3. tétel

Ebben a tételben a játékfejlesztéssel járó problémákat, a főbb verziókezelő rendszereket, illetve a programváltozatok frissítését fogom kifejteni.

Saját fejlesztése során egy programozó csapatnak rengeteg problémával kell szembenéznie. A legelső, hogy a csapat nem megfelelő fejlesztői környezetet választ, az általuk elképzelt játékuknak. Minden motornak van egy bizonyos korlátozása, amit már nem képes végrehajtani. A második probléma lehet a játék helyes optimalizációja, hogy a kód minél kevesebb erőforrást használjon, de emellett jól használja a ki a hardver adta lehetőségeket. A harmadik ilyen probléma a kódban elkövetett hibák, más néven BUGOK. Nem ezeknek a javítása okoz problémát a programozóknak, hanem a megkeresésük a több 100.000 sorból.

Játékot keretrendszerrel vagy anélkül tudják fejleszteni a programozók. Manapság már kevésbe népszerű a keretrendszer nélküli programozás, annak a bonyolultsága miatt. Ilyenkor a játékot a nulláról kell felépíteni, főképp alacsonyszintű programozási nyelvet használva, mint például C++. A megjelenítéséhez pedig API-kat használnak, megkönnyítik ezzel a programozók feladatát. Ezek a programrendszer lehet pl. OpenGL, segít a 3D-s grafikában. 0-ról kell felépíteni a programot, így a megejtett hibákat a kódban könnyebben kitudják javítani. Kezdő játékfejlesztőknek ez a megoldás általában hátrány lehet, a fejlesztési folyamat sokkal időigénysebbé válik.

Keretrendszereket a programozók a 2000 körüli években kezdték el használni. Két részre tudjuk szedni ezeket. Grafikusmotor, ami csak a képi megjelenítésben segíti a programozót. A másik pedig a játékmotor, ami nagyobb eszközteret biztosít nekik. Az utóbbi a népszerűbb, mivel tartalmaz különböző alrendszereket, például a bevilágítás, fizika, mesterséges intelligencia. Hátránya a magas ár, kevesebb rugalmasság, szabadság és nehezebb kijavítani a hibákat benne, viszont gyorsabb a bennük lévő fejlesztés, mint a keretrendszer nélküliben. A mai legismertebb motorok a Unity, Unreal Engine, CryEngine, Source.

A legelső fázis a játékfejlesztés során az ötletelés és az alapötlet pontos meghatározása. Ez után következik, hogy milyen platforma szeretnék fejleszteni a játékot. Lehet ez PC, web, konzol, telefon és főképp ez fogja meghatározni a játékmotort is. Ha ezekből minden teljesült, a fejlesztők nekiállhatnak egy prototípust gyártani, ami tartalmazza az alapmechanikákat. Ilyenkor még a játéknak nincsen designja, csak egyszerű szürke grafikája van. Az a célja, hogy kialakuljon a végső design, a játékos cselekedetei, illetve a játékszabályok. Ilyenkor egy elég hosszú dokumentumot kell készíteni a fejlesztésről, leírni a játékmenetet és milyen alapvető változásokat kéne meghozni. Következő fázis a leghosszabb, úgy hívják gyártás. Ilyenkor már nem történik nagy változtatás, a fejlesztők megkapják a feladatukat és annak a határidejét. Részletes dokumentum alapján megcsinálják a grafikai asseteket és ezzel párhuzamosan pedig előhívják a játékmenetet, szép lassan elkészül a játék. Ennél a résznél a játékosok már láthatnak bemutatót a közelgő alkotásról és ilyenkor kezdi marketinget a kiadó. A negyedik fázis a karbantartás, a

megjelenést követő hibák javítása. A fejlesztők akár későbbi kiegészítőket is csinálhatnak a meglévő játékmenethez, úgynevezett DLC-ket.

A játékszoftverek életciklusát nagyban meghatározza a játéksok érdeklődése az adott játékkal kapcsolatban. Egy átlagos játékot a kiadó akár 2-3 évig is támogat, ezt főképp az előbb említett dolog határoz meg. Azért is kell ennyi idő, hogy a fejlesztés és a marketing árát teljesen visszahozza a játék és profitálni is tudjon a kiadó. Rengeteg ember használ kalózjátékokat, amivel megkárosítja a bevételtől.

A frissítések a felhasználói visszajelzésekből jönnek létre. A játékosok tudomást adnak a fejlesztőknek a hibákról. Ilyenkor a csapat megpróbálják kijavítani és biztosítani a megfelelő játékélményt a játékosok számára. Frissítések lehetnek még játékmenet újítások, hogy minél több ideig lehessen fent tartani az érdeklődést a program körül.