

ESZTERHÁZY KÁROLY FŐISKOLA
MATEMATIKAI ÉS INFORMATIKAI INTÉZET

Blender és Unity cikkek feldolgozása

Készítette:

Gregus Tamás
programtervező informatikus BSc

Témavezető:

Dr. Kovásznai Gergely
adjunktus

EGER, 2011

Tartalomjegyzék

1. Blender cikkek	4
1.1. Modellezés	8
1.1.1. Subdivide	8
1.1.2. Decimate	9
1.1.3. Transzformációk	11
1.1.4. Boolean műveletek	12
1.1.5. Tükrözés	12
1.2. Digitális szobrászkodás	12
1.3. Textúrázás	14
1.4. Animálás	19
1.4.1. Kulcsképkockákkal történő animálás	19
1.4.2. Importált fájl segítségével történő animálás	20
1.5. Exportálás	24
2. Unity cikkek	26
2.1. Ütközési zónák	28
2.2. Scriptek	30
2.3. Kapcsolók	32
Irodalomjegyzék	36

1. Blender cikkek

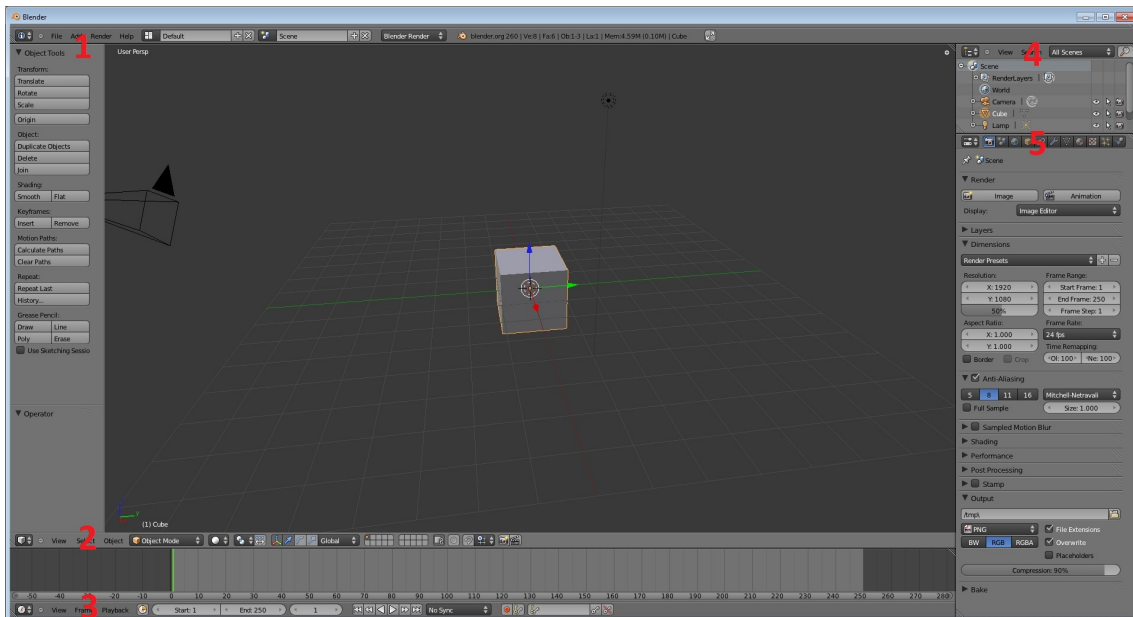
A Blender Foundation által fejlesztett *Blender* nevű 3D grafikai programot a következőkben szakmai cikkeken keresztül mutatom be, így az Olvasó betekintést nyerhet a program bizonyos funkcióinak használatába. Lásd [1] és [2].

A Blender képességei hasonlóak más, kereskedelmi forgalomban kapható zárt forráskódú programokhoz, sőt, teljesen ingyenes és nyílt forráskódú, ennek ellenére rendszeresen fejleszti a Blender Foundation, és számos dologban kiemelkedő tulajdonságokkal rendelkezik, mint az UV kicsomagolásnál használt fejlett algoritmusok (lásd az 1.3. fejezetben).

Emellett a Blender hírhedt arról, hogy nehéz megtanulni. Ez leginkább abból adódik, hogy sokáig zárt szoftver volt egy cégen belül. Mióta nyílt lett, sokat foglalkoznak azzal, hogy logikusabbá, átláthatóbbá tegyék felületét és egyre több dokumentáció készül róla, mely egy vagy több funkcióját elemzi, taglalja működését.

Számos projektben használták, úgy, mint a Pókember 2 című filmben, ahol képes forgatókönyvet terveztek meg segítségével. Ezenkívül a Sintel, Big Buck Bunny és Elephants dream című animációs filmeket is a programnak köszönhetjük. Rendeléreért (képekért) lásd a [2, 500. oldaltól].

Indítás után az 1. ábrán látható képernyő fogad minket:



1. ábra. A Blender felülete

Mint a legtöbb program, úgy a Blender felülete is több kisebb ablakból áll. A különbség annyi, hogy a Blender felülete szabadon változtatható. Az „action zone”-okkal (vonalkázott rész) lehet új ablakokat felhúzni, illetve a meglévőket egybemosni

a régiekkel. Az alapbeállítás szerint a kezdőképernyőn a képen látható módon helyezkednek el az ablakok, melyek meg vannak számozva a képen (egyéb ablakokhoz lásd a [2, 1. fejezet] és az [1, 2. fejezet]). Viszont ezek akármikor átválthatóak másikká. Ez alapján az adott ablak funkciója:

1. Info

Gyakorlatilag a főmenü opciókat tartalmazza. Itt lehet menteni, betölteni, importálni, exportálni, új objektumot hozzáadni a projekthez, képet vagy animációt renderelni, ezenkívül lehetőségünk van bugot bejelenteni, manuált olvasni, elérni a honlapot, stb.

2. 3D View

Ez az ablak foglalja el a legnagyobb teret. Ezen keresztül jobb oldalon látjuk magát a színpadot vagy jelenetet egy adott kameraállásból. Bal oldalon helyezkednek el az objektum(oka)t valamilyen szempontból módosító eszközök, például (a gyorsbillentyűket nem számítva) itt tudunk transzformációkat végrehajtani, árnyalni, kulcsképkockákat (keyframe) beszúrni, mozgásirányt kiszámítani, stb. A tényleges színpad mellett jobb oldalon láthatunk egy kis „+” jelzést, amellyel további opciók nyithatók meg. Itt mindenféle koordinátákat láthatunk, ami az éppen aktuálisan kijelölt objektumra vonatkozik. A 3D View legalján a nézettel, kijelöléssel, illetve objektumokkal kapcsolatos attribútumokat állíthatunk be, melynek egy része a 3D View bal oldalán is állítható volt. Emellett váltható a 3D View módja is, az alábbi módok közül dolgozhatunk a számunkra kedvezőben: *Object Mode* alapvető objektumtranszformációkra, *Sculpt Mode* „faragásra”, *Weight Paint* „súlyozásra”, *Edit Mode* csúcsok transzformációjára. Emellett az árnyalást és a kijelölt objektum pivot pontját is áthelyezhetjük.

3. Timeline

Gyakorlatilag animációkezelést tudunk itt végezni. Láthatjuk, hol tartunk a képkockák között, hány képkockával dolgozunk az adott animációban, látjuk, hol vannak a kulcsképkockák, és akármikor ráléphetünk. Bekapcsolhatunk automatikus kulcsképkocka rögzítést, illetve *Keying Setet* választhatunk.

4. Outliner

Itt gyakorlatilag a hierarchiát láthatjuk, mely a felhelyezett objektumok neveit jeleníti meg. Beállíthatjuk, melyik látszódjon, illetve melyik ne, illetve az adott objektum kijelölhető legyen-e, és alkalmazzunk-e rá valamilyen rendelési megkötést.

5. Properties

Ebben a részben a kijelölt objektum tulajdonságait állíthatjuk be. Rendelési, transzformációs attribútumok konfigurálhatók kedvünk szerint, ezenkívül megkötéseket, módosítókat illetve csúcs csoportokat is létrehozhatunk. Textúrákat kezelhetünk, anyagot hozhatunk létre, illetve fizikát is alkalmazhatunk az egyes objektumokon.

A programban számtalan *hotkey* (billentyűkombináció, gyorsbillentyű) található, mellyel a fentebb említett funkciókat gyorsabban is elérhetjük (a hotkey-kről további információhoz lásd a [2, 1. fejezet] és az [1, 2., 3. fejezettől kezdve]). Kezelésük módfüggő, így egy-egy dolgot csak bizonyos módból használhatunk. A teljesség igénye nélkül felsorolhatóak a legfontosabbak:

Object Mode-ból használható hotkey-k:

Kamerakezelő hotkey-k:

- 1: y tengellyel egyirányú kamera
- 3: x tengellyel egyirányú kamera
- 7: z tengellyel egyirányú kamera
- 4: kamerát balra forgatjuk az origó körül
- 6: kamerát jobbra forgatjuk az origó körül
- 2: kamerát lefelé forgatjuk az origó körül
- 8: kamerát felfelé forgatjuk az origó körül
- 5: perspektivikus nézet és párhuzamos vetítés között válthatunk
- 0: kamera perspektíva és felhasználói perspektíva között válthatunk. A kamera perspektívánál a jelenetben (scene-ben) létrehozott kamera objektum képét fogjuk látni
- egér görgőgombja: ha nyomva tartjuk, ennek segítségével szabadon forgathatjuk a kamerát
- egér görgőgombja+SHIFT: segítségével a képsíkot (kamera síkja) alkotó tengelyek mentén történő eltolás történik az egér mozgásának irányába

Objektumkezelő hotkey-k:

- G: eltolás (translation) az egérrel
- G+X G+Y G+Z: eltolás az adott tengely mentén
- R: forgatás (rotation) az egérrel
- R+X R+Y R+Z: forgatás az adott tengely mentén
- S: skálázás (scale) az egérrel
- S+X S+Y S+Z: skálázás az adott tengely mentén

CTRL+A: a transzformációs adatoknak megfelelően transzformációkat lehet végezni az objektumon, hogy az adott transzformáció szinkronba kerüljön az adattal. Ezt főleg exportálás előtt célszerű elvégezni.

CTRL+S: mentés

CTRL+J: egyesítés (join). Olyan, mintha boolean módosítót unió művelettel alkalmaztunk volna két objektumon, annyi különbséggel, hogy nem marad meg a kiválasztott objektum eredeti állapotában.

SHIFT+D: duplikáció. használata után az egérrel pozicionálhatjuk az objektumot, majd kattintással helyezhetjük el a kívánt pontban.

CTRL+Z: visszavonás.

CTRL+P: két objektum esetén van értelme használni, a másodikként kijelölt objektumot szülőnek beállítja az elsőként kijelölt objektumnak.

H: elrejtés. Az Outlinerből kikapcsolható.

Animációkezelő hotkey-k:

I: képkocka beszúrás végezhető el, ami az objektum különféle transzformációját tárolhatja el.

Rendereléses hotkey-k:

P: renderelés, ESC-el lehet kilépni belőle

Módváltó hotkey:

TAB: Object Mode és Edit Mode között váltogat

Kijelöléses hotkey-k:

A: mindent kijelölni/mindenről a kijelölést leszedni

B+BAL EGÉRGOMB: Lenyomása után kijelölhetünk poligonokat, éleket, illetve csúcsoakat egy másik beállítástól függően.

B+GÖRGŐGOMB: Az egérrel kijelölt poligonokról, élekről, illetve csúcsokról leszedhetjük a kijelölést.

Edit Mode-ból használható hotkey-k:

Szerkesztéses hotkey-k:

E: A kijelölt dolog extrudálása az egér mentén. Kattintással rögzíthető az újonnan létrejött dolog helye. Extrudálható él, csúcs, illetve poligon is.

Kijelöléses hotkey-k:

ALT+SHIFT+JOBBKLIKK: Ha ezt a kattintást egy élre végezzük el, akkor a vele közel egy irányba mutató, folytatólagos (megosztják csúcsukat az előző

élel), párhuzamos élek kijelölődnek.

ALT+JOBBKLIKK: Az adott élel párhuzamos, de nem folytatólagos élek jelölődnek ki (de egy tengely mentén).

CTRL+JOBBKLIKK vagy SHIFT+JOBBKLIKK: több él jelölhető ki a segítségükkel (poligonból és élből többet csak a SHIFT+JOBBKLIKKel lehet).

Az alap kijelölések gyorsbillentyűi azonosak az Object Mode-ban használttal.

Transzformációs hotkey-k:

Ezek azonosak az objektumkezelő hotkey-kenél megemlített transzformációs hotkey-ekkel.

Speciális hotkey-k:

W: Itt oszthatunk fel poligonokat, egyesíthetünk csúcsokat, poligon normálvektorának irányát válthatjuk illetve ha egy pontban több csúcs van, azokból csak 1-et hagyjunk meg.

