Debreceni Egyetem

Informatikai Kar

Háromdimenziós komputergrafika a modern webböngészőkben

Belső nézetű lövöldözős játék AngularJS és three.js segítségével

Témavezető: Balla Tibor Készítette: Martyin Kornél  
Egyetemi tanársegéd a Programtervező Informatikus szak hallgatója

Tartalomjegyzék

[Szójegyzék 1](#_Toc449280996)

[Betűszavak 1](#_Toc449280997)

[Kifejezések 1](#_Toc449280998)

[Bevezetés 3](#_Toc449280999)

[Célkitűzés 4](#_Toc449281000)

[Történelmi áttekintés 5](#_Toc449281001)

[Matematikai alapok 9](#_Toc449281002)

[Testek leképezése 9](#_Toc449281003)

[A virtuális tér megjelenítése 10](#_Toc449281004)

[Homogén koordináták 10](#_Toc449281005)

[Ponttranszformációk 10](#_Toc449281006)

[Középpontos vetítés 11](#_Toc449281007)

[A játék 12](#_Toc449281008)

[A játék implementációja 12](#_Toc449281009)

[A szoftver architektúrája 13](#_Toc449281010)

[Szerver 13](#_Toc449281011)

[Kliens 15](#_Toc449281012)

[A játékkliens szoftveres megvalósítása 16](#_Toc449281013)

[Modulok felépítése 16](#_Toc449281014)

[Erőforrás-betöltő modul (fps\_game.loaders) 16](#_Toc449281015)

[Generátor modul (fps\_game.generators) 16](#_Toc449281016)

[Képalkotó modul (fps\_game.rendering) 18](#_Toc449281017)

[Játékos modul (fps\_game.player) 18](#_Toc449281018)

[Játék modul (fps\_game.game) 22](#_Toc449281019)

[Hálózat modul (fps\_game.network) 23](#_Toc449281020)

[Modulok hierarchiája és illeszkedése 23](#_Toc449281021)

[Konklúzió 24](#_Toc449281022)

[Irodalomjegyzék 25](#_Toc449281023)

[Függelék 26](#_Toc449281024)

# Szójegyzék

## Betűszavak

3D – Térbeli, háromdimenziós

ACM – Számítástechnikai szövetség (Association for Computer Machinery)

API – Applikációfejlesztési interfész (Application Programming Interface)

CNC – Programozható szerszámgép (Computer Numerical Control)

CRT – Katódsugárcsöves megjelenítési technológia (Cathode Ray Tube)

CSS – Kaszkádolt stíluslapok (Cascading Style Sheets)

DirectX – A Microsoft saját fejlesztésű API-ja, multimédiás alkalmazások készítéséhez.

GPU – Grafikus számítási egység (Graphics Processing Unit)

GPGPU – Grafikus feldolgozóegység használata általános célú számítások elvégzésére (General-Purpose computing on Graphics Processing Units)

HTML – szabványosított nyelv, HTML oldalak implementálásához (HyperText Markup Language)

HTTP – Kommunikációs protokoll HTML kiszolgálásához (HyperText Transfer Protocol)

JS – Szkriptnyelv, weboldalak, webes applikációk fejlesztésére alkalmas nyelv, melynek interpretere a böngészőben fut.(Javascript)

MIT – Massachusetts Institute of Technology

PC – Személyi számítógép (Personal Computer)

RAM –Írható-olvasható memóriaegység (Random Acces Memory)

SIGGRAPH – a kifejezés grafikus és interaktív technológiák specialistáinak társasága címeként kezdte, mára már nagy számban látogatott fesztivállá nőtte ki magát (Special Interest Group on GRAPHics and Interactive Techniques)

WEB

WEBGL – böngészők által nyújtott interfész (fejlesztői API) a képalkotó hardver erőforrásaihoz ad hozzáférést.

## Kifejezések

vertex – egy térbeli pontot képviselő objektum.

poligon – több képpont között kialakított háromszögekből vagy négyszögekből álló felület

transzformációs mátrixok – egy lineáris transzformáció függvényének, vagy több kompozíciójának hatását leíró szám-mátrix

skin – modell manipulációs technika, csontváz-alapú metamorfózist valósít meg.

# Bevezetés

Az informatikai eszközök elterjedése, fejlődése most már a mindennapi élet részévé tette a kisebb – nagyobb méretű számítógépek használatát, ha csak a mobiltelefonokra, táblagépekre, okosórákra vagy laptopokra gondolunk. Moore törvénye, miszerint a számítási kapacitás minden évben duplázódni fog, kiállta az idő próbáját, manapság a zsebeinkben olyan készülékek vannak, amik másodpercenként több milliárd alapművelet végrehajtására képesek.

A számítási kapacitás növekedésével a komputergrafika is egyre nagyobb teret nyert. A számítógéppel előállítható grafikák egyre szofisztikáltabbak, részletgazdagabbak lettek, ma már fotorealisztikus képek előállítása sem okoz gondot egy középkategóriás játékkonzol sőt, egy okostelefon számára sem.



. ábra – Fotorealisztikus komputergrafika

Nem csak a grafikák előállítási technológiája fejlődött robbanásszerű mértékben az elmúlt három évtizedben. A web technológiai fejlődése következményeként az interaktív tartalmakban is egyre nagyobb szerepet játszik a 3D megjelenítés. Sorra jelennek meg az olyan webes alkalmazások, mint például termékek promóciós weboldalai, lakástervező szoftverek, autóverseny szimulátorok vagy vizuális művészeti alkotások, melyek térbeli megjelenítést alkalmazva gazdagítják a felhasználói élményt.

Az ember agya a tér érzetét a két szemével észlelt, fény által keltett - szemenként különböző – ingerület feldolgozásával állítja elő. Manapság a háromdimenziós média térbeli szemléltetésén dolgozik szinte minden vizuális eszközöket gyártó cég, sőt az előállított mesterséges képet némely eszköz a valós életben észlelt képre vetíti, így kibővítve a valóság látványát a virtuális térelemekkel.

# Célkitűzés

Szakdolgozatom a webes böngészők által támogatott WEBGL technológia képességeit hivatott bemutatni egy erre a célra elkészített webalkalmazáson keresztül. Mielőtt belevágnék a szoftver elemzésébe, egy rövidebb történelmi áttekintő során bemutatom a számítógépes grafika fejlődéstörténetét majd a matematikai háttér alapjait is, amivel képesek vagyunk a számítógép segítségével előállítani a tér látszatát.

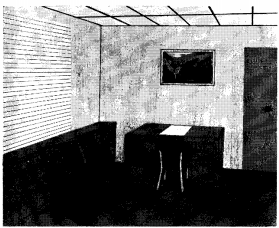
A dolgozat elméleti jelentősége a térbeli ábrázolás matematikai alapjainak bemutatása, és a komputergrafika történelmének feltárása, míg gyakorlati szempontból a bemutatott elmélet alapján megvalósított interaktív szoftver előállítása.

Célom a technológiában rejlő potenciál demonstrációja, melyet a szoftver elkészítésével és a forráskód valamint a felhasznált médium elemző leírásával kívánok elérni.

# Történelmi áttekintés

Az áttekintés során a számítógéppel előállított grafikus média fejlődését fogom bemutatni a jelentősebb mérföldköveken keresztül. Fontos megjegyezni, hogy csak egy szűk keresztmetszetét mutatom be a technológiának, mégpedig az ember által elkészített térbeli modellek vagy számítógéppel segített modellezéssel előállított grafikák vizualizációs technológiáját.

