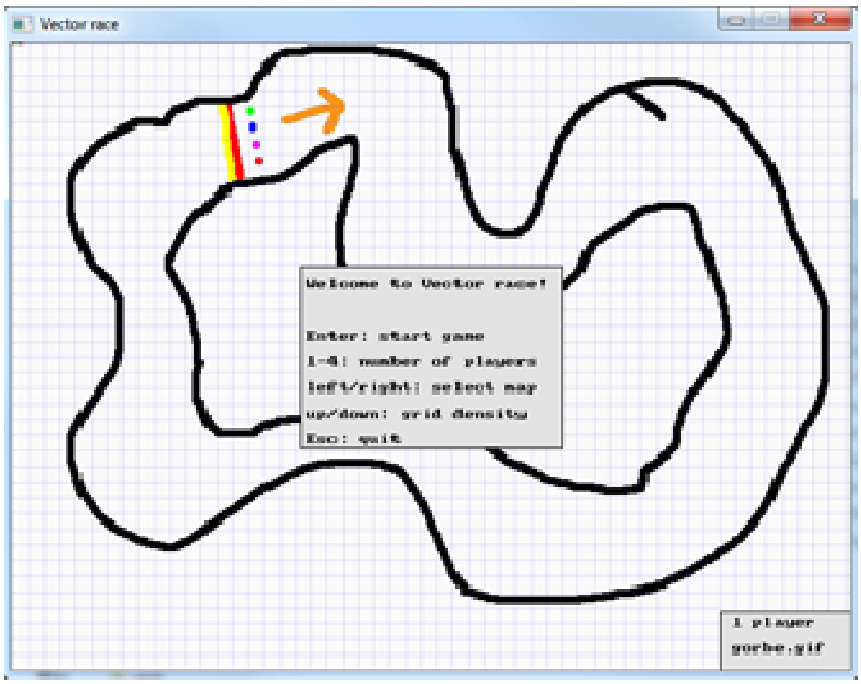
Felhasználói dokumentáció

A “Vector race”, a hagyományos kockás papíron játszható egyszerű játék számítógépes változata.



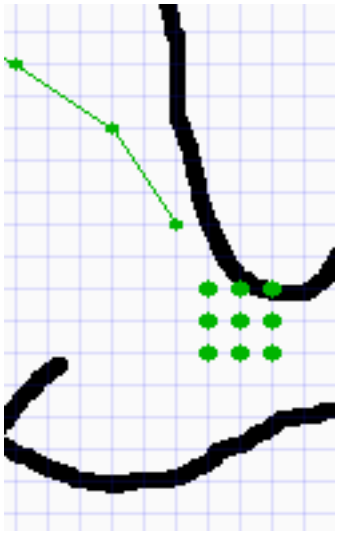
A játék indításához szükséges az autoverseny.exe file-on kívül néhány .dll is (ugyanabban a könyvtárban) illetve egy maps nevű könyvtár, amely legalább egy gif-et tartalmaz (a játék helyes működéséhez egy megfelelő struktúrájú és tartalmú palettás gif szükséges).

A játék indítása után a nyitóképernyő fogad minket.

Itt lehetőségünk van:

- beállítani a játékosok számát 1 és 4 között a megfelelő számok megnyomásával
- kiválasztani a pályát a jobb és bal nyilak segítségével (a lista körbefordul)
- beállítani a rácsméretet a fel és le nyilak használatával
- kilépni az ESC billentyűvel
- a játékot az Enter gombbal indíthatjuk

A kiválasztott pálya nevét és a játékosok számát a jobb alsó információs panelen ellenőrizhetjük.



A játék indítása után az első játékos kerül lépésre. A felületen megjelennek az érvényesen választható rácspontok, melyeket a jobb láthatóság kedvéért ki is színez a program. A számbillentyűzet(keypad) 1-9 gombokkal választhatunk, melyik mezőre szeretnénk lépni a felajánlottak közül (természetesen, ha valamelyik nem elérhető, azt nem választhatjuk). Megjegyzendő, hogy olyan lépést viszont választhatunk, mellyel kicsúszunk a pályáról.

A választás értelemszerűen a gombok keypadon való elhelyezkedéséből adódik, tehát ha a bal felső pontot szeretnénk választani a 7-est kell megnyomjuk, míg ha a jobb oszlop középsőjét, akkor a 6-ost.

A lehetőségek mindig úgy vannak felajánlva, hogy az utolsó lépésünk vektorát (kezdetben 0-át) meg kell ismételjük és ahová az vezet, azt, vagy valamely szomszédos rácspontot lehet kiválasztani. Ez viszonylag jól szimulálja az autóversenyekre jellemző fizikát.