Szeparációs hotkey-k:

P: A kijelölt poligonokat tudjuk egy új objektumba áthelyezni a segítségével.

A gyorsbillentyűk ismeretében könnyebben hozzá tudunk látni valamilyen művelethez. Megkötéseink ingyenes program lévén nem lesznek, tehát csak az erőforrásaink (idő, gép teljesítménye) szabhatnak határt kitűzött feladatunknak.

1.1. Modellezés

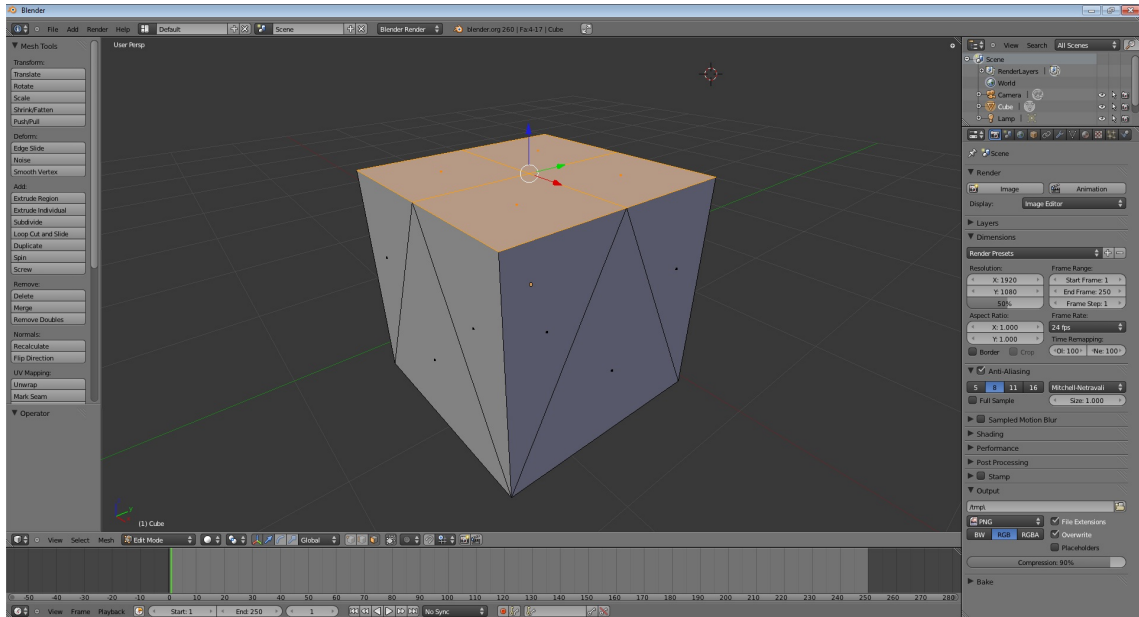
A *modellezés* hagyományos módja általában a következők szerint szokott zajlani. Általában modellezéskor már tervben van az, amit le szeretnénk modellezni. Ez vagy fejben elképzelt dolog, vagy egy előre elkészített rajzról, képről történhet. Az utóbbi segít bennünket a méretarányok meghatározásában is, amellet, hogy kevesebb korrekciót kell majd az elkészült modellen végrehajtani.

A kép/rajz, vagy egyéb dolog megfigyelése után (melyet egyébként akár egyenesen a Blenderbe is bevihetünk, jócskán megkönnyítve ezzel a modellezést) próbáljunk meg olyan alapértelmezett *mesh-eket* (hálót, felületet) keresni, amely az objektum egy adott részletéhez a leghasonlóbb. Ezt könnyedén megtehetjük az Info ablak Add opciója alatt. Az objektumot G billentyűvel könnyedén pozicionálhatjuk.

1.1.1. Subdivide

Amennyiben valamely poligonját részletesebbé akarjuk tenni, váltsunk Edit Mode-ba a TAB segítségével, majd kattintsunk a 3D View-ban Face Select Mode opcióra. A

felosztani kívánt poligon, másnéven face kijelölése után nyomjunk W-t és válasszuk a *Subdivide* opciót. Így ha egy poligon volt kijelölve, akkor a középpontjából merőleges és függőleges tengely mentén lesz szétvágva a poligon, más szóval 4 negyedakkora poligon jön létre belőle. A vele szomszédos poligonok is módosulnak e szerint, hogy illeszkedni tudjanak hozzá. Lásd a [2, 2. fejezet] és az [1, 4. fejezet]. Hatása a 2. ábrán látható.



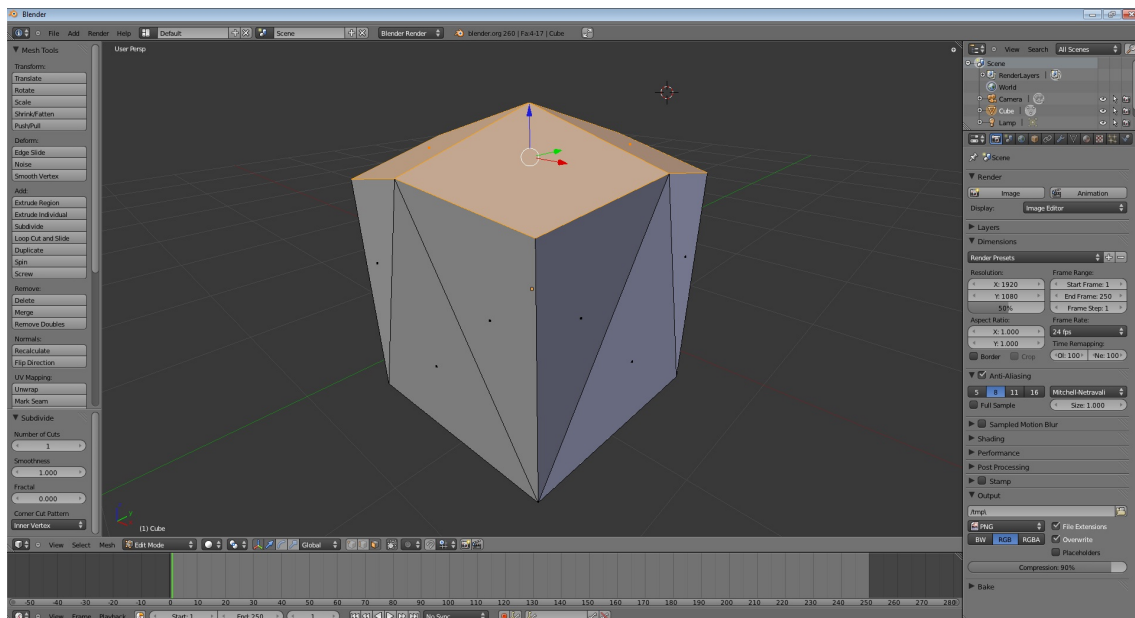
2. ábra. Subdivide után megfigyelhető a szomszédos poligonok felosztódása is

A *Subdivide Smooth* opcióval a felosztódás mellett a poligon által alkotott felület domborodik, tehát a létrejött felület középpontja lesz a legtávolabb a felület síkjához képest, és minél messzebb található egy csúcs ettől a középponttól, annál mélyebben fog elhelyezkedni a felületen. Végrehajtása után a 3. ábrán látható eredményt kapjuk.

A Subdivide ellentettje a Decimate.

1.1.1.2. Decimate

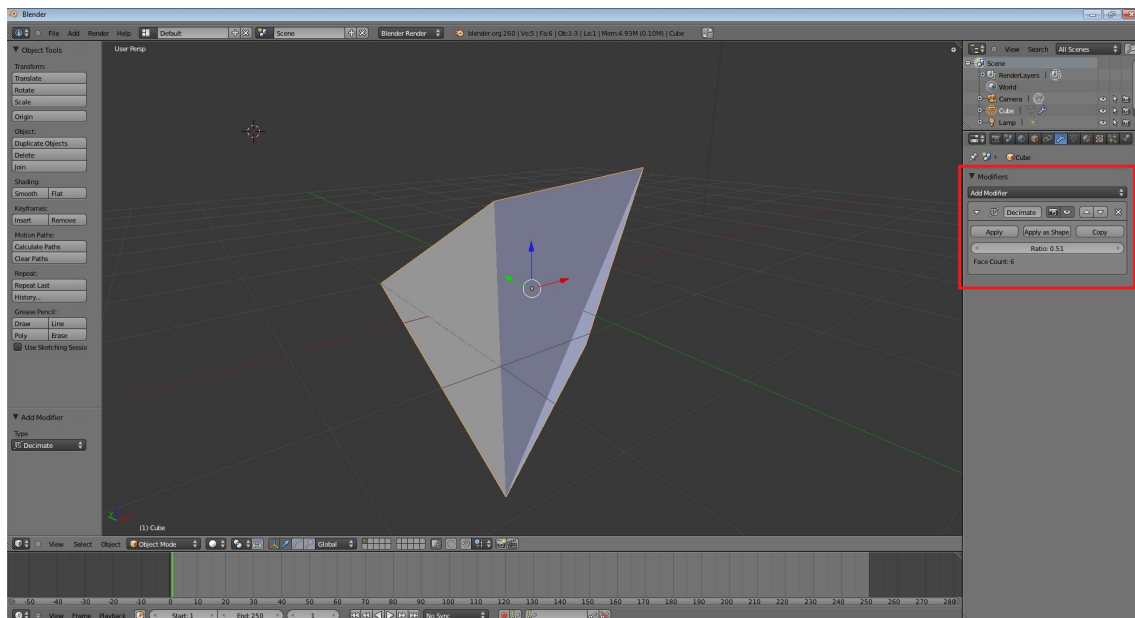
A *Decimate* valójában egy *módosító* (modifier). Alkalmazni akkor tudjuk, ha Object Mode-ra váltunk, és az objektum kijelölése mellett megkeressük a Properties ablakot. Itt egy csavarkulcs kinézetű ábrát kell keresnünk, ahol hozzáadhatunk módosítókat az aktuálisan kijelölt objektumhoz. A legördülő listában láthatjuk a Decimate-et. Ezután a Ratio (arány) sávnál tudjuk a poligonok számának új arányát nyomon követni. A módosító alján látható, hogy az épp megadott arányban hány poligon



3. ábra. Subdivide Smooth hatása

vesz részt a mesh felépítésében. Az Apply segítségével fogadhatjuk el az objektum meshjének változásait.

A Decimate használata viszont sajnálatos módon kerülendő, ugyanis okozhat előre nem várt dolgokat. Viszont ezeket aránylag könnyű megtalálni és kijavítani. A 4. ábrán a Decimate használat közben. Egyéb módosítókhoz lásd a [2, 2. fejezet] és az [1, 4. fejezet].



4. ábra. Decimate

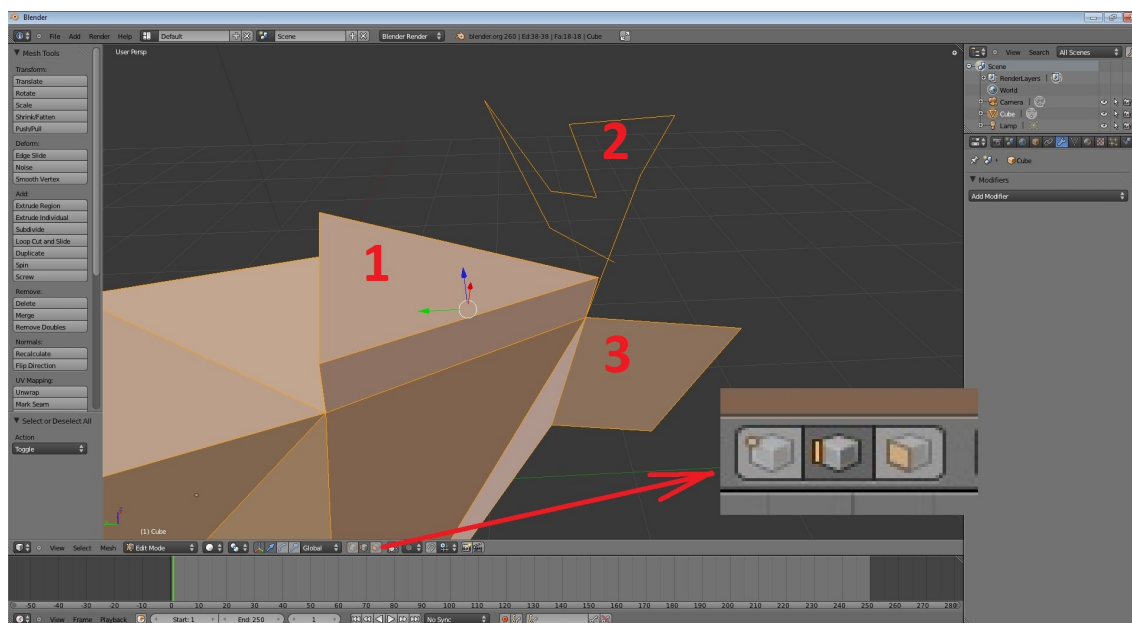
1.1.3. Transzformációk

Mint minden 3D modellező programban, így a Blenderben is transzformációk hajthatók végre egy adott objektumon, és/vagy a mesh felületében. Lásd az [1, 3. fejezet].

