**Eötvös Loránd Tudományegyetem**

**Informatikai Kar**

[**Információs Rendszerek**](http://www.inf.elte.hu/karunkrol/szervezet/tanszekek/Lapok/Inform%C3%A1ci%C3%B3sRendszerekTansz%C3%A9k.aspx) **Tanszék**

**Portal 2D**

**Témavezető:**

Dr. Gombos Gergő

adjunktus, Ph.D.

**Szerző:**

Kiss János

Programtervező informatikus BSc.

Budapest, 2020

Tartalomjegyzék

[1. Bevezetés 3](#_Toc41318879)

[1.1 Rövid ismertető 3](#_Toc41318880)

[1.2 Motiváció 3](#_Toc41318881)

[2. Felhasználói dokumentáció 4](#_Toc41318882)

[2.1 A játék telepítése és telepítési előfeltételei 4](#_Toc41318883)

[2.2 Szerver 4](#_Toc41318884)

[2.2.1 Konfigurálása 4](#_Toc41318885)

[2.2.2 Elindítása 5](#_Toc41318886)

[2.3 Kliens 6](#_Toc41318887)

[2.3.1 Elindítása 6](#_Toc41318888)

[2.3.2 Használat 6](#_Toc41318889)

[3. Fejlesztői dokumentáció 11](#_Toc41318890)

[3.1 Szerver 11](#_Toc41318891)

[3.1.1 Fizika 12](#_Toc41318892)

[3.1.1.1 Entity 12](#_Toc41318893)

[3.1.1.2 Ball 13](#_Toc41318894)

[3.1.1.3 Wall 15](#_Toc41318895)

[3.1.1.4 RoundWall 15](#_Toc41318896)

[3.1.1.5 Portal 16](#_Toc41318897)

[3.1.1.6 Fizika motor 16](#_Toc41318898)

[3.1.2 Játéktér 17](#_Toc41318899)

[3.1.3 Kommunikáció 17](#_Toc41318900)

[Hivatkozások 17](#_Toc41318901)

# Bevezetés

## Rövid ismertető

Szakdolgozatom egy a Portal [1] című játékhoz hasonló 2 dimenziós többszemélyes játék megvalósítása webes alkalmazásként. A játék fő eleme, hogy a pályák során különböző fejtörőket kell megoldani, teleportáció segítségével, portálokat lehet falakra helyezni. A játék fizikai rendszere szerint amilyen lendülettel áthalad egy tárgy az egyik portálon, az olyan lendülettel távozik a másikon, ez fontos szerepet fog játszik az egyes feladatok megoldásánál. A játék fizikai rendszerét magam valósítottam meg, ezzel egy egyedi fizikai szimulációt létrehozva. Az alkalmazást javascript nyelvben valósítottam meg, a többszemélyességet pedig TCP alapú web socketet [2] alkalmazva Nodejs [3] szerverrel. Többszemélyesség a játékban oly módon nyilvánul meg, hogy együttesen próbálhatják megoldani a fejtörőket a játékosok.

## Motiváció

A 2007-ben megjelent Portal című játékkal még általános iskolában ismerkedtem meg és első pillanattól kezdve el voltam ámulva tőle. Egy számítógépes játékban valódi fizikát láttam méghozzá olyan sci-fi elemekkel, mint a teleportáció egybevetve. Abban a pillanatban tudtam, hogy aki egy ilyen dolgot meg tud valósítani az bármire képes, persze ekkor még nem tanultam programozni, így el sem tudtam képzelni, hogyan lehetne valami hasonlót megalkotni. Most, hogy egyetemi tanulmányaim vége felé közeledem egy személyes próbatételként élem meg ezt a témát a többszemélyesség megalkotása pedig csak hab a tortán, hiszen soha nem dolgoztam még sem fizikával, sem többszemélyes játékkal.

# Felhasználói dokumentáció

Ez a fejezet bemutatja a játék indításának előfeltételeit, rendszerkövetelményeit, az elindításához szükséges lépéseket és további instrukciókat, majd részletesen leírja a játék működését és használtatát.

## A játék telepítése és telepítési előfeltételei

A program egy webalkalmazás, így nincs szükség külön telepítésre, viszont szükségünk van a kliens oldalon egy modern böngészőre (pl.: Google Chrome, Microsoft Edge) valamint szerver oldalon Node.js [3] keretrendszerre. A szerver és a kliens természetesen lehet azonos eszközön, viszont, ha a két külön eszközt használunk akkor biztosítani kell a hálózaton való kommunikációt.

