Debreceni Egyetem Informatikai Kar  
INFORMÁCIÓTECHNOLÓGIA TANSZÉK

Háromdimenziós komputergrafika a modern webböngészőkben

Belső nézetű lövöldözős játék AngularJS és three.js segítségével

Témavezető: Balla Tibor Készítette: Martyin Kornél  
Egyetemi tanársegéd Programtervező Informatikus hallgató

Tartalomjegyzék

[Bevezetés 1](#_Toc448768401)

[Történelmi áttekintés 3](#_Toc448768402)

[Matematikai alapok 7](#_Toc448768403)

[Testek leképezése 7](#_Toc448768404)

[A virtuális tér megjelenítése 8](#_Toc448768405)

[Homogén koordináták 8](#_Toc448768406)

[Ponttranszformációk 9](#_Toc448768407)

[Középpontos vetítés 9](#_Toc448768408)

[A szoftver architektúrája 10](#_Toc448768409)

[Szerver 10](#_Toc448768410)

[Kliens 10](#_Toc448768411)

[A játékkliens szoftveres megvalósítása 11](#_Toc448768412)

[Modulok felépítése 11](#_Toc448768413)

[Erőforrás-betöltő modul (fps\_game.loaders) 11](#_Toc448768414)

[Generátor modul (fps\_game.generators) 11](#_Toc448768415)

[Képalkotó modul (fps\_game.rendering) 12](#_Toc448768416)

[Játékos modul (fps\_game.player) 13](#_Toc448768417)

[Játék modul (fps\_game.game) 14](#_Toc448768418)

[Hálózat modul (fps\_game.network) 15](#_Toc448768419)

# 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 jóslata a technológiai fejődés sebességéről kiállta az idő próbáját, manapság a zsebeinkben olyan készülékek vannak, amik másodpercenként több milliárd alapvető művelet 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 3 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 szoftver, autóverseny szimulátor 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.

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.

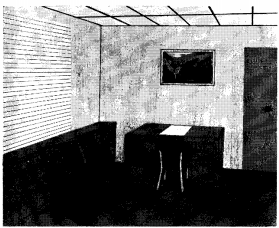
A dolgozat elméleti jelentősége a térbeli ábrázolás matematikai alapjainak bemutatása, és a komputergrafika történelmének feltárása, míg gyakorlati szempontból a bemutatott elmélet alapján megvalósított interaktív szoftver előállítása.

Célom a technológiában rejlő potenciál demonstrációja, melyet a szoftver elkészítésével és a forráskód valamint a felhasznált médiumok 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 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. Ezek a grafikák mechanikus vagy analóg komputereket használva készültek el, de nem nevezhetőek tisztán számítógépi grafikának, 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 megjelenítőeszköz, a képet Munsell mintapapírokból kivágva, összeragasztva állították elő, így demonstrálva matematikai modellüket.

Ben Laposky 1950-ben analóg számítógép és oszcilloszkóp segítségével hullámformákat jelenít meg és fotóz le, 1951-ben vektorszkóp típusú CRT megjelenítőt 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 megmunkálógépek irányítására a rajzolt 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 pontjai 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 elkészítik az első híres komputergrafikus játékot, a Spacewar-t.

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

1965: Jack Bresenham feltalálja az „idális vonal” rajzolásának algoritmusát (kör rajzolása pixelgrafikus megjelenítőn).

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

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

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

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

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

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

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

1979: a Bell Labs és a Cornell Egyetem közös munkájával kifejlesztik a fénysugár követésének technikájával felépített képet (raytracing).

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

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

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

1988: Aranykorát éli a videojáték; megjelennek az első dedikált valósidejű háromdimenziós grafikus egységek, ami az asztali számítógépekben található videokártyák elődjei. A személyi számítógépek elterjedésével felvirágzik a digitális művészet is, megjelenik a demoscene, terjednek 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 használják a térbeli megjelenítést. A korábbi komputergrafikára specializált Silicon Graphics munkaállomások helyét erőteljes Microsoft Windows vagy Apple Macintosh rendszerű gépek váltják fel, megugrik az olyan modellezőszoftverek népszerűsége, mint például az Autodesk 3D studio.

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

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

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

1999: az Nvidia értékesíti 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 ú.n. „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).

