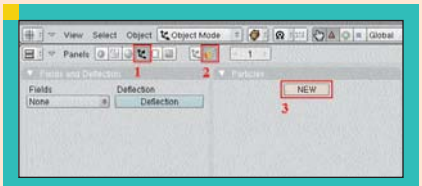
A Blender használata (10. rész)
Finomságok

Az elmúlt egy évben kilencszer írtam bevezetőt. Hogy őszinte legyek, mind a kilencszer bajban voltam és most sincs könnyebb dolgom. A mostani a tizedik, és minden valószínűséggel a legutolsó cikkem a Blender sorozatban, ami szintén csak nehezíti a dolgom.

Afféle búcsúzás képen megpróbáltam néhány igazi csemegét összegyűjteni, amelyekre mindezidáig nem jutott idő, de úgy érzem túlságosan sokról maradnánk le.
Néhány egyszerű példán keresztül bemutatom a ***Blender*** „részecske motorját", amivel könnyen tudunk tűzijátékot, esőt, füvet, stb. modellezni, majd a hasonlóan hasznos folyadék szimulációval is megismerkedhetünk. A cikk második felében betekintést nyerhetünk a ***Blender Python*** nyelven programozható felületébe, megtudjuk mi az amiben a ***Blender*** messze túlszárnyalja a „professzionális", fizetős (és egyplatformos) alkalmazásokat, és végül vetünk egy pillantást a ***Blenderbe*** épített játékmotorra is, amivel a programozó lelkületűek a ***Blender*** adottságait kihasználva gyönyörű játékokat írhatnak.

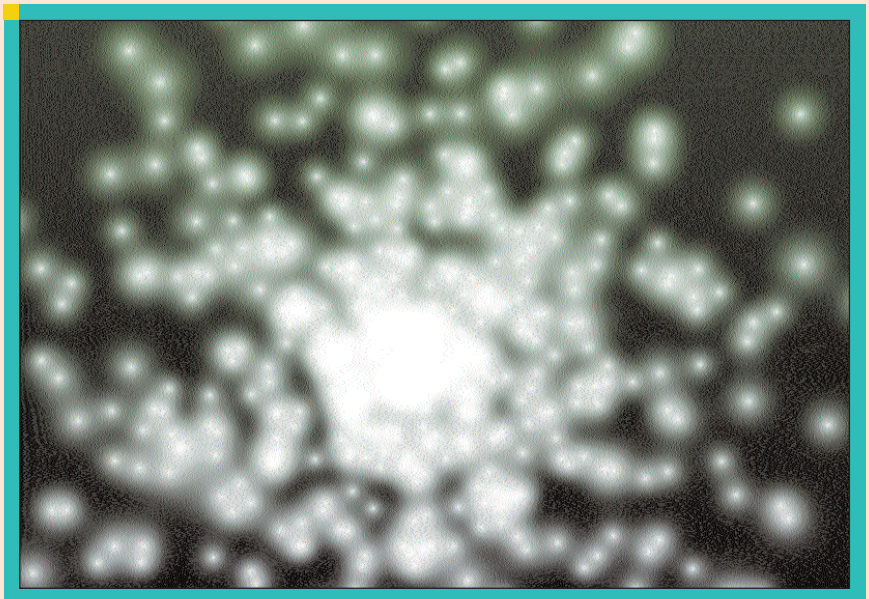
Particle engine

A név kicsit csalóka, ugyanis a ***particle*** szó magyarul részecskét jelent, de a ***Blenderben*** ez tulajdonképpen bármilyen objektum lehet. Az alapvető felállás szerint egy általunk megadott felületről induló részecskékre ható fizikai erőhatásokat szimulálhatjuk, így gyönyörű effektekkel bővíthetjük az animációnkat. Az egész akkor válik igazán gyönyörűvé, mikor rájövünk, hogy a részecskének nevezett mozgó objektum gyakorlatilag bármi lehet (tényleg bármi). Mindezt kombinálva a ránézésre is sok beállítási opcióval (*2. ábra*), az eredmény tényleg csak a fantáziánkon múlik. Legelső lépésként létre kell hoznunk egy objektumot, ami létrehozza a részecskéket. Ő lesz az emitter. Tetszőleges mesh lehet, de ajánlott mindig a célnak legmegfelelőbbet használni. Ha például egy szörnyecskének szeretnénk bundát varázsolni (igen, olyat is lehet), maga a szörny lehet a részecskéket kibocsátó objektum, de egyszerűbb esetekben egy ***plane*** (egy egyszerű négyszöglap) is megteszi. Hogy a részecskék a mesh felületén (face), vagy a vertexeken (vagy esetleg mindkettőn) keletkeznek, a ***Particles*** nevű panelen ***„From"*** részénél a ***Verts***