Az optimális játékélményért az alábbi rendszerkövetelmények ajánlottak:

* Op. rendszer: Windows 10
* Processzor: 2 Ghz
* Memória: 2 GB RAM
* Tárhely: 50 MB szabad hely

## Szerver

A játék működéséhez egy dedikált szerverre van szükségünk, amit futtathatunk a saját gépünkön vagy akár egy távoli gépen is, amit elérünk az interneten keresztül.

## Konfigurálása

A szerver konfigurálása a config.js fájlban történik.

Paraméterei:

* server\_host: segítségével megadható, hogy a szerver milyen ip cím alatt fusson. Ha helyi szervert szeretnénk futtatni akkor ez nyugodtan lehet a ’localhost’ érték
* server\_port: segítségével megadható, hogy a szervert melyik porton szolgáljuk ki. Itt érdemes egy 1024-nél nagyobb számot megadni, hisz az ez alatti portok a rendszer számára kitüntettek portok.

## Elindítása

A szerver elindításához Node.js [3] keretrendszerre van szükségünk, ami letölthető innen: <https://nodejs.org/en/>.  
Ha már rendelkezünk Node.js-el a gépünkön akkor a szervert indíthatjuk manuálisan parancssorból: a szerver főmappájába navigálunk es beírjuk a következő parancsot:  
„node gameserver.js”, vagy Windows 10 alatt használhatjuk a local\_server.bat scriptet, ami ugyanezt eredményezi.



1. ábra: sikeres szerverindítás képe

Ha a szerver nem dobott hibát (1. ábra) akkor készen áll a kliensek fogadására.

Lehetséges hibák:

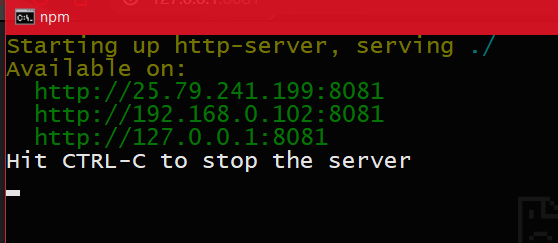
|  |  |
| --- | --- |
| Hibaüzenet | Kiváltó ok |
| ’node’ is not recognised… | Nincs Nodejs feltelepítve a gépünkre |
| address already in use … | A portot már egy másik alkalmazás foglalja |

## Kliens

## Elindítása

A kliens elindításához ki kell szolgálnunk a client mappa tartalmát egy http-szerveren amit egyszerűen elvégezhetünk az npm http-server moduljával, amit itt szerezhetünk be: <https://www.npmjs.com/package/http-server>  
Ha rendelkezünk, az előbb említett http-kiszolgálóval akkor a Windows 10 alatt a client mappába lévő local\_server.bat scriptet futtatva elindíthatjuk a kiszolgálót, vagy konzolból az „npx http-server” paranccsal.

Sikeres futtatás után a következőt (2. ábra) kell látnunk a konzolban:

