Game Engine

Követelmény-specifikáció

Piller Imre

1. Általános leírás

A cél egy kétdimenziós játékok működtetéséhez, illetve megtervezéséhez használható környezet létrehozása. Ebbe tehát beletartozik a játék futtatásához szükséges programrész, illetve mind azok az egyéb eszközök, amelyek majd a játékok készítését segítik elő.

Kétdimenziós játékokból nagyon sokféle van. Ez esetben azokra kell gondolni, amelyekben a játékos egy karaktert irányít, és a térképet oldalról lehet látni. A játékmenet is hagyományosnak mondható, vagyis bizonyos pontokba kell eljutni, tárgyakat összeszedni, illetve legyőzni a különféle ellenfeleket.

Miért is van szükség ilyen játékmotorra ?

Már számos hasonló játékmotor van a piacon, illetve nyílt forráskódú változatok is elérhetőek. A következő pontokban össze van szedve, hogy mik azok a fő jellemzők, amiben ez majd több próbál lenni. Alapvetően egy játékmotorról van szó, nem pedig egy konkrét játékról, viszont nehéz teljesen általánosságban bemutatni a jellemzőket, ezért helyenként vannak utalások konkrét használati esetekre, amely csak a játékmotor lehetőségeinek bemutatására szolgálnak majd, nem pedig egy készítendő játékot írnak le.

Realisztikus, csontváz alapú grafika

A megjelenítés alapját a karakterek (általánosságban entitások) csontozata jelenti majd. A másik elterjedt megoldás az egyszerűbb *sprite* alapú megközelítés, vagyis amikor a karaktereket, mint külön rajzokat készítik el, majd a játékmotor ezeket rétegekben mozgatja. Ezekben a karakterek és a környezetük jól elkülöníthetőek, az ütközésvizsgálat is viszonylag egyszerűen megoldható, viszont az animálásukhoz meg kell rajzolni a lehetséges állapotaikat.

Ennél egy rugalmasabb megoldást jelent, ha a karakternek a részei külön készülnek el, majd ezeket egy csontozat fogja össze. Bizonyos értelemben ezek a külön részek is tekinthetők sprite-oknak, viszont nem ezek jelentik az entitások alapegységét. Ennél a megoldásnál a játék fizikájára már a karakterek létrehozásánál figyelni kell, a karakter mozgását befolyásoló tényezőknek hasonlóaknak kell lenniük, mint azoknak amelyek a karakter és a környezete viszonylatában vannak.

Problémát jelent egy karakter megfordítása. Sprite-ok esetében elegendő tükrözni az adott képet, viszont szélesebb karaktereknél ott is gondot jelenthet. Két dimenzióban ez a fajta forgatás kivitelezhetetlen, a háromdimenziós megjelenítés viszont felemássá teheti a játék összképét. Egy kompromisszumos megoldás az, hogy a csontozatnak bizonyos, két dimenzióban nem természetes mozgásokat is megengedünk, majd a karakter kirajzolását az így kapott pozíció függvényében végezzük. A játék ilyen értelemben nem lesz teljesen realisztikus, viszont egy kétdimenziós játékban, ahol a valóságoshoz hasonló karaktereket szeretnénk kezelni ez nem is lehetséges. A realisztikus megvalósítás elsődlegesen majd a játék fizikájára vonatkozik, illetve arra, hogy nem a klasszikus sprite-alapú megjelenítést használja.

Szerepjátékbeli elemek

Teljesen új műfajt kitalálni manapság már szinte lehetetlen, viszont vannak olyanok amelyekből eddig nagyon kevés készült. Platform játékokban időnként megjelennek szerepjáték (**R**ole **P**lay **G**ame) elemek, de nem annyira jelentősen mint akár csak a felülnézetes játékokban. A játékmotor majd az oldalnézetes akció RPG típusú játékok fejlesztéséhez lesz megtervezve. Szükséges lesz tehát, hogy a játékban:

- a karakterek fejlődéséhez tartozó információk külön kezelhetőek legyenek,
- legyen sok felszedhető tárgy (pl. fegyverek), amik azután már a karakterhez kell, hogy tartozzanak,
- előre megadható akár logikai feladványok, csak a megfelelő kulcsokkal nyíló ajtók és egyéb speciális elemek.