Az objektum Object Mode-ban skálázható az S billentyűvel, eltolható a G-vel és forgatható az R-rel. Ha mellette nyomva tartjuk például az X-et, akkor az X tengely mentén tudjuk például G nyomva tartása mellett eltolni az objektumot.

Az Edit Mode-ban az egy adott objektumot alkotó egységeket, úgy mint a csúcsokat, éleket, polygonokat tudjuk transzformálni. A az 5. ábrán látható az a három opció, melyek segítségével állítható, hogy mely komponense legyen kijelölhető az adott objektumnak.

Ebben a módban használható még az E billentyű is, mely segítségével növeszteni (extrudálni) lehet az adott komponenst. Szintén az 5. ábrán 1-essel jelölve egy polygon *extrudációját* láthatjuk, mely során annyi új polygon jön létre, ahány éle volt a kijelölt poligonnak. 2-essel jelölve egy csúcs extrudálás látható, mely során új élek képződnek. A 2-es mellett 3-assal jelölve egy él extrudációja vehető ki, melynek mellékhatása pontosan egy polygon képződése. Bővebb információért lásd a [2, 2. fejezet] és az [1, 4. fejezet].



5. ábra. Transzformációk és Extrude hatása.

Mind az Object Mode-ban, mind az Edit Mode-ban használható a DEL. Ez előbbiben az objektum, utóbbiban a kijelölt komponens törlését jelenti. Ezáltal magával vonhatja több komponens törlését is, bár kiválasztható, hogy pontosan mit szeret-

nénk a kijelölt részen belül törölni (ugyanis mondjuk ha már egy poligont jelöltünk ki, akkor párhuzamosan vele kijelölődik az őt alkotó élek és csúcsok halmaza is.

Viszont attól, hogy Edit Mode-ban a Delete menüben Allt választunk (mindent törölni akarunk), attól az objektum még létezni fog az Outlinerben, csak semmiből se fog állni. Ezért ha a teljes objektumot törölni akarjuk, váltsunk Object Mode-ra és onnan hajtsuk végre a törlést.

1.1.4. Boolean műveletek

Ezek gyakorlatilag az unió, különbség és metszet *halmazműveleteknek* felelnek meg. Azonos helyen található, mint a Decimate, ez is módosítóként érhető el, annyi különbséggel, hogy ki kell választanunk az aktuálisan kijelölt mellé egy objektumot, amivel elvégezzük a műveletet. A modifier-en keresztül választott objektum a művelet elvégzése után az eredeti állapotában is megmarad. Két objektumot úgy jelölhetünk ki, ha a másodiknak kijelölt mellé a SHIFT-et nyomva tartjuk. A CTRL+J billentyűkombináció segítségével két kijelölt objektumot *join-olhatunk* (egyesíthetünk). Ennek azonos hatása lesz az unió műveletével, annyi különbséggel, hogy nem marad meg egyik objektum eredeti állapota sem (persze CTRL+Z-vel vagy szeparációval visszavonható a művelet).

1.1.5. Tükrözés

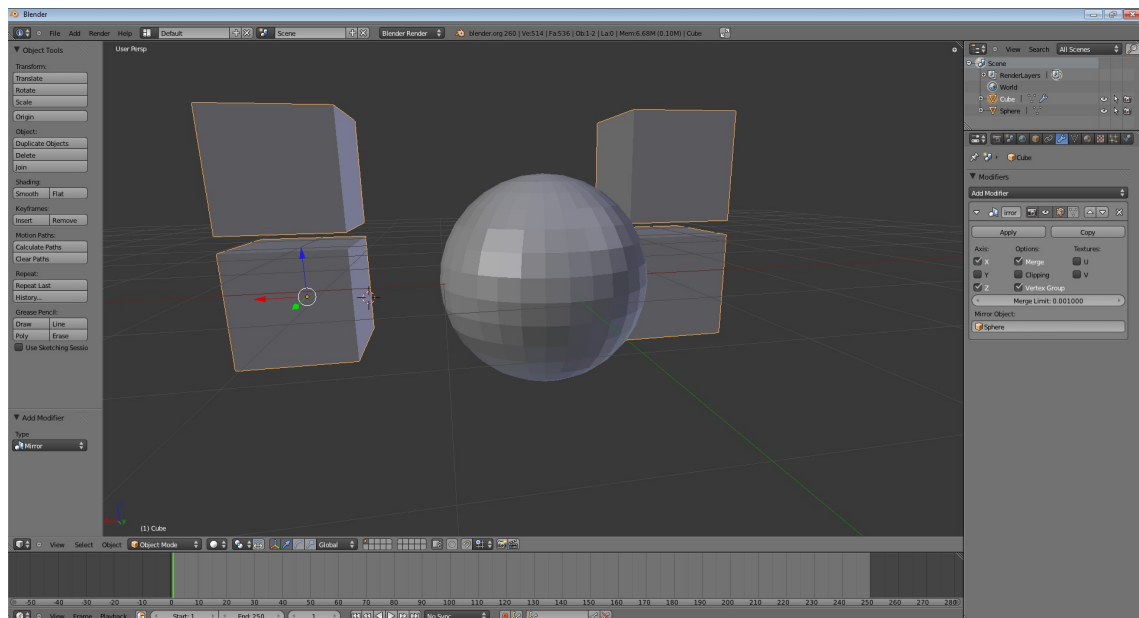
A *mirror* (tükrözés) művelete is egy módosító. Használatához egy adott objektumot kell kiválasztani, majd mellé egy úgynevezett tükröző objektumot, aminek a középpontjára fog történni a tükrözés. Ha ezt kiválasztottuk, a *modifier Axis* részben azt a tengelyt választhatjuk ki, amelyik irányba akarjuk, hogy az aktuális objektum képe kerüljön. Ha még egy tengelyt választunk ki, akkor a már meglévő aktuális objektumot és kép(ek)et is tükrözi a módosító.

A 6. ábrán a sphere (gömb) a tükör objektum, és azt a cube-ot (kockát) tükröztük, amelyikben a tengelyek jelzése van feltüntetve.

1.2. Digitális szobrászkodás

A *sculptozás* (másnéven szobrászkodás) gyakorlatilag a modellezés részeként tekinthető, viszont egy sajátos technológiának fogható fel, direkt ilyen célra fejlesztett programokkal. Ilyen például a ZBrush, illetve a Sculpttris is, de számos más program is támogatja, mint a Blender, a 3D Studio Max, illetve a Maya.

Sculptozáskor egy poligonokból álló felületet tudunk könnyedén változtatni, legalábbis könnyebben, mintha külön kijelölgetnénk a poligonokat és transzformálgat-



6. ábra. Tükrozés X és Z tengelyekre.

nánk őket. A szobrászkodás segítségével szabályosabb eredményt vagyunk képesek elérni. Minél nagyobb felbontású egy felület, annál részletesebb, annál több részlet tudjuk megváltoztatni. a könnyítést az okozza, hogy gyakorlatilag egyszerre több *face* (poligon) pozícióját tudjuk módosítani.

Sculptozni Blenderben a Sculpt Mode-ban lehetséges (lásd az [1, 8. fejezet]). A módba belépéskor bal oldalon különféle opciók jelennek meg. Ezek közül a legfontosabbak:

Brush:

A *Radius* sávban állíthatjuk be az egér körül megjelenő kör méretét. Gyakorlatilag ez az ecsetünknek a mérete lesz, vagyis erre a területre fog hatni sculптоzásunk eredménye.

A *Strength*-ben határozható meg, hogy mekkora intenzitással változtassa a poligonokat egy egérekattintás. Ez viszont függ a választott ecsettípustól is.

Tool:

Ebben a részben a Sculpt Toolt (gyakorlatilag az ecsettípust) választhatjuk ki. Igen változatos eszközök közül választhatunk, a 7. ábrán a teljesség igénye nélkül láthatunk pár eszközt és a hatásait:

- 1: draw - add
- 2: draw - subtract
- 3: grab
- 4: inflate - deflate

5: snake hook

6: clay

7: pinch

Az eszközök hatása persze egy-egy kattintásra nem lesz látványos változás, ezért, és hogy szemléletesen látható legyen a poligonok térbeli helyzetváltozása, egy adott pozícióra többször kattintottunk. A *Draw*-nál minden kattintásra a *Strength*-be megadott értéktől függően többször történik változás az adott felületben. Viszont a *Grab*-nál semmilyen változás nem történik egy kattintásra, ott az egér pozícióját kell változtatni.

Curve:.

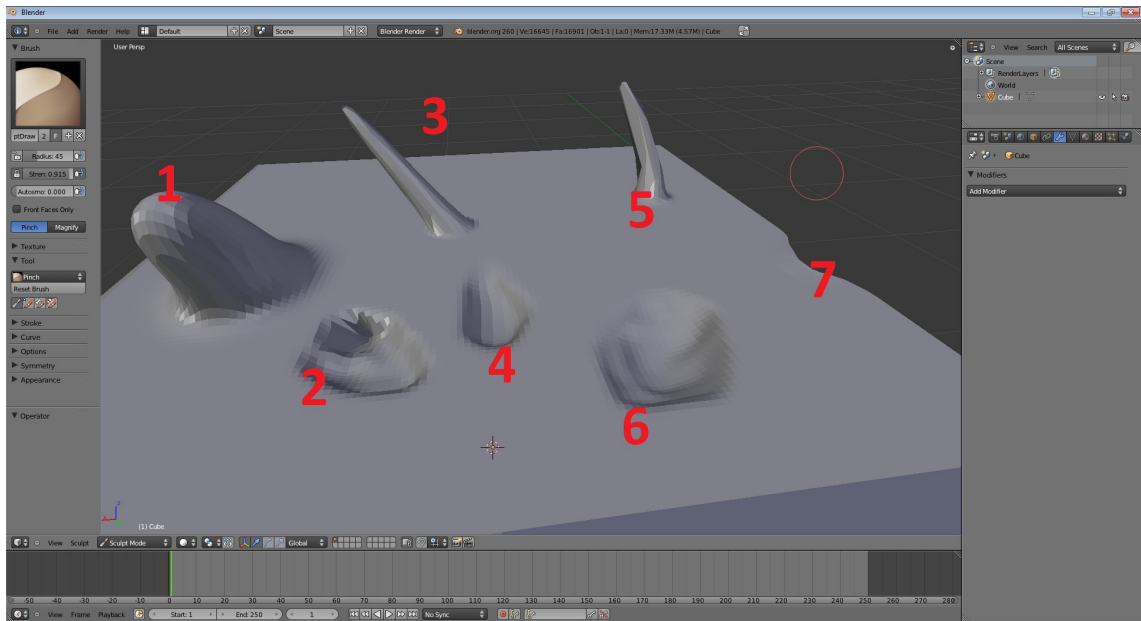
Ebben a részben állítható a sculптоzás görbéje. Gyakorlatilag a poligonok szobrászat utáni elrendeződését írjuk le a görbével. Változtatásának hatásai jól kivehetők a 8. ábrán. Itt a számokkal nem megjelölt részek a görbe alatti kis ikonok által kiválasztott alapértelmezett görbetípusok eredményei, míg az 1-essel illetve 2-essel jelölt alakzat a kép bal szélén látható görbealakból jött létre. A különbség közöttük, hogy az 1-es *Grab*-bal, a 2-es *Draw*-val lett létrehozva. Megfigyelhető, hogy például ilyen görbe mellett a *Draw*-val történő „rajzolás” mennyivel másabb hatást kelt a módosítatlan *Draw*-hoz képest. Az alapértelmezett görbét könnyedén visszaállíthatjuk a *Reset Brush* opció segítségével, melyet a *Tool*-ban találunk.

Symmetry:.

