© Kiskapu Kft. Minden jog fenntartva

Játékmotor a Blenderben (1. rész)

Ismét a Blender képességeit fogom ismertetni, az elmúlt egy hónapban azonban alkalmam volt jobban átgondolni a tanmenetet. Ezért ebben a részben a játékmotorban használható érzékelőket (Sensors) és végrehajtó szerveket egyesével mutatom be, megalapozva a további játékkészítést.

Folytassuk a tanulást az elméleti anyag elsajátításával – a szükséges ismeretek birtokában már könnyebben érthető lesz játékunk elkészítésének a módja is. Mivel azonban a tárgyakhoz rendelhető változókat már az előző részben megismertük, folytassuk a sort a különféle érzékelőkkel!

A Blender játékmotorja tizenegyféle érzékelőt kínál, ezekből természetesen tetszőleges számút rendelhetünk egy-egy tárgyhoz. Az első ilyen az `Always` nevet viseli, működése szerint a kimenetén meghatározott időközönként jelzést ad. Minden érzékelőnek három olyan alapvető tulajdonsága van, amelyekkel beállíthatjuk a kimenet viselkedését. Az egyik ilyen tulajdonság az ***Active***, amelyet az érzékelő neve alatt található két kapcsolóval választhatunk ki. Értéke `True` vagy `False` lehet. Ezzel határozhatjuk meg, hogy amikor az érzékelő kimenete az alapállapottól eltérő lesz, annak logikai értéke igaz vagy hamis legyen. Fontossága abban rejlik, hogy minden érzékelő kimenete egy feltételt vizsgáló csomópont bemenetéhez kapcsolódik, ahol a feltétel egy logikai vizsgálat is lehet. A másik jellemző tulajdonság a – ***freq:***, vagyis az érzékelő működési frekvenciája. Ezzel határozhatjuk meg, hogy a játékmotor milyen gyakran vizsgálja az adott csomópont feltételeinek a teljesülését. Ezt az értéket 1/50-ed másodpercben mérve kell megadni, és a beállítására szolgáló mező az érzékelő neve alatt középen található. A harmadik alapvető tulajdonság pedig a név alatt jobbra található ***Inv***, amellyel meghatározhatjuk, hogy a kimenet állapota a feltételek teljesülésekor vagy azok nem teljesülésekor váltson-e aktív jelszintre.

A `Keyboard` érzékelővel, miként a neve is mutatja, a billentyűzet eseményeit figyelhetjük – az alábbi beállításokat figyelembe véve. A ***Key*** az elsődleges billentyű, amelyre az érzékelő válaszolni fog. A ***Hold*** felirat mellett található két mezőben adhatjuk meg azt a két módosító billentyűt, ami a jelzés kiváltásához elengedhetetlen. Természetesen amennyiben nem bonyolult billentyűkombinációkban gondolkodunk, ezek megadása nem kötelező. Ezeket a mezőket könnyedén beállíthatjuk, ha az egérrel a mezőre kattintunk, ezután pedig a ***Press a key*** felirat megjelenésekor lenyomjuk a kívánt billentyűt. Módosító billentyű lehet a bal és jobb oldali Ctrl, az Alt, a Shift és természetesen bármelyik másik billentyű is, de célszerű a fentiek közül választani, mert ezek eredeti működése is a többi billentyű működési módjának a megváltoztatása. A ***LogToggle*** felirat melletti mezőben beállíthatjuk, hogy a megadott billentyű lenyomásakor valamilyen szöveges üzenetet is átadjunk a ***Target*** mezőben megadott tárgynak.

A `Mouse` érzékelővel az egér eseményeire válaszolhatunk a játékban. Kevés beállítási lehetősége közül az első három az egérgombokra vonatkozik, míg a ***Movement*** az egér mozgatásakor hozza működésbe az érzékelő kimenetét, a ***Mouse Over*** pedig akkor, ha az egérmutató a tárgy fölé kerül. Ezt az utóbbit csak abban az esetben tudjuk majd kipróbálni, amikor a Blenderrel futtatható állományt készítünk a játékból.