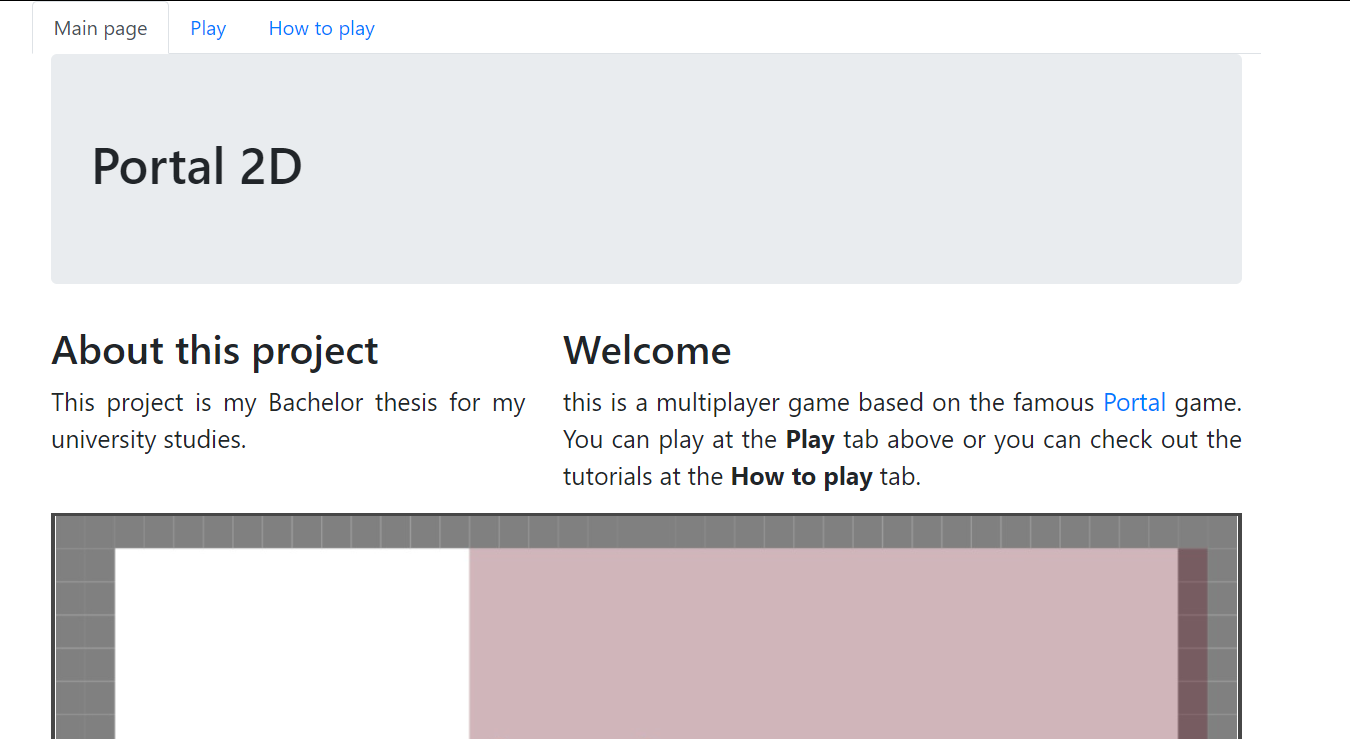


2. ábra: Sikeres kliens indítás a 8081-es porton

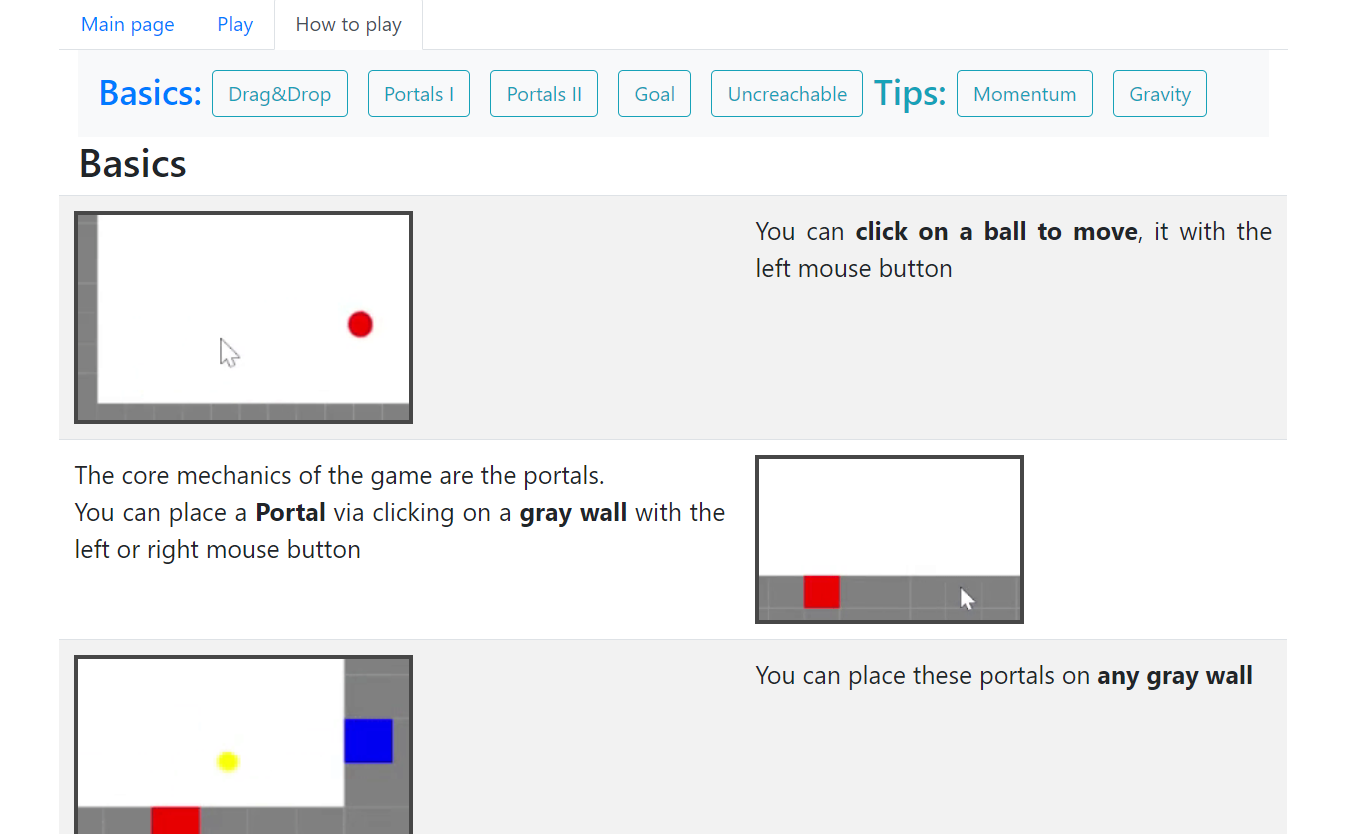
Ha elindítottuk a kliens, akkor egy modern böngészőt megnyitva (pl. Google Chrome, Microsoft Edge) alapértelmezetten a localhost:8080 címet felkeresve, elérhetjük az applikációt.