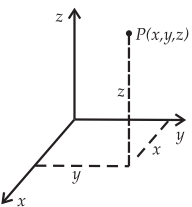
2004: Az Apple, a Mozilla és az Opera munkatársai munkacsoportot hoznak létre a HTML5 kifejlesztéséhez, 2008-ban a Firefox böngésző lesz az első, aki támogatja a HTML5-öt, így a 3D hivatalosan is betört a WEB-es technológiák közé, a canvas HTML elem révén a böngésző képes a számítógép grafikus erőforrásait használni térbeli megjelenítésre.

Szakdolgozatomban az imént említett, böngészőkben használható technológiát fogom bemutatni.

# 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 pontot a Descartes-féle koordinátarendszerben egyértelműen meghatározzák.

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 háromszög legalább egy éle egy másik háromszög 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.

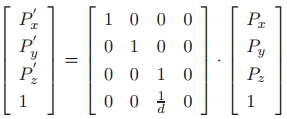
Ezeket a transzformációkat tulajdonságaik alapján különböző kategóriákba sorolhatjuk, köztük a leggyakrabban használtak:

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

Ezeket a 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 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 szolgálja ki, valamint egy websocket kiszolgálóra a valós idejű kapcsolathoz a kliensek és a szerver között.

A webszerver a GET metódussal érkező HTTP hívások esetén megkeresi a fájlrendszerben a kérésnek megfelelő fájlt, és a tartalmával válaszol.

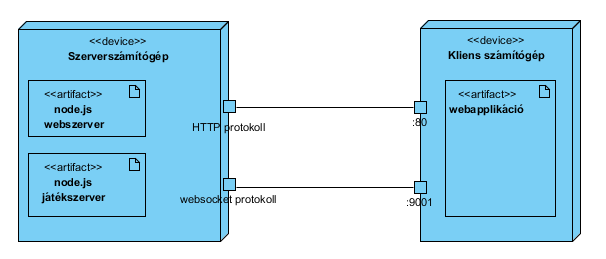
A játék kiszolgálóegysége a csatlakozni kívánó klienseknek egyenként websocket kapcsolatot nyit, melyet nyilván tart és egyedi azonosítóval lát el. Ezt az egyedi azonosítót elküldi a kapcsolódó kliensnek, így az azonosítás mindkét oldalon egyértelmű lesz. A webapplikáció az azonosítás után elküldi a játékszervernek a játékos objektum egyszerűsített példányát - melyet a szerver regisztrál - majd elkéri a jelenleg csatlakozott játékosok hasonló példányait. Az aktuális kliens ezután aktívvá válik, minden frissítéskor elküldi a játékszervernek a játékos állapotát, melyet a szerver üzenetszórással továbbít minden további kliensnek, így frissítve a játékosok pozícióját minden csatlakozott példánynál.

## 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 ú.n. data-binding technológia biztosítja, hogy a modell minden változása valós időben tükröződik 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 webGL technológia kihasználásához a three.js nevű csomag biztosít olyan fejlesztői könyvtárat, ami jelentősen mind a matematikai számítások mind a térbeli objektumok manipulációs és interakciós feladatainak megoldása során nagy segítséget nyújt.



. ábra - Architekturális vázlat

# A játékkliens szoftveres megvalósítása

## Modulok felépítése

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

Az erőforrás-betöltő modul felelős a játéktér betöltéséért és felépítéséért. A játék szcénáját JSON formátumban tárolt objektumként írtam le, ez a fájl 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 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. A fájlban a játéktér elemeit képező modellek neve, elérési útja és a rajtuk végrehajtandó transzformációk találhatóak egy vektorban sorakozó objektumok formájában. A loadScene metódus meghívásakor először kezdeményezi 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 modelléből épülnek fel, melyet a housingGenerator függvénye sokszoroz. A függvény négy bemeneti paramétere rendre: maga a modell, 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 koordinátarendszerének 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.

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 koordinátarendszerben a 0 magasságú síkra érintő lesz (ezt neveztem ki a föld magassági szintjének).