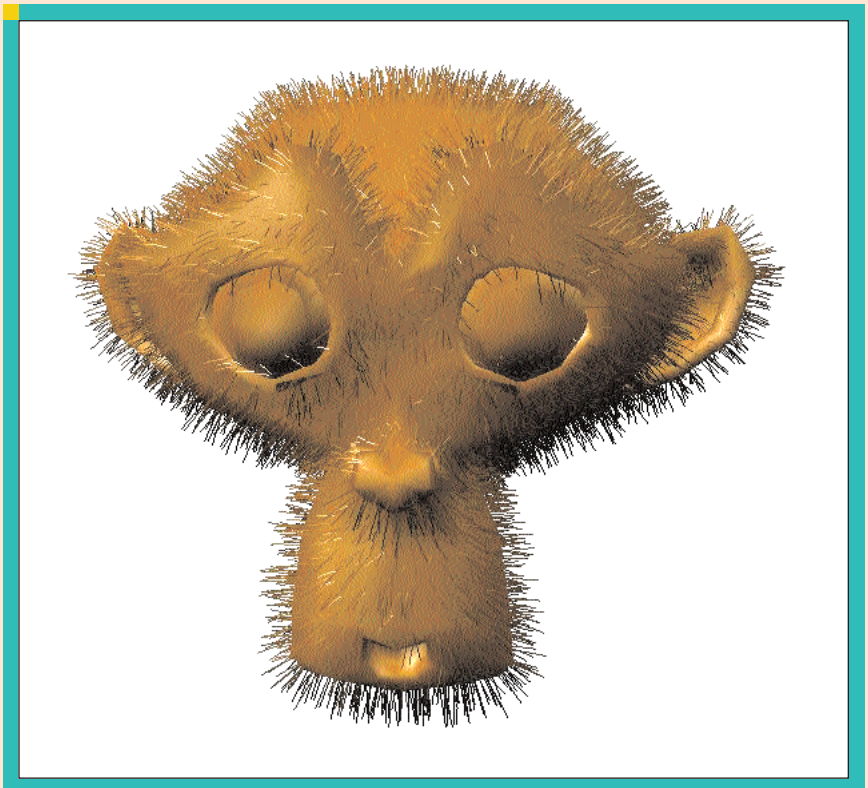


1. ábra Particle Engine bekapcsolása

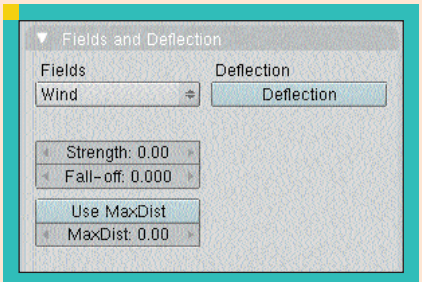
2. ábra Beállítások (Particles, Particle Motion)



3. ábra Példa részecske szimulációra



4. ábra Static particles



5. ábra Fields and Deflection panel

illetve ***Faces*** gombokkal állíthatjuk be. Ugyanitt beállíthatjuk az is, hogy vertexek esetén melyik ***Vertex Group***-ot (vertexek egy csoportja) kívánjuk használni. Ha nem adunk meg csoportot, akkor az összes vertex emitterként fog működni.