A játékmotornak segítenie kell azt, hogy a játékban az ellenfelek legyőzése inkább logikai és ügyességi feladat legyen, nem minthogy kinek sebez nagyobbat a fegyvere, és kinek van nagyobb pajzsa. Ehhez tartozik még az is, hogy a játéknak története is van, amit valamilyen formában közölni kell, például a beszélgetések alapján. Az RPG jelleg miatt egy, a hasonló játékokból már jól ismert felhasználói felületek létrehozása is szükséges lesz.

Mesterséges intelligencia

Az entitások mesterséges intelligenciájára az alábbi okok miatt van szükség:

- **Karaktermozgás**
  A fizikai megvalósításhoz hozzátartozik, hogy az entitások mozgása valahogy rendezett legyen. Ez megoldható lenne animációval is, aminél sajnos elég sok esetre külön ki kellene térni, és figyelmet kellene arra fordítani, hogy az ezek közötti váltások zökkenőmentesen menjenek végbe. E helyett az a megoldás tünik célszerűbbnek, hogy a karakter vezérlése úgy történjen, hogy az csak egy utasítást kap valamilyen feladat elvégzésére, majd ezt megpróbálja valahogyan megoldani. Ilyen lehet például az, hogy induljon el egy adott irányba. Ekkor tudnia kell, hogy éppen milyen pozícióban van, illetve azt, hogy hogyan próbálja elkerülni az akadályokat, ha egyáltalán lehetséges. Meg kell tudniuk oldani továbbá olyan feladatokat is, mint például ha elesnek, vagy megsérülnek.

- **Kooperáció az ellenfelek között**
  Azért, hogy élvezhető legyen a játék szükséges, hogy az ellenfelek között legyen valamilyen szintű együttműködés. A legyegyszerűbb eset az, hogy tudják, hogy kiket kell megtámadniuk, hogy reagálják le, ha egyikükkel történt valami, hogy próbáljanak egymásnak segíteni.

- **Segítő karakterek**
  Az előző mintájára meg kell oldani majd azt is, ha esetleg a játékosnak van szüksége segítségre egy feladat megoldásához. Ez annyiban különbözik az ellenfelek esetétől, hogy itt már lehetőséget kell adni arra, hogy a játékos valahogy befolyásolja a másik karaktert is, viszont az még közben önállóan is tudjon cselekedni.

Érdemes összeszedni azokat a dolgokat is, amelyek a játékmotor fejlesztésénél nem szerepelnek elsődleges célként. Ezekre a tervezésnél érdemes tekintettel lenni, hogy azok a későbbiekben is még egyszerűen megoldhatóak legyenek, viszont az első változat fejlesztésénél ezek még nem kulcsfontosságúak.

Többjátékos mód, online elérhetőség

Az RPG elemek miatt természetes elvárás, hogy a játék majd többjátékos módban, online is legyen játszható. Ezzel az a fő probléma, hogy a játék akció jellege miatt a szinkronizálás nagyon nehezen oldható meg. Több olyan problémát is felvet továbbá ez a lehetőség, amellyel már csak akkor érdemes foglalkozni, ha az offline változat stabilan működik.

2. A játékmotor részei

A játékmotor alatt többféle szoftverbeli megvalósítást is szoktak érteni. Ebben az esetben az alábbi fő részek tartoznak bele:

- A játék működéséhez szükséges program, amely betölti magát a játékot (pl.: térkép, karaktereket, zenét), megjeleníti azokat, illetve kezeli a felhasználói beavatkozásokat.

- A térképek, karakterek és egyéb erőforrások megtervezéséhez szükséges programok.

- A fejlesztéshez használt kisebb tesztprogramok.

Az utóbbi kettő nem tartozik szigorúan a játékmotorhoz, viszont a játék fejlesztéséhez mindenképp szükségesek, és így érdemes azokat a motorral együtt párhuzamosan fejleszteni, és később is együtt kezelni. Ezek alapján a készülő játékmotor az alábbi részekből fog állni előreláthatólag:

- **Game Library**
  A szükséges függvénykönyvtárak, amelyek majd a további programrészek alapjául szolgálnak.

- **Engine**
  A program, amely a játék egyes részeit kezeli; amiből később maga a játék lehet. Ideális esetben az egyes játékok fejlesztésénél elég ezt konfigurálni, megadni a szükséges erőforrások helyét, majd a motor beolvasva azokat el tudja indítani a játékot.

