Kígyójáték
fejlesztői dokumentáció

Kövecs Roland (GPZN1J)

2020. november-december

Fejezetek:

Bevezető, a program funkciói

Ez a dokumentáció a Programozás 1 nevű, fizika alapképzés tárgy keretein belül megvalósult beadandó nagy házifeladat teljesítésére megírt C nyelvű kígyó játék program részleteit kívánja bemutatni.

A játék a klasszikus „snake game” mintájára íródott Windows operációs rendszerre, kiegészítve egy kétjátékos móddal. A szabályok a megszokottak: ha a kígyó falnak ütközik, vagy a saját testének nekimegy, akár visszafelé indulva, akkor meghal; ha eszik egy eledelt, a testmérete nő egyel. A kétjátékos módban szintén hasonló szabályok érvényesek, kiegészülve a két játékos testének ütközésével, ami ugyancsak a játék végét jelenti.

Az indításkor megjelenő menü lehetőséget nyújt a felhasználónak a játéktér méretének módosítására és a játékmód kiválasztására. Ha véget ér a játék, az eredményét elmentheti egy, az eszközön játszott mindenkori eredmények rekordtáblájába. Új játék is kezdhető mindkét játékmód esetén. Ha az új játék kezdését választja a felhasználó az indításkor megjelenő menüvel az előbb leírt folyamat megismétlődik. A játék intuitív, de tartalmaz utasításokat a felhasználóval való kommunikáció céljából (menü, játékmód- és pályaméret választás, illetve az elért eredmény elmentése az egy játékos módnál), ezekre egyszerű eldöntendő kérdések formájában, vagy adatok megadásával(pálya méret) kell válaszolnia a felhasználónak. Ezek pontosabban a felhasználói dokumentáció részben olvashatók.

Futási környezet, fordítás

A konzolos megjelenítés miatt a képernyőre írást a gotoxy nevű függvény segítségével valósítottam meg(részletek az adatstruktúrák, függvények és főbb változóik részben), amihez a windows.h API-t használtam, így a program **kizárólag Windows operációs rendszeren futtatható**.

A program minimális rendszerkövetelményekkel is elfut, ami legfőképpen a konzolos megjelenítésnek köszönhető. Nem szükséges a standard könyvtárakon kívül mást telepíteni a számítógépre, hogy játszhassunk ezzel a remek játékkal. Az alkalmazás mérete 75 KB, átlagosan 1 MB memóriát használ. A folyamat során a képkockák között legalább 100ms (1 játékos), illetve 150ms (2 játékos mód) szünet van ezért, alacsony CPU teljesítménnyel is élvehető a program. A konzolos megjelenítés minimális GPU követelményeket támaszt.

A forráskód fordítása megegyezik bármely egyszerű, C nyelvű kód fordításával, külső standard könyvtárakon kívüli könyvtárak hiányában nem szükséges plusz tevékenység.

A program felépítése

Az egyszerűség kedvéért az összes forráskód egy .c file-ban van, nincsenek külsőleg definiált header-ök, objektumok. Ennek ellenére a kód áttekinthetőségét és strukturáltságát a különböző folyamatok függvényekre bontása segíti.

A forráskód elején találhatók a standard könyvtárak inkludálása: stdio.h, windows.h, conio.h, stdlib.h, time.h, stdbool.h. Ezt követik a típus definíciók: egy enum *hdir* –kígyó fej irány beállításának és egy struktúra, ami a kígyó testrészeinek felépítését határozza meg (láncolt lista „láncszemei”). A típus definíciók után a függvények definiálása következik a végrehajtási sorrendtől független sorrendben (fejlesztési sorrendben). A main függvényben a az előbb definiált függvények hívása történik kontroll függvények irányítása szerint (újraindítás, egy- vagy kétjátékos mód), illetve a játék során használt változók deklarálása (ezeket a függvények **cím szerinti paraméterátadással** használják fel és módosítanak értékeiken (ahol nincs érték átállítás ott simán érték szerint adódnak át paraméterként)).