Az 1940- es évek hajnalán jelentek meg az első olyan képek, melyek számítógép közreműködésével jöttek létre. Ezen képek megalkotását mechanikus vagy analóg komputerek segítették, de nem nevezhetőek tisztán számítógépes grafikáknak, általában más fotografikus vagy rajzművészeti technika elegyével készültek el.

Az első kép, melynek elkészítése tisztán számítógépes kalkulációk alapján készült el, 1946-ban jelent meg, amit az észak-amerikai világítástechnikai mérnökök konferenciáján mutattak be.

2. ábra - Luminous-Ceiling Lighting - Parry Moon, Domina Eberle Spencer, 1949 Augusztus

A képet Parry Moon és Domina Eberle Spencer, az MIT alkalmazott matematika professzorai állították elő. A kép egy szoba megvilágítását illusztrálja, így bemutatva a fénysugárzás - képleteik alapján számolt – megvilágító erejét. Mivel a korabeli számítástechnikai eszközök között nem volt megfelelő megjelenítő eszköz, a képet Munsell mintapapírokból kivágva, összeragasztva állították elő, így demonstrálva matematikai modelljüket.

Ben Laposky 1950-ben analóg számítógép és oszcilloszkóp segítségével hullámformákat jelenített meg és fotózott le.

1951-ben vektorszkóp típusú CRT kijelzőt kapcsolnak az MIT Whirlwind nevű számítógépére. Ekkor már a General Motors kutatás-fejlesztési részlege is felfigyel a számítógépes grafika jelentőségére, tanulmányt készítenek a számítógéppel támogatott grafikai alkalmazások szerepéről. Ez a tanulmány vezet 1959-ben a DAC-1 (Design Augmented by Computers) kifejlesztéséhez, mely az első számítógéppel támogatott rajzoló rendszer. Ez a rendszer képes volt számítógép vezérlésű CNC-munkagépek irányítására a megrajzolt tervek alapján.

1960: William Fetter, a Boeing alkalmazottja először használja a komputergrafika szóösszetételt, melyet az általuk létrehozott, emberi testet és egy repülőgép pilótafülkéjét illusztráló 3 dimenziós modell megalkotása során talált fel. A repülőgép főbb pontjainak koordinátáinak bevitele után perspektivikus nézetet állított elő, melyet a számítógép számolt ki. Egy évvel később, Steve Russell, Slug Russell, Shag Graetz, és Alan Kotok, az MIT hallgatói elkészítették az első híres komputergrafikus játékot, a Spacewar-t.

A magyar származású Charles Csuri 1965-ben elindította a komputergrafikai kurzust az Ohio Állami Egyetemen. A Smithsonian magazin 1995. februári számában Paul Trachtman így írt Charles-ról: „a digitális művészet és számítógépes animáció szülőatyja”. Rendkívül érdeklődő és nyitott ember, a számítógépes grafika minden területén úttörő munkát végzett, a kalligráfia vonalak térbeli megjelenítésétől a számítógéppel generált szobrokon át a kerámiaművészetig.

1965: Jack Bresenham feltalálta az „ideális vonal” rajzolásának algoritmusát - kör rajzolása pixelgrafikus megjelenítőn.

1966-ban a Harvard egyetem professzora, Ivan Sutherland néhány diákja segítségével megalkotta az első virtuális valóságot, bár az csak a szoba körvonalait és a négy égtájat jelenítette meg.

1967-ben Kepes György megalapította a Fejlett Vizuális Tanulmányok Központját az MIT egyetemen.

A 70-es években a képpuffer RAM modulok árának relatív csökkenésével bővült a számítógéppel generált képek piaca. 1973-ban megalakult az ACM SIGGRAPH, mely akkor csak néhány specialistából álló csoport, napjainkra nemzetközi közösséggé nőtte ki magát a komputergrafika és az interaktív eszközök terén.

1974: Phong Bui-Toung kifejlesztette a Phong árnyalási metódust.

1975-ben, 20 évnyi munka után, Dr. Benoit Mandelbrot publikálta elméletét a fraktál sorozatokról.

1977: Larry Cuba előállította a halálcsillag szimulációját a Star Wars c. filmhez.

1978: Jim Blinn szakdolgozata részeként publikálta a hepehupás felszín árnyalási technikáját (bump mapping).

1979: a Bell Labs és a Cornell Egyetem kooperációjának eredményeként született meg a fénysugár követésének technológiája, a raytracing.

Eközben a SIGGRAPH konferencián évről évre növekvő számú audiencia és kiállító jelenik meg, 1981-ben ez a szám 14000 látogatót és 124 kiállítót jelentett.

1982: Dan Drake és John Walker megalapították az Autodesk céget, majd nemsokára bemutatják az AutoCAD 1.0 szoftvert,

1984: A Cornell egyetemen, Don Greenberg által vezetett csapat bemutatott egy új globális megvilágítási technikát (Radiosity lighting).

1988: Aranykorát élte a videojáték; megjelentek az első dedikált valósidejű háromdimenziós grafikus egységek, ami az asztali számítógépekben található videokártyák elődjei voltak.  
A személyi számítógépek elterjedésével felvirágzott a digitális művészet is, megjelent a demoscene, terjedtek a shareware szoftverek.

A 90-es években a PC-k képessé váltak a háromdimenziós képek előállítására. Tömegesen elterjedtek a modellezőszoftverek, a képi minőség hatalmas fejlődésen ment keresztül. Egyre fontosabbá vált a videovezérlők teljesítménye, a játékszoftverek és a multimédiás tartalmak is előszeretettel kezdték el használni a térbeli megjelenítést, ezáltal a hardwaret is. A korábbi komputergrafikára specializált Silicon Graphics munkaállomások helyét erőteljes Microsoft Windows vagy Apple Macintosh rendszerű gépek váltották fel, megugrott az olyan modellezőszoftverek népszerűsége, mint például az Autodesk 3D studio.

1992: A Silicon Graphics fejlesztéseként megjelent az OpenGL alkalmazásfejlesztési interfész, mely kifejezetten 3D megjelenítésű programok implementációjához készült.

1995: Megjelent a DirectX, a Microsoft saját fejlesztésű API-ja is, mely a háromdimenziós megjelenítés mellett audió és hálózati erőforrások használatához is támogatást nyújt Windows alapú rendszereken történő fejlesztéshez.

1996-ban Krishnamurty és Levoy feltalálták a normal mapping árnyalási eljárást, a bump mapping továbbfejlesztett változatát.

1999: az Nvidia értékesítette az első GPU (Graphics Processing Unit) típusú asztali számítógépekbe szánt videokártyáját, a GeForce 256-ot.

A 2000-es években a számítógéppel előállított képek annyira mindennapossá váltak, hogy a televíziós reklámok körében is nagy számban megjelent a technológia használata. A PC-s grafikák elérték az esztétika hipotézise szerinti ún. „uncanney valley” tartományt, ez az a lélektani határ, amikor a virtuális kép annyira hasonlít az igazihoz, hogy némely emberben a valósághoz történő asszociáció miatt undort, félelmet válthat ki a látvány. A tudomány is profitált a GPU-k számítási kapacitásából: kidolgoztak egy olyan technológiát, mellyel a CPU és a GPU két irányban oszthat meg nagy adatmennyiséget, így felgyorsítva a számítógépes szimulációk számításait (GPGPU).