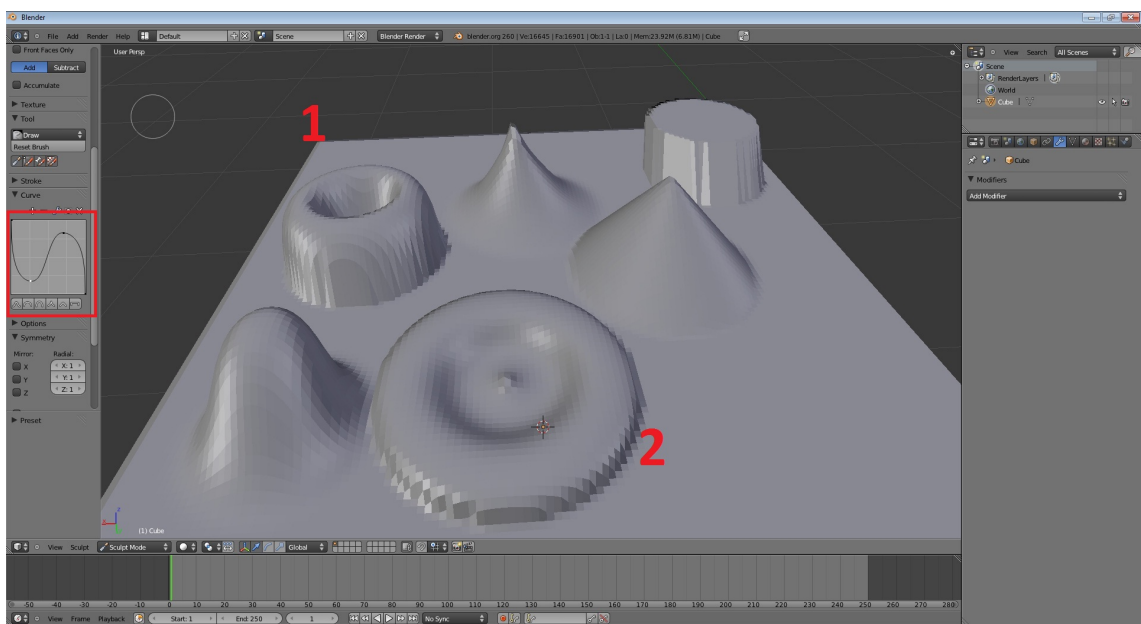
Ennek a segítségével a sculптоzott felületet biztosító objektum középpontja határozza meg a három tengely közös metszéspontját (origóját). Tehát ha egy kocka egyik oldalán szobrászkodunk és az *X* tengelyre tükrözzük rajzunkat, akkor a 9. ábra alapján fog kinézni az adott tengelyes tükrözés.

1.3. Textúrázás

Most tekintsük meg a Blenderben történő textúrázás folyamatát, mely *UV Mapping* segítségével fog történni (lásd az [1, 7. fejezet] és a [2, 3. fejezet]). Először *SHIFT+ALT+JOBBKLIKK* segítségével jelöljük körbe a textúrázni kívánt területet. Kattintsunk a *3D View* bal oldalán, az *UV Mapping* részben található *Mark Seam* opcióra, lásd a 10. ábrán. Ezt követően a *3D View* jobb felső sarkában található csíkozott részt húzzuk el az egérrel, hogy új ablakunk jöjjön létre. Eltűntetése egyébként úgy történik, hogy kiválasztjuk, mely ablakot szeretnénk eltüntetni, és a vele egy sorban/oszlopban lévő ablakban található hasonló csíkozott részt ragadjuk



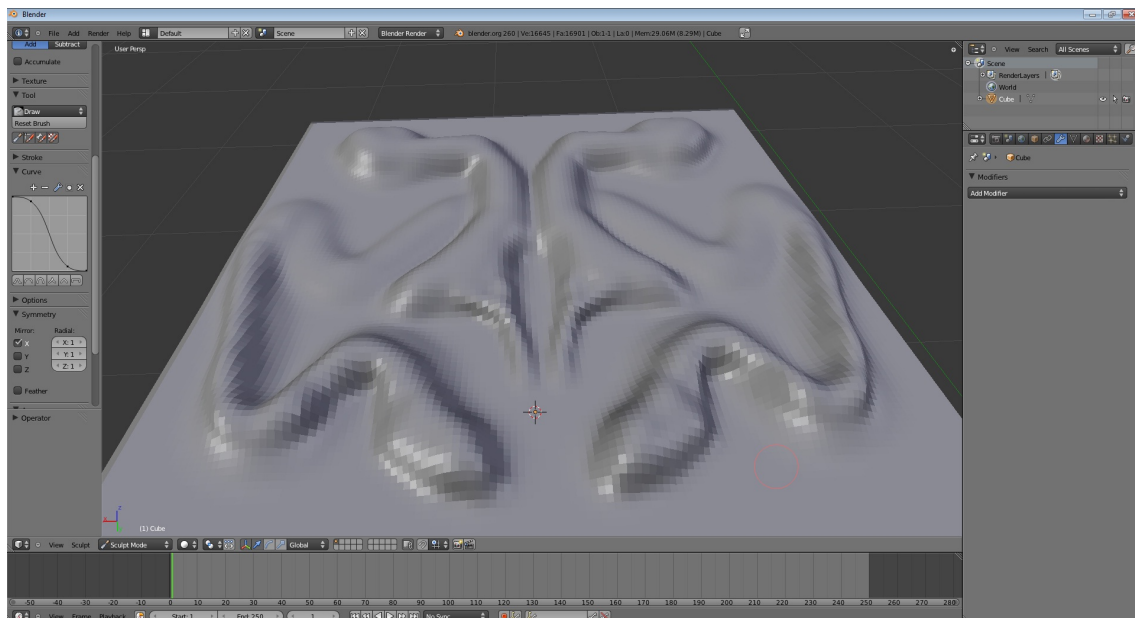
7. ábra. Sculpting Blenderben.



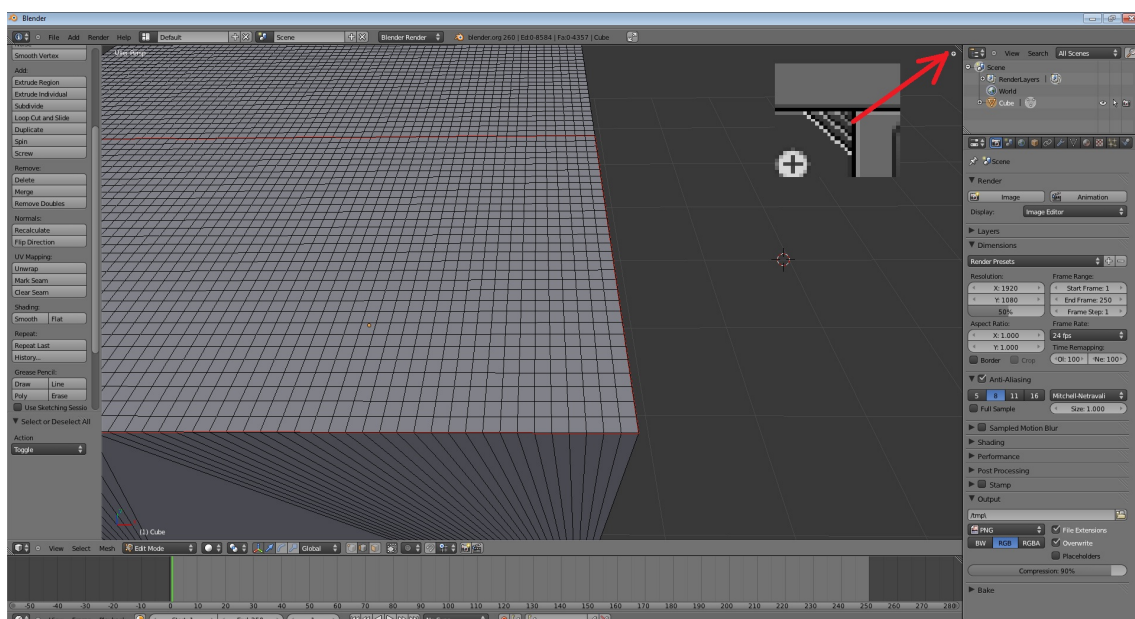
8. ábra. Curve változtatásának hatásai.

meg az egérrel, majd húzzuk rá az eltüntetni kívánt ablakra. A megjelenő nyíl és a világos ablak fogja jelezni, hogy az az ablak fog megmaradni és amelyikre mutat, az nem.

Az új ablakot váltsuk át *UV/Image Editor* módba. A 3D Viewban jelöljük ki a korábban körbekerített területet, majd a P-vel válasszuk a *Separate Selectiont*, ezzel egy új objektumot hozva létre. Váltsunk ugyanitt Object Mode-ba és jelöljük ki az újonnan létrejött objektumot. Az UV/Image Editorban kattintsunk a New



9. ábra. X tengelyre történő Draw tükrözés.



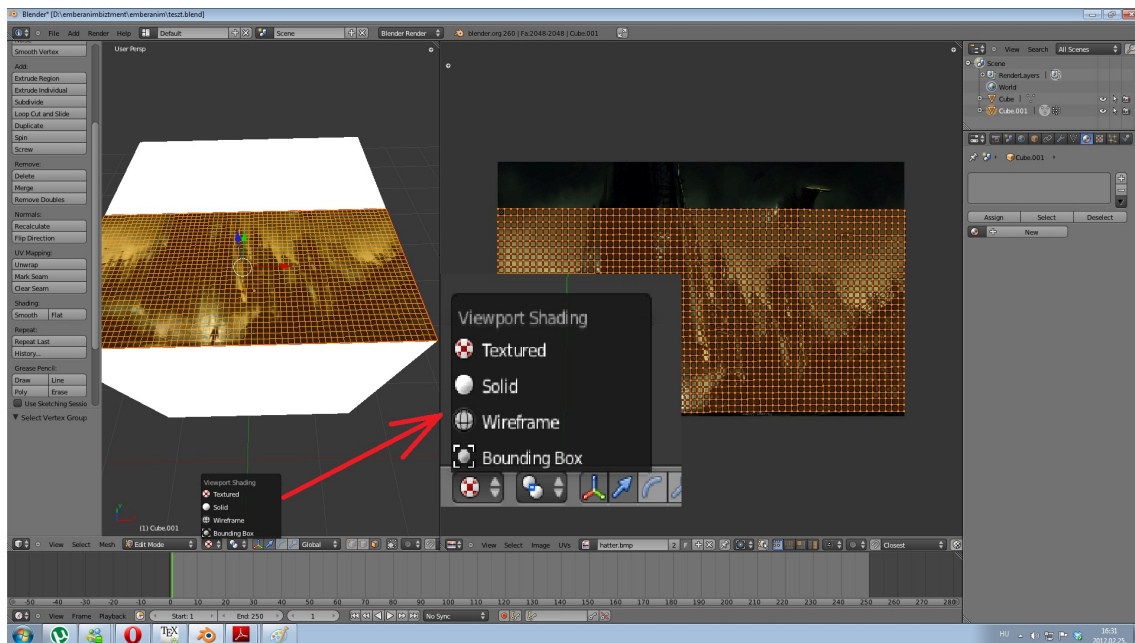
10. ábra. Seam megjelölés, új ablak nyitás

opcióra, ahol elég csak egy nevet megadni az új képnek, majd okézzuk le. Ekkor egy nagy feketeséget fogunk kapni. Ezután kattintsunk az Image opcióra, majd az Open Image-ben tallózzunk be egy képet, melyet textúráként szándékozunk használni. A 3D View bal szélén használjuk az UV Mapping *Unwrap* nevű opcióját (lásd a 11. ábrán), mellyel a mesht *kiterítve* láthatjuk az UV/Image Editor módban. Amennyiben a mesh nem sík felület volt, hanem egy gömb felülete, úgy inkább válasszuk a Sphere

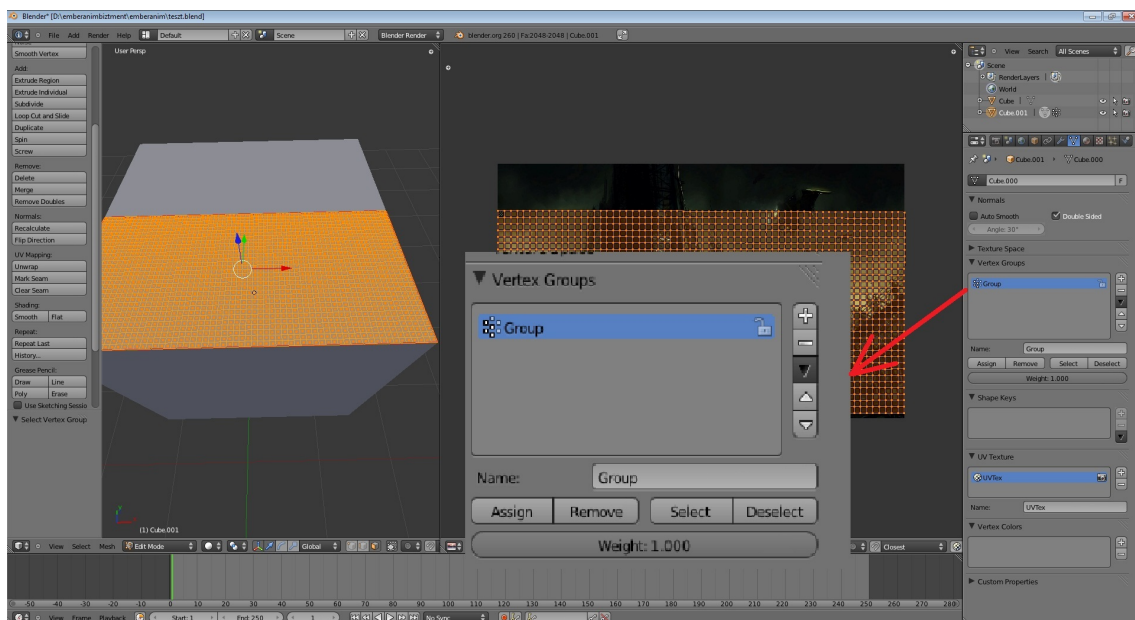
The screenshot shows the Blender 2.80 interface. The top status bar indicates the current file is 'Blender (.org 280) [F:\2020-2048] | Cube.001'. The left sidebar contains the 'Tools' shelf with the 'Unwrap' button highlighted. The main 3D viewport shows a cube with a yellow mesh. The right sidebar shows the 'Properties' panel for the 'Cube.001' object, with the 'Mesh' tab selected. The 'Unwrap' button is located in the 'Tools' shelf on the left, under the 'Mesh' tab. The 'Unwrap' button is highlighted, and a tooltip is visible, indicating that the 'Unwrap' button is used to 'Unwrap the mesh of the object being edited'.