## Használat

Az oldal betöltése után a következő képnek (3. ábra) kell fogadnia minket:

  
3. ábra: kezdőoldal

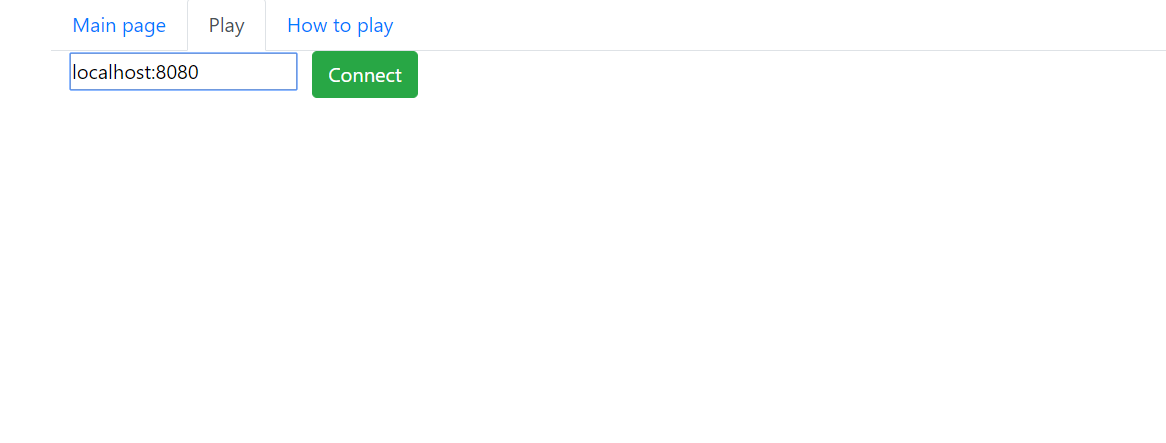
Itt a felső menüsávban válthatunk az oldalak között. Külön oldal van a játék elmagyarázására és külön magára a játékra. Ha szeretnénk megismerkedni a játék menetével és szabályaival akkor a „How to play” menüpontra kell kattintanunk, ami elnavigál minket a következő (4. ábra) oldalra:



(4. ábra: betanító oldal)

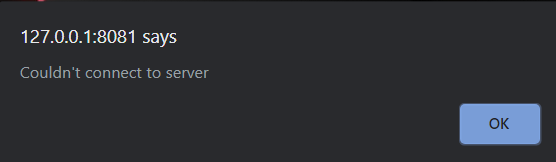
Ezen az oldalon leírásokat találunk a játék mechanikájáról, és a leírásokat videók kísérik a könnyebb megérthetőség érdekében.

A felső menüsávban a „Play” gombra kattintva érhetjük el azt a felületet, ahol a valódi játék történik. Itt egy egyszerű szövegmezővel találkozunk, ahová a szerver elérési útvonalát kell megadnunk, majd a „Connect” gombra kattintva csatlakozhatunk is a szerverhez (5. ábra).