Mivel a részecskék születnek, élnek, majd halnak, ezért meg kell adnunk hogy pontosan mikor és mennyi részecske keletkezzen, és azok milyen sokáig létezzenek. Az *„Emit”* felirat alatt található ***Amount*** gombbal az összesen megjelenő részecskék számát szabályozhatjuk. Ezek az animációban a ***Start*** és ***End*** beállításoknál megadott idő alatt keletkeznek, és a ***Life*** gombon beállított ideig léteznek (*3. ábra*). Alapállapotban, ha egy mesh-t emitterré avatunk, az a mesh a renderelt animáción nem látszik. Ha mégis ezt szeretnénk, a ***Display*** felirat alatt kapcsoljuk be a ***Mesh*** gombot. Ugyanitt található a material nevű beállítás, ami kicsit trükkösen működik. A ***Mesh*** anyagbeállításainál több csoportot hozhatunk létre, és megadhatjuk, hogy az egyes face-ek melyik csoporthoz tartozzanak. Itt ugyanezekből az anyagokból választhatunk ki egyet, ami majd a részecskékre vonatkozik. A ***Static*** beállítással (mint a neve is mutatja) statikus részecskéket hozhatunk létre. Ezek a részecskék a *4. ábrán* látható módon helyezkednek el, és a ***Vector*** gombbal kombinálva kiválóan alkalmasak szőr vagy fű szimulálására.

Részecskék mozgása

Megpróbáltuk, de nem működik? Nem mozognak a részecskék? Nos, ez csupán annyit jelent, hogy a ***Blender*** a valóságnak megfelelően működik. ***Newton*** törvénye kimondja, hogy minden test (a részecskét tekintsük pontszerű testnek) nyugalomban marad, vagy egyenes vonalú egyenletes mozgást végez mindaddig, amíg egy másik test vagy erő nem hat rá. Létrejövő részecskéinknek nincs kezdeti sebessége, és mivel nem hat rájuk semmiféle erőhatás, ezért egyáltalán nem mozognak.

Sok lehetőségünk van, hogy a részecskéket mozgásra bírjuk. A ***Particle Motion*** panelen kezdősebességet adhatunk a részecskéknek és statikus erőhatásokat (például gravitáció) állíthatunk be, illetve létrehozhatunk különböző mezőket, amik dinamikusan hatnak a részecskékre.

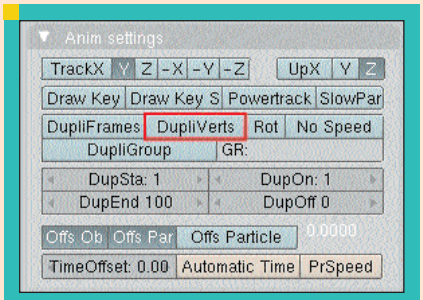
A statikus beállításokat a ***Particle Motion*** panelen (*2. ábra*) végezhetjük el. A ***Force*** résznél beállíthatjuk a részecskékre ható X, Y és Z irányú erőket, a ***Velocity*** résznél pedig a részecskék kezdősebességét, és kezdő haladási irányát. A ***Normal a*** részecskék indulási sebessége. Nagyobb szám nagyobb sebességet jelent. Ha azt szeretnénk, hogy adott irányba gyorsabban mozogjanak a részecskék, használhatunk egy textúrát is, ami megadja az indulási sebességet. A kezdő haladási irány általában az emitter középpontjából kifelé mutató vektor vagy a face normálvektora, de ha unalmasnak találjuk, a ***Random*** érték állításával véletlenszerű irányba indíthatunk részecskéket. Minél nagyobb ez az érték, a kezdő haladási irány annál jobbal eltérhet a normálistól.

Dinamikus erőhatások

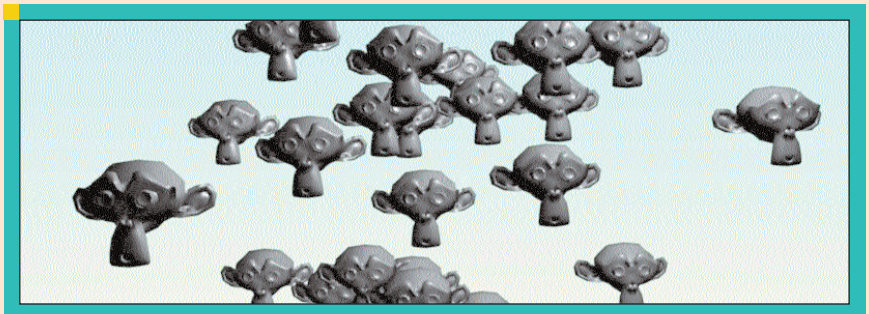
A statikus, mindig ugyanazt az útvonalat bejáró részecskék unalmasak. A ***Random*** beállítással javíthatunk ezen a hiányosságon, de az igazi megoldás az úgynevezett mezők (***fields***) használata. ***Blenderben*** ez tetszőleges objektumot jelenthet. Ezek beállításait a ***Fields and Deflection*** panelen végezhetjük el. A szabadságot az adja, hogy ezeket a mezőket objektumok szimbolizálják, vagyis animálhatók. Így hozhatunk létre például változó erősségű és irányú szelet, ami változó irányba fújja a füstöt vagy a felhőinket.