Ha ez megtörtént, a 12. ábra szerint a 3D View alján látható kis fehér gömbre kattintva válthatunk *Viewport Shading*ek között. Ez az általunk látott nézőpont árnyalása, gyakorlatilag annak a kamerának a képe, amelyből alapértelmezetten látjuk a jelenetünket. A fehér gömb a Solid árnyalást jelöli, vagyis szürke tónusú objektumokat látunk a segítségével. Ha viszont átváltunk Texturedre, láthatjuk, hogy az objektumunk mely részei vannak textúrázva, illetve milyen textúrát tartalmaznak objektumaink.

17



12. ábra. Viewport Shading átváltás Texturedre.



13. ábra. Vertex Group

Lehetőségünk van a Properties ablakban *Materialokat* (anyagot) is létrehozni, melynek többféle textúrát adhatunk. Így ha valamelyik textúrát több helyen kívánjuk felhasználni, egyszerűen csak a számára szükséges anyagot elég kiválasztani. Viszont a textúrákkal gondunk lehet, ha exportáljuk az egész jelenetünket vagy csak részleteit egy másik programba, így a plusz munka elkerülése végett elég csak szétválogatni az objektumokat, majd az új programban feltextúrázni őket.

Unity-be történő exportálás során megmaradnak a materialok és textúrák, csak újra el kell helyezni őket az objektumon.

1.4. Animálás

A Blenderben képesek vagyunk animálni is (lásd a [2, 6., 7. fejezet] és az [1, 3. fejezet]). Ezt többféleképpen megtehetjük:

- Keyframe-eket (kulcsképkockákat) alkalmazva animálunk
- Valamilyen importált fájlból olvassuk ki az animációt és azt használjuk fel

Akármelyik módot is válasszuk, mintkét módszernek lesznek előnyei és hátrányai is.

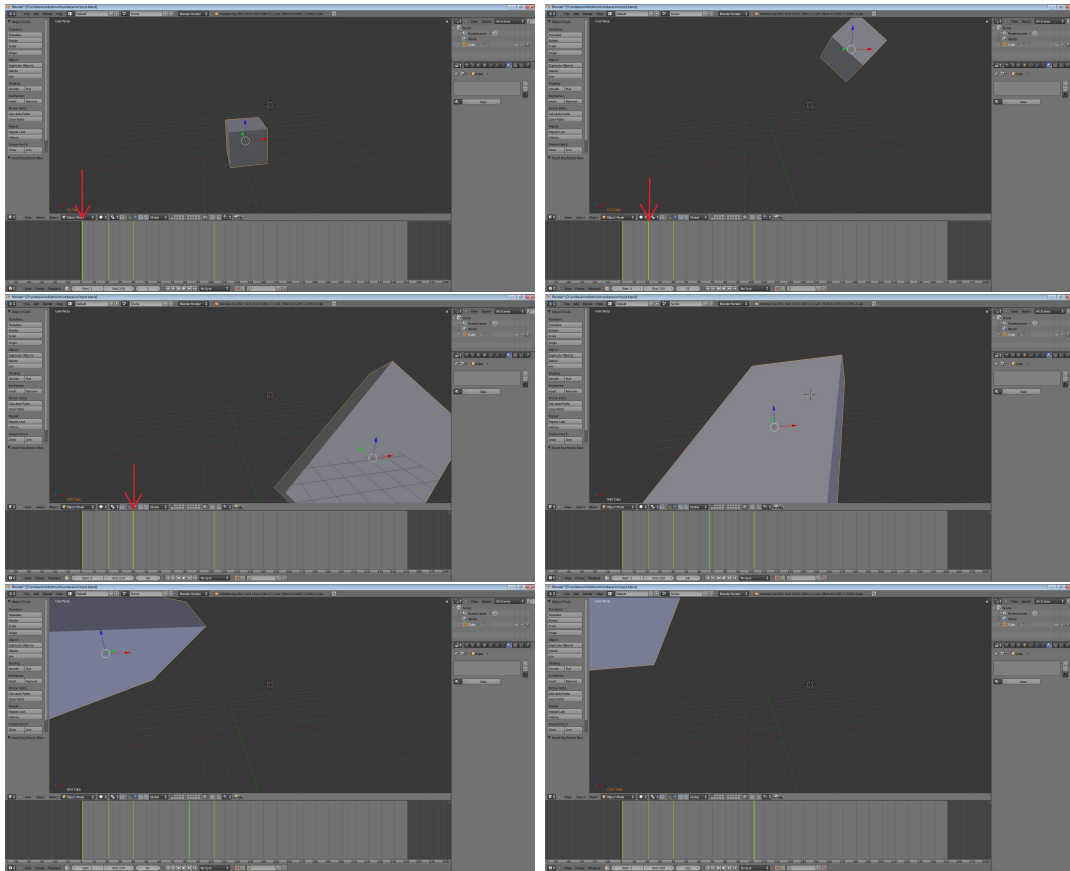
1.4.1. Kulcsképkockákkal történő animálás

- Kevésbé valóság-hű animációkat hozhatunk létre vele
- Akármilyet létrehozhatunk
- Az animáció bonyolultságától függően sok időt elvehet a létrehozása

Jelöljük ki az animálandó objektumot. A *Timeline* ablakban álljunk a *x*. képkockára, ahol *x* az első és az utolsó képkocka közötti *frame* sorszáma. Ezután vigyük az egeret a 3D View felé, majd nyomjunk I-t kulcsképkocka létrehozásához, ahol kiválaszthatjuk, mely tulajdonságait rögzítsük az objektumnak az adott kulcsképkockára (nem véletlenül ez a neve, ugyanis a Blender a kulcsképkockák közötti frame-ekben saját maga számolja ki az eltéréseket, hogy egyenes arányban változzon meg az objektum minden paramétere, amíg az egyik kulcsképkockától a másikhoz ér a lejátszófej). Ha ez megtörtént, álljunk egy ezután következő képre, majd a 3D Viewban transzformáljuk valamilyen módon az objektumot. A transzformációnak megfelelően rögzítsük a számunkra lényeges adatokat a képkockára. Egy ilyen létrehozott animáció képei láthatóak a 14. ábrán.

Miután ezt elvégeztük párszor, álljunk a legelső képkockára, majd a Timeline alján lévő lejátszás ikon segítségével megtekinthetjük animációnkat. Láthatjuk, hogy az így létrehozott animációnk messze van az élethű mozgástól, emellett ha ezzel a módszerrel egy bonyolultabb filmet szeretnénk létrehozni, igen sok munkánk lesz vele. Viszont könnyedén módosíthatjuk a már meglévő kulcsképkockáinkat is.

Az adott *keyframe-et* el is tudjuk távolítani, ha a Timeline-on ráállunk a törölni kívántra, majd a 3D View bal szélén a Keyframes résznél a Remove-ra (törlésre) kattintunk.



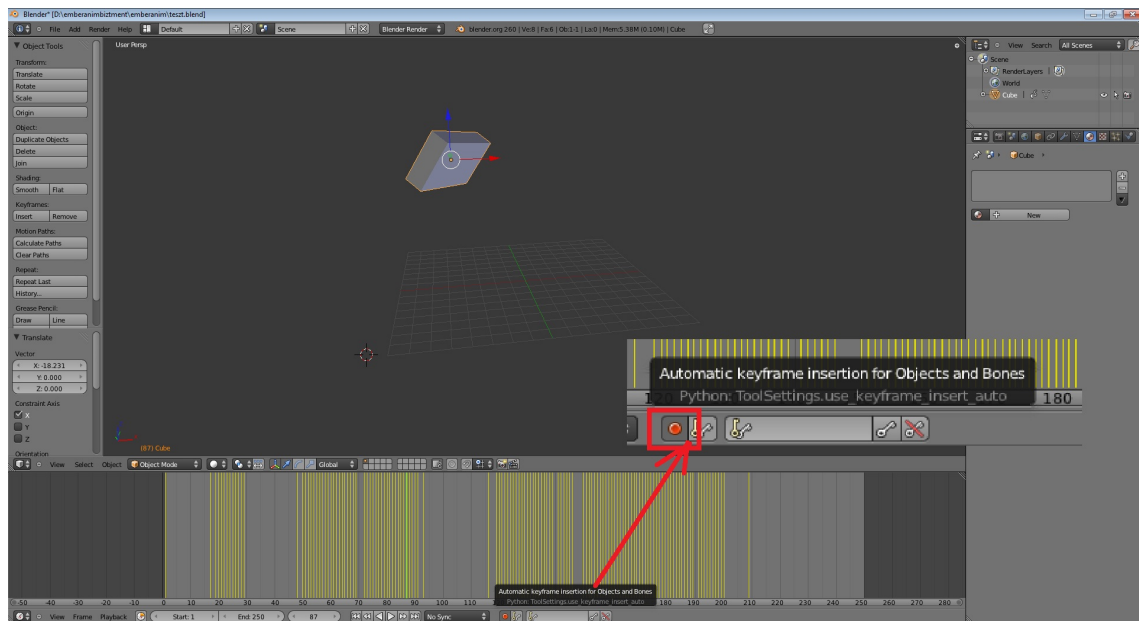
14. ábra. Keyframe Animation

Arra is van mód, hogy ne mi magunk hozzunk létre egyesével keyframe-eket, hanem ez mind automatikusan történjen. Ehhez válasszuk a Timeline alján lévő kerek és vörös *REC* gombot, majd indítsuk el az animációt. Ekkor ha transzformációkat végzünk az aktuálisan kijelölt objektum(ok)on, azok adatai rögzítésre kerülnek kulcsképkockaként arra a helyre, ahol épp a lejátszófej áll. Lásd a 15. ábrán.

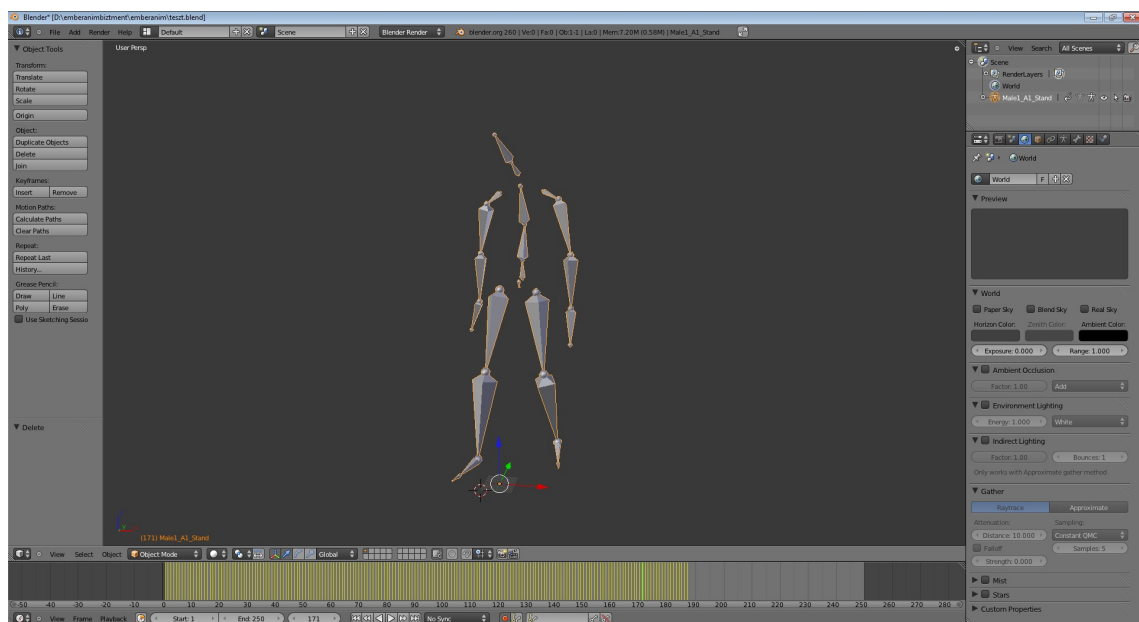
1.4.2. Importált fájl segítségével történő animálás

Az Info ablakban a File-ban található Import opció segítségével importáljunk be egy *Motion Capture* (.bvh kiterjesztésű) fájlt. Ennek eredménye a jelenetben egy *armature* (váz), melyet egy hozzá hasonló alakú mesh-hez tudunk illeszteni (lásd a 16. ábrán).