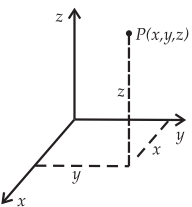
2004: Az Apple, a Mozilla és az Opera munkatársai munkacsoportot hozntak létre a HTML5 kifejlesztéséhez, 2008-ban a Firefox böngésző volt az első, ami támogatta a HTML5-öt, így a 3D hivatalosan is betört a WEB-es technológiák közé, a canvas HTML elem révén a böngésző képessé vált a számítógép grafikus erőforrásait használni térbeli megjelenítésre (webGL).

Szakdolgozatomban az imént említett, böngészőkben használható technológia képességeit mutatom be.

# Matematikai alapok

A matematikai háttér bemutatása során a térbeli elemek kijelzőn történő megjelenítésének alapjait, valamint a szoftver elkészítése közben használt technikákat fogom részletezni.

## Testek leképezése

A modellek vektorgrafikus leírásához a legkisebb alapegység a pont. Egy P pont meghatározásához a Descartes térben három komponenst használunk, így: P(x,y,z) | x,y,z , ahol a három komponens az origótól a saját tengelyén mért távolság, így a P pont a Descartes-féle koordinátarendszerben egyértelműen definiálható.

3. ábra - Pont a Descartes - térben

Legyen P és Q a fentiek szerint konstruált térbeli pont. Tegyük fel, hogy a két pont nem egyenlő, azaz legalább egy komponensük nem egyezik meg. P és Q ekkor egyértelműen meghatároz egy egyenest, egy szakaszt és ha P és Q sorrendje is meghatározott, akkor egy vektort is.

Vegyük P, Q és R pontokat, melyek mind különbözőek. Ez a három pont meghatároz egy síkot, és a síkon egy háromszöget. Ha ezeknek a pontoknak a sorrendje is meghatározott, akkor a háromszög síkjának normálvektora is kiszámolható, egy csúcsból kiinduló élei vektorként történő keresztszorzatából. A háromszög síkjának normálvektora fogja meghatározni a háromszög azon felületét, mely a beérkező fényt visszaveri. Ilyen háromszögeket felhasználva, ha egynél több háromszöget veszünk és feltételezzük, hogy minden háromszögnek legalább egy éle egy másik háromszögnek legfeljebb egy élével megegyezik, akkor poligonokat konstruálhatunk. Ezek a poligonok alkalmasak bizonyos testek felszínének leírásához és gyakran használatosak a számítógépi grafikában. A dolgozatomhoz megalkotott szoftver is poligonokból építi fel a testek felszínét, a görbe felületek is nagyobb részletességű poligonokból származtathatóak.

## A virtuális tér megjelenítése

### Homogén koordináták

A számítástechnikai megjelenítő eszközök nagyrészt sík képernyőn ábrázolnak képpontokat. Ahhoz, hogy a háromdimenziós testeket szemléltessük, a tér pontjait a síkra kell leképeznünk. Mielőtt a leképezést elvégezhetnénk, a térbeli pontokat jellemző három komponenst ki kell egészítenünk egy újabb, negyedik komponenssel. Erre azért van szükség, mert a leképezés egy projektív transzformáció lesz, ami nem párhuzamosságtartó, így a parallel vonalak kezelése végett bevezetjük a homogén koordináták használatát. A negyedik komponens segítségével a párhuzamos vonalak perspektivikus ábrázolása kezelhetővé válik. A probléma leginkább vasúti sínekkel szemléltethető:



. ábra - Párhuzamos vonalak problémája

Tudjuk, hogy a sínek valójában párhuzamosak, mégis minél távolabbra tekintünk, olyan, mintha egy távoli pontban metszenék egymást. A negyedik komponens bevezetésével meghatározhatunk egy olyan speciális pontot, ahol a valójában párhuzamos egyenesek a projekción metszik egymást a végtelenben.

### Ponttranszformációk

A ponttranszformációk olyan leképezések, melyek egy p ponthoz egy p’ pontot rendelnek, ahol p, p’ | n > 1.

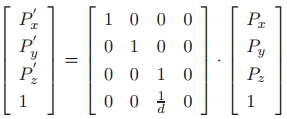
Ezeket a transzformációkat tulajdonságaik alapján különböző kategóriákba sorolhatjuk, köztük a leggyakrabban használtak:

* Egybevágósági (szakasztartó) transzformációk:
  + Eltolás
  + Forgatás
  + Tükrözés
* Hasonlósági transzformációk
* Affin (párhuzamostartó) transzformációk:
  + Skálázás
  + Nyírás
* Projektív transzformációk

Ezen transzformációkat leíró függvényekkel nehézkes dolgozni, viszont mivel ezek lineáris transzformációk, transzformációs mátrixokkal reprezentálhatóak. A transzformáció végrehajtásához a p pontot mátrixszorzással p’ pontra képezzük le.

### Középpontos vetítés

A tér síkra történő leképezése projektív transzformációval történik, melyet általában az emberi szemmel észlelt valódi tér érzetének analógiájára, középpontos vetítéssel teszünk meg. Vegyük P térbeli pontot és d távolságot (a képsík távolsága az origótól), ekkor a leképezés az alábbi mátrixművelettel írható le:



Ez a transzformáció dimenzióvesztő, a Descartes alakra történő alakítás után látszik, hogy a kapott pont z komponense mindig d lesz. Ez azt jelenti, hogy a tér a d távolságú síkra lett vetítve origó középpontú vetítéssel. Ebben az esetben az origót tekinthetjük a kamera fókuszpontjaként.

A matematikai alapok áttekintése során megismerhettük azokat a matematikai fogalmakat és számításokat, melyeket a játék fejlesztéséhez elengedhetetlenek.

# A játék

A webGL technológia képességeinek demonstrációjához egy belső nézetű lövöldözős játékot készítettem el. A játékot egynél több játékos játssza, a célja: minél több pontot szerezni. Egy játékos akkor szerez pontot, ha egy másik játékos életpontját 0-ra csökkentette.

Minden játékos kezdetben 100 életponttal rendelkezik. Amikor az egyik játékos lő és eltalálja a másikat, a találat helyétől függően csökken az eltalált játékos életpontja. Ha a lövedék a fejet találta el, 100 életponttal, ha a törzset 40 életponttal, minden más helyen 20 életponttal csökkenti a találat az eltalált játékos aktuális életpontját.

Ha egy játékos életpontja 0-ra csökken, akkor a játékos ”meghalt”, klikkelés után 5 másodperccel egy véletlenszerűen választott ponton ”újraéled” és folytathatja a pontszerzést.

A játék célja: az öt perces időkeretben minél több pont összegyűjtése.

# A játék implementációja

Ahhoz, hogy felépítsem a játékot először a megjelenítendő térbeli modelleket kellet megalkotnom, összeszednem. A játéktér modelljét lakóhelyem utcarészlete alapján készítettem. A modellek elkészítése során a Blender modellező szoftverben volt segítségemre. Először a lakótömbök egy szintjét modelleztem. A lakások falait egyenként, téglatestből készítettem el, megpróbálva az eredeti méretarányban megtenni ezt. A falakon ablakok és ajtók nyílásait alakítottam ki egy téglatesttel, melyet a térbeli logikai kivonás művelete segítségével értem el.