DupliVerts

Az előbb azt állítottam, hogy részecske bármi lehet, ennek ellenére még mindig csak pontoknál és vonalaknál tartunk. A ***DupliVerts*** opció segítségével azonban akár angyalkák is repkedhetnek a képernyőn részecskék helyett. Semmi más dolgunk nincs, mint elkészíteni a kívánt objektumot, szülőnek kiválasztani az emittert és ez utóbbi objektumon bekapcsolni a ***DupliVerts*** opciót a *6. ábrán* látható módon.



6. ábra DupliVerts



7. ábra DupliVerts (működés közben)

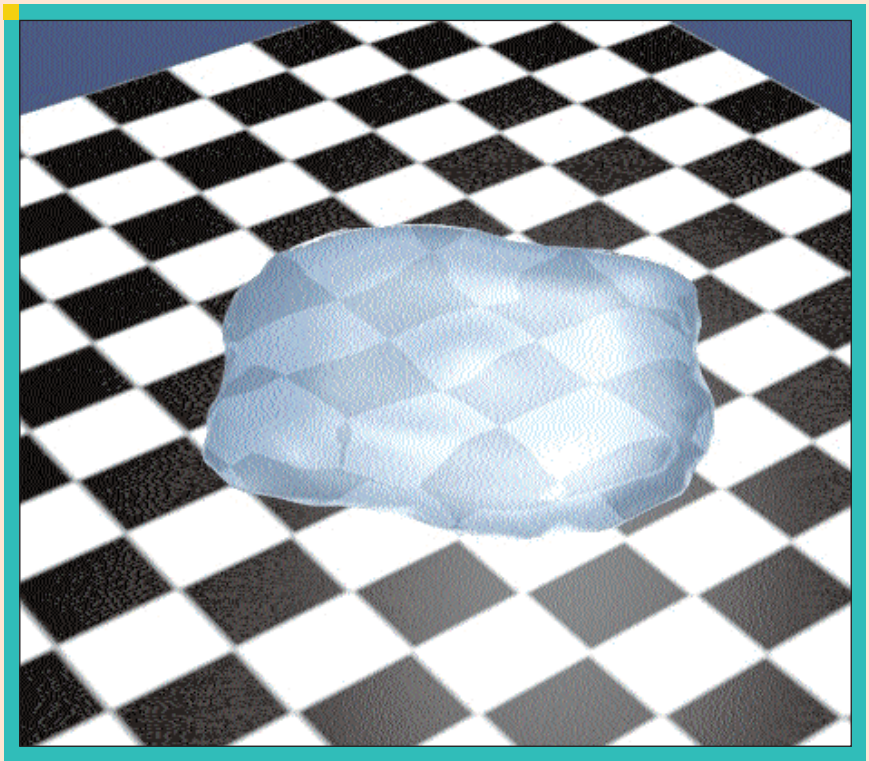
Fluid Simulation

A 2.4-es verzió újdonsága a ***Fluid Simulation***, amely kiválóan alkalmas sör,bor,víz,vér és hasonló folyékony állagú anyagok szimulációjára, és ami azt illeti mindezt egész ügyesen csinálja. Az egyetlen negatívum az ehhez szükséges memória és ***CPU*** idő, ugyanis egy összetettebb folyadék-animáció kiszámolása legalább akkora gépigényt igényel, mint egy bonyolultabb raytracing. De lehet hogy csak én számítógépem maradi, és elkelne egy kis bővítés.
A ***Fluid Simulation*** panelen (*8. ábra*) összesen öt gomb közül választhatunk. Az itt kiválasztott opció határozza majd meg, hogy az épp kijelölt mesh milyen szerepet fog betölteni a szimuláció során.
A legszükségesebb a ***Domain***.
A ***Domainnek*** kikiáltott objektum (legtöbbször egy kocka) mintegy behatárolja a szimuláció színterét, így ami a ***domain*** objektumon kívül van, egyáltalán nem vesz részt a szimulációban. Ez maga után vonja azt is, hogy minden bizonnyal a ***Domain*** objektum lesz a legnagyobb.
A ***Fluid***-ként megjelölt objektum(ok) fogják betölteni a folyadék szerepét.
A szimuláció kezdetekor egész egyszerűen elkezdenek vízként viselkedni.
Az ***Obstacle*** magyarra fordítva valamiféle akadályt jelent. A mi esetünkben olyan objektumot, amin a víz nem, vagy csak részben folyik át.
A fenti háromféle objektummal már neki is állhatnánk pancsolni, de van még két beállítási mód, ami sokat segíthet. Az ***Inflow***-nak titulált objektum folyadékforrás – vizet hoz létre –, az ***Outflow*** pedig afféle lefolyóként funkcionál, vagyis vizet von el a szimulációból.
Minden szimulációban résztvevő objektum beállítását külön szabályozhatjuk, továbbá a szimuláció részletességét, a folyadék fizikai viselkedését (víz, olaj, stb) és a gravitációt is mi állíthatjuk be.
