenfelül minden egyes pillanatban tárolnia kell a kígyó haladásának aktuális irányát, amelyet a játékos vagy játékosok természetesen a nyilak segítségével (két játékos esetén a WASD billentyűk segítségével is) tudnak befolyásolni.

A játéknak egy játékos üzemmód esetén akkor van vége, ha a játékos kígyójának fej pozíciójának x- vagy y-koordinátájának értéke 0-25 értéken kívüli értéket vesz fel, vagy ha a kígyó fejének és valamelyik test elemének a koordinátája megegyezik.

Két játékos esetén annyival bonyolultabb a program megvalósítása, hogy figyelembe kell venni mind a fej-fej, a fej-saját test, fej–idegen test, illetve a fej–fal ütközéseket is. Fej-fej ütközés esetén a játék véget ér mind a két játékos számára, míg fej-test ütközés esetén csak annak a játékosnak ér véget a játék, amelynek feje a másik kígyó testébe ütközött. Az életben maradó kígyó addig folytathatja a játékot, amíg a falba vagy saját testébe nem ütközik.

A játék végét követően a játékos nevével ellátott pontszámot egy szövegfájlban lesznek elmentve, amely a főmenü „Leaderboard" menüpontja alatt bármikor elérhetőek a felhasználó számára grafikusan megjelenítve. A program ezután promptolja a felhasználót, hogy szeretne-e új játékot játszani.

Emellett fontos megemlítést tenni az „alma" nevű osztályról, amely az alma pozícióját illetve az alma kirajzolásának függvényét tartalmazza. A programnak figyelnie kell arra, hogy az étek elfogyasztása után az almát egy olyan helyre generálja, ami nem esik egybe a kígyó vagy kígyók testével.

1.2. Menüvezérelt események logikája

A program struktúrája az *1.1*- illetve az *1.2 ábrákon* láthatóak. A menüvezérelt programok által használt függvényeket a ***„menu_functions.py"*** program különböző osztályai tartalmazzák. Ennek részletes bemutatása látható az alábbiakban:

- ***Snake_main.py:*** A játék megnyitása a ***„Snake_main.py"*** programon keresztül történik. Ez a program felelős a PyGame modul nyitásáért és zárásáért is egyaránt. A PyGame modul a ***„Snake_main.py"*** alprogramjai közül sehol nem inicializálódik újra vagy záródik be.
- ***main_menu.main():*** A ***„Snake_main.py"*** közvetlen alprogramja ami a program főmenüjeként szolgál. Innen érhető el a ***„Start"*** gombra kattintva a játék üzemmódjának kiválasztására szolgáló ***„one_or_two.main()"*** program is. . ***„Leaderboards"*** menüpontra kattintva az egy, illetve a kétszemélyes ranglisták grafikus megjelenítései érhetőek el a felhasználó számára a ***„Leaderboards_one.main()"*** és ***„Leaderboards_two.main()"*** programokon keresztül. A főmenü ***„Credits"*** gombjára kattintva a játék készítőjének adatai, míg a ***„How to Play"*** gombra kattintva a program

felhasználásának rövid útmutatója érhető el rendre a ***„Credits.main()”*** és ***„how_to_play.main()”*** programokon keresztül. A főmenü bezárása után a program a *„return”* paranccsal visszatér a főmenübe.