A `Collision` típusú érzékelő alkalmas rá, hogy egy meghatározott tulajdonsággal vagy anyaggal rendelkező tárggyal való érintkezést észleljünk a játékban. Az ***M/P*** kapcsolóval állíthatjuk be, hogy tulajdonságot vagy mintázatot szeretnénk-e figyelni. Az érzékelő akkor fog jelzést adni, amikor a kijelölt tárgy azzal a meghatározott tulajdonsággal vagy anyaggal rendelkező tárggyal érintkezik, amelyet itt beállítottunk.

A `Touch` érzékelő működése csak annyiban tér el az előzőtől, hogy nem adhatunk meg tulajdonságot, csakis a tárgyak anyagát figyeltethetjük vele.

A `Near` érzékelővel akkor adhatunk jelzést, ha a tárgyunk egy adott tulajdonsággal rendelkező másik tárgy közelébe ér. A `Dist` értékkel a bekapcsoláshoz szükséges távolságot adjuk meg, míg a `Reset` értékkel azt a távolságot, amelynél ha a két tárgy messzebb kerül egymástól, az érzékelő kimenete passzívvá válik.

A `Radar` típusú érzékelő működése megegyezik a kinti világban is használatos rádiólokátorokéval, vagyis egy kúp alakú csóván belül lévő tárgyak érzékelésére szolgál. Itt is megadható, hogy milyen tulajdonsággal rendelkező tárgyakat keressen, a `X`, `Y`, `Z` kapcsolókkal pedig megadhatjuk, hogy a forrásként szolgáló tárgynak (amelyikhez az érzékelőt hozzárendeljük) melyik tengelye alkossa a kúp tengelyét. Az érzékelő kúpjának a csúcsa egybeesik a tárgy érvényben lévő középpontjával. Az `Ang` értékkel adhatjuk meg a kúp nyílásszögét, míg a `Dist` értékkel a hatótávolságot.

A `Property` érzékelő alkalmas arra, hogy figyeljük a tárgy egy adott tulajdonságát. A beállításoktól függően jelzést kaphatunk, ha a ***Prop*** mezőben beállított tulajdonság egyenlő (`Equal`) vagy nem egyenlő (`Not Equal`) a ***Value*** mezőben beállított értékkel, vagy akkor is, ha az érték megváltozott (`Changed`), esetleg egy adott tartományon belül van (`Interval`). A működést kiváltó feltételt a legnyíló listából választhatjuk ki, az előbbieknek megfelelően. Amennyiben az `Interval` feltételt választjuk, a lista alatt két mezőt találhatunk, amelyekben a tartomány alsó és felső határát adjuk meg.

A `Random` érzékelő működése minden jel szerint megegyezik az `Always` típusúéval, semmilyen szemmel látható különbséget nem fedeztem fel.

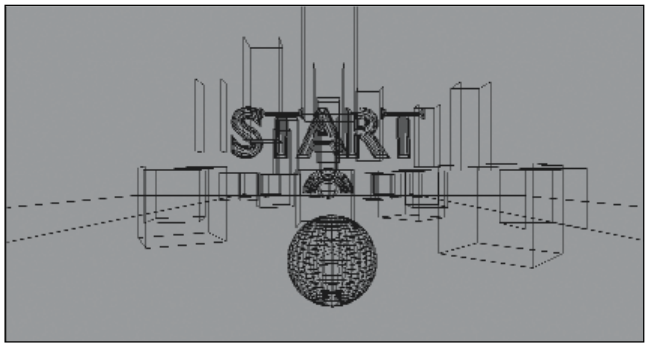
A `Ray` érzékelő hasonlít a korábban bemutatott `Radar`-hoz, itt azonban az érzékeny terület nem kúp alakú, hanem egyenes vonal. Ennél is beállítható, hogy az érzékelő állapota anyagtulajdonságra vagy játékmotorbeli tulajdonságra változzon-e, továbbá a `Range` értékkel a hatótávolságot is meghatározhatjuk. A lenyíló listában szereplő tengelyek szabják meg, hogy az érzékelő sugár a kiindulási tárgynak melyik tengelyével párhuzamosan haladjon, és ebben az esetben természetesen az előjel határozza meg a kiindulási irányt. Ez az érzékelő alkalmas például annak megállapítására is, hogy egy lézersugár eltalált-e egy adott tárgyat vagy a főhős belépett-e az infravörös kapu sugarába.