A szimulációt a ***Domain*** objektum ***BAKE*** gombjával indíthatjuk el. Ilyenkor a ***Blender*** a megadott ideig próbálja fizikailag modellezni a folyadék mozgását. Mindeközben a jobb felső sarokban láthatjuk, pontosan hányadik frame-nél tart a szimuláció.
Nincs más dolgunk, mint hátradőlni és várni, vagy erre az időre beidőzíteni az ebédet (mint én tettem), ugyanis a szimuláció frame-enként egy úgy mesh-t hoz létre, ami majd a folyadékot reprezentálja és ez bizony eltarthat egy darabig…

Soft Body

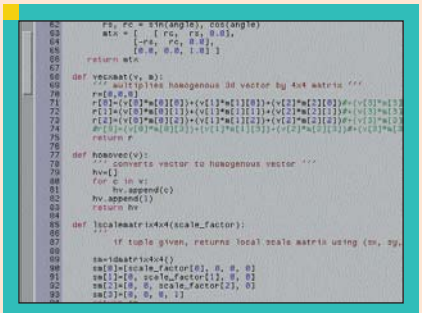
A ***Soft Body*** a ***2.37***-es verziótól található meg a ***Blenderben***. A puha, rugalmas anyagok fizikai viselkedését modellezi úgy, hogy animáció közben kiszámolja

8. ábra Fluid Simulation panel



9. ábra Fluid Simulation

10. ábra Soft Body



11. ábra A Blender python editora

a vertexekre ható fizikai hatásokat, és aszerint mozgatja el azokat.
Az *10. ábrán* a ***Soft Body*** panelt láthatjuk. A felső gombokkal állíthatjuk be a szimuláció fizikai tulajdonságait.
A ***Fraction*** az általános közegellenállást, a ***Mass*** a tömeget (nagyobb tömeg lassabban mozog) a ***Grav*** pedig a -Z irányba ható gravitációs erőt jelenti. A ***Speed*** beállítással a szimuláció sebességét, az ***ErrorLimit***-el pedíg a precizitást (kisebb szám, nagyobb precizitást, de több számolást jelent) állíthatjuk be.
Mivel semmi sem nyúlik akármeddig, ezért kell bizonyos korlátozásokat bevezetni. A ***Goal*** gomb bekapcsolásával, a rugalmas anyagokhoz híven minden részecske igyekszik a fizikai hatások ellenére visszatérni az animáció által meghatározott pozícióba.
Megadhatunk egy ***Vertex Groupot***, amire nem hat a fizika, csak az animáció, ilyenkor a többi vertex ehhez a csoporthoz igyekszik igazodni.
További korlátozásokat jelentenek a mesh élei. Ezek mint a rugalmas gumikötelek, próbálják összetartani a vertexeket. Ráadásul ezeknek a beállításait is szabályozhatjuk.
A ***Blender*** egyetlen hiányossága, hogy a ***Soft Body***-t csak mesh-re lehet alkalmazni. Vertexekkel dolgozik, így nem lehet statikus részecskéket sem animálni (a mozgó szőrszálakat is máshogy kell megoldani). Ettől az apróságtól eltekintve azonban a ***Soft Body*** egy igencsak kezes apróság, persze csak a megfelelő kezekben.