- ***one_or_two.main():*** A program, ami lehetőséget biztosít a felhasználónak, arra, hogy kiválaszthassa a játék üzemmódját, vagy arra, hogy a ***„Vissza”*** gomb segítségével visszatérjen a főmenübe. Az ***„Egy játékos”*** gombra kattintva elindul a ***„player_names.main()”*** program. Amennyiben a felhasználó a ***„Két játékos”*** gombra kattint elindul a ***„two_name.main()”*** program.
- ***player_names.main():*** Ez a forráskód egy a felhasználó számára módosítható szövegdobozt tartalmaz. A szövegdobozba beírt (min. 3, max. 15 karakter, szóközt nem tartalmazó) szöveg lesz a játékos azonosítója a játék során, illetve az elért pontszámával ezen a néven fog szerepelni a ranglistán. A megfelelő formátumú név megadása után a ***„Mehet”*** feliratú gombra kattintva elindíthatja az egy játékos üzemmódú kígyó játékot. Amennyiben a ***„Vissza”*** gombot nyomja meg a függvény *„Igaz”* értékkel tér vissza a függvény a ***„one_or_two.main()”*** program futását ezzel nem szakítva meg.
- ***one_player.main(PlayerName):*** Az egy játékos üzemmódban működő kígyó játék forráskódja, melynek működési elve leírása került az *1.1 fejezetben*. Egyetlen bementei paramétere a „***player_name.main()”*** programtól átvett string típusú változó. A kígyó játékon kívül a játékos nevét és pontszámát tartalmazó fejléc grafikáját is tartalmazza.
- ***game_over_p1.main(pont, name):*** A játék végét jelző grafikai megjelenítést tartalmazó forráskód. Kiírja a „Játék vége!” feliratot, a játékos nevét és elért pontszámát. Az ***„Új Játék”*** gombra kattintva „main” függvény *„Igaz”* értékkel tér vissza ezzel nem szakítva meg a ***„one_or_two.main()”*** függvény futását, míg a ***„Főmenü”*** gombra kattintva a „main” függvény *„Hamis”* értékkel tér vissza, ezzel megszakítva a ***„one_or_two.main()”*** függvény futását visszatérve a ***„main_menu.main()”*** függvénybe. Ezen a forráskódon belül fut le a ***„ranglista_one.txt.main()”*** is.
- ***two_names.main():*** Ez a forráskód a felhasználók által módosítható szövegdobozokat tartalmaz. A szövegdobozokba beírt (min: 3, max: 15 karakter, szóközt nem tartalmazó) szövegek lesznek a játékosok azonosítói a játék során, illetve az elért pontszámokkal ezen a néven fognak szerepelni a ranglistán. A megfelelő formátumú név megadása után a ***„Mehet”*** feliratú gombra kattintva a felhasználók elindíthatják a két játékos üzemmódú kígyó játékot. Amennyiben a ***„Vissza”*** gombot nyomja meg a függvény

„Igaz” értékkel tér vissza a függvény a ***„one_or_two.main()”*** program futását ezzel nem szakítva meg.

- ***two_players.main(Name1, Name2):*** A két játékos üzemmódban működő kígyó játék forráskódja, melynek működési elve leírása került az *1.1 fejezetben*. Bementei paraméterei a „***two_names.main()”*** programtól átvett string típusú változók. A kígyó játékon kívül a játékosok neveit és pontszámait tartalmazó fejléc grafikáját is tartalmazza.
- ***two_player_end_scene(pont1, pont2, name1, name2):*** A játék végét jelző grafikai megjelenítést tartalmazó forráskód. Kiírja a ***„Játék vége!”*** feliratot, a játékosok neveit és elért pontszámaikat. Az ***„Új Játék”*** gombra kattintva „main” függvény *„Igaz”* értékkel tér vissza ezzel nem szakítva meg a ***„one_or_two.main()”*** függvény futását, míg a ***„Főmenü”*** gombra kattintva a „main” függvény *„Hamis”* értékkel tér vissza, ezzel megszakítva a ***„one_or_two.main()”*** függvény futását visszatérve a ***„main_menu.main()”*** függvénybe. Ezen a forráskódon belül fut le a ***„ranglista_two.txt.main()”*** program is.
- ***ranglista_one_txt.main(name, pont):*** Beolvassa a ***„ranglista_one_player.txt”*** szövegfájlban található neveket és pontokat egy listába, majd ellenőrzi, hogy a pareméterként kapott *(name, pont)* tuple nem szerepel-e már a listában. Ha szerepel, akkor visszatér a függvény, azonban, ha nem szerepel, akkor a *(name, pont)* tuple-t hozzáfűzi a listához. A lista sorbarendezésre kerül *„selection sort”* módszerrel, majd a lista elemein végigiterál. Minden egyes elemből készít egy a *rank(pos, name, pont)* által definiált objektumot, ahol „*pos”* a nullától kezdődő ciklusváltozó eggyel megnövelve. A program ügyel arra, hogy az azonos pontszámú versenyzők azonos *pos*, azaz helyezés értéket kapjanak. Az így kapott objektumok egy új listához lesznek hozzáfűzve, amelynek elemeit kiírásra kerülnek a már előzőleg törölt ***„ranglista_one_player.txt”*** szövegfájlba a megfelelő formátumban.
- ***ranglista_two_txt.main(name, pont):*** Beolvassa egy listába a ***„ranglista_two_player.txt”*** szövegfájlban található neveket és pontokat egy listába, majd ellenőrzi, hogy a paraméterként kapott *(name, pont)* tuple nem szerepel-e már a listában. Ha szerepel, akkor visszatér a függvény, azonban, ha nem szerepel, akkor a *(name, pont)* tuple-t hozzáfűzi a listához. A lista sorbarendezésre kerül *„selection sort”* módszerrel, majd a lista elemein végigiterál. Minden egyes elemből készít egy a *rank(pos, name, pont)* osztály által definiált objektumot, ahol „*pos”* a nullától kezdődő