A térbeli testekhez anyagot rendelünk, melyet a számítógép annak paraméterei alapján fog megjeleníteni. Ilyen attribútumok a felületi fényesség, fénykibocsátás, árnyékvetés, a különböző árnyalási metódusok, textúrafájlok és textúrakoordináták. Miután a modell elkészült, következhetett az anyagok mintázatát illusztráló képek betöltése és pozicionálása. A külső falak téglatestjein UVWmapping alkalmazásával a beton mintázatának képét helyeztem el, a belső falakra szintén ezzel a metódussal vakolat-szerű anyagot készítettem el (**)**.

A környezet kialakításához egyszerű, négyzet alakú felület hozzáadásával kezdtem el a munkát. Ezt a négyzetet felosztottam 9 kisebb, egyenlő méretű felületre. Az utca úttesti felületének kialakításához a felület bal oszlopában található 3 négyzet bal pontjait az X koordináta mentén eltoltam, kialakítva az úttest szélességét. A modellt alkotó térbeli pontok további másolás, eltolás műveleteivel, valamint az így kapott vertexek között felületeket kialakítva elkészíthető az utcarészlet egy kereszteződése, egy elágazódása és végül egy kisebb gyepesített park is.

A játéktérhez tartozik az égbolt is. Hogy elkészítsem, egy gömbből indultam ki. A test felezése és poligonjaniak kifordítása után elhelyeztem rajta a textúrafájlt, melynek applikációs eljárásként gömb projekciós koordinátákat használó vetítést választottam. Az égbolt mintázata egy halszem-optikával készült fotó, az előbb ismertetett technika használatával valódi égbolt látszatát kelti. (**)**

A játékosok alapmodelljét külső forrásból szereztem be, viszont a modell még nem volt játékra kész. A karakterhez létrehoztam egy csontvázat, amely a lábfejek, lábszárak, lábcombok, törzs, vállak, felkarok, alkarok csontjaiból áll és a modell mozgatásához a skin nevű deformáló eszközt használtam, így a karaktert beállíthattam olyan pózba, ami a puskát fogja. Szintén a csontváz segítségével készítettem el a lépések animációját, melyet a láb csontjainak ízület körüli forgatásával értem el.

A fegyver modelljét a karakter kezébe illesztettem, valamint készítettem egy torkolat-tűz modellt is, ezzel szemléltetve a lövéseket. A karaktert elláttam két további segédmodellel, ami a fej és a törzs körüli lövedék-becsapódások azonosítását szolgálja.

Miután a modellek elkészültek, elkezdhettem a játék fejlesztését.

# A szoftver architektúrája

## Szerver

A webes applikáció kiszolgálásához Node.js szervert használtam. A Node.js erősen moduláris felépítésű, telepítéskor a node package manager (npm) nevű komponens-menedzsment applikáció települ, ezzel tölthetjük le a különböző építőelemeket, melyeket használni szeretnénk. Ahhoz, hogy a játék működőképes legyen, szükség van egy webszerverre, ami az applikáció fájljait kezeli, valamint egy websocket kiszolgálóra a valós idejű kapcsolathoz a kliensek és a szerver között.

A webszerverhez a http, url, path és fs nevű komponenseket használtam fel, ezek sorra: a HTTP protokollon történő kommunikációt, URL-ek feldolgozását, elérési utak kezelését és a fájlrendszerhez való hozzáférést szolgálják. A szerverhez GET metódussal érkező HTTP hívások esetén megkeresi a fájlrendszerben a kérésnek megfelelő fájlt, és ha létezik, a tartalmával válaszol a szerver, egyébként http hibával tér vissza.

A játékszerver websocket technológia segítségével kommunikál a kliensekkel. Ennek megvalósításához a nodejs-websocket nevű komponenst használtam. A játék kiszolgálója a csatlakozni kívánó klienseknek egyenként kapcsolatot nyit, melyet nyilván tart és egyedi azonosítóval lát el. Ezt az egyedi azonosítót elküldi a kapcsolódó kliensnek is, így az azonosítás mindkét oldalon egyértelmű lesz. A webapplikáció az azonosítás után elküldi a játékszervernek a játékos objektum egyszerűsített példányát - melyet a szerver regisztrál - majd elkéri a jelenleg csatlakozott játékosok hasonló példányait és megjeleníti őket a játéktérben. Az aktuális kliens ezután aktívvá válik, minden képfrissítéskor elküldi a játékszervernek a játékos állapotát, melyet a szerver üzenetszórással továbbít minden további kliensnek, így frissítve a játékosok pozícióját minden csatlakozott példánynál.

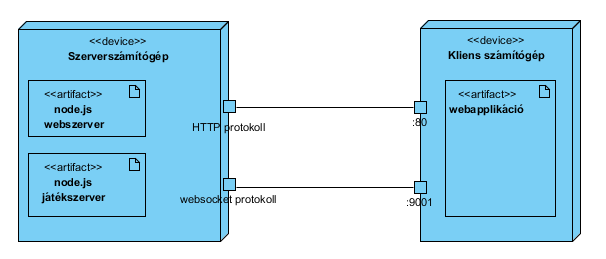
A nodejs példány elindításakor mind a webszerver, mind a játékszerver elindul, a játék futtatása a klienseken elkezdődhet. A kliensek először a webszerverhez kapcsolódnak, ahonnan letöltik az erőforrásokat (HTML, CSS, JS, képek és modellek fájljai). Az erőforrások letöltése után a kliens kezdeményezi a websocket kapcsolat felállítását.

## Kliens

A játékkliens webböngészőben futó applikáció mely AngularJS keretrendszeren alapszik. Azért választottam ezt a keretrendszert, mert jelentősen egyszerűsíti a fejlesztést:

* HTML formátumú sémákkal dolgozhatunk, melyek direktívákkal programozhatóak. A sémákat az Angular lefordítja és az ún. data-binding technológia biztosítja, hogy a modell minden változása valós időben tükröződjön a felhasználói felületen.
* Több programozási mintához is, mint például a singleton (egyke), factory (gyár), controller (vezérlő), directive (direktíva), olyan eszközöket nyújt, melyek jelentősen egyszerűsítik, felgyorsítják a fejlesztést
* A függőségek injektálhatósága nagyban segíti a célközpontú kódszeparációt, ezáltal az átláthatóságot
* Szilárd alapot ad a Modell-View-Controller vagy a Model-View-Viewmodell szoftverarchitektúrához.

A grafikus megjelenítést szolgáló webGL technológia kihasználásához a three.js nevű csomag biztosít olyan fejlesztői könyvtárat, ami mind a matematikai számítások mind a térbeli objektumok manipulációs és interakciós feladatainak megoldása során nagy segítséget nyújt. Ez a csomag olyan absztrakciót biztosít, ami teljesen elfedi a webGL nyelvét, így minden művelet javascript nyelven implementálható.



. ábra - Architekturális vázlat

# A játékkliens szoftveres megvalósítása

## Modulok felépítése

### Erőforrás-betöltő modul (fps\_game.loaders)

Az erőforrás-betöltő modul felel a játéktér betöltéséért és felépítéséért. A játék szcénáját JSON formátumban tárolt objektumként írtam le, ami felsorolja a betöltendő fájlokat, a meghívandó generátorfunkciókat és a modelleken elvégzendő transzformációkat.

A játékterek a resInfo.json nevű fájlban vannak felsorolva, azok megnevezésével és gyökérkönyvtáruk elérési útjával. Ezt a json formátumú fájlt a resourceFetcher nevű szolgáltatás dolgozza fel.

