2. A számítógépes játékfejlesztésről

A játékfejlesztés kétségkívül a programozás egyik legcsábítóbb területe. Napjainkban sokan azzal a szándékkal kezdenek megismerkedni a programozással, hogy létrehozzanak valami igazán különleges és lenyűgöző játékot. Ez mindenképpen szép elhatározás, azonban ahhoz, hogy lelkesedésünk ne fulladjon kedvetlenségbe, ne érjenek bennünket csalódások, és elkerülhessük a kudarcokat, érdemes elolvasni a következő sorokat. A leírtak nagy része nemcsak játékok, hanem bármilyen alkalmazás fejlesztésére érvényes.

2.1. Néhány tanács játékfejlesztőknek

Az első komolyabb program – tehát nem feltétlenül játék – elkészítése során igen sokat tanulhatunk. Észre sem vesszük, és a munka előrehaladtával újabb és újabb ismereteket szerzünk. Folyamatosan problémákba ütközünk, amelyeket jó esetben meg is tudunk oldani. Ha mégsem, akkor lehetséges, hogy túl nagy fába vágtuk a fejszénket! Következzen hát az első jótanács:

Csak olyasmibe kezdjünk, amit képesek vagyunk befejezni!

Ugyanis mindegy, mennyire bravúros megoldásokat, forradalmi újításokat tartalmaz a forrásunk, ha a programunkat soha nem fejezzük be! Egy félkész program akkor is félkész, ha amúgy briliáns a kód. Tehát néha bármennyire jónak is tűnik egy-egy ötlet, ne sajnáljuk elvetni, ha valóban azt akarjuk, hogy játékunk elkészüljön. Leginkább talán a kezdő játékkészítők hajlamosak arra, hogy a fellegekben járjanak: ragyogó fények, színorgia, realisztikus mozgás, millió ötlet; de ki fogja ezt lekódolni?

Tehát legyünk szerények – különben hamar rájövünk arra, hogy egyedül nem vagyunk képesek arra, amire egy csapatnak is több évre lenne szüksége. Sajnálatos módon mennek tönkre amúgy ígéretes kezdeményezések, garázsprojektek csak azért, mert túl magasra helyezik a lécet. Legjobb, ha olyan célokat tűzünk magunk elé, amit belátható – mondjuk 6 hónapos vagy annál rövidebb – határidő alatt meg tudunk valósítani. Ha nem így teszünk, nagy valószínűséggel lankadni fog lelkesedésünk, és előbb-utóbb abbahagyjuk az egészet.

Ne találjuk fel újra a spanyolviaszt!

Érthető, hogy az ember örül, ha a semmiből hoz létre valamit. Most azonban a lényeg, hogy belátható időn belül valami eredményt tudjunk felmutatni. Természetesen nem arról van szó, hogy valahonnan letöltsünk egy kész forrást, és aztán pár képet kicserélve úgy állítsuk be, mintha saját fejlesztés lenne! Ez plágium – mondjuk ki: lopás! –, ami a mi szakmánkban a legnagyobb bűn.

Azonban nyugodtan körülnézhetünk a neten, és átvehetjük programozótársaink vívmányait anélkül, hogy újra feltalálnánk Bresenham vonalrajzoló algoritmusát, vagy komplex matematikai függvénygyűjtemények létrehozásával bíbelődnénk. Ilyesmiket nyugodtan vegyünk kölcsön, de törekedjünk arra, hogy megértsük, átlássuk a működésüket (hiszen ha nem így lenne, valószínűleg úgyis képtelenek lennénk használni a megszerzett „kincseket”).

Mindenképpen ajánlott kisebb próbaprojektben tesztelni az átvett kódot, mielőtt saját, gondosan felépített programunkba illesztenénk, hiszen senki sem garantálja, hogy a kód valóban jól működik, s az sem biztos, hogy minden szolgáltatására szükségünk van. Érdemes azt is megvizsgálni, hogy az adott forrás valóban szabadon felhasználható és módosítható-e, továbbá illik valahol megemlíteni a szerző nevét (például a forrásban, köszönetnyilvánító részben, súgóban).

Ne fektessünk energiát saját motor készítésébe!