ciklusváltozó eggyel megnövelve. A program ügyel arra, hogy az azonos pontszámú versenyzők azonos *pos,* azaz helyezés értéket kapjanak. Az így kapott objektumok egy új listához lesznek hozzáfűzve, amelynek elemeit kiírásra kerülnek a már előzőleg törölt ***„ranglista_two_player.txt”*** szövegfájlba a megfelelő formátumban.

2. Adatszerkezetek

2.1. Snake_functions.py

Class Color: Színeket tartalmazó mellékosztály.

Class Grid(Color): A pálya tulajdonságait tartalmazó egyik főosztály. A Color() osztály egy példányát használja.

> ***def __init__(self):*** A Grid(Color) osztály konstruktor függvénye. Tartalmazza a cellák számát és méretét (amik a felépülő ablak méreteit fogja majd adni), az ablak nevét, illetve fejléc és a betűtípus adatait. Ennek az osztálynak szintén attribútuma az első és második játékos pontszáma, ami a megadott betűtípusban fog kiíródni a fejlécre.
>
> ***def draw_grid_p1(self, color):*** Egy játékos esetén a függvény meghívásával a program kirajzolja a pályát és a fejlécet az éppen aktuális pontszámmal.
>
> ***def draw_grid_p2(self, color):*** Két játékos esetén a függvény meghívásával a program kirajzolja a pályát és a fejlécet az éppen aktuális pontszámokkal.

Class Snake(Grid, Color): A játékos kígyójának attribútumait tartalmazó osztály. A Grid() és a Color() osztályok egy-egy példányát használja.

> ***def __init__(self, grid, color):*** A Snake(Grid, Color) osztály konstruktor függvénye. Listaként tartalmazza a kígyó testelemeinek koordinátáit (cella_x, cella_y) formában, a kígyó színét (alapértelmezett szín a zöld), a kígyó fejének a pozícióját és a kígyó éppen aktuális haladási irányát. Szintén attribútomként tartalmazza a kígyó testelemeinek pozícióját listaként.. Emellett attribútuma hogy a kígyó éppen növekszik-e (bool, akkor válik igazzá ha a fej és az étek pozíciója megegyezik), illetve, hogy a kígyó még él és mozog, vagy falnak, testnek, két játékos esetén másik kígyótesttel történő ütközés következtében meghalt-e.

def event_handler(self, event, snake): A billentyűzet lenyomása során keletkező esemény kezeléséről gondoskodó függvény. A megfelelő billentyűzetek lenyomásakor frissíti a kígyó haladási irányát a lenyomott gombnak megfelelően.

def clear_snake(self, grid, color): A függvény meghívásának hatására törli a kígyó előző pozícióját képernyőről.

def is_dead(self, grid): Leellenőrzi, hogy a kígyó feje a pályán belül található, illetve, hogy a kígyó fejének a pozíciója nem egyezik meg a kígyó bármely saját testelemének a pozíciójával.

def rearrange(self): Újrarendezi a kígyó testelemeinek a pozícióját. A haladási iránynak megfelelően eggyel nagyobb (x vagy y koordinátákon való pozítív haladási irány esetében) vagy eggyel kisebb (x vagy y koordinátákon való pozítív haladási irány esetében) pozíciót vesz fel a kígyó feje, amit a pozíció lista elejére beszúr a program a függvény meghívásakor. Abban az esetben ha a kígyó nem találkozik étekkel, a lista utolsó elemét törli a program (így érhető el az a látszat, hogy a kígyó vonszolja maga után a testét), ellenkező esetben egy újrarrendezés idejéig nem törli a pozíciólista utolsó elemét, ekkor egy elemmel nő a kígyó testének hossza.

def move(self, grid): Az újrarendezett pozíciólista elemeinek megfelelően kirajzolja a kígyó testének elemeit a játékablakra.

class Apple(Grid, Color): A játékmezőn megjelenő étek (alma) attribútumait tartalmazó osztály. A Grid() és a Color() osztályok egy-egy példányát használja.

def __init__(self, grid, color): Az Apple(Grid, Color) osztály konstruktor függvénye. Tartalmazza az alma aktuális pozícióját, illetve az alma grafikus megjelenítésének elemét. Az alma alapértelemzett színe piros.

def draw_food(self, grid): A függvény hívásának hatására kirajzolja a játékmezőre az alma az éppen aktuális pozícióban.