A `Message` típusú érzékelő a sorban az utolsó, egy üzenet elfogására szolgál. A ***Subject*** mezőben határozhatjuk meg azt a szöveget, ami a tárgyhoz érkező üzenetek azonos nevű

mezőjében szerepelhet. Amennyiben megadunk egy ilyen értéket, a tárgyhoz rendelt érzékelő csak ilyen üzenetek érkezésekor reagál majd.
Az előző részben ismertettem már a vezérlők (***Controllers***) típusait, így ennek ismétlésétől eltekintve a hatásokat kiváltó csomópontok (***Actuators***) tárgyalására térek át.

1. kép Hangállomány betöltése

Az első ilyen hatás a `Movement`, amivel a tárgy mozgását tudjuk befolyásolni. Minden értéket a három térbeli tengelyre vonatkoztatva háromdimenziós vektorként adhatunk meg.
A `Force` a tárgy mozgási energiáját határozza meg, a `Torque` pedig a forgatónyomatékát. A `dLoc` a tárgy helyzetének változtatására szolgál, míg a `dRot` a tárgy elfordulását változtatja meg. A `linV` a tárgy haladási sebességére vonatkozik, és ide tartozik még a három koordináta után található ***add*** kapcsoló – ennek alkalmazásával a tárgyat a megadott értékeknek megfelelően gyorsíthatjuk vagy lassíthatjuk. Ha ezt a kapcsolót használjuk, akkor a beállított értékek hozzáadódnak az érvényben lévő sebességhez. Az `angV` vektorral határozhatjuk meg a tárgy szögsebességét. A fent felsorolt vektorokra vonatkozóan beállíthatjuk, hogy a Blender az értékeket a tárgy helyi koordinátarendszerében vagy a jelenet koordinátarendszerében értelmezze-e. Amennyiben a tárgy helyi koordinátarendszerét szeretnénk használni, akkor az adott vektor sorának végén lévő ***L*** kapcsolót is be kell kapcsolni.
A `Constraint` hatás alkalmazásával elérhetjük, hogy tárgyunk mozgása valamely tengely mentén korlátozott legyen. A kívánt tengelyt a legördülő listából választhatjuk ki, majd a `Min` és a `Max` értékekkel adhatjuk meg a korlátozás mértékét.
A `Camera` hatás alkalmazásával a kamerának adhatunk utasítást arra, hogy a ***OB:*** mezőben megadott tárgy felé forduljon. Itt is használhatunk különféle megkötéseket. Az egyik ilyen a ***Height***, amellyel a kamera magassága állítható be, a másik pedig az ***X*** és ***Y*** kapcsolók egyike, ezzel azt érhetjük el, hogy a kamera a céltárgy kiválasztott tengelye mögött próbáljon maradni. Természetesen előfordulhat, hogy a kamera nem tud megfelelően a tárgy felé fordulni, például akkor sem, ha az X tengelyt beállítjuk, és a kamera már alaphelyzetben is a céltárgy X tengelyének meghosszabbításán helyezkedik el.
A `Sound` meglepő módon hangállomány lejátszására alkalmas. Amikor azonban ezt a hatást választjuk ki, alaphelyzetben hangfájl nem található a legördülő listában. Hogy itt is megjelenjen valamilyen választási lehetőség, először a nézetek váltására szolgáló gombsoron ki kell választanunk a hangszóró ikonnal jelzettet, és az anyagtulajdonságok hozzáadásához hasonlóan a nézet fejlécének segítségével be kell töltenünk egy hangállományt. Ennek megértését szolgálja az *1. kép* is, amelyen piros és zöld keretek jelölik a megfelelő gombokat.
A Blender képes WAV-állományok betöltésére és lejátszására.
A `Property` hatást (effect) alkalmazhatjuk olyan esetekben, amikor a tárgy valamely játékmotorbeli tulajdonságát szeretnénk megváltoztatni. A ***Prop*** mezőben megadott változó értékét a legördülő listából kiválasztott ***Assign*** menüponttal változtathatjuk meg, vagy hozzáadhatunk egy előre meghatározott értéket, feltéve, hogy a listából a ***Add*** pontot választjuk ki. Amennyiben ebből a listából a ***Copy***-t választjuk, akkor a létrehozott tárgy örökli az előző tárgy tulajdonságait. Ebben az esetben a ***OB:*** mező tartalmazza a tulajdonság forrását, emellett a ***Prop*** határozza meg a forrástulajdonságot, míg a cél a korábban beállított ***Prop*** értéke lesz.
Az ismertetést a könnyebben érthető hatásokkal folytatom: a `Message` alkalmas arra, hogy egy adott tárgynak