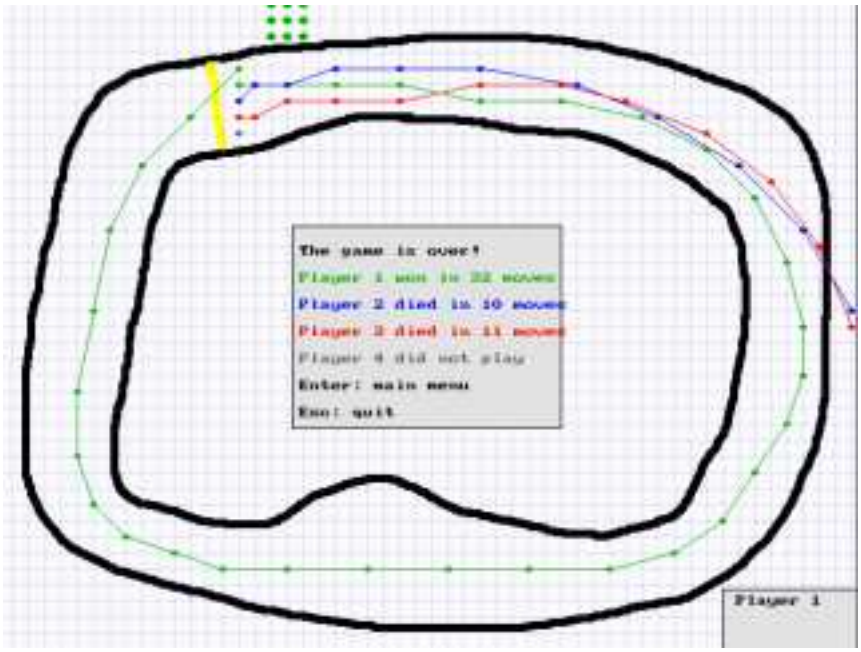
A játék célja, hogy a pálya vonalának (fekete vonal) érintése nélkül megtegyünk egy kört és a célvonalon (a megfelelő irányból) haladjunk át. Ha a játékos keresztezi a pálya vonalát, akkor a játékból kiesik. Amennyiben rossz irányból hajt át a célvonalon, szintén kiesik. Amennyiben a helyes irányból keresztezi a célvonalat, akkor a játékát befejezte. Ha valaki egy lépésen belül több

“különleges” elemet is érint (például átmegy a célon, de rögtön utána falba is csapódik) akkor az számít, amelyiket először keresztezte (a példa szerint nyert).

A játékosok sorban egymás után következnek, őket más-más színnel jelöljük. A játék véget ér, ha valamennyi induló játékos kiesett vagy sikeresen letudta a pályát. A cél természetesen minél kevesebb lépéből körbeérni.

Hogy melyik játékos következik, a jobb alsó infó panelen is látszik, valamint a kis körökkel megjelölt lehetséges lépések színe is segíti a tájékozódást.

A játék során az ESC billentyűvel bármikor visszaugorhatunk a főmenübe (viszont ekkor minden eddigi lépésünk elveszik!).



A játék végeztével egy ablak tájékoztat minket az elért eredményekről. Információkat láthatunk, hogy hány lépés után és milyen eredménnyel zárták a játékosok a kört (mindenki aki beért győztesnek minősül ☺). ESC-el kilépünk, Enter-el visszajutunk a főmenübe.

A játék lehetőséget biztosít saját pályák használatára. A pályákat elkészíthetjük szinte bármely képszerkesztő program segítségével. A követelmény hozzá annyi, hogy palettás képet használjunk (lehetőség szerint átlátszó hátterű gif-et). A pálya első 9 pixele (1,1;2,1;3,1;4,1;5,1;6,1;7,1;8,1;9,1) tartalmazza ugyanazt a paletta információt, amivel rendre a következőket jelöltük ki:

1, pálya vonalát

2, start vonalat

3, célvonalat

4-5-6-7, első-második-harmadik-negyedik játékos kezdő pozícióját

8, meta információkat (menetirány, pálya készítőjének neve… stb)

9, átlátszó színt

A program a főmenüben a gif-et jeleníti meg, azonban játék közben megjelenítéskor már leveszi a kezdő pozíciókat, a rajtvonalat és a meta adatokat. Ügyeljünk rá pályaszerkesztésnél, hogy a start vonal arra való, hogy meggátolja a célon rossz irányból való áthaladást, ugyanakkor ne látszódjon játék közben (így tulajdonképp egy láthatatlan fal), ezért jó oldalán helyezzük el a célnak!