A main függvényben először a változók deklarálása történik (egyszer fut le a program futása során), és néhánynak az új játék indításakor a változók értékét alapértelmezettre állító setup függvénytől függetlenül is értéke állítódik be.

Ezután elindul egy „**do-while**” ciklus, ami a játék újra nem indításáig fut, tehát minden játékmenet indításakor végrehajtódnak a benne meghívott függvények: mainmenu fv (játékmód, pályaméret):

1. **egyjátékos mód:**
    a. setup fv. – változók alapértékének visszaállítása az előző játékmenetből
    b. draw fv. – pálya és a fej kirajzolása a játék legelején
    c. while(!game end) amiben a halálig történő folyamatotok futnak le:
        i. control fv. – billenytűnyomás érzékelés -> hdir(fej irány) beállítás
        ii. logic fv. –a játék magja, változók értékének változtatása, logikai döntések(ütközés, evés), test elem létrehozás és hozzáadás a testhez
        iii. pontszám kiírás frissítés a pálya aljára
    d. wipesnake fv. – test elemek törlése a fej kivételével
    e. save fv. – eredmény lementése a rekordtáblába és a rendezett tábla kiírása

2. **kétjátékos mód**

    A kétjátékos módban meghívott függvények megegyeznek az egyjátékos módban meghívottakkal, azzal a különbséggel, hogy itt a *control*, *logic és wipesnake függvények* kétszer hívódnak meg, a második játékos paramétereire is, illetve lefut még egy *crash függvény* ami a két kígyó testének összeütközését vizsgálja.

    Fontos megjegyezni, hogy a logic függvény másodszori lefutásakor egy multi nevű bool változó igazra állításával fut le a második játékosra eltérő folyamat (fej kiírás). A setup és draw függvények is hasonlóan járnak el a kétjátékos mód alatti lefutáskor. Ennél a játékmódnál nincs lehetőség mentésre, mert úgy gondoltam, hogy az egymással versengés ennek a módnak a célja és nem a pontszám.

A két játékmód két if függvény segítségével fut le. Igaz, lehetett volna egymás után közvetlen if-ekkel végrehajtani a másodszor végrehajtandó függvényeket, viszont így átláthatóbb, hogy szeparált a két mód.

A do-while ciklus végén történik a játék újra indítására kérdezés és az esetleges újra lefutás tesztelése egy restart bool változó segítségével.

Adatstruktúrák, függvények és főbb változóik

Adatstruktúrák és a main függvény lokális változói

A kígyó(k) teste **láncolt listával** van ábrázolva, amiben csupán az egyes elemek x és y koordinátái vannak eltárolva adatként. Ez az adatstruktúra kitűnően jelképez egy elemekből álló, növekvő testű kígyót. Egy elem mérete 0,07 KB méretű, minden evésnél egy újabb ilyen elem van mallocolva, majd a testhez fűzve (logic fv-ben történik), végül a wipesnake fv.-el törölve a játék vége után.

- **snake** - A játék legfontosabb struktúrája a *snake* nevű struktúra ami a kígyó testét jelképező láncolt lista elemeinek felépÍtését határozza meg. Adatai az x,y integerek, a next pointer a következő elemre mutat. A fejre nem mutat semmi, az utolsó NULL-ra mutat.
- **dir és dir2** – A kígyók fejének irányát tároló változó. Értékei: 0(stop), L, R, U, D (bal,jobb,fel,le), használja: control, logic
- **game_end** – játék vége bool, alapból false,
- **restart** – új játékmenet bool, alapból false
- **eaten és eaten2** – evett-e a kígyó a körben bool, alapból false
- **dsize** – alapértelmezett pályaméret bool, alapból true
- **multi** – kétjátékos bool, használja: setup, draw, logic
- **head, head2** – játékosok legelső test elemei a láncolt listában, típus: snake*
- **x,y és x2,y2** – első és második játékos fej integer koordinátái, külön vannak módosítva az átláthatóságért, módosítás után megkapják a head(2).x és head(2).y az értékeiket
- **fx, fy –** étel (forráskódban kaja) integer koordinátái, értéküket random sorsolja a logic függvény, ha bármelyik fej megeszi
- **score és score2** – játékosok pontszámai, egyenlő a fej utáni részek számával, értékét sok folyamat használja a végrehajtások száma miatt
- **height és width** – pálya méretei, integerek alapból height=20, width=40, de a felhasználó be is állíthatja a mainmenu fv-futásakor

Függvények (algoritmusaik) és főbb változóik

A mainen belüli lefutás sorrendjében.