2. kép A játék kezdete

(ezt határozza meg a ***To*** mező értéke) – vagy amennyiben nem adunk meg céltárgyat, akkor minden tárgynak – üzenetet küldjünk. A ***Subject***-ben megadhatjuk az üzenet tárgyát, ez alapján a `Message` érzékelő el tudja dönteni, hogy kell-e foglalkoznia a beérkező üzenettel. A ***T/P*** kapcsolóval választhatunk szöveges üzenet vagy egy tulajdonság értékének az elküldése között.
A `CD` hatással zenei CD-t tudunk lejátszani a játékban. Lejátszhatjuk az egész lemezt, egy kiválasztott számot, a hatás segítségével pedig módosíthatjuk a lejátszás hangerejét, megállíthatjuk, majd újra elindíthatjuk a lejátszást.
A `Game` alkalmas a játék viselkedésének a megadására. Segítségével kiléphetünk a játékból, újraindíthatjuk azt, vagy egy másik játékot tölthetünk be. Amikor egy másik játék betöltése mellett döntünk, természetesen a játékot tartalmazó fájlnevet is meg kell adni.
És végül a `Visibility` hatás segítségével a tárgy láthatóságát szabályozhatjuk, például eltüntethetjük a már kilőtt és felrobbant űrhajót.
Nos, ebben a hónapban megtárgyaltuk a játékmotor legfontosabb összetevőit, ami ugyan egy kissé száraz és nagyon is elméleti anyag volt, ám a következő részben befejezzük a játékot (a működés szempontjából nézve), és a most kimaradt hatásokkal is megismerkedünk. Mint azt korábban már említettem, játékunk célja az lesz, hogy egy gömbszerű főhőssel anélkül jussunk át az akadálypályán, hogy a kalapácsok összelapítanának minket vagy leesnénk a pálya szélén. A játékot nehezíti az is, hogy a főhős a nem mozgó akadályokkal ütközve pattog, így könnyen előfordulhat, hogy egy ütközést követően pont egy kalapács alá kerül. A külső kinézetre vonatkozóan mellékeltem egy képet (*2. kép*). A távolban látható a kijárat boltíve, míg a főhős a nézőponthoz legközelebb eső gömbszerű alak.
A decemberi viszontlátásig mindenkinek kellemes ismerkedést és alkotást kívánva búcsúzom.

Fábián Zoltán (dzooli@freemail.hu, dzooli@yahoo.com) 25 éves, jelenleg programozóként dolgozik. Szabadidejében szívesen kirándul, túrázik. Emellett szeret rajzolni, érdekli a 3D-grafika és a Linuxszal kapcsolatban minden olyan program és programnyelv, amit még nem ismer vagy nem próbált ki.

© Kiskapu Kft. Minden jog fenntartva