*(5. ábra csatlakozás szerverhez)*

Sikertelen csatlakozás után a következő hibaüzenet fog várni minket (6. ábra):

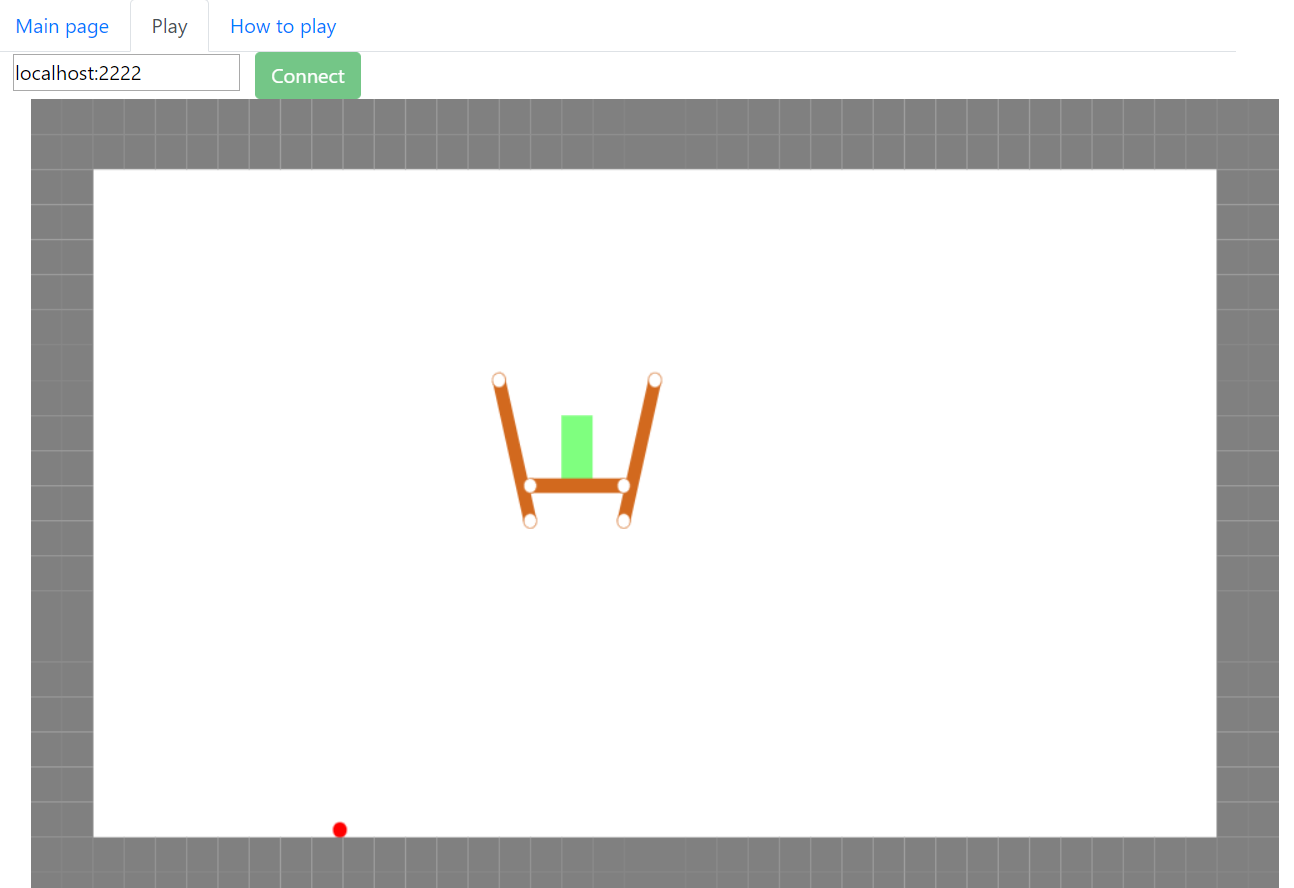


(6. ábra: sikertelen csatlakozás)

Sikertelen csatlakozást 2 hiba okozhat:

* Rossz szerver elérési útvonal
* A szerverre nem tud több kliens csatlakozni

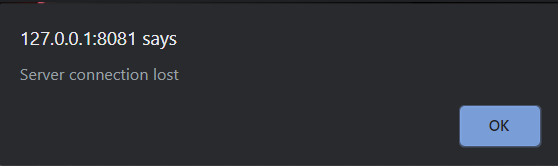
Miután sikeresen csatlakoztunk egy szerverhez, kezdődhet is a játék, az első pálya fog fogadni minket (7. ábra).