Maga a *bvh* fájlkiterjesztés a Biovision Hierarchy rövidítése. Gyakorlatilag csontváz hierarchiáról tartalmaz információt (milyen csontok, hol és hogy helyezkednek el, stb.), illetve minden egyes csonthoz saját animációt kulcsképkockák százaival, ezreivel. Maga a fájl egy szöveges dokumentum, melynek részletei a következő formában néznek ki:



15. ábra. Automatikus kulcsképkocka illesztés



16. ábra. Importált armature

HIERARCHY

ROOT Hips

{

OFFSET 13.6774 106.653 -34.4427

CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation

JOINT ToSpine

{

```

OFFSET -2.69724 7.43032 -0.144315
CHANNELS 3 Zrotation Xrotation Yrotation
JOINT Spine
{
...
}
MOTION
Frames: 187
Frame Time: 0.0333333
13.6653 106.685 -34.2172 -5.63641
...

```

Importálás után kattintsunk az armature-re, majd a Properties ablak Object data részben válasszuk a Rest Positiont, hogy az armature-t nyugalmi állapotba (emberalkatnál T-alak) állítsuk. Amennyiben nem T-alakú a Rest Position, valószínű, hogy másik bvh fájlt kell beszerezniünk. Ha ez megtörtént, létre kell hoznunk a korábban taglalt módszerekkel egy mesh-t, vagy szereznünk valahonnan, melynek alakja nagyjából hasonló az armature-höz. Úgy hajtsunk végre transzformációkat a mesh-en és/vagy az armature-ön, hogy - például emberalaknál - a láb csontjai a lábban helyezkedjenek el, stb. Ezt elvégezve jelöljük ki először a mesh-t, majd SHIFT nyomva tartása mellett az armature-t is. Ekkor nyomjunk CTRL+P-t. Válasszuk a Set Parent to Armature Deform with Automatic Weights opciókat, mellyel az armature-t alkotó minden egyes csont a hozzá legközelebb eső mesh részletet (az azt alkotó csúcsokat) fogja a legnagyobb erővel mozgatni, míg a tőle egyre távolabb esőket a legkevésbé, vagy egyáltalán nem.

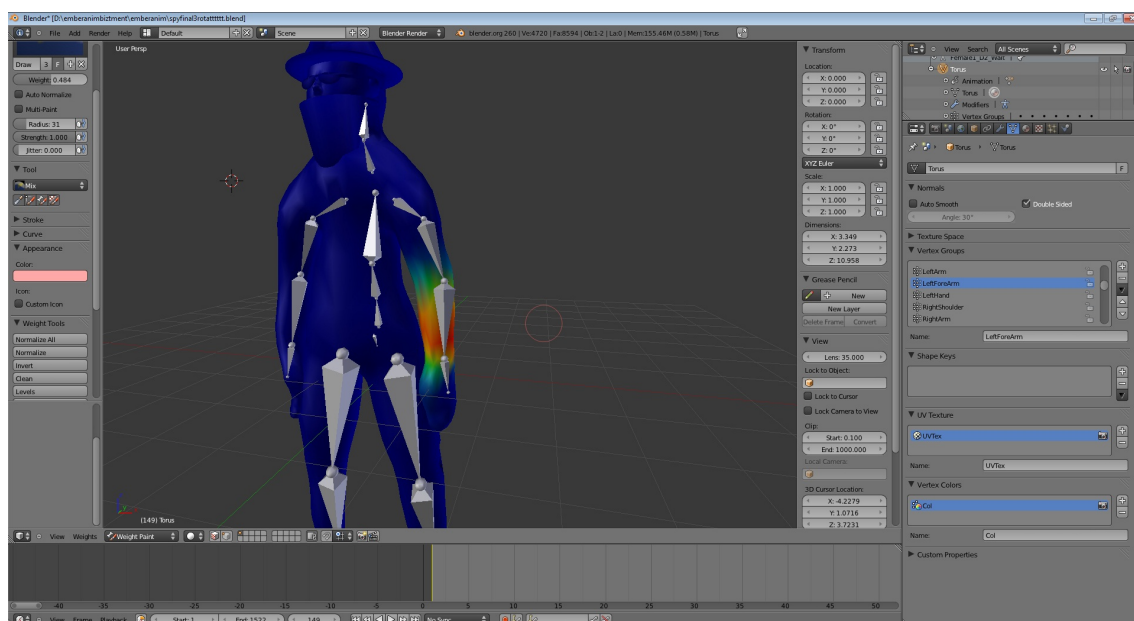
Ezt végrehajtva kattintsunk a mesh-re, és a Properties ablak Object data opciójában láthatjuk, hogy sok vertex csoportunk jött létre. Ezeknek nevei meg fognak egyezni a csontokkal. Figyeljük meg, hogy minden csonthoz tartozik egy vertex csoport.

Váltsunk át a 3D View alján a *Weight Paint* opcióra. Ekkor mesh-ünk átvált egy elég tarka színű felületté. Ahogy a vertex csoportokon végigmegyünk és sorban jelöljük ki őket, láthatjuk, hogy a csontokhoz tartozó vertex (csúcs) csoportok sorban vörösödnek el a mesh felületén.

Ennek egyszerű oka van. Weight Paint módban láthatóvá válik, mely csúcsokra hatnak a legjobban az adott csont helyzetének változásai. Vörössel vannak jelölve azok a részek, melyekre a leginkább hat, melyek együtt mozognak az azonos nevű csonttal, illetve narancssárga, sárga színnel vannak jelölve azok a csúcsok, melyekre

már csak fele olyan erővel hat a csont mozgása. A kékkel jelölt csúcsokra egyáltalán nincs hatással a változás.

Amennyiben valamelyik vertex csoport nem megfelelően néz ki és furcsa dolgokat művelnek azok a csúcsok, melyek a csont közelében helyezkednek el (például az egész objektum elindul, de egy-egy csúcs hátul marad, vagy más testrésszel együtt mozog), lehetőségünk van korrigálni manuálisan a csúcsok súlyozását. Weight Paint módban a 3D View bal oldalán hasonló opciók jelennek meg, mint Sculpt mód esetén. A Tool opciót hagyjuk változatlanul, helyette egyszerűen Weight, Radius és Strength opciókat változtassuk. Weighttel gyakorlatilag a festés színét állítjuk be, másnéven a súly erősségét. Radiusban az ecsetünk nagyságát, vagyis hogy egyszerre mekkora felületet tudjunk festeni. A Strength opció segítségével a festés gyorsaságát, hirtelenségét állíthatjuk, hogy egy kattintásra milyen sebességgel változzon a felület színe. Hatását lásd a 17. ábrán.

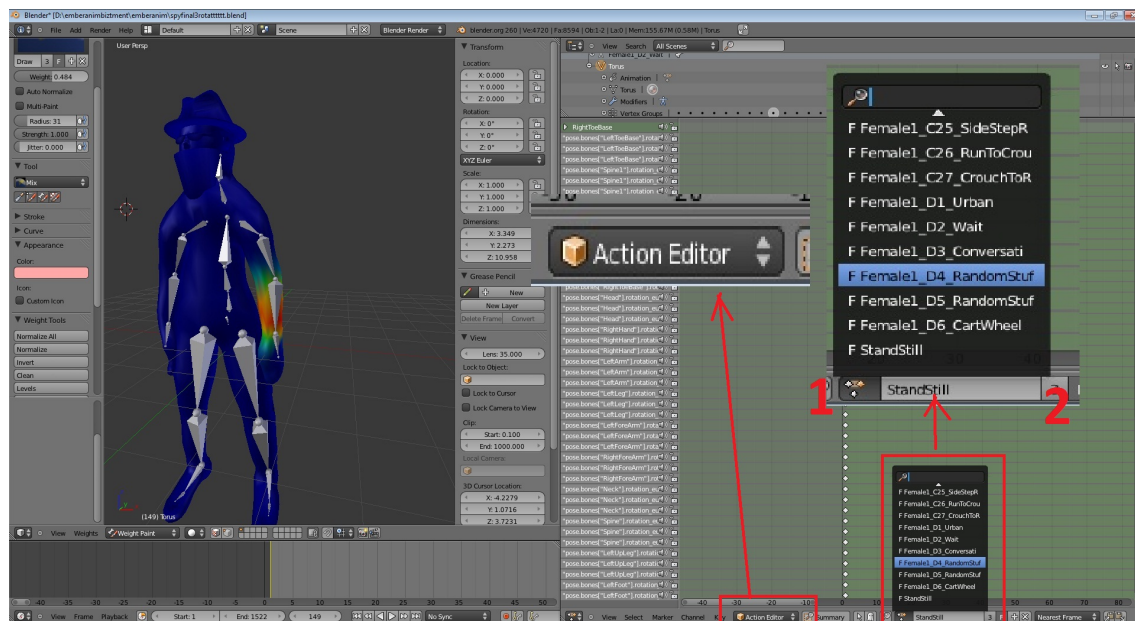


17. ábra. Csúcs csoportok súlyozása

Ha új animációt akarunk importálni egy már meglévő armature-höz, abban az esetben hozzunk létre egy új Blender jelenetet. Ehhez indítsuk el újra a Blendert és importáljuk be a bvh fület, ezután mentjük el. Az aktuális jelenetünkből jelöljük ki az armature-t, majd az Info ablak File opciójának Append részében válasszuk ki az elmentett jelenetet (a .blend fület), melyben az Action mappában megtaláljuk ezt az *action fület*.

Amennyiben ez az *append* (csatolás) megtörtént, a Properties ablakot váltsuk át *Dopesheet* ablakra, amiben láthatjuk a csontok kulcsértékeit elmentve különféle Node-okként (csomópontokként). Az ablak alján váltsunk Action Editor módba,

ahol a 18. ábrán 1-essel jelölt ikonra kattintva tudunk az aktuálisan linkelt action-ök között váltani. Átváltás után viszont a 2-essel jelölt F betűre kattintva mentsük le az adatblokkot, a gomb segítségével ez akkor is meg fog történni, ha nincs felhasználója a linkelt animációnak (tehát ha épp nincs hozzákötve egy armature-höz sem). Ha az F mellett bal oldalon 2-es vagy nagyobb szám áll, biztosak lehetünk abban, hogy van felhasználója az adatblokknak, létezik rá hivatkozás jelenetünkben.

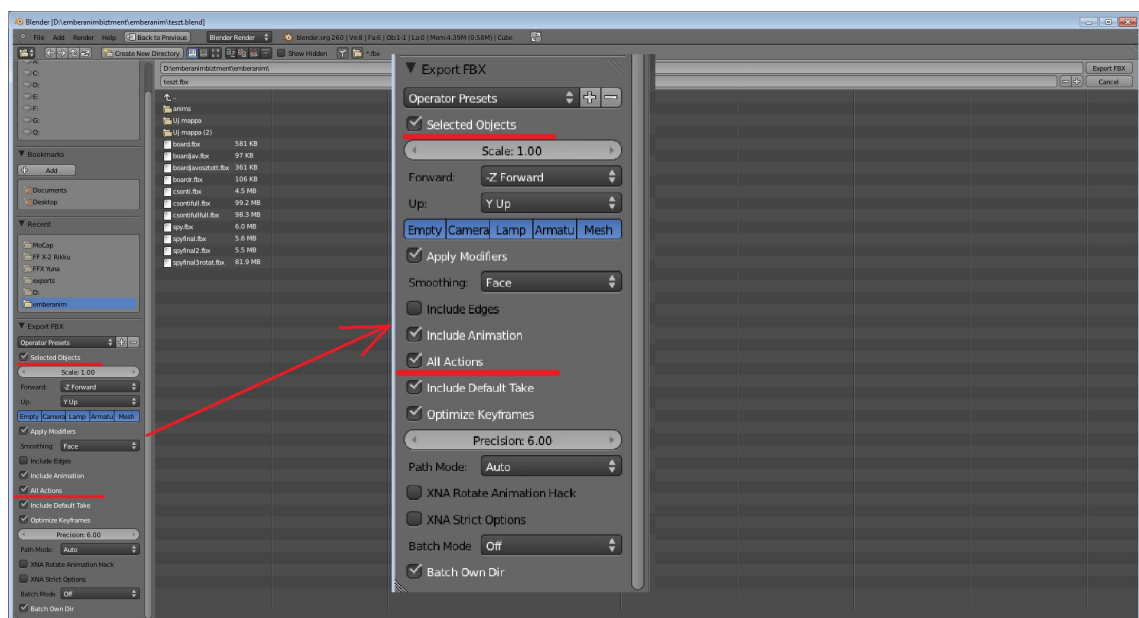


18. ábra. Action váltogatás

1.5. Exportálás

Exportálás előtt ne felejtsük el az objektumon elvégzett transzformációkat a tárolt adatokkal egyenlővé tenni a CTRL+A opción belül található transzformációk segítségével, máskülönben az új programban anomáliákat tapasztalhatunk.