Amikor játékprogramozásról esik szó, sokan 3D-, felületgeneráló stb. motorokra gondolnak. Egy kisebb projekt esetében értelmetlen ilyesmibe fogni, hiszen valószínűleg úgysem használnánk sziszifuszi munkával létrehozott motorjainkat. Ez akkor is igaz, ha valóban az újrafelhasználhatóság a cél – ehhez nem kell feltétlenül saját motor, hiszen ötletes megoldásainkat átvehetjük forrásszinten vagy függvénykönyvtárak formájában is. Igazából csak nagyobb cégek esetében van értelme a motorok fejlesztésének, hiszen ők több játékba is beleépítik ezeket. Természetesen senkit sem akarunk lebeszélni saját motor írásáról, amennyiben azért fogna hozzá, mert meg szeretne ismerkedni a tervezésének, kivitelezésének és tesztelésének szakaszaival.

A következő csapda, amit érdemes elkerülni:

Ne fordítsunk túl sok időt a különleges grafikai megoldásokra!

Bármennyire fájó, szembe kell nézni a valósággal: különleges megoldások terén nem tudjuk felvenni a versenyt az id Software méretű cégekkel. Célunk, hogy jól játszható, jópofa játékot hozzunk össze, nem pedig az, hogy megfőzzük a legújabb videokártyák processzorát! Ha mégsem így lenne, akkor játék helyett kezdjünk inkább (technológiai) demó írásába.

Összpontosítsunk a játékmenetre, kezelhetőségre, és törekedjünk arra, hogy a játékosoknak kellemes perceket szerezzünk. Számtalan olyan számítógépes játék létezik, aminek csodálatos grafikája van, de egyszerűen játszhatatlan vagy lapos a játékmenet – rossz esetben mindkettő együtt.

Egy utolsó tanács:

Fejezzük be, amit elkezdtünk, és legyünk igényesek!

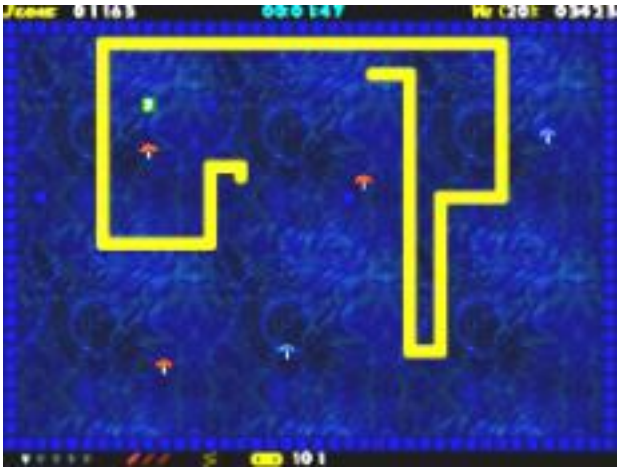
Ne hagyjuk félbe elkezdett játékunkat például azért, mert egy újabb játékötletünket akarjuk megvalósítani. Az sem mindegy, hogy néz ki a végeredmény: minimális követelmény, hogy rendelkezzen egy nyitó és egy záróképernyővel, tartalmazzon utasításokat a játék kezeléséhez, illetve a játékmenethez. Esetleg rendelkezzen néhány olyan szolgáltatással, amely kényelmesebbé és felhasználóbaráttá teszi a játékunk kezelését.

Következő alfejezetünkben megpróbáljuk összefoglalni, mik az ajánlott fokozatok, ha játékírásra adjuk a fejünket.

2.2. Hogyan fogjunk saját játék írásához?

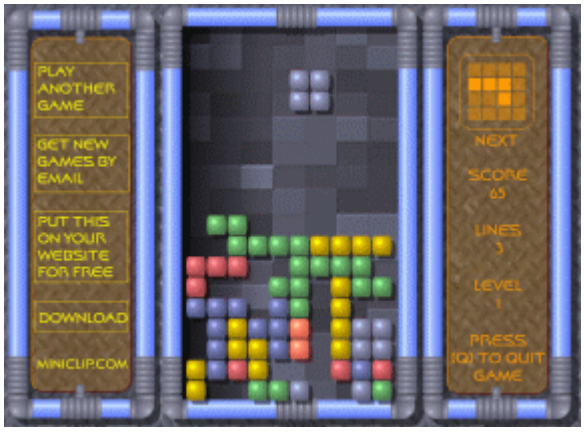
Ahhoz, hogy alkotókedvünk megmaradjon, és valóban elérjük a kitűzött célt, nemcsak az előzőekben felsorolt szempontokat, hanem a fokozatosság elvét is figyelembe kell vennünk. Játékírás esetén ez azt jelenti, hogy ne kezdjünk egyből egy 3D FPS- vagy kalandjáték írásába. Lássunk néhány konkrét példát arra, hogy mit értünk fokozatosság alatt.