*(7. ábra: a játék első pályája)*

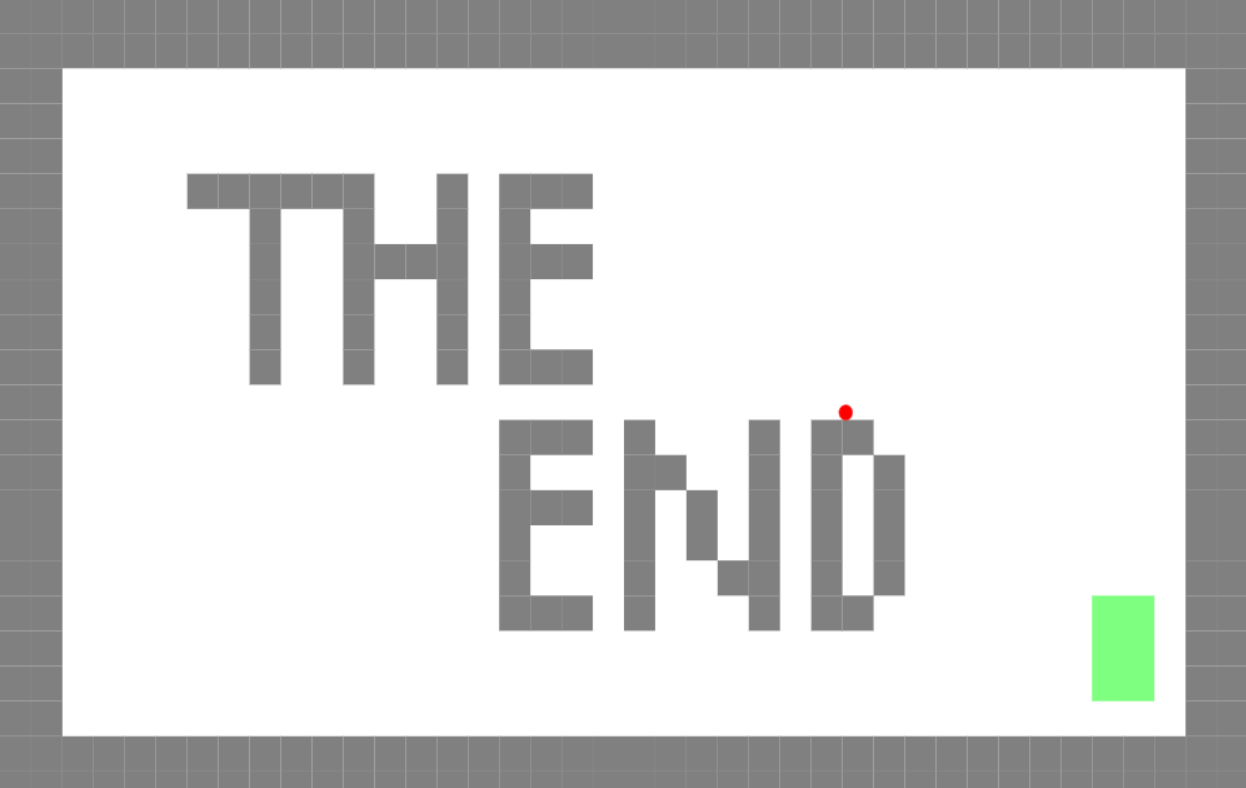
Itt pedig alkalmazhatjuk a betanító oldal által tanult technikákat a pálya teljesítéséhez.

Ha a játék, működése közben valami oknál fogva megszakadna a kapcsolat a szerverrel, a következő hibaüzenet fog fogadni minket (8. ábra):



(8. ábra: szerver elvesztése)

Ha pedig elértük az utolsó pályát, akkor a következő kép fog fogadni minket (9. ábra):



(9. ábra: játék vége)

# Fejlesztői dokumentáció

Ez a fejezet bemutatja a játék megvalósítását és annak eszközeit.

## Szerver

A szerver felel a játék működéséért, tehát a fizikai objektumok viselkedéséért, a játéktér létrehozásáért, elpusztításáért, valamint a kliensek fogadásáért és a velük való kommunikációért.

A szerver mappaszerkezete a következő (10. ábra):