A játék pályájának betöltését a sceneLoader szolgáltatás fogja elvégezni a loadScene metódusban. Ez a metódus megkapja a játéktér elérési útját, ahonnan legelőször a models.json fájlt dolgozza fel, amiben a játéktér elemeit képező modellek neve, elérési útja és a rajtuk végrehajtandó transzformációk találhatóak egy vektorban sorakozó objektumok formájában. A loadScene metódus, meghívásakor, először elindítja az összes modell betöltését. A modellek betöltése egyúttal a modellhez tartozó textúrák betöltését is kezdeményezi, amint az összes modell megérkezett a generátorfüggvény meghívódik, majd a betöltő modul végrehajtja az utolsó transzformációt az így kapott modellen és elhelyezi azt a térben.

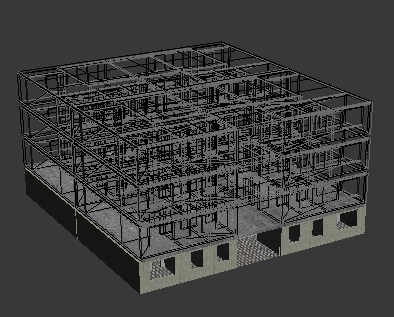
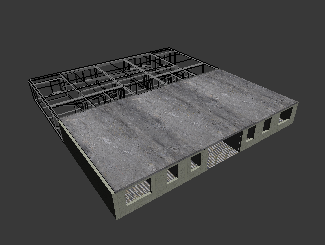
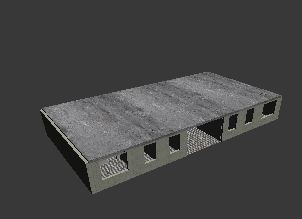
### Generátor modul (fps\_game.generators)

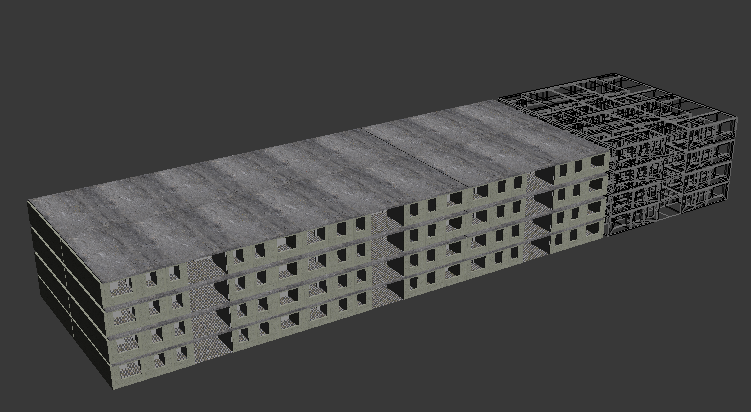
A generátormodul olyan szolgáltatásokat tartalmaz, melyek a betöltött modelleket transzformálják. A játékban látható lakótömbök például egyetlen emelet modelljéből épülnek fel, melyet a multiplyGenerator függvénye sokszoroz. A függvény négy bemeneti paramétere rendre: maga a modell, valamint az x, y és z tengely mentén történő sokszorozás mennyisége.

Egy ötemeletes, három tömbből álló lakótömb például az alábbi függvényhívással hozható létre:

*housingGenerator*.generate(<THREE.Object3D>,3,5,1);

A sokszorozás az alábbi módon történik: feltételezzük, hogy a modell középpontja egyben a lokális koordinátarendszernek az origója. Meghatározom a modell méreteit boundigBoxHelper segítségével, majd a modell másolatát eltolom az adott koordinátán mért mérettel. Ezt a műveletet ismételjük mindhárom koordináta mentén, mindig az előző művelet eredményeképp kapott modellből kiindulva.





. ábra - Modell sokszorozása

Miután elkészült a kompozíció, az Y tengely mentén eltolással transzformálom a kész modellt, annak magasságának felével, így a modell lokális koordinátarendszerében a minimum Y sík a globális koordinátarendszer 0 magasságú síkjára érintő lesz - ezt neveztem ki a föld magassági szintjének.

### Képalkotó modul (fps\_game.rendering)

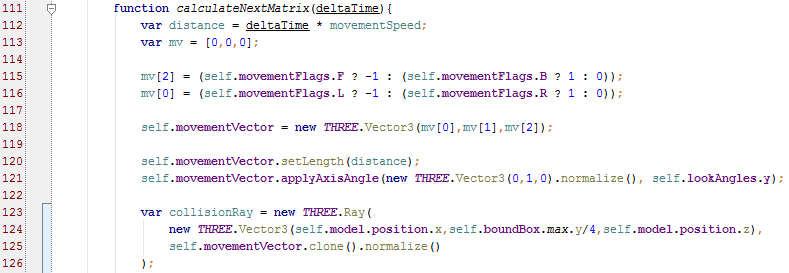
A webGL vezérlését szolgáló modul. A modul három komponenst tartalmaz:

* az ngWebgl direktíva összekapcsolja felhasználói felületen a megjelenítésért felelős HTML elemet a vezérlővel (renderingCtrl),
* a renderingCtrl a képalkotó osztály egy példányát készíti el a renderModelFactory segítségével. A példányosításkor a konstruktornak átad egy minimális konfigurációt, mely a kamera látószögét, a kívánt képfelbontás és a képarány beállításait tárolja. Ezután a vezérlő további általános beállításokat eszközöl a már létrehozott példányon, mint például az árnyékok vagy a fényforrások megjelenítése.
* a renderModelFactory olyan objektumot példányosít, melyek a three.js által szolgáltatott komponensekből a képalkotáshoz szükséges minimumot felépíti. Ez magába foglalja egy kezdő szcéna létrehozását, egy globális fényforrás beállítását és egy kamera létrehozását. Ezt az objektumot további metódusokkal láttam el, melyek a modellek betöltését, szcénához történő hozzáadását és törlését valamint a kép rekurzív újrarajzolását végzi.

### Játékos modul (fps\_game.player)

Ez az egység a játékosok megjelenítéséért és vezérléséért felelős.

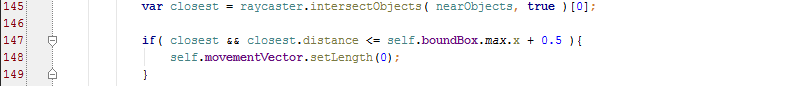
A játékos egyedeit a fps\_game.player.Player elnevezésű objektumgyár készíti el. Ezek az objektumok tárolják a játékosok attribútumait. Inicializáláskor betöltődik a játékos modellje, majd az objektum figyelni kezd a képfrissítésekre. Minden képfrissítés alkalmával a bejövő vezérlési adatok alapján újraszámolja a játékos modelljének transzformációs mátrixát, így megvalósítva a játékos irányítását:



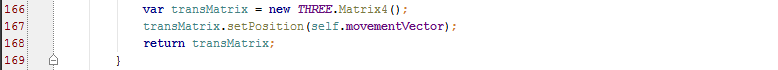
…



…



…



A forrás jól demonstrálja a three.js kifejezőerejét. Meghatározom az utolsó képkocka rajzolása óta eltelt időből a következőig megteendő távolságot, majd az aktuálisan igaz értékű mozgásirányt jelölő bitekből egységvektort építek, ami a mozgás irányába mutat (bal, jobb, előre, hátra). Beállítom az irányvektor hosszát a kiszámolt távolságra, majd az Y tengely körül - melyet a 0,1,0 egységvektor határoz meg (ez a vertikális koordináta a three.js kontextusában) - elforgatom a vektort a tekintet aktuális forgási szögével, létrehozva azt a vektort, mely a karakterünk egy képkocka alatti eltolását fogja jelenteni, így megalkotva a megszokott belső nézetes vezérlés.