Játék a kígyóval

A ***Python*** névre hallgató találmány egy nagyon jól használható szkript nyelvet takar. Legfőbb jellemzői az objektum orientáltság, különböző platformok támogatása, a java-hoz hasonlóan működő bájtkódos fordítás, az elegáns szintaktika és a bővíthetőség. Ezek a tulajdonságai tették elterjedté, és emiatt választották a ***Blender*** szkript nyelvévé.
A legtöbb professzionális, kereskedelmi ***3D*** animációs program rendelkezik valamilyen saját szkriptnyelvvel, de mindnek korlátozottak a lehetőségei. Azt hiszem nem állítok valótlant, ha azt mondom: a ***Blender Python API*** bizony fényévekkel megelőzi a versenytársakat, és a folyamatos fejlesztések révén a lehetőségek száma csak nőni fog.
A ***Blender*** minden indításkor ellenőrzi, hogy megtalálható-e a számítógépünkön a ***Python*** futtatókörnyezet. Ennek hiányát vagy meglétét közli, de a program futtatását nem teszi ettől függővé (nélküle is fut), azonban a ***Python*** futtatókörnyezet nélkül nem használhatjuk ki a ***Blenderben*** rejlő rejtett lehetőségeket. A ***Blender*** biztosít ugyanis számunkra egy programozható függvénykönyvtárat, aminek segítségével szinte bármit megtehetünk.
Objektumokat, mesheket hozhatunk létre, azok beállításait tetszőlegesen megváltoztathatjuk, mozgathatjuk őket, stb. mindeközben pedig saját készítésű felhasználói felülettel tarthatjuk a kapcsolatot a felhasználóval.
Játékot írunk, és saját formátumba szeretnénk exportálni a modelleket? A ***Python API***-val probléma nélkül megoldható. Esetleg importálni szeretnénk egy szöveges formában tárolt ***motion capture*** animációt? Semmi akadálya.
A ***Pythonban*** írt bővítményekhez saját felhasználói felületet alakíthatunk ki, így az együgyű felhasználók is használni tudják. Számos, a ***Python API***-ra textúra illetve mesh generátort tölthetünk le az internetről, amik könnyebbé (de legalábbis élvezetesebbé) teszik a munkánkat.

„Ha már a kezünkben van egy jól használható 3Ds függvénykönyvtár, miért ne használhatnánk fel interaktív alkalmazások írására?"

Valami ilyesmi gondolat motoszkálhatott a fejlesztők fejében, amikor megalkották a ***Blender*** játékmotorját. Ennek segítségével valós időben renderelt interaktív játékokat hozhatunk létre a ***Blender*** grafikai lehetőségeit felhasználva. Természetesen figyelembe kell vennünk a számítógép sebességét, ugyanis a másodpercenkénti 25-30 kép megjelenítéséhez ugyanennyi renderelésre is van szükség, ami eleve kizárja erőforrásigényes effektek és raytracing alkalmazását.

Végszó

A ***Blender*** egy nagyon sokrétű, összetett alkalmazás. Egyetlen cikksorozatban nem lehet bemutatni minden részletét, és ha lehetne is, a sorozat elkészültekor már elavult lenne. A nyílt forráskódú programokhoz híven a ***Blender*** folyamatosan fejlődik, minden verzióban találhatóak újdonságok és sajnos néha hibák is. Mindezekkel azonban együtt lehet élni és amint azt az ***Orange Project*** is bebizonyította, a ***Blender*** alkalmas professzionális munkára, és tudásban felveszi a versenyt a versenytársaival.

Szalai András
(sly87@freestart.hu)

Jelenleg középiskolába jár, ahol informatikát tanul. Jövőre érettségizik. Hobbija a programozás és a biztonságtechnika, és a továbbtanulási szándékai is ilyen irányúak.

KAPCSOLÓDÓ CÍMEK

A Blender hivatalon honlapja:
➲ http://www.blender.org

Python scriptek:
➲ http://www.blender.org/cms/Python_Scripts.3.0.html

További információk:
➲ http://mediawiki.blender.org/