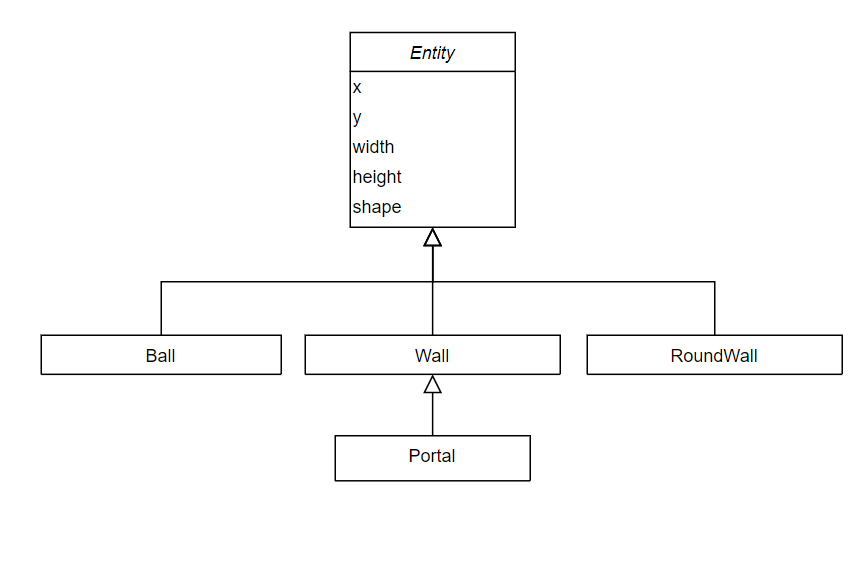


(10. ábra: szerver mappaszerkezete)

A gameserver.js fájl az egész szerver magja, ezt az egy fájlt kell futtatnunk a működéséhez, és nem tesz mást, mint felkészül a kliensek fogadására, betölti az első pályát és elkezdi a fizikai szimulációt, mindezt a server.js, mapLoader.js és az engine.js fájlok segítségével. Erre a három fő mozgatóelemre épített bontást szeretném bemutatni az elkövetkező oldalakon.

## Fizika

A fizika néhány általam megvalósított objektumon működik, amelyeket a következő diagram (11. ábra) szemléltet:



(11. ábra objektumok)

## Entity

Ez a fizikai ősosztály melyből az össze többi származik, ez az osztály van olyan alapvető attribútumokkal és metódusokkal felruházva, amelyekre minden egyéb fizikai szimulációban résztvevő egyednek szüksége lesz.

Attribútumok:

* objID

azonosító szám, amely egyedi minden objektumnál

* x, y

Az objektum pozíciója

* width, height

Szélesség, magasság

* shape

Az objektum formája (pl.: négyzet vagy téglalap)

* vx, vy

horizontális és vertikális sebesség

* mass

tömeg, egy képzeletbeli hozzárendelt tömeg az objektumhoz, fontos szerepet játszik például két labda ütközésénél.

Metódusok:

* getCenter

Az objektum középpontját visszaadó metódus

* getDir

ha mozgásban van az objektum akkor visszaadja a mozgás irányát

* getLeft/getTop/getWidth/getHeight

Az objektumot körül határoló téglalap széleinek koordinátáit visszaadó metódusok

* physicsUpdate(deltaTime)

ebben a függvényben kell megadnunk, hogy milyen fizikai változások történjenek az objektummal. És ez a fizikai szimuláció alapvető függvénye, ez fut le minden objektumra amikor egy szimulációs iteráció végbemegy. Van 1 paramétere is a „deltaTime”, aminek segítségével megmondhatjuk, hogy mennyi idejű mozgást kell elvégezni az adott függvényhívás alatt.

## Ball

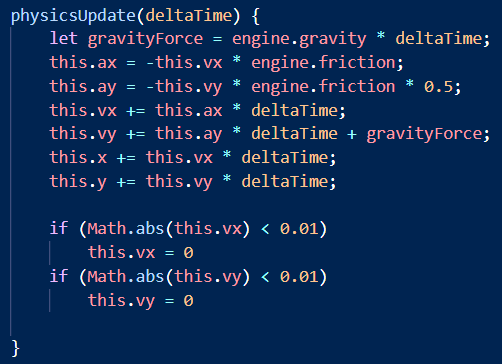
Ez az osztály valósítja meg a közismert labdákat. Egy labdára hat a gravitáció, ha meglökik elgurul, és a súrlódás hatására lelassul majd megáll. Ha egy labda ütközik egy másik labdával akkor azok dinamikusan ütköznek, ha pedig egy szilárd felülettel (egyszerű és lekerekített fal) akkor arról visszapattan.

* Konstruktor (x, y, r, vx, vy)

Egy labda szabályos létrehozásához az első három paraméterre van szükségünk, amelyek megadják a labda középpontját és a sugarát. Az utolsó két paraméterrel pedig kezdősebeséget adhatunk meg a labdánknak, ezek alapértelmezetten 0 értéket vesznek fel.

* Mozgás

Egy labda mozgását a fentebb említett physicsUpdate metódusában valósítottam meg a következő módon (12. ábra):



(12. ábra: egy labda mozgása)

Itt a függvény első sorában kiszámoljuk mekkora gravitációs erő hasson a labdára, a következő két sor megmondja, hogy a sebessége mennyivel változzon. Majd az elkövetkezendő 4 sorban alkalmazzuk is a kapott értékeket. Ez mind az eltelt idő „deltaTime” függvényében történik, és a függvény végén található 2 elágazás, amelyek segítenek abban, hogy a nagyon kicsi mozgásokkal ne törődjünk, azokat úgy sem tudjuk megjeleníteni.  
Ez a mozgásmegvalósítás nem valódi fizikai képletek alapján működik, csak egy látszólagos hihető közelítése.