Egy sugárvető és egy sugár segítségével megkeresem a közeli térelemeket. Ezt úgy teszem meg, hogy a mozgásvektor másolatát a karakter magasságának az ¼ részéhez transzponálom, normalizálom, majd a sugárvető objektumtól elkérem az ezzel irányvektorral érkező sugár által metszett objektumokat. Ha valamely objektum túl közel kerül karakterünkhöz (10 cm-en belül), beállítom a mozgás vektorának hosszát nullára, így megakadályozom, hogy a játékosok szellemek módjára a falakon keresztül közlekedhessenek.

Hogy miért a karakter magasságának a negyedét választottam? A válasz egyszerű: feltételezem, hogy a karakter ekkora magasságú objektumot még képes megmászni (180cm magas játékos esetén ez 45cm). Azért, hogy valóban képes legyen a tereptárgyak megmászására, újabb sugár segítségével a játékos talpához legközelebb eső objektum magasságához transzformálom a karaktert az Y tengely mentén eltolással, egy olyan vektorral, ami a figura fejétől a föld irányába mutat. Ez a transzformáció fogja a karaktert mindig a lába alatt levő legmagasabb pontra emelni.

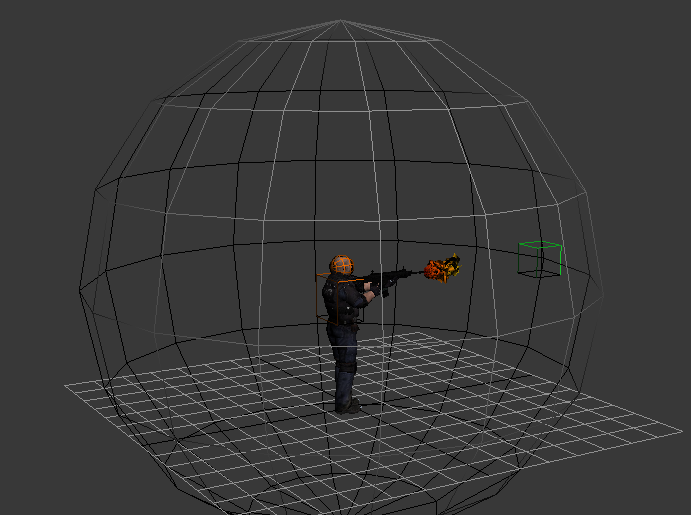
Amint az irányvektor elkészült, generálok egy transzformációs mátrixot a setPosition() metódus segítségével, mely az adott vektornak megfelelő eltolást fogja reprezentálni, ezzel modifikálható a játékos modellje.

A belső nézetes vezérlés része még a kamera mozgatása is. Ehhez az egér kétdimenziós koordinátáiból kell kameramozgást létrehoznom. Megvalósításához egy segédmodellt hoztam létre, mely nem látszik a képernyőn, de az egérmozgás hatására egy olyan gömb felületén mozog, melynek középpontja a játékos tekintete (a kamera fókuszpontja), a modell pontos pozíciója pedig az a felületi pont, mely a középponttól a lokális koordinátarendszer Z tengelye mentén pozitív irányban rajzolt sugárra mért Y és X tengely körüli forgatással számolható ki. Ezt az objektumot a kamera célpontjaként beállítva elkészült a teljes karaktervezérlés.

Ez a mátrix minden képfrissítésnél kiszámolódik és transzformálja játékosunkat.  
A modell manipulációja után értesítjük a szervert a változásról az aktuális karakterállapot elküldésével.

Ahogyan a játék ismertetőjében olvashattuk, a játékosok lövöldözni fognak egymásra. A lövéseket szintén sugárvető segítségével implementáltam. A sugár ezúttal a kamera fókuszpontjából mutat az imént említett segédobjektum irányába. Amikor a játékos bal egérgombbal klikkel, felvillantom a torkolattűz modelljét, majd megvizsgálom a sugár útjába eső modelleket. Ha a legközelebbi modell egy játékosé, akkor újabb vizsgálattal ellenőrzöm, hogy a sugár metszi-e a fejen vagy a törzsön elhelyezett becsapódási objektumok egyikét kiszámolva ezzel a találat által okozott életpont-vesztés mértékét.

Miután a számítások megtörténtek, a játékszervert értesítjük a találatról, az ellenfél elszenvedi a sérülést. Ha a sérülés a játékos ”halálához” vezet, akkor megszakítjuk a kapcsolatot a vezérlés és a játékos között, így az inaktív játékosok mozgásképtelenné válnak és lőni sem lesznek képesek, egészen az ”újraéledés” pillanatáig.



. ábra - A játékos modellje, a tekintet segédmodellje (zöld) a tekintet által bejárható gömb felszíne (szürke) valamint a fejet és a törzset ért lövedékek becsapódási segédmodelljei (narancs)

Szintén a játékos objektuma vezérli a lépések animációját, mely haladás közben lejátssza, álló helyzetben leállítja a lábak mozgatását.

Ebben a modulban találhatóak a vezérlési interfészek is.  
Az fps\_game.player.mouseControl direktíva az egérmozgás eseményeire figyel, minden ilyen esemény bekövetkezésekor frissíti az irányított játékos tekintetének szögét. Horizontális mozgás esetén a tekintet Y tengely körüli elfordulási szöge módosul, vertikális mozgáskor a tekintet lokális koordinátarendszerének X tengely körüli forgásszöge változik -90° és 90° között.

Az fps\_game.player.keyboardControl a billentyűzetről érkező bemenet eseményeinek hatására a játékos haladási irányának jelzőbitjeit állítja. Belső nézetű játékok esetén megszokott vezérlés szerint : [W] az előre, [S] a hátra, [A] a balra illetve [D] billentyű a jobbra haladást irányítja, a mozgás addig folyamatos, míg a billentyűk valamelyike lenyomott állapotban van. Két különböző – nem ellentétes – irányú gomb nyomva tartása esetén keresztirányú mozgás is lehetséges.

### Játék modul (fps\_game.game)

Ebben találhatóak a játékmenet vezérléséért felelős komponensek. A játéktér és a játékosok inicializálását valamint a játékmenet irányítását az fps\_game.game.gameDriver szolgáltatás végzi. A start metódus elkéri a játéktereket, majd az első játéktér betöltését elindítja. Mikor a betöltés befejeződött, inicializálja a játékot, ekkor összegyűjti a játékosok újraéledési pontjait, majd példányosítja a vezérelhető játékost. A gameDriver végzi a játékosok újraélesztését is, ekkor indulóhelyzetbe állítja a vezérelt játékos példányát, majd egy véletlenszerűen kiválasztott újraéledési pontra mozgatja azt.