Jó néhány olyan játék létezik, amely elsősorban egyszerűségével és játszhatóságával érdemelte ki népszerűségét. Klasszikusnak számít a „kígyós” játék, amelyben ügyesen kell vezetnünk az egyre növekedő állatkánkat (2.1. ábra).



2.1. ábra: DaSnake: egyszerű, mégis jópofa

Valószínűleg mindenki, aki játszott már számítógépes játékkal, ismeri a Tetrist, vagy találkozott valamilyen Tetris-klónnal (2.2. ábra).



2.2. ábra: Egy Tetris-klón

Egy Tetris-klón írása kezdésnek kifejezetten jó, mivel programozási szempontból gyakorlatilag mindent magában foglal, ami általában közös a játékokban:

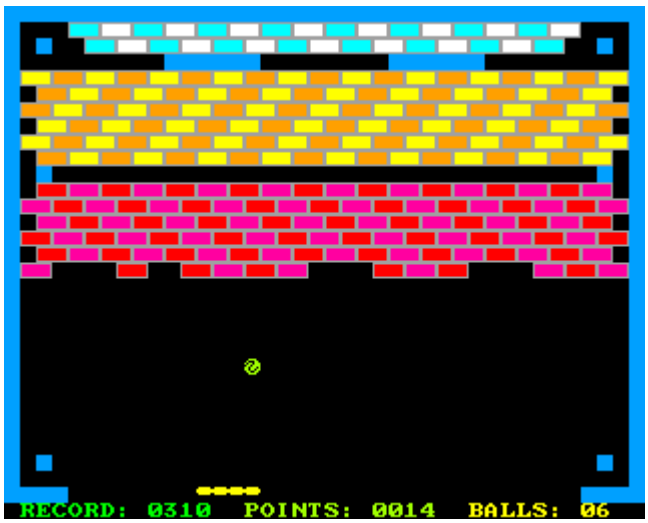
- fő ciklussal rendelkezik, amely folyamatosan végrehajtódik, míg fut a program;
- elvégzi a felhasználó által bevitt billentyűk beolvasását és feldolgozását;
- különböző grafikai műveleteket végez a játék elemeivel (forgatás, eltolás);
- kiértékeli az eredményt, nyilvántartja a magas pontszámot elérő játékosok listáját.

Egyszerűsége ellenére a Tetris élvezetes játék, amit nem lehet megunni. Nem elhanyagolható szempont az sem, hogy a grafikai elemekkel nem kell sokat bíbelődnünk, hiszen a lényegi elemeket négyzetekből lehet összeállítani, amit a legegyszerűbb rajzolóprogrammal is létrehozhatunk.

Már az első, egyszerű játék elkészítése igen tanulságos: egyrészt büszkeséggel tölti el az alkotót, másrészt a realitások talajára helyezi. A megszerzett tapasztalatok alapján fokozatosan javul az ítélőképessége, azaz egyre pontosabban meg tudja határozni az új projektek elkészítéséhez szükséges időt. Ez különben alapprobléma a szoftverfejlesztésben, és néha még a „profik” is súlyos becslési hibákat követnek el.

Megjegyzés: A „Tetris” elnevezés, sőt a „tris” szó is védett, ezért ügyeljünk arra, hogy ne használjuk játékunk elnevezésekor.

Valamelyest összetettebbek a Breakout-stílusú játékok, ugyanis itt megjelenik néhány újabb elem, mint az ütközések érzékelése, valamint egy kis fizika a labda ütőről történő visszapattanásánál (2.3. ábra).