- **Material editor**
  Program a játék környezetének kialakításához szükséges elemi egységek tervezéséhez. Itt az anyag arra vonatkozik, hogy az a játékban milyen tulajdonságokkal bír majd, nem korlátozva csupán a fényvisszaverődési jellemzőkre.

- **Entity Editor**
  A játékban az alapegységet az entitások jelentik. Ezek már magukban is több olyan tulajdonsággal bírnak, ami miatt külön egy tervezőprogram kell nekik. A viselkedést is itt kell majd megadni, illetve bizonyos szintű tesztelési lehetőségre is szükség lehet.

- **Map editor**
  A környezet megtervezéséhez szükséges program. Ebben kell majd az egyes entitásokból felépíteni a térképet, illetve meghatározni azok viszonyát. Szükség esetén ehhez tartozhat még egy **Champaign Editor** is, amelyel már teljes küldetéseket fel össze lehetne állítani.

- **GUI editor**
  Egy egyszerűbb program, amelyel meg lehet tervezni a játék felhasználói felületét. (Ebbe csak a felület létrehozásához feltétlenül szükséges elemek kell hogy beletartozzanak, mivel egy teljes, felhasználói felület kezeléséhez szükséges rendszer kifejlesztése nehézkes lenne, és nem is tartozik az elsődleges feladatok közé.)

A tervezés során még újabb eszközök szóbajöhetnek, ezek csak azok, amelyek ebben a kezdeti stádiumban szükségesnek látszanak.

3. Szerkesztőeszközökkel szembeni elvárások

A szerkesztőeszközök célja, hogy lehetőleg a játékot úgy lehessen fejleszteni az adott játékmotorhoz, hogy ne kelljen a játékmotor kódjába belenyúlni, azt csak mint egy különálló komponenst lehessen kezelni. A szerkesztőknek grafikus felületet kell majd biztosítaniuk az egyes elemek megtervezéséhez, viszont emellett (főleg az entitások bonyolultabb beállításai miatt) szükség lesz helyenként arra, hogy szkripteket is megadjon a játék fejlesztője. (A szkriptnyelv egyelőre még nem rögzített, mivel a tervezés során dől el, hogy melyik nyelv is lenne hatékonyabb. Új szkriptnyelv fejlesztése megintcsak sok időt vonna el a feladattól, ezért valószínűleg *Python*, *JavaScript* esetleg *LUA* szkriptelési lehetőség fog majd belekerülni az eszközökbe.)

A képek elkészítéséhez nem szükséges külön eszközt biztosítani, mivel ilyenek már szabadon is hozzáférhetőek. Fontos viszont, hogy a játékmotor minél inkább támogassa majd a szabványos formátumokat, emiatt a képek kezelése *PNG*, a hangok kezelése pedig *WAV* illetve *MP3* fájlok formájában fog történni.

A programok a térképek és egyéb modellek kisebb méretének érdekében, illetve a logikai összetartozást segítendő azokat a szintén szabványos *ZIP* formátumban fogja tömöríteni.

4. Eszközök a fejlesztéshez

A játékmotor egyik célja, hogy egy absztrakciós réteget biztosítson, és a játék fejlesztőjének már csak a játék megjelenésével és logikájával kelljen foglalkozni, nem pedig a megjelenítéssel vagy például a fájlok betöltésével. Főleg bővítési és kompatibilitási kérdések miatt azonban érdemes szót ejteni arról, hogy milyen környezetben fog elkészülni a szoftver.

A motor *C++* programozási nyelven fog készülni, mivel így könnyebben lehet illeszteni a már rendelkezésre álló komponensekkel.

A megjelenítés az *OpenGL* függvénykönyvtár segítségével fog történni. Kétdimenziós játékok fejlesztéséhez erre nem feltétlenül lenne szükség, de mivel a megjelenítéshez szükséges transzformációk ezzel kényelmesen megoldhatók, illetve ma már gyakorlatilag az összes videókártya támogatja, ezért mégis ez tünik a legjobb választásnak.

Azért, hogy a játékmotort egyszerűbben át lehessen vinni a különféle platformokra, ezért az OpenGL inicializálására, a felhasználói események kezelésére, illetve a képek és hangok belöltéséhez a **S**imple **D**irectMedia **L**ayer kerül felhasználásra.