Exportáláskor számtalan formátumba menthetjük ki jelenetünket (vagy bizonyos részeit). A leghasznosabb formátum az Autodesk FBX, melyet választva majdnem biztosak lehetünk abban, hogy sikeres lesz az exportálás és az új programba történő importálás. Figyeljünk arra, hogy az All actions, illetve a Selected Objects be legyenek pipálva. Lásd a [2, 17. fejezet] és a 19. ábrán.



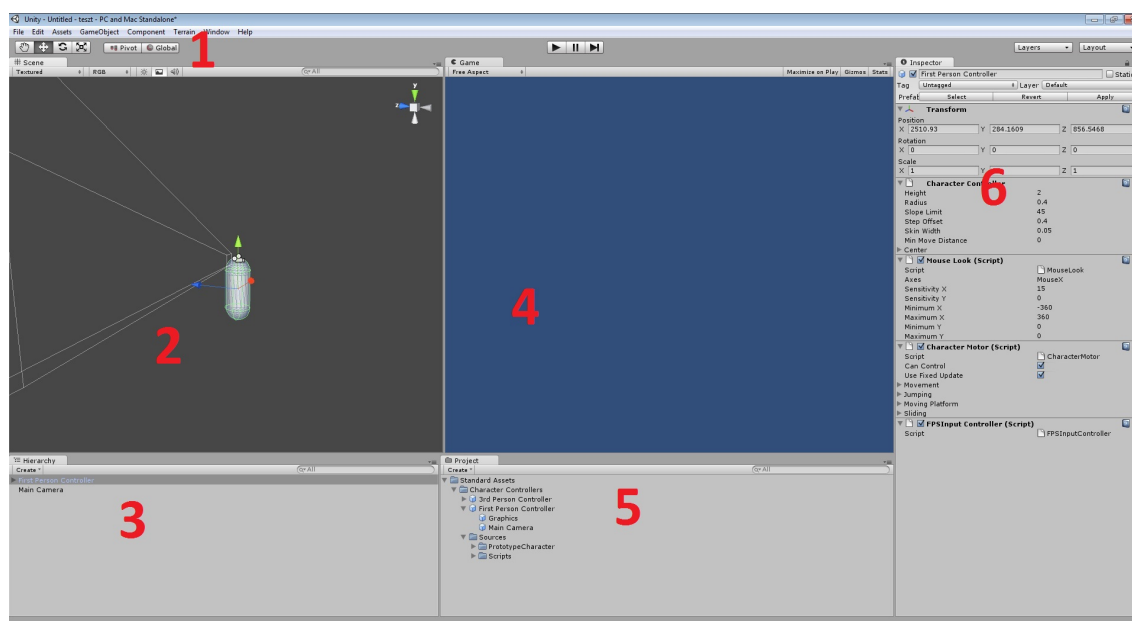
19. ábra. FBX Exportálás

2. Unity cikkek

A Blender után betekintést nyerhetünk a Unity Technologies által fejlesztett *Unity game engine*-be (játékmotorba), melyet egy, a Unityről (ejtsd: juniti) szóló cikken keresztül fogom bemutatni az Olvasó számára, rátérve a konkrét játékfejlesztés főbb mozzanataira. Lásd [3].

A Unity valójában egy integrált szerzői eszköz, mint maga a Blender, melynek a segítségével videójátékokat, valósídejű 3D animációkat és építészeti látványterveket is létrehozhatunk. Magában foglal egy szerkesztőt fejlesztésre, tervezésre és egy játékmotort, amivel az elkészült terméket futtathatjuk. A Unity ezen kívül számos díjat zsebelt be, például a Wall Street Journal 2010-es Technology Innovation Award díjat szoftver kategóriában. Számos platformra lehet vele játékot írni, úgy mint Playstation 3, Xbox 360, illetve fejlesztés alatt van a Playstation Vita támogatás is.

A Unity képernyője a következőképpen néz ki (lásd a [3, 1. fejezet]):



20. ábra. A Unity felülete

A 20. ábrán látható képernyő számozott részei a következőket jelentik:

1. Vezérléssel kapcsolatos opciók, transzformációk:

A File részben a jelenettel kapcsolatos beállításokat találjuk, amely a szokásos mentés, betöltés opciókon túl a célplatformra történő exe létrehozás képességét is magában foglalja.

Az Edit (szerkesztés) részben található megszokott gyorsbillentyűk mellett (másolás, kijelölés, duplikáció, stb.) láthatjuk, hogy a projekt beállításában

lehetőség van az irányítással kapcsolatos (Input), hangért felelős (Audio) és minőséggel foglalkozó (Quality) opciókat is szerkeszteniük.

A Quality opcióban nagyon fontos, hogy a beállított minőségben belül a Pixel Light Count értékét például 100-nak adjuk meg, ugyanis ha ez az alapértelmezetten van, könnyen lehet, hogy bizonyos fényforrások nem lesznek hatással a poligonok színének változására.

Az Assets részben előre legyártott illetve saját vagy mások által készített package-eket (csomag) importálhatunk, illetve exportálhatunk, valamint *assetekkel* (dolgokkal) is megtehetjük ugyanezt. Az asset egy csomag része lehet. Tartalmazhat textúrákat, objektumokat, animációs file-okat, stb.

A GameObject részben *gameobjecteket* (játékelemeket, objektumokat) hozhatunk létre. Ezek előre legyártott és saját objektumok is lehetnek, illetve lehetnek üresek is. Ezek hasonlóak, mint a Blender Add opciójának Mesh részében található elemei, de létrehozhatunk 3D Textet illetve GUITexture-t is. Utóbbi jó például a kamera mozgását követve élet vagy lőszer kijelzőként is.

A Component részben érdemes megemlíteni a Physics és Audio opciókat. Az Audio-ban Audio listener tulajdonsággal ruházhatunk fel egy gameobject-et, a Physicsből különféle colliderekkel (ütközési zónákkal) ruházhatjuk fel a kijelölt objektumot, valamint rápakolhatunk olyan *Joint-okat* (illesztési helyeket), melyekkel két vagy több objektumot rögzíthetünk egymáshoz.

A Terrain részben létrehozhatunk egy olyan síkot, melyet könnyedén formálhatunk (gyakorlatilag egy sculптоzható felület, de pályarészek kialakítására van kitalálva). A Window részben *Lightmappingot* (fénytérképet) hozhatunk létre, gyakorlatilag a fényforrásoktól árnyékolt felületek lenyomatait bake-elik (rásütik) egy képre, mely később térképként szolgál és optimálisabb megoldást nyújt, mintha valós időben kellene a grafikus motornak kiszámítania, mely felületrészek vannak árnyékban és melyek nem.

A Help rész általános segítségnyújtást, információkat nyújt a felhasználó számára. A menüsáv és eszköztár alatti részben bal oldalon a kamera mozgatására szolgáló kéz ikont láthatjuk (melyre GÖRGÖGOMB segítségével gyorsan válthatunk), valamint az alapvető transzformációkat láthatjuk, balról jobbra eltolás, forgatás illetve skálázás.

2. Scene:

Ez az ablak gyakorlatilag egyenértékű a Blender 3D View ablakával.

3. Hierarchy:

A Blender Outliner ablakához hasonlít. Itt viszont semmi nem állítható, csak kijelölhető.

4. Game:

Az aktív kamerából ezt a képet látjuk játékon belül. Ha fel van helyezve egy *First Person Controllerünk* a jelenetre, akkor az abban lévő kamera lesz az aktív kamera és azt láthatjuk a Game ablakban, amit ő lát. Ezenkívül itt nem láthatók a hivatalosan *Gizmo-knak* (bigyóknak) elnevezett jelölések sem, mint a fényforrást jelző kis nap képe, vagy a kamera ikonja.

5. Project:

Itt találhatóak azok a package-ek, textúrák, materialok, objektumok (és mindenféle asset) illetve scriptek („forgatókönyv”, valójában forráskód), melyeket felhasználhatók a tettünk projektünkben.

6. Inspector:

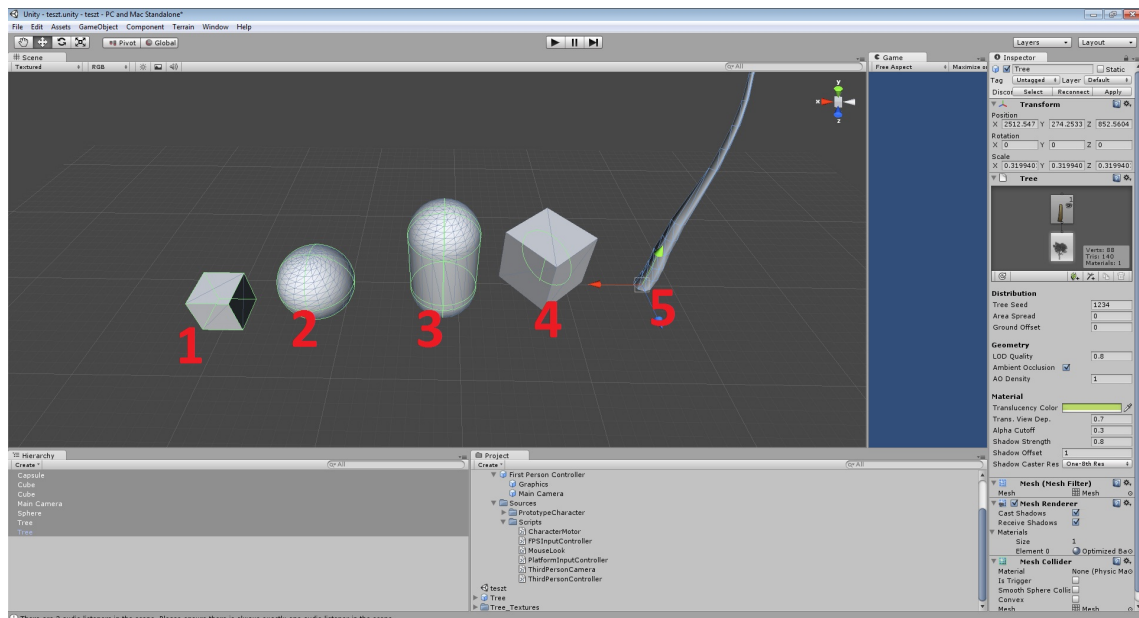
Az aktuálisan kijelölt objektumról kaphatunk részletes információkat. Az ablak felső részében az objektum neve mellett láthatóvá illetve láthatatlanná tehetjük. címkét tehetünk rá, illetve ráhelyezhetjük egy *layerre* (rétegre). Előbbi scripteknél az objektumra hivatkozásnál, utóbbi például fényforrások szeparációjánál jelentős. Alatta transzformációs értékeket láthatunk (ezek lehetnek helyi koordináták is, nem csak világkoordináták, amennyiben ez egy gyerekobjektuma egy szülőnek, vagyis ha a hierarchiában valamelyiben helyezkedik el). Ezalatt az objektumra húzott különféle scriptek publikus mezőit állíthatjuk be, illetve egyéb tulajdonságok lehetnek, saját opciókkal.

Ezek ismeretében rá tudunk térni a program használatára.

2.1. Ütközési zónák

Másnéven *colliderek*. Az ütközési zónák segítenek abban, hogy ha egy objektumra hat a gravitáció (Inspectoron belül láthatjuk, hogy a *Rigidbody*, másnéven merev test komponensében a Use Gravity be van pipálva), akkor ne zuhanjon a végtelenségig, ne essen át a pályán, hanem álljon meg, ütközzön neki valaminek (lásd a [3, 1. fejezet]).

Ütközési zónából többféle van, de alakja, elhelyezkedése függhet attól, hogy egy impoltált objektumot az exportfájlt készítő programban hogy transzformáltunk. Ugyanis előfordulhat, hogy az ütközési zóna transzformált lesz, ekkor a szülő programban kell transzformációkat végrehajtanunk. Alapvetően hatféle collider érhető el



21. ábra. Különféle colliderek

Unityben, ahogy a 21. ábrán is látható, a *character controller* (3), amely egy speciális collider, *box* (1), *sphere* (2), *capsule* (3), *mesh* (5) és *wheel collider* (4). Ezeket akkor láthatjuk, ha kijelölünk egy objektumot. Amennyiben nincs rajta semelyik collider, rátehetünk egyet vagy többet is. Ehhez válasszuk a Component opcióból a Physics részt, melyben megtaláljuk a különféle colliderek listáját. Ha már van egy collider az adott objektumon, a Unity rá fog kérdezni, hogy szeretnénk-e helyettesíteni az előzőt az új colliderrel vagy hozzá szeretnénk adni az előző(ek) mellé.