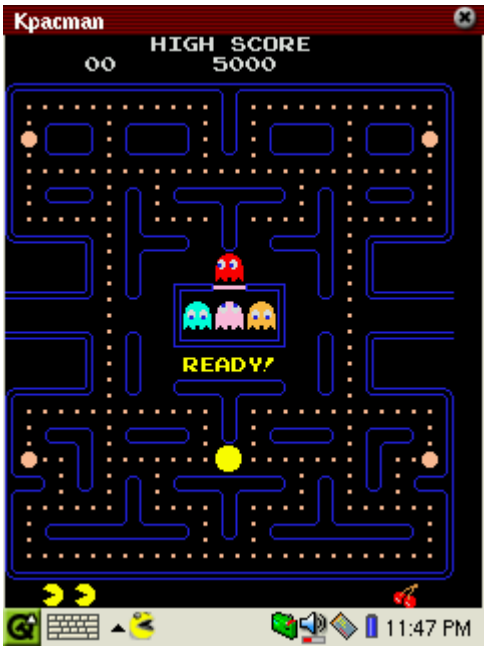


2.3. ábra: Ismerős Breakout-képernyő

Megjelennek a szintek is, amely újabb fejlődést jelent a Tetrishez képest (az utóbbinál ez a játék sebességének növelését jelenti az elért pontok vagy a játékidő függvényében). Minden több szinttel rendelkező, valamirevaló játékban ajánlott – vagy inkább kötelező – megadni az elért szint(ek) elmentésének, illetve visszaállításának lehetőségét; ez fontos szempont, ami nagyban befolyásolja a játék használhatóságát. Hiánya bosszantó, és elkedvetlenítheti a játékost, hiszen ki szeretné újra és újra átküzdeni magát a már lejátszott pályákon kizárólag azért, hogy onnan folytathassa, ahol abbahagyta. Vannak egyszerűbb alternatívák is, mint például az adott szinten felvillanó kód, amelynek begépelésével a játékos egyből az adott szintre jut. Összetettebb játékokban azonban ez nem kielégítő megoldás, hiszen megalapozott lehet az igény arra, hogy akár egy adott szinten belül a játékmenet pillanatnyi állapotát lehessen elmenteni (stratégiai vagy FPS-játékoknál ez alapkövetelmény).

Eddig két alapjátékmodellt mutattunk be, de még nem említettük a mesterséges intelligencia divatos fogalmát. Legyen szó alap- vagy – enyhe túlzással – a felhasználóéval vetélkedő MI-ről, jelenléte nagymértékben növeli a játékélményt. Természetesen itt is ügyelni kell az arányokra, ugyanis senki sem szeret állandóan veszíteni a „géppel” szemben.

A Pacman volt az első játék, ahol a mesterséges értelem legegyszerűbb formája megjelent (2.4. ábra).



2.4. ábra: Pacman

A játékost szellemek veszik üldözőbe, mindegyik egyedi stratégiával próbálja elvágni annak útját. Bár nem kell hozzá fuzzylogika, mégis programozási kihívásnak tekinthető.

Sokunk kedvence a Bomberman, amely tovább fokozza a Pacman izgalmait (2.5. ábra). Itt a küzdelem egy vagy több játékostárs, gépi ellenfelek, esetleg mindkét kategória ellen folyhat. A hanghatások és a poénos megoldások még szórakoztatóbbá teszik a játékot. Bár az Atomic Bomberman tartalmaz néhány hibát, idegeket és billentyűzetet nem kímélő játékélményben lehet részünk – kiváltképp, ha edzett Bomberman-játékosok csapnak össze.

Az úgynevezett side-scroller, vagyis a görgetős játékok kissé már kimentek a divatból, de azért ki ne emlékezne például Super Marióra és társaira. Az ilyen játékokban már több ellenfél, díszesebb és változatosabb háttérelemek jelennek meg, többféle mozgási lehetőségünk van.

Ezek után a manapság oly divatos 3D-műfaj kerülhet sorra. Azonban nem minden 3D, ami annak látszik: a Warkrafthoz, Starcrafthoz, Red Allerthez hasonló stragégiai játékok gyakorlatilag kétdimenziós elemekből (úgynevezett sprite-okból) állnak, amelyek háromdimenziós érzetet keltenek (2.6. ábra).