Az fps\_game.game.networkGameDriver végzi a hálózaton csatlakozott játékosok vezérlését. Ez az egyke szolgáltatás a szerverkomponenstől érkező információ alapján transzformálja a játékos modelleket és vezérli azok animációit. Ezen üzenetek beérkezésekor úgynevezett figyelő függvények hívódnak meg, amiket a hálózat modulban található fps\_game.network.webSocket nevű szolgáltatás fog meghívni azok beérkezésekor. Minden egyes üzenet a típusának megfelelő figyelőfüggvény meghívását indukálja. Az üzenetek típusai és adatai lehetnek:

* getAllPlayers : a jelenlegi online játékosok listája
* playerConnect : újabb játékos kapcsolódása a játékhoz
* playerDisconnect : egy játékos megszakította a kapcsolatot
* playerUpdate : egy játékos attribútumai frissültek
* playerTakeDmg : egy játékost találat ért
* playerScore : egy játékos pontot szerzett

A getAllPlayers üzenet érkezésekor a játéktérhez az összes jelenlegi játékos hozzáadódik. Ezt a hívást csak inicializáláskor hívom meg, minden további beérkező, csatlakozni kívánó játékost a playerConnect fog kezelni. playerDisconnect üzenet esetén az adott játékost töröljük a játéktérről.

A playerUpdate üzenet egy egyszerűsített játékos objektumot tartalmaz, mely a főbb tulajdonságok értékeit tartalmazza. A játékos azonosítója alapján ezeket az attribútumokat frissítjük a játéktérben megfeleltethető játékos példányán.

Ha playerTakeDmg üzenet érkezett, megnézzük, hogy a vezérelt játékos szenvedett-e el a sérülést, ekkor az már ismertetett szabályok alapján csökkentjük az életpontját. Amennyiben ez a sérülés végzetes, a playerScore üzenet fog aktiválódni, ami a találatot ejtő játékos pontszámait fogja növelni.

### Hálózat modul (fps\_game.network)

Ebben a modulban található a websocket architektúrán történő kommunikáció megvalósításáért felelős szolgáltatás, az fps\_game.network.webSocket. Ezen a szolgáltatáson keresztül küldi a kliens az üzeneteket a játékszervernek. Minden üzenettípushoz implementáltam egy küldő metódust, mely interfész-implementációhoz hasonlítható.  
A szerveroldalon ezen üzenetek fogadó oldali párjai szintén elkészültek. A szolgáltatás példányosításakor implicit módon regisztrál egy setClientID üzeneteket fogadó metódust, ez fogja a szerver által kiosztott azonosítót a példány privát változójába írni.

Az objektumot az összes, korábban a játékos tárgyalásakor felsorolt üzenettípus mellett további három metódussal bővítettem:

* connect: kapcsolat létesítése a WS szerverrel
* close: aktív kapcsolat bontása
* sendCommand: privát metódus az üzenetek küldéséhez

## Modulok hierarchiája és illeszkedése

Ahogy azt a függelék 8. ábra - Hierarchia és kommunikáció a modulok között képén láthatjuk, három nagy csoportra oszthatjuk a modulokat:

* a játékos – játék – hálózati kapcsolat
* a betöltők – generátorok – közös eszközök
* és a képalkotó modulok.

Megfigyelhető, ahogy a játék komponens játssza a központi szerepet, és azt is, hogyan épül fel a virtuális tér implementációja a játék, játékos és betöltő modul közreműködésével, amit a képalkotó komponens fog a megjelenítőre rajzolni. Az ábra többé-kevésbé fedi az időrendiséget is az egyes modulok aktiválódását tekintve.

A játék modul egy játéktér inicializálását kéri. A fájlok megérkezését követően a betöltő modul a játékteret a képalkotónak átadja, majd példányosít egy játékost. A játékos saját modelljének betöltése után a játéktérbe kerül, szintén a képalkotó révén. Legvégül a játék a hálózati modulon keresztül vezérli a többi komponenst.

# Konklúzió

A számítógépes grafika története és az eddig eltelt évek alatt elkészült eszközök elképesztőek, rendkívül izgalmas és változatos technológiai megvalósításokat ismerhettem meg a kutatómunkám végzése során. Érdekes volt megfigyelni, hogyan befolyásolta a számítástechnikai eszközök fejlődése a hétköznapi életet, és vice-versa, ahogyan a különböző képalkotó matematikai képletek és algoritmikus megvalósításukat is nagyon figyelemre méltónak találom.

A matematikai alapfogalmak ismertetését elengedhetetlennek tartottam, a témakör és az implementáció részletesebb megértése végett.

Sokrétű, érdekes tapasztalatot szereztem szakdolgozatom elkészítése során.  
A javascript vezérelte webGL technológia nagyon szép, részletesen kidolgozott képek megalkotására is képes, viszont a fejlesztés során egy igen mérvadó tényező nehezítette fantáziám elengedését: a teljesítmény. A three.js fejlesztője is meglepődve értesült a THREE könyvtár használatával 100 000 poligon rajzolása során mért viszonylag alacsony képfrissítési rátáról, hiszen egy középkategóriás videokártyával is sokkal jobb eredményeket érhetünk el, ami egy belső nézetes lövöldözős játék esetében kényes érték. A játék, bár megközelítőleg 90 - 110 ezer poligont rajzol ki egy képen, 1080p felbontású monitor esetén az élvezhető játékélmény megköveteli a pixelsűrűség felezését. Az implementáció tartalmazza a napfény keltette árnyékok rajzolását is, ám ennek aktiválása újabb teljesítménybeli csökkenést okoz.

Véleményem szerint a technológia hatékony lehet a webböngészés élményének gazdagításához, de még fejlődnie kell egy részletgazdag, élethű játékélmény vagy virtuális valóság megjelenítéséhez (a legújabb béta verziójú böngészők már ehhez is biztosítanak fejlesztői eszközöket).

# Irodalomjegyzék

*Blender official site．* **https://www.blender.org/**

**designinfo.in** Munsell grayscale paper．[Online][Hivatkozva: 22 April 2016] https://designinfo.in/2505/munsell-neutral-value-scale-glossy-finish.jpg．