1. **Mainmeu:** Játékmód választás: char be változóval, ezt is adja vissza függvény ami alapján lefut az egyjatékos mód függvény tömb vagy a másik. Pályaméret választás: bool dsize átállítása false-ra, majd méretek bekérése height és width átállításával. Szélső értékben 5x5-ös lehet a pálya(két fal, két játékos, egy kaja). A a játékmód kiválasztásán kívül (itt csak számokat lehet beírni, különben egy végtelen ciklus lesz) az összes bemeneti válasz korlátozott, ha nem megfelelő karakterekkel válaszol a felhasználó, a kérdés megismétlődik.

2. **Setup:** A main változóinak alapértelmezett értékadása(minden áll, játéknak nincs vége, pontok nullák, első játékos feje a pálya felénél, másodiké a negyedénél van kiírva, illetve a legelső kaja generálás is megtörténik)

3. **Gotoxy:** Odaviszi a kurzort a megfelelő koordinátákra(x és y integer bemenetek alapján)

4. **Draw:** Falak kiírása, és a játék kezdetén a fej(ek) és a kaja kiírása: a width és height értékek alapján. A falakat ciklusok, az egyedi karaktereket pedig a gotoxy fv. segítségével írja ki.

5. **Control és Control2:** hdir típusú dir és dir2 bemenetek értékét állítja át a `GetAsyncKeyState('billentyu')`

fv. segítségével, ami igazat ad vissza ha a vizsgált billentyű le van-e nyomva. Első játékos esetén(awsd) második játékos esetén(jikl). Ha nincs billentyűnyomás, dir és dir2 értéke marad, így irányt tud tartani a kígyó.

6. **Logic:** A program magja, itt történnek a fő változó értékállítások, logikai döntések(falnak vagy testnek ütközés, evés). A függvény mindkét játékos paramétereit be tudja állítani, az átadott értéken kívül a kígyók megegyeznek (kivéve a fej (O/M)).

   a. Először az **evés vizsgálata** történik (fej koord == kaja koord). A vizsgálat eredményét az eaten bool változó értéke tárolja, amit a fv. később felhasznál. Ha történt evés akkor a score int változót egyel növeli. A score értékét később felhasználjuk.
   Az új kaja koordinátáit random sorsolja ki, és négynél nagyobb testméret esetén(ez alatt túl kevés az esélye az alapértelmezett pályaméretnél – kisebb területnél viszont már van vizsgálat 4 testrész alatt is, megegyező valószínűséggel már egy testrész esetén) megvizsgálja, hogy nem a testbe sorsolta-e a kaját. A vizsgálat egy snake pointerrel történik, ami végigszalad a test részein a fejtől kezdve. A ciklus lefutását a score értéke határozza meg. Az esetleges falba sorsolás vizsgálata mindig lefut. Ha „beletalált egy objektumba", újat sorsol. A kiírás gotoxy-val történik.