A Character Controller olyan collider, melynek kapszula alakja van. Lényegében ez egy capsule collider. A character controller a First Person Controller (első személyű vezérlő) és Third Person Controller (harmadik személyű vezérlő) colliderje. Mivel mindkét dolog egy emberalakot feltételez, emiatt kapszula alakúak.

A Capsule collider és a character controller colliderek között viszont alakjuk megegyezése mellett van egy nagy különbség, mégpedig az, hogy más opcióik vannak. Mindkettőnél állíthatjuk a magasságot és sugarat (szélességet), viszont amíg a character controllernél különféle mozgással kapcsolatos opciókat találunk (az irányíthatóságból adódóan), addig a capsule collidernél meghatározhatjuk azt, hogy trigger-e (kapcsoló). A triggerrekről a későbbiekben lesz szó.

A Box, Sphere, Mesh és Wheel colliderek esetében is létezik a trigger jelenlétére irányuló opció. A box collider egy téglatestet alkot, amelyen belül fog elhelyezkedni az objektum. Ezenkívül x, y illetve z koordinátában adhatóak meg méretei.

A Sphere collider egy gömbalakot formál. Sugarát lehetséges egyedül szabályozni. Szélessége miatt, illetve keskenység esetén előforduló, objektumból kilógó részek miatt csak gömb alakú objektumokra érdemes használni.

A Mesh collider az objektum alakját veszi fel, illetve pont annyi poligonból fog állni, mint amennyiből az objektum áll. Célszerű bonyolult felépítésű, animációnélküli objektumokhoz, viszont amelyek valamilyen mozgást hajtanak végre, azoknál nem ajánlott, ugyanis az objektum alapállapotának a képét fogja felvenni a collider és nem fog az objektummal együtt változni.

A Wheel collider kör alakú, és a sugara mellett megadhatjuk, hogy mennyi legyen a suspension distance (kerékfelfüggesztési távolság), a suspension spring (felfüggesztés rugalmassága), amin belül a rugalmasság, a csillapítás és a felfüggesztés nyugalmi állapotának a helyzetét adhatjuk meg. Ezenkívül szabályozhatjuk a kerék tömegét, illetve a súrlódás milyenségét. Ezekből az attribútumokból következtethető, hogy a wheel collider tipikusan előnyös autók kerekeinél, mely egy könnyen módosítható felületet biztosít például autószimulátorok fejlesztéséhez.

2.2. Scriptek

A scriptek, másnéven forgatókönyvek gyakorlatilag események leírására szolgálnak, ezért hívják így őket. A scriptek programkódok, ebből adódóan megkövetelik a programozói tudást. Programoktól függően számtalan programozási nyelv segítségével hozhatunk létre scripteket, melyek az adott programon belül, a programhoz adott *API-t* (application programming interface - alkalmazásprogramozási felület), vagyis az előre megírt metódusokat, függvényeket felhasználhatják, illetve újakat tartalmazhatnak. Megírásuk egy, a kód programozási nyelvéhez optimalizált *IDE-ben* (integrated development environment - integrált fejlesztői környezet) történik.

Az IDE-k főbb tulajdonságai a következők lehetnek:

Code completion:

Kódkiegészítés

Code folding:

Kód blokkok egységbe zárása

Code browser:

Segítségével meg lehet nézni, milyen osztályok, metódusok, property-k vannak projektünkben

Code intantation:

Sorok behúzása

Search & replace:.

Segítségével bizonyos szavakra, szórészekre rákereshetünk a kódra, illetve változók, metódusok neveit lecserélhetjük

Debugger:.

A kód soronkénti futtatását teszi lehetővé, segítségével figyelni lehet a változásokat

Kommentezés:.

Kiegészíthetjük kódunkat magyarázatokkal

Syntax highlight:.

Kódszínezés az átláthatóság érdekében

Code snippet:.

Rövid (mondjuk kétbetűs) szavak kóddá kiegészítése

A Unity IDE-je a Scintilla nevű ingyenes kódszerkesztő Unity-re módosított változata, a *UniSciTe*. Segítségével írhatunk a Unity-s projektünkhöz C# (Unityscript), Java, Boo nyelveken is scripteket. Természetesen ezekhez a scriptekhez az adott nyelvhez alkalmasabb IDE-t is választhatunk (C# script esetén például a Microsoft Visual Studiot, mely egy igen fejlett IDE-ként van számontartva), a Unity csupán egy alternatívát kínál a kódok írásához (lásd a [3, 1. fejezet]).

A UniSciTe tulajdonságait tekintve tartalmaz code completion-t, code folding-ot, code intantiationt, search & replace-t, kommentezést, syntax highlightot is.

Scriptet könnyűszerrel létrehozhatunk, ha a Project ablakban jobb egérgombbal kattintunk és kiválasztjuk a Create opció belül a C Sharp scriptet (vagy a másik kettő közül valamelyiket).

Magukat a scripteket tekintve a Unityscriptek alapértelmezetten a következők szerint épülnek fel:

```
using UnityEngine;
/*namespace, amely a Unity API-ját tartalmazza*/

using System.Collections;
/*különbféle adatszerkezeteket tartalmazó névtér*/

public class ScriptNeve : MonoBehaviour {
/*publikos osztály, melyet a MonoBehaviour osztályból
származtattunk le*/
```

```

void Update()
{
    /*minden egyes képkocka kirajzolásakor
    lefutó metódus. Ciklusnak fogható
    fel. Emiatt nem érdemes egyéb
    ciklusokat beleírni, mert az az FPS
    (frame per screen – képkocka per
    másodperc) érték csökkenését
    eredményezheti*/
}

void Start()
{
    /*a script lefutásának legelején fut
    le, itt változódeklarációkat (változók létrehozását)
    végezhetünk.*/
}

}

```

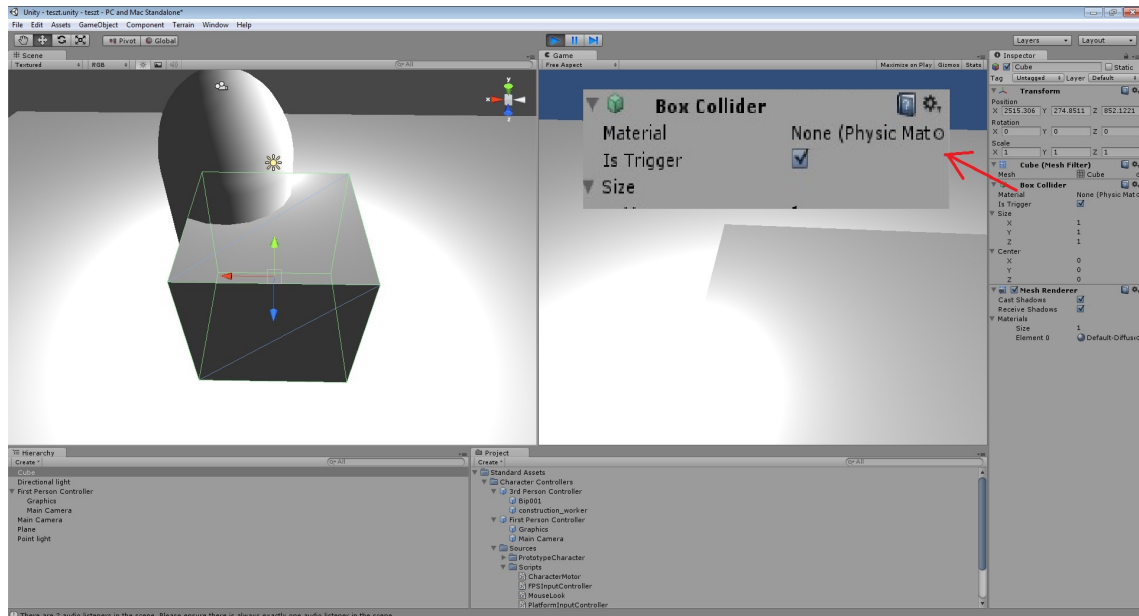
Az így létrehozott script egy szöveges állomány lesz, melynek (C# script esetén) „cs” kiterjesztése van. Természetesen szabadon módosítható IDE nélkül is, viszont akkor kevésbé fogjuk tudni átláthatóan szerkeszteni illetve hibákat keresni benne.

2.3. Kapcsolók

A *triggerek* (kapcsolók) olyan colliderek, amelyeken áthatolhat egy másik collider vagy trigger. Erre az áthatolásra viszont már tudunk valamilyen, az objektumra aggatott scripten belül hivatkozni (a hozzárendeléshez elég egyszerűen az objektumra húzni a megfelelő scriptet). Segítségével elérhető az, hogy a behatóló objektumnak a környezet számára elérhető (publikus) tulajdonságaira hivatkozhassunk, máshogy megfogalmazva megváltoztathassuk a behatóló vagy más objektumok belső állapotát. Könnyebben elképzelhető, ha úgy tekintünk rá, mint egy konkrét kapcsolóra, melyet ha átkapcsolunk, valamilyen eseményt vagy eseményeket indíthatunk el (mint egy villanykapcsoló).

Jelentősége akkora, hogy gyakorlatilag új és régi játékokban egyaránt használják. Triggerekkel valósítható meg egy mozgásérzékelős ajtó szimulációja, egy *scrip-*

telt event (esemény) elindítása, ebből adódóan mondhatjuk azt, hogy a legtöbb, valamilyen történéshez köthető, ezáltal létrejövő esemény (mely akár látható, akár láthatatlan, háttérben zajló is lehet) triggerelt esemény, ugyanis valaminek hatására hajtodik végre feltételesen. Lásd a [3, 4. fejezet] és a 22. ábrán.



22. ábra. Trigger

Működésének konkrét bemutatásához tekintsük meg a következő scriptet. Ezt egyszerűen drag & drop módszerrel (bal egérgombot nyomva tartva elengedni a kívánt objektum felett) húzzuk arra az objektumra, amely a triggerünk.

```
using UnityEngine;
using System.Collections;

public class Script1 : MonoBehaviour {

    public bool b=true;

    /*egy b bool típusú változónk van, mely igaz értéken áll*/
    void OnTriggerEnter(Collider other){
        if(other.name=="First_Person_Controller"){
            Script2 s;
            s=other.gameObject.GetComponent<Script2 >();
            s.b=this.b;
            /*
            Ez a metódus akkor fut le, amikor
```

```

        egy tetszőleges colliderrel
        rendelkező objektum behatol
        a scriptet tartalmazó objektum
        colliderébe. Itt a megfelelő kezelés
        érdekében szükséges megvizsgálnunk,
        hogy az az objektum hatolt-e be,
        amelyikre a feladatunk
        szempontjából figyelniünk kell.
        Ekkor meghívjuk azt a Script2-t,
        ami a behatoló objektumon van,
        majd megváltoztatjuk a b nevű
        attribútumát (tulajdonságát) a
        saját értékünkre
        */
    }
}

void OnTriggerExit(Collider other){
    if(other.name=="First_Person_Controller"){
        Script2 s=other.gameObject.GetComponent<Script2>();
        s.b=!this.b;
        /*
        Ez értelemszerűen egy behatoló
        objektum elhagyásakor fut le,
        mely lehet, hogy épp az előzőleg
        behatoló volt, de
        az is lehet, hogy egy régebben
        behatolt objektum. Ezért
        fontos itt is megvizsgálni, hogy
        az hagyta-e el
        az objektumot, amelyikre mi
        gondolunk.
        Ezután az előzőleg megváltoztatott
        tulajdonságát ellentétes
        értékűre állítottuk.
        */
    }
}

```

```
}
```

A behatoló objektumnak a trigger szempontjából teljesen mindegy a felépítése, csupán az a fontos számára, hogy a feltételvizsgálatkor a neve megegyezzen a vizsgált névvel, valamint számára elérhető (publikus) legyen az az attribútum, melynek értékét át szeretné állítani a sajátjára. Ezáltal az egyetlen, működéshez szükséges dolgot fog példánkban tartalmazni ez a kód:

```
using UnityEngine;
using System.Collections;

public class Script2: MonoBehaviour {

    public bool b;

    /*a mező, melynek értéke behatoláskor
    megkapja a trigger értékét, majd
    kilépéskor az ellentétes értéket kapja
    meg
    */
}
```

Irodalomjegyzék

- [1] Roland Hess, Blender Foundations: *The Essential Guide to Learning Blender 2.5*, Elsevier, Inc., 2010.
- [2] Tony Mullen: *Introducing Character Animation With Blender*, Wiley Publishing, Inc., 2007.
- [3] Will Goldstone: *Unity Game Development Essentials*, Packt Publishing Ltd., 2009.