2.2. Menu_functions.py

class Mouse: Az egér pozícióját tartalmazó segédosztály melynek példányát egyéb osztályok fogják majd használni.

class Window: Az éppen aktuális menüablak attribútumait tartalmazó osztály. Tartalmazza az ablak méretét, színét és címét.

class Button(Window, Mouse): Az éppen aktuális menüablakon található gombok attribútumait tartalmazó osztály. Használja a Window() és a Mouse() osztályok egy-egy példányát is.

> ***def __init__(self, window, mouse, width = 100, heigth = 50, pos = (312.5, 337.5), color_unclicked = (255, 0, 0), color_click = (0, 255, 0), text = "Start", fontsize = 40):*** A Button() osztály konstruktor függvénye. Attribútuma a gomb szélessége és hossza, a gomb felülete amin majd a gomb megjelenik a képernyőn, a pozíciója, a gomb színét amikor nincs rajta illetve mikor rajta van a kurzor. A gomb feliratának betűtipusát, annak szövegét és a szöveg pozícióját (ami megegyezik a gomb felületének pozíciójával.) Bool változoként tárolja, hogy a kurzor éppen a gomb felett van-e.
>
> ***def is_on_button(self, mouse):*** A függvény megvizsgálja, hogy a kurzor a gomb felett van-e. Ha a téglalap objektum határain belül van a kurzor, akkor a *self.on_button* értéket igazra változtatja. A függvény az egér pozíciójának megtalálása miatt használja a Mouse() osztály egy példányát.
>
> ***def draw(self, window):*** Kirajzolja a gombot a éppen aktuális menüablak felületére a megadott pozíciónak megfelelően. Amennyiben a kurzor rajta van a gombon, akkor a gomb színe zöldre változik, ellenkező esetben a gomb színe piros.