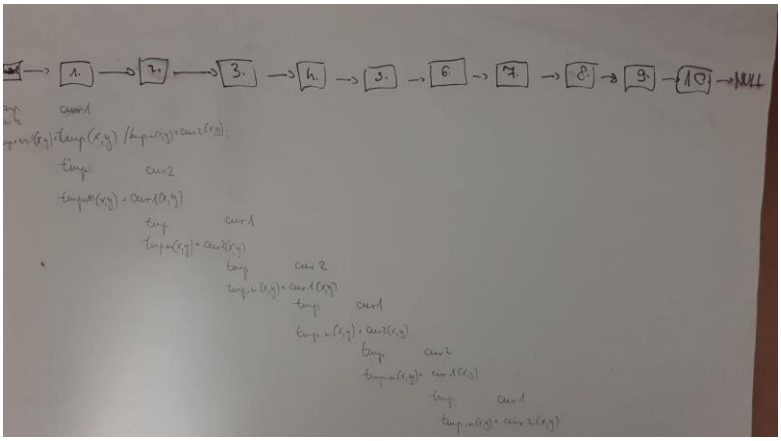
   b. Ha evett és az eaten bool értéke igaz, egy test elem foglalódik le a memóriában és hozzáfűződik a testhez. Az uj snake típusú elem next pointere NULL, ezután megkeresi az addigi legutolsót a last pointerrel és annak a következő eleme lesz az uj.

   c. A legbonyolultabb része az egész programnak az adatok léptetése a kígyó testén belül. Ehhez három segédváltozót használtam, egy tmp pointert (érték változtatás) és két snake típusú segéd elemet cur1 és cur2 (érték tárolás). A fejtől kezdve indulva egyel hátra csúsztatja az eljárás az elemek koordinátáit, így léptetve a kígyót.

      <u>Az eljárás:</u>

      i. tmp-vel mutat az elemekre és azzal megy végig az összesen amíg a ciklus változó a score értékénél kisebb, így mindegyik elemet meg tud vizsgálni és átállítani az értékét.
      ii. A tmp által mutatott elem utáni elem értékét eltárolja a cur1, majd a tmp utáni elem felveszi a tmp értékét. A tmp lép a következő elemre, aminek az értékét a cur2 tárolja el, majd felveszi az előző lépésben cur1 által felvett elem értékét amit az előző lépésben módosítottunk. A tmp megint lép a következő elemre aminek megint a cur1 tárolja el az értékét, majd felveszi a cur2 értékét, és így tovább. Észrevehető, hogy a páros sorszámú (0-tól kezdve) elemeknél a cur1 tárol, a páratlanoknál a cur2. Ezt esetszétválasztással oldottam meg: a ciklus változót osztjuk kettővel, így megtudjuk, hogy az elem páros vagy páratlan.
      iii. Az adatmódosítás előtt gotoxy-val letörli az elemet, majd az értékváltozás után kiírja az elemet az új koordinátára szintén gotoxy-val. Így a test elemek kiírása helyben történik. Lehetett volna külön függvénnyel a struktúráltság miatt, viszont így helyben szerintem könnyebben megoldható.

Egy kép a léptető algoritmus megértéséhez:



d. A függvényben ezt követően a dir változó alapján megtörténik a fej x és y koordináták értékváltoztatása, majd átadása a fej elemnek. A játékmódtól függően az egyes játékos feje helyére 'O', a kettes játékoshoz pedig 'M' karaktert ír ki a függvény a megváltozott koordinátákra. Így az elemek léptetése után minden elem elé kiíródik a kígyó feje.

e. A függvény végén van a falba és a farokba ütközés vizsgálata. A farokba ütközés ellenőrzése 2 ponttól indul, tehát fej+1elemnél még nem ütközhet bele a saját testébe a kígyó. Ezt azért tartom hasznosnak, hogy a játékkal először játszó felhasználó megérthesse a játék szabályait és ne haljon meg már az elején.

7. **Wipesnake:** A kígyó lefoglalt test elemeit törli a fej nélkül, hogy az esetleges újra indításkor ne kellejen megint lefoglalni azt. A program bezárásával törlődik csak a kígyó feje. A törlés két segédpointerrel történik a fej címének átadásával.