2.5. ábra: Reakcióidő-teszt Bomberman-módra

2.6. ábra: Starcraft: háromdimenziós látszat valódi 3D-modellek nélkül

Amint a PC-hardverek lehetővé tették a megfelelő minőségű és sebességű háromdimenziós képalkotást és animációt, egyre inkább divatba jöttek a háromdimenziós játékok, amelyek szó szerint új dimenziót adnak a játékélménynek. Eleinte talán túlzott reményeket is fűztek hozzá, olyannyira, hogy kizárólag a szebb, meggyőzőbb grafikára, kidolgozott modellekre összpontosítottak, viszont elhanyagolták a játékélményt.

Nem szabad elfelejteni, hogy a modellek, grafikák, textúrák elkészítése nagyon sok időnket rabolhatja el, főleg ha mindent egyedül akarunk csinálni. A 3D-játékok írását is kezdjük valami egyszerűbbel, például próbáljuk meg a már említett Tetrist átültetni három dimenzióba.

Még egy utolsó tanács: ne sértődjünk meg, ha egyesek keményen kritizálják művünket, amelyen annyit dolgoztunk. Leginkább azoktól ne számítsunk objektivitásra, akik kizárólag felhasználók, és nem konyítanak a programozáshoz. Egy fejlesztő sokkal megértőbb, mivel nagyjából átérzi a nehézségeket, értékeli a befektetett energiát. Hibáinkat természetesen javítsuk ki, azonban az alaptalan kritikák ne szegjék kedvünket.

2.3. A számítógépes játék főbb elemei

A játékprogram is „csak” egy szoftver, amely a többi programhoz hasonlóan bizonyos alkotóelemekből áll. Felépítését illetően azonban akadnak bizonyos sajátosságok, apróbb eltérések.

A következőkben egy játékprogram leegyszerűsített modelljét mutatjuk be. Modellünkkel egy játék főbb, általános elemeit szemléltetjük, amelyeket gyakorlatilag minden játéknak tartalmaznia kell.

Megjegyzés: Vegyük figyelembe, hogy most elvonatkoztatunk az operációs rendszer és a programozási nyelv sajátosságaitól, ezáltal a felsorolt fogalmak függetlenek ezektől.

1. Inicializálás

Programunk ezen eleme hajtódik végre először; az inicializálás során történik az erőforrások lefoglalása, objektumaink létrehozása, különböző állományok, grafikai elemek betöltése stb.

2. Főciklus

Gyakorlatilag a játék főciklusában történik minden, miután elindítjuk a játékot, és befejeződött az inicializálás. A főciklust a kilépés zárja le.

3. Irányítás

Játékosaink különböző beviteli eszközökkel – billentyűzet, egér, joystick, gamepad stb. – irányíthatják a játékot. A beviteli eszközök által szolgáltatott értékeket fel kell dolgozni, és annak megfelelően különböző tevékenységeket végrehajtani, megfelelő választ generálva. Ez utóbbi lehet például egy fegyver elsütése, a kamera körbeforgatása, vagy éppenséggel a beépített mesterséges intelligenciát utasíthatjuk megfelelő válaszra.

4. A játék logikájának végrehajtása

Kódmennyiség szempontjából általában ez a játék legméretesebb és legösszetettebb része, mivel itt történik minden, amit az eddigiek során nem említettünk meg: a jelenet megjelenítése, szinkronizálási feladatok végrehajtása, objektumaink mozgatása, a játék logikájának és – ha van – a mesterséges intelligenciának a végrehajtása, kiértékelés, mentési-visszaállítási műveletek stb.

5. Kilépés

Ez a pont lényeges és elmaradhatatlan eleme minden egyes alkalmazásnak. Az inicializálás során lefoglalt erőforrások felszabadítására mindenképpen illik odafigyelni, mielőtt visszatérünk az operációs rendszerhez.

Kissé leegyszerűsítve ugyan, de bemutattuk egy játékprogram felépítését és működési logikáját – végül is ez volt a célunk. Könyvünk későbbi fejezeteiben számos gyakorlati példával találkozunk majd, amelyekben alaposabban megvizsgálhatjuk az egyelőre csak elméletben bemutatott funkcionális egységeket, illetve tovább finomítjuk ezt az általános, felülnézeti képet.