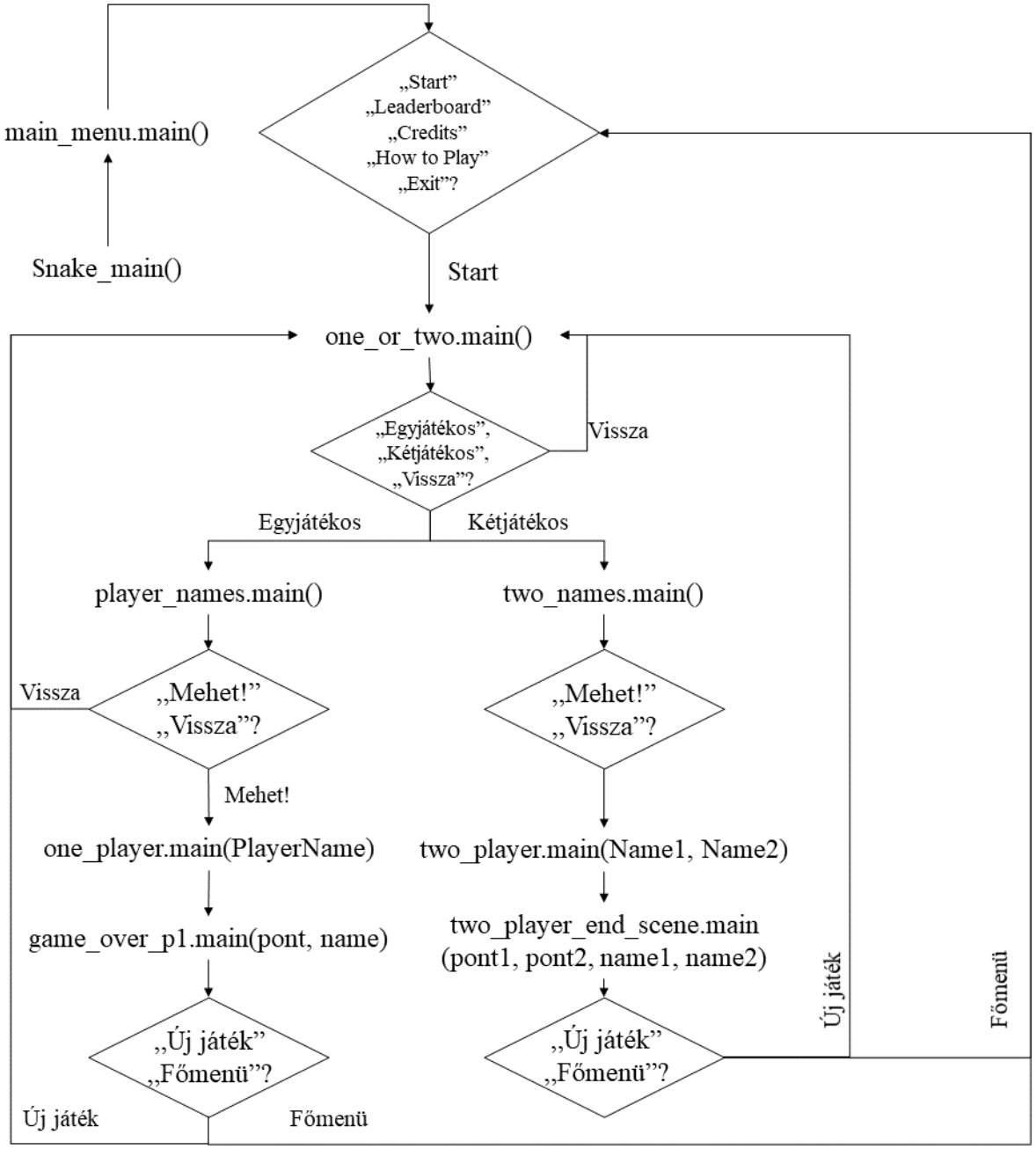
class Textbox(Window): Az éppen aktuális menüablakon található szövegdoboz adatait tartalmazó osztály. Használja a Window() osztály egyik példányát is.

> ***def __init__(self, window, width, heigth, pos, text, max = 15, min = 3):*** A Textbox(Window) osztály konstruktor függvénye. Tartalmazza a szövegdobozba írható karakterek minimális és maximális hosszát, a karakterszám túllépése során keletkező hibaüzenetek betűtípusát, szövegét, a szövegdoboz magasságát és szélességét, illetve szövegdobozként szolgáló téglalap típusú objektum pozícióját. Emellett tartalmazza a szövegdoboz szövegének betűtípusát és magát a kiirandó szöveget. *Bool* változóként tárolja azt, hogy a szövegdobozba éppen belekattintott a felhasználó vagy sem. Egy bool változóban tárolja, hogy a szöveg tartalmaz-e szóköz karaktert, illetve az ehhez tartozó hibaüzenet szövegét. Tartalmazza a szöveg létrehozásakkor létrejövő felület pozícióját, ami megegyezik a szövegdoboz keretének pozíciójával.
>
> ***def event_handler(self, event):*** Egérgomb vagy billentyűzet lenyomása esetén, esetleg felhaszálói esemény kezeléséről gonodoskodó függvény. Abban az esetben ha az egérgomb lenyomásakor az egér pozíciója megegyezik a szövegdoboz téglatest