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

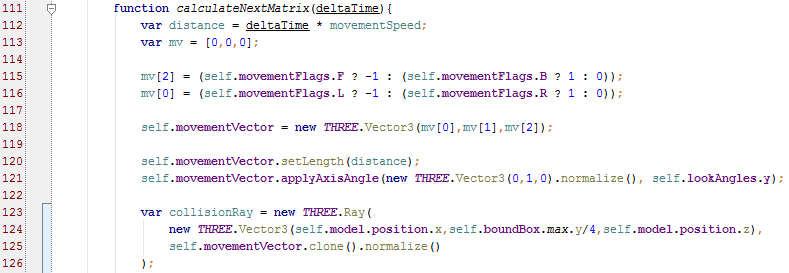
A webGL vezérlését szolgáló modul. A modul három komponenst tartalmaz:

* az ngWebgl direktíva összekapcsolja felhasználói felületen a megjelenítésért felelős HTML elemet a vezérlővel (renderingCtrl),
* a renderingCtrl egy képalkotó osztályt példányosíttat a renderModelFactory segítségével. A példányosításkor a konstruktornak átad egy minimális konfigurációt, mely a kamera látószögét, a kívánt képfelbontást és a képarányt állítja be. A példányosítás után a vezérlő további általános beállításokat eszközöl a már létrehozott példányon.
* 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 példányosítását. Ezt az objektumot további metódusokkal láttam el, melyek a modellek betöltését, szcénához történő hozzáadását és törlését valamint a kép rekurzív úrjarazolását végzi.

### Játékos modul (fps\_game.player)

Ez az egység a játékosok megjelenítéséért és vezérléséért felelős.

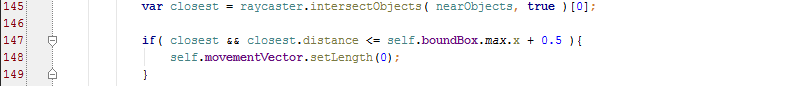
A játékos objektum példányosításakor betölti a játékos modelljét. A betöltés befejeztével az objektum figyelni kezd a képfrissítésekre. Minden képfrissítés alkalmával a bejövő vezérlési adatok alapján újraszámolja a játékos modelljének transzformációs mátrixát:



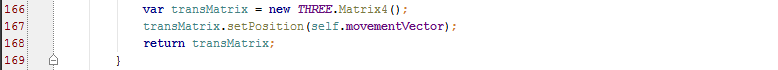
…



…



…



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, mely 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, így létrehozva azt a vektort, mely a karakterünk egy képkocka alatti eltolását fogja jelenteni, így létrehozva a megszokott belső nézetes vezérlés

Egy sugárvető és egy sugár segítségével megkeresem a közeli objektumokat. Ezt úgy teszem meg, hogy a mozgásvektor másolatát a karakter magasságának az ¼ részéhez transzponálom, normalizálom, majd a sugárvető objektumtól elkérem az ebből az irányvektorból képzett sugár által metszett objektumokat. Ha valamely objektum túl közel kerül, 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 (175cm magas játékos esetén ez 43cm). De hogy a képessége valóban meglegyen, újabb sugár segítségével, ami a figura talpától a föld irányába mutat, a legközelebbi objektum magasságához transzformálom a karaktert az Y tengely mentén eltolással. Ez a transzformáció fogja a karaktert mindig a lába alatt levő legmagasabb pontra tenni.

Amint az irányvektor elkészült, létrehozok egy transzformációs mátrixot, mely az adott vektornak megfelelő eltolást fogja reprezentálni, ezzel transzformálhatom a játékos modelljét.

A belső nézetes vezérlés része még a kamera mozgatása is. Ehhez az egér kétdimenziós koordinátáiból kell kameramozgást létrehoznom. Hogy megvalósítsam, 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 transzformáció után értesítjük a szervert a változásról az aktuális karakterállapot elküldésével.

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. A 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.

A 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.

A networkPlayerControlService végzi a hálózaton csatlakozott játékosok vezérlését. Ez az egyke szolgáltatás a szerverkomponenstől érkező információ alapján transzformálja a játékos modelleket és vezérli azok animációit.

### Játék modul (fps\_game.game)

Itt találhatóak a játékmenet vezérléséért felelős komponensek. A játéktér és a játékosok inicializálásáért a gameSetupController nevű vezérlő a felelős, a játékszerverrel történő kommunikáció pedig a networkGameDriver nevű szolgáltatáson keresztül történik, mely csatlakozik a szerverhez, az aktuális játékost regisztrálja, hozzáadja a játéktérhez a többi játékost vagy törli a játéktérről a lecsatlakozottakat.

### Hálózat modul (fps\_game.network)

Ebben a modulban található a websocket kommunikációt megvalósító szolgáltatás. Ezen a szolgáltatáson keresztül küldi a kliens az üzeneteket a játékszervernek.