Jó szórakozást kívánok! ☺

Tesztelői dokumentáció

Játékosok számának állítása, a programban valóban megfelelő számú játékost állít be. Visszajelzést a jobb alsó infópanelen kaphatunk a főmenüben. Játék közben pedig ez az induló játékosok számát jelenti. Figyelem! Hibásan tervezett pályánál lehet, hogy valaki eleve a rajtvonalról kezd, így már a legelső lépésben ki fog esni. Ez nem programhiba, hanem pályahiba.

Rácssűrűség állítása a racssuruseg globális változót állítja, és valóban minden fv ezt a változót használja a számításokhoz. Teszteléséhez játsszunk különböző rácsméreteken és figyeljük a helyes működést.

Pályák betöltése egy önmagába záródó listát kellett, hogy eredményezzen, ha sikerül a jobbra-balra nyilakkal a főmenüben lefagyasztani, vagy elérni, hogy egy jobb és egy bal nyíl megnyomásával ne ugyanazt a pályaképet lássuk, az mindenképp baj.

Játék közben keresztezzük a rajtot, vagy keresztezzük a falat. Ha továbbra is tudunk lépni az a hibás működés jele. Haladjunk át a célon. Ha továbbra is tudunk lépni az hiba.

Számoljuk a lépéseinket (jobb alul is kijelzi nekünk játék közben) ha bármikor eltérést észlelünk a kiírt és a valós lépésszám között az hiba.

Amíg nem keresztezzük sem a pálya falát, sem a rajtot, sem a célt, bármennyit léphetünk. Ettől eltérő működés hibára utal.

A játék végén lévő információk helyességét ellenőrizzük, hogy megfelelnek-e a valóságnak (mely játékosok játszottak egyáltalán, hány lépésben és milyen eredménnyel zárult a játékuk). Valóságtól eltérő esetben hibás működést állapíthatunk meg.

Fejlesztői dokumentáció

A fejlesztésnél a cél az eredeti hangulat visszaadása volt, így a pályán (szándékosan) csak diszkrét pontokon lehet lépkedni, pedig a vektorműveletekkel, ennél pontosabb “hatást” is el lehetne érni.

A program a következő fő struktúrákat tartalmazza:

Pályalista, pálya, játék, játékos, validmoves. A játék típus (majdnem)mindenki mást is tartalmaz(legalább közvetve), mert a fv-ek így tudták passzolgatni az adatokat cím szerint könnyen.

Globális változónak van kitéve a rácsméret és a képernyő felbontása (ami elegánsan konfig file-ban is lehetne, sőt a felhasználónak meg lehetne engedni a változtatását… végül a rácsméret beállítható paraméter lett.)

A pályák listáját a /maps könyvtár .gif –jeiből építjük fel, mint egy önmagába záródó duplán láncolt listát (egy eleműnél is). Ha nem sikerült egy pályát sem betölteni, a program nem indul el.

A main elején inicializálunk egy játékot (game) és beállítjuk a default értékeket (játékosok neveit, színeit, hogy egy játékos aktív…).

Létrehozunk egy SDL Surface-t (ez is része lesz a játék structnak, egy pointeren keresztül), egy resx és resy globális változóba várjuk a képernyő felbontást, amiben működni fogunk (így a pályáknak ekkoráknak vagy ennél kisebbeknek kell lenniük, ezt nem ellenőrizzük, de egy értelmes funkció lenne).

Van néhány rajzoló függvényünk, ezek mind Draw_ -al kezdődnek és közös tulajdonságuk, hogy nincs bennük SDL_Flip hívás, tehát ráteszik a surface-ra ami a dolguk, de külön frissíteni kell utánuk a hívónak. Egyetlen kivétel a Draw_refresh_gamescreen, az minden lépés után a játék ciklusban magától újra rajzolja a szükséges surface-eket.

Az eseményhurokban a gombok nyomkodását figyeljük csak. A játéknak van egy állapota (0:főmenü, 1:játék, 2:játék vége) ez, mint állapotgép befolyásolja, hogy a megnyomott gombok mit eredményeznek.

A menüben az állítgatás viszonylag egyszerű, viszont minden változtatás után a rajzolásokat újra el kell végezni, ez sok sor, helyenként igencsak sorminta jellegű, lehetne szebbíteni rajta.

A játék alapvetően az event loop-ban amikor a játék státusza 1 akkor zajlik. A logikája a következő, amikor elindítottuk a játékot (a menüben az enterrel) akkor minden játékos pozíció listájának első két elemét beállítjuk a kezdőpozíciójára (mintha az első lépést ugyanoda tették volna ahol eddig is álltak), átállítjuk a játék státuszát futásra (ami alapvetően a lenyomott gombok hatását befolyásolja) és meghívjuk a Draw_refresh_gamescreent, hogy az első képünk meglegyen. Az első játékos az aktív, így ő van lépésen.