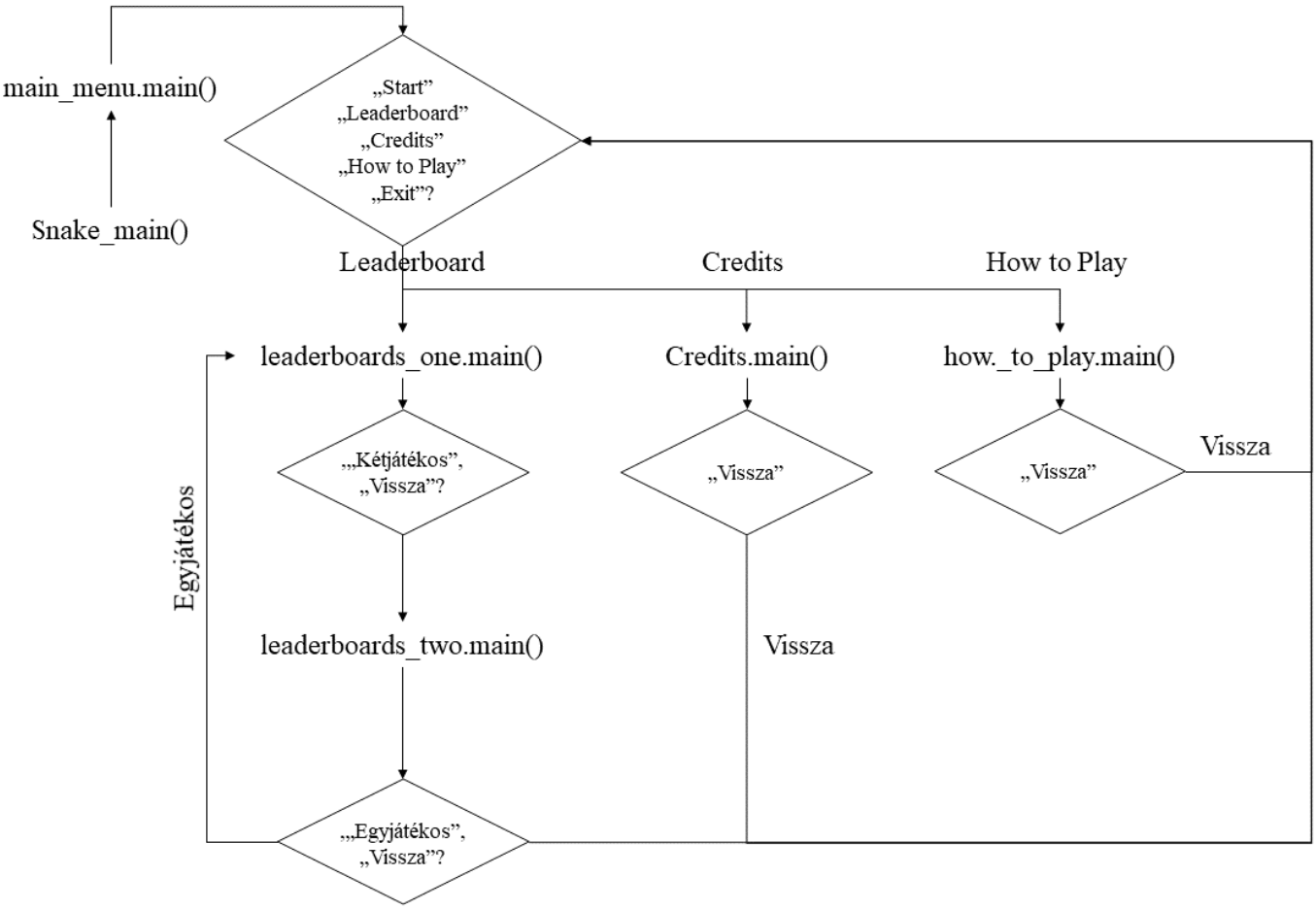
objektumaként szolgáló pozícióval, akkor a gombhoz tartozó *self.clicked* értéket negálja. Abban az esetben ha a lenyomott billentyűzet a backspace volt, akkor törli a szövegdobozban lévő utolsó karaktert. Bármilyen más billentyűzet lenyomása esetén az annak megfelelő unicode karaktert fűzi hozzá a szövegdoboz objektum szövegéhez Felhasználói esemény hatására leellenőrzi, hogy a szövegdoboz tartalmaz-e szóköz karaktert. A szövegdoboz méretét igazítja a beírt szöveg méretéhez.

def draw(self, window): Kirajzolja a szövegdobozt, alá pedig az esetlegesen hozzá tartozó hibaüzeneteket. 3 karakternél kisebb vagy 15 karakternél nagyobb, esetleg szóközt tartalmazó karakter esetén figyelmezteti a felhasználót, hogy javítson. Hibás bemenet esetén a szövegdoboz színe piros. Abban az esetben ha a szövegdobozra a felhasználó rákattintott és az érték helyes a szövegdoboz kék. Ha a szövegdobozban szereplő string értéke helyes és a felhasználó nem kattintott bele a szövegdobozba, akkor annak színe fehér

3. Melléklet



1. ábra: A menüvezérelt program felépítése I.



2. ábra: A menüvezérelt program felépítése II.