**DoobMr.** THREE FramePerSec drop．[Online][Hivatkozva: 22 April 2016] https://github.com/mrdoob/three.js/issues/25．

**threejs.org** THREE.BoundingBoxHelper．[Online][Hivatkozva: 22 April 2016]

http://threejs.org/docs/index.html#Reference/Extras.Helpers/BoundingBoxHelper．

**—．**THREE.Matrix4．[Online][Hivatkozva: 22 April 2016] http://threejs.org/docs/index.html#Reference/Math/Matrix4．

**—．**THREE.Ray．[Online][Hivatkozva: 24 April 2016] http://threejs.org/docs/index.html#Reference/Math/Ray．

**—．**THREE.Raycaster．[Online][Hivatkozva: 24 April 2016] http://threejs.org/docs/index.html#Reference/Core/Raycaster．

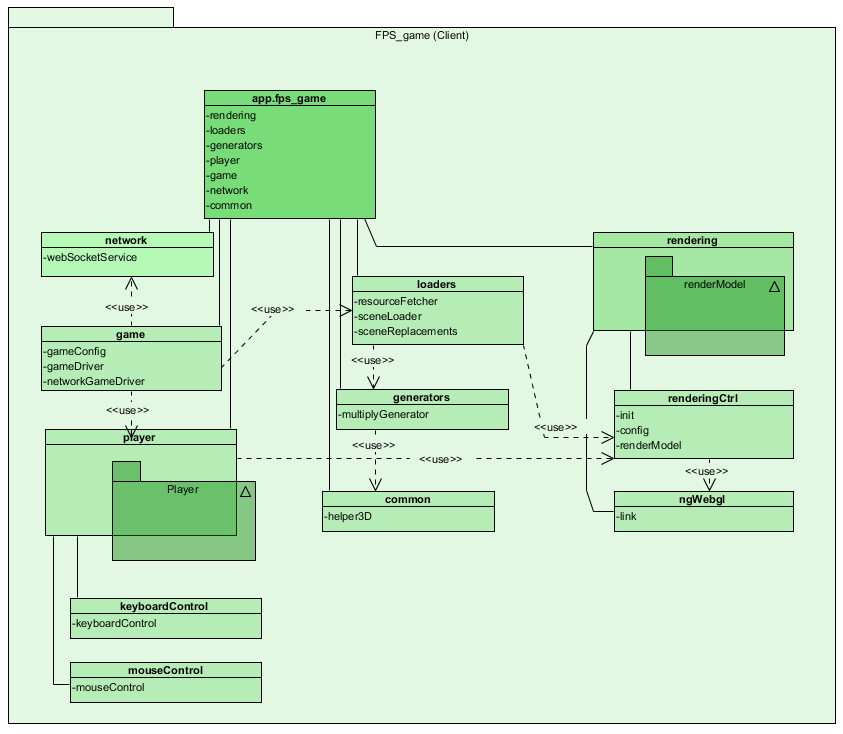
**—．**THREE.Vector3．[Online][Hivatkozva: 22 April 2016] http://threejs.org/docs/index.html#Reference/Math/Vector3．

**Wikipedia** AutoCAD 1.0．[Online][Hivatkozva: 24 April 2016] https://hu.wikipedia.org/wiki/AutoCAD．

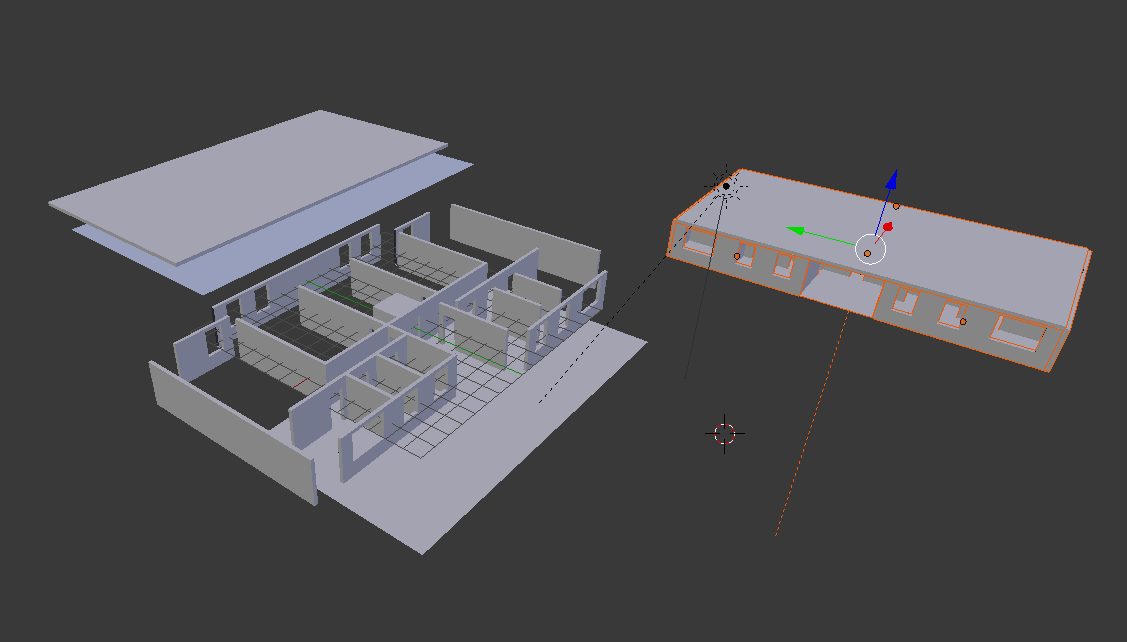
# Függelék



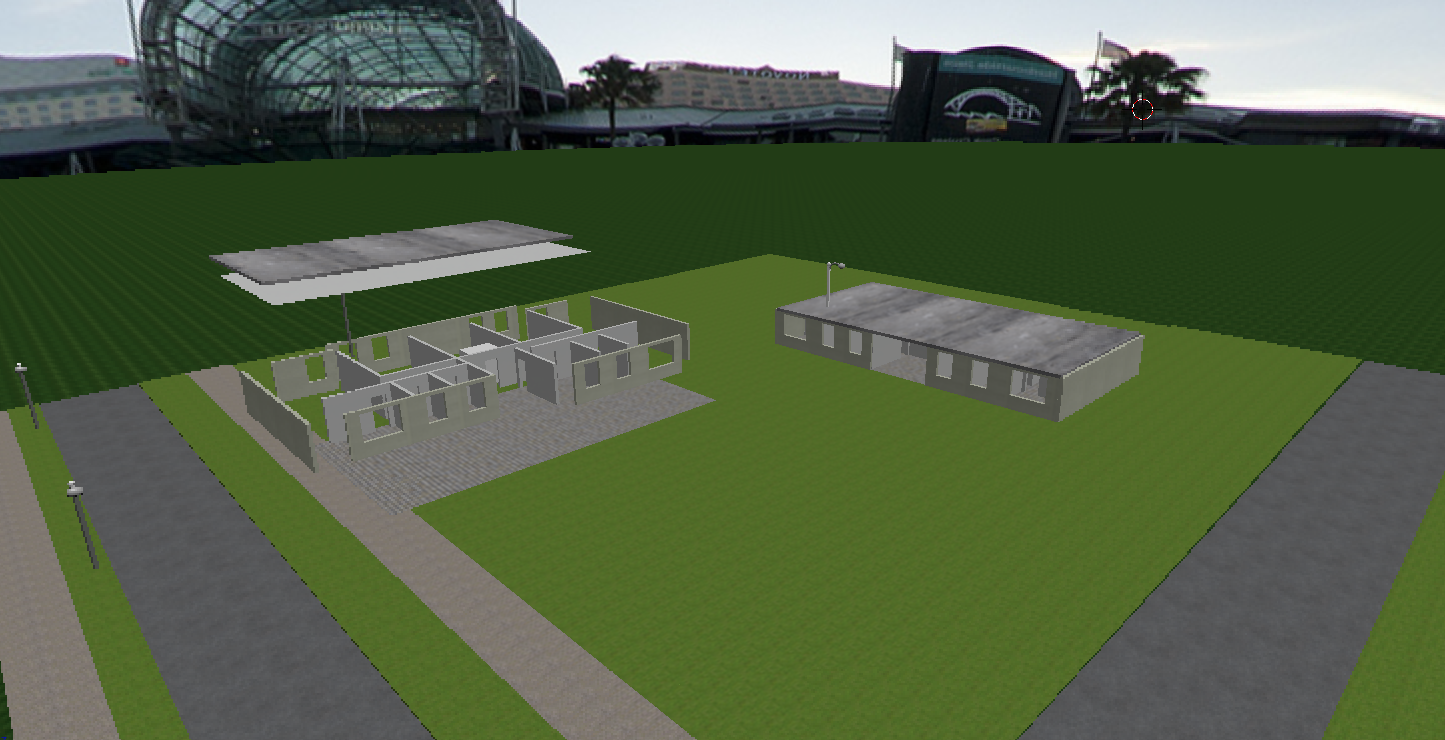
. ábra - Képernyőfelvétel



9. ábra - Hierarchia és kommunikáció a modulok között



10. ábra - Lakótömb egy emelete, robbantott ábra



11. ábra - Égbolt és környezet a játéktérhez