A lépés előtt le kell fusson a set_valid_moves függvény, ami a CurrentValidMoves struktúrában beállítja (az éppen lépésen lévő játékoshoz), hogy mely lépései lesznek érvényesek a 9 lehetőségből és azok mely játékkoordinátákra mutatnak. Ezeket ki is rajzoljunk aztán a Draw_valid_moves-al.

Utána várunk valami gomblenyomásra. A nyomás hatására meghívódik a lép fv, megkapja a játékot és a gombot, amit nyomtak.

A lépés menete: a kiválasztott koordinátáról mivel tudjuk, hogy szabályos lépés (set_valid_moves által generáltakat választhatott csak) ezért a játékoshoz épített pozlistához (ami egy duplán láncolt lista) hozzáadjuk a választott koordinátát gondolkodás nélkül (ezt az add_koord_to_player végzi). Meghívjuk a do_last_move fv-t, ami pedig a játékos pozíció listájából előveszi az utolsó kettőt. A köztük lévő képpontokat(!) végig vizsgálja, hogy van-e valami speciális feltétel, mint pl. fal vagy cél. Ha talál, akkor a játékos státuszát átállítja játszikról valami másra. A fv az első különleges pixel hatására véget ér (ha valaki a cél után falnak megy, az attól még győz, aki pedig fal után robog át a célon az halott), ez szándékosan van így. Mivel a lépés befejeződött és a játékos státusza megváltozott, ha kellett, visszatérünk a lép fv-be, ami ezek után meghívja a set_next_active_playert.

A set_next_active_player egy lényeges fv, beállítja a következő játékost lépésre (ez a játék struktúra int kovetkezik változója). Veszi az az utánit, mint akin éppen állunk és ha a státusza játszik akkor ő a következő, ha nem akkor veszi a következőt és így tovább. Mod 4 van használva ebben a függvényben (hogy ne mutassunk ki a 4elemű játékosok tömbjéből) és korántsem elegánsan végzi a feladatát (todo comment van is mellette), de működik. Ami még ebben fontos, hogy ha nincs több játszik státuszú játékos, ez a fv. vet véget a játéknak. (Draw_end_menu hívással és játék státuszának állításával)

Ha sikerült beállítani egy következő játékost, akkor hívódik a set_valid_moves (erre a játékosra) aztán várunk gombnyomásra és kezdődik a játék ciklusa elölről. Kilépni ebből a ciklusban a loop-ban adott ESC-el, de amúgy csak a set_next_acive_playerrel fogunk.

A játék végén - nagyon hasonlóan, mint a főmenünél - kitesszük az információkat (Draw_end_screen).

A játék végén megszámoljuk, hogy a játszó játékosok pozíció listája hány elemű volt, hogy kiírhassuk hányat léptek. A játékosok státuszából jön, hogy miként fejezték be a játékot (esetleg egyáltalán el se kezdték). Ami még fontos, hogy miután megszámoltuk és kiírtuk a láncolt lista elemszámát, utána le is bontjuk őket (reset_koords_of_player fv intézi), felszabadítva a malloc-olt helyeket, egy esetleges új játék előtt.

Nyilvánvaló hibák a program struktúrájában:

A sormintákat (főként rajzoló fv-ek) lehetne csökkenteni.

A kód sok helyen épít arra, hogy fix 4 játékos van. Ez még akkor is igaz, ha néhány ezek közül nem aktív, gyakorlatilag úgy kezeljük, mint egy speciális kiesési feltételt, hogy el sem indult.

A fv-ek elnevezése kissé átgondolatlan, sokszor keresgélni kell.

Az alap koncepció nem volt kellően átgondolt a paraméterek passzolgatásával kapcsolatban. Egy ideig a fejlesztés a globális változók használata felé ment, de az teljesen rossz volt, így maradt az az ötlet, hogy a játék (mint struktúra) tároljon mindenki mást (legalább mutató szinten) és a fv-ek csak a játékot passzolgassák, így mindenki mindenhez hozzáfér. Ez ekkora méretben nagyon kényelmes is

így, de mivel nem az elejétől van így pl. a pályalista nem is került be a játék-ba, mert nem kellett éppen semelyik fv-nek. A később írt fv-ek mind jatek_t* típust várnak, a korábbiak össze-vissza mindenfélét ☹.

Általánosítási ötletek:

A játékosok számát lehetne (sőt kéne) dinamikusa kezelni és lehetőséget biztosítani a felhasználóknak saját név/szín választására.

További ötletek a program továbbfejlesztéséhez:

Lehetne undo/redo funkció.

Lehetne az autó mozgása animált (hiszen most is közbeeső képpontonként nézegeti, volt-e fal, ennyi erővel végig is mozgathatna valami kis autó-szerű képet ott, persze a fordulás animálása már nehezebb ügy).

Lehetne benne gépi ellenfél.