Ha kívülről szeretnénk befolyásolni egy adott labda mozgását akkor csak a labda vx és vy attribútumait kell módosítanunk a kívánt értékekre.

## Wall

Egyszerű téglalap alakú fal megvalósítása, amelyen a labdák nem tudnak áthaladni, hanem visszapattannak róla.

* Konstruktor (x, y, w, h)

Első két paraméter a fal bal felső sarka, majd a szélesség és magasság. Ha nincs magasság megadva akkor alapértelmezetten négyzetnek fogja megvalósítani a fal alakját.

* getCorners

Egy segédfüggvény amely egy A,B,C,D négyessel tér vissza, amelyek rendre a fal sarok koordinátáit jelölik jobb felülről kezdve óramutatóval ellentétes bejárással.

## RoundWall

Lekerekített fal, a sima téglalap alakú fallal megegyezően viselkedő fal rész, annyi különbséggel, hogy a végpontjai le vannak kerekítve és állása lehet ferde is (13. ábra).



(13. ábra: egy példa a lekerekített falra)

* Konstruktor (sx, sy, ex, ey, r)
  + sx és sy a fal kezdetének x és y koordinátája
  + ex és ey pedig a fal végének a koordinátái
  + r pedig a fal szélessége

## Portal

A portál a játék fő eleme, ez egy olyan objektum, amelyekből egyszerre 2-őt tudunk falakra (Wall) helyezni és ha az egyikbe „belemegy” egy labda akkor az a másikon kijön megfelelően elforgatva a sebességvektoraival.

* Konstruktor (x, y, w, playerID, color)

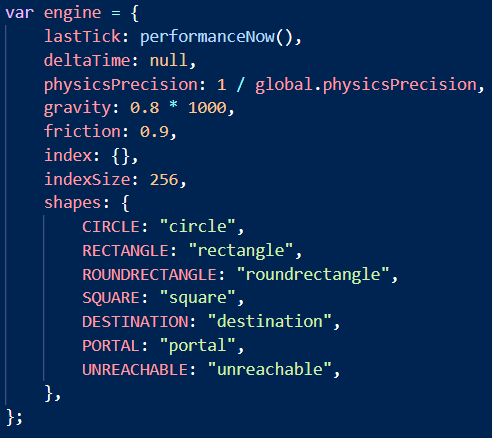
Egy portál kinézet ügyileg megegyezik egy négyzetes fallal, így az első 3 paraméter megegyezik a Wall konstruktorával, ezen kívül megadható, hogy melyik játékos birtokolja, és hogy milyen színű legyen a portál  
A játékban most egyelőre csak a kék és a piros színű portálok játszanak szerepet, ők alkotnak egy part, ha a ugyan azon játékoshoz tartoznak.

A portál konstruktorában hozzá rendelünk egy kisebb négyzetet, ez lesz az a négyzet, amivel vizsgálni fogjuk, hogy találkozik-e vele labda, ha igen akkor úgy tekintjük, hogy az belement a portálba. Ennek az attribútumnak a neve „portalRect”.

## Fizika motor

A játék fizikai szimulációja saját kezűleg lett megvalósítva a fentebb említett engine.js fájlban.

Az engine.js fájl néhány beégettet paraméterrel (11. ábra) kezdődik, amelyek teljes mértékben meghatározzák a fizika viselkedését a szimuláción belül:



(11. ábra: a fizika paraméterei)

Néhány fontosabb paraméter elmagyarázása:

|  |  |
| --- | --- |
| Paraméter | Hatás |
| gravity | milyen gyorsan esnek le a labdák |
| friction | Milyen gyorsan vesztik el a horizontális sebességüket |
| physicsPrecision | Másodpercenként hányszor számoljon fizikát |

A többi paraméterre később kerül sor használat közben a könnyebb érthetőség végett.

## Játéktér

## Kommunikáció

# Hivatkozások

|  |  |
| --- | --- |
| [1] | „Portal,” 11 05 2020. [Online]. Available: https://en.wikipedia.org/wiki/Portal\_(video\_game). |
| [2] | „WebSocket,” 10 05 2020. [Online]. Available: https://javascript.info/websocket. |
| [3] | „Nodejs,” 11 05 2020. [Online]. Available: https://nodejs.org/en/about/. |