8. **Save:** Az *eszközön játszott* játékok (így nem szükséges egy txt.-t is hordozni és menteni a játékhoz) pontszámot menti el névvel együtt egy rekordtáblába. A scoreboard.txt nevű file tárolja az eddigi pontszámokat. A függvény ha nem bírja megnyitni olvasómódban a file-t kreál egy ilyen nevűt a gépen, így biztosítva a file jelentlétét amivel dolgozhat. A név (max. 10 karakter) bekérése után beírja azt és a pontot a táblába a file hozzáfűzés módban való megnyitásával, így nem kavarva össze az adatokat. Ezután megszámolja a beolvasott adatokat, majd létrehoz egy akkora dinamikus tömböt ami player struktúrákat (név és pontszám) tartalmaz és beolvassa őket a tömbbe. A tömb adatait, már a mostani pontszámmal együtt buborékos rendezés algoritmusával pontszám szerint csökkenő sorrendbe rendezi. A file-t törli az eszközről, majd egy ugyan olyan nevűt kreál amibe visszaírja a rendezett a datokat, eközben kiírja a konzolra kocsmai boksz gép szerűen. Ez az eljárási mód furcsának tűnhet de ezt találtam a legegyszerűbbnek a file adatainak módosítására.

9. **Crash:** A két játékos módnál a kígyók ütközését vizsgáló függvény. Logikailag a logic függvény része lenne, és meg is lehetne valósítani ott plusz paraméterek átadásával, viszont nekem így jobban tetszett, hogy egymás után látható a két eset(egyes ütközik kettes testének és fordítva). Ha csak két fej van a pályán, akkor is ütközhetnek egymással a testek hiányának ellenére. (Valószínűleg a két játékos egyből ezt próbálja majd ki).

Tesztelési eredmények

A kétjátékos mód irányítása nehéz feladatnak tűnt, de megoldottam egy aszinkron billentyűlenyomás figyeléssel (control fv.). A játékmenet során a lenyomott billentyűk bemenetet generálnak a pufferbe, amit a FlushConsoleInputBuffer() függvénnyel töröltem, viszont ennek ellenére a kétjátékos módnál valamiért néha előfordul, hogy egy irányító billentyű lenyomására elindul a másik játékos karaktere is egy teljesen random irányba. A problémát orvosolandó, próbáltam közvetlen a játékmenet alatt futó while ciklus előtt lenullázni mindkét dir változó értékét, viszont ez sem segített, a hiba még mindig fennáll, kiszámíthatatlanul előfordul esetenként.

Ezen kívül, a játék megfelelően működik és élvezhető.

Felhasználói dokumentáció/segédlet

1. A játék a klasszikus „snake game” mintájára íródott Windows operációs rendszerre, kiegészítve egy kétjátékos móddal.

2. A játék futtatásához indítsa el az snake.exe file-t, majd kövesse a képernyőn megjelenő instrukciókat.

3. A játékmód kiválasztásához írjon be 1(egyjátékos mód) vagy 2(kétjátékos mód) számokat!

4. Ha ezután pályaméretet kíván változtatni írjon be n betűt, majd adja meg a választott pálya méreteit (fontos, hogy legalább 5x5-ös legyen)!

5. A játék irányítása a megszokott awsd billentyűkkel történik, illetve jikl a második játékos számára.

6. A szabályok a megszokottak: ha a kígyó falnak ütközik, vagy a saját testének nekimegy, akár visszafelé indulva, akkor meghal; ha eszik egy eledelt, a testmérete nő egyel. A kétjátékos módban szintén hasonló szabályok érvényesek, kiegészülve a két játékos testének ütközésével, ami ugyancsak a játék végét jelenti. A kígyó nem tud diagonálisan haladni.

7. Az egyjátékos módban a játék mentésére is van lehetőség. A program képes eltárolni az eszközön végzett játékok eredményeit, ha ön is szeretné. Ehhez válassza a mentést, majd adja meg a nevét. Ezt követően a program elmenti az eredményét és kilistázza az eszközön mentett eredmények rekord tábláját.

8. Ha újra kívánja indítani a játékot válaszoljon y betűvel erre a kérdésre. Ha nem szeretne többet játszani ezzel a király játékkal, üssön n betűt a megfelelő kérdésre.

Jó szórakozást kíván a játék fejlesztője:

Kövecs Roland

2020. 11-12. hó