Szakdolgozat

Tartalom

[Bevezetés 4](#_Toc415189422)

[Mitől jó a játék 4](#_Toc415189423)

[Elterjedt játéktípusok 4](#_Toc415189424)

[Többjátékos online harci aréna (MOBA) 4](#_Toc415189425)

[Háborújáték 5](#_Toc415189426)

[Vidd haza a zászlót (CTF) 5](#_Toc415189427)

[Játék bemutatása 5](#_Toc415189428)

[A saját elképzelés 5](#_Toc415189429)

[Több játékos, nem csak a pályán 5](#_Toc415189430)

[Tervezés 6](#_Toc415189431)

[Elvárások a játékkal szemben 6](#_Toc415189432)

[Fejlesztési platform megválasztása 6](#_Toc415189433)

[Használt technológiák 7](#_Toc415189434)

[Irányelvek 7](#_Toc415189435)

[Szerveroldal 7](#_Toc415189436)

[Node.js 7](#_Toc415189437)

[Websocket 7](#_Toc415189438)

[Kliensoldal 7](#_Toc415189439)

[HTML5 Canvas 7](#_Toc415189440)

[Felületi tervek 8](#_Toc415189441)

[Megvalósítás előkészületei 9](#_Toc415189442)

[Futtatókörnyezet telepítése 9](#_Toc415189443)

[Fejlesztőkörnyezet megválasztása 9](#_Toc415189444)

[Verziókövetés 10](#_Toc415189445)

[Megvalósítás 11](#_Toc415189446)

[Modulokra bontás 11](#_Toc415189447)

[Kapcsolatkezelő 11](#_Toc415189448)

[Parancsértelmező 11](#_Toc415189449)

[Csoportkezelő 11](#_Toc415189450)

[Játékvezérlő 12](#_Toc415189451)

[A vezérlők közti kommunikáció 12](#_Toc415189452)

[A játékmotor 13](#_Toc415189453)

[A játékmotor célja 13](#_Toc415189454)

[Rugalmas ütközése 13](#_Toc415189455)

[Ütközés fallal 13](#_Toc415189456)

[Ütközés játékossal 13](#_Toc415189457)

[Pontszerzés és annak logikája 13](#_Toc415189458)

# Bevezetés

A szórakozás fogalma generációról generációra változik. Modern világunkban számítógépek segítik munkánkat. A mozgalmas hétköznapok közepett el is felejtjük, hogy milyen lehetőségek vannak karnyújtásnyira. Így számítógépeink szórakozató képességeit hajlamosak vagyunk figyelmen kívül hagyni.

A fejlődő technika újabb és újabb vívmányai lehetővé teszik az egyre látványosabb grafikai szoftverek hétköznapi használatát. Ennek a legnagyobb piaca a Játékipar.

De a játékoknak nem kell feltétlenül lélegzetelállító grafikai elemeket mutatnia ahhoz, hogy sikeresek legyenek.

## Mitől jó a játék

Nem a grafika határozza meg a játék élvezeti értékét, ezt napjaink egyik legnépszerűbb játéka a Minecraft is igazolja. Alapvető grafikai elemeket használhatunk ötleteink megépítésére. Akár barátainkkal vagy más játékosokkal közreműködve.

Most eljutottunk mindenidők legsikeresebb játékaihoz. Ezek a játékok versenyszerűek, csapatok harcolnak csapatok ellen. A játék az ellenfél kicselezésével nyerhető meg. Az efféle játékoknak mesterévé válásához rengeteg gyakorlás szükséges. Név szerint a Quake 3, StarCraft 2 és a League of Legends képviselik legjobban ezt a kategóriát.

## Elterjedt játéktípusok

A versenyszerű játékoknak néhány fő fajtája ismert, mindegyiknek rengeteg alváltozata, újraértelmezése létezik. A legjelentősebb tulajdonsága az efféle játékoknak, hogy a játék célja jól ismert és egy ponton a nyertes egyértelműen kihirdethető. Másik közös tényező a csapatokra bontottság. A csapat egységes célért küzd, ennek nevében minden csapattagnak megvan az egyedi szerepe. A játéktól függően a csaptok méretében jelentős különbségek lehetnek. A továbbiakban a leglényegesebb játéktípusokat ismertetem

### Többjátékos online harci aréna (MOBA)

A játékosok csapatokat alkotnak és egy kijelölt területen az „arénában” küzdenek meg. A játék célja lehet egyes pontok elfoglalása, vagy az összes ellenfél megsemmisítése. Leggyakoribba az 5-5 és a 3-3 felállás. A játékban fontos a csapat együttműködése és a választott stratégia.

### Háborújáték

Itt a csapatok helyett seregekről beszélhetünk, melyek nyílt területen mérkőznek meg. A játék során a seregek célja, az ellenséges bázis elfoglalása. Ebben a játékmódban szerepe van a stratégiának, de a csapat toborzás sokkal fontosabb.

### Vidd haza a zászlót (CTF)

Mindkét csapat a bázisán lévő zászlót védi, és az ellenfél zászlóját próbája megszerezni. A játékosok szabadon mozognak a pályán, de ellenfelek akadályozhatják egymást. Az ellenséges zászló könnyen elrabolható, a zászlót csak meg kell érinteni. Amint ez megtörtént a zászló hordozója sérülékennyé válik, a zászló könnyen visszaszerezhető. A rablási kísérlet megállításáért pont jár. De ha ez nem sikerül és az elrabolt zászlót saját bázisáig juttatja egy játékos. Azzal a csapat pontot szerez.

## Játék bemutatása

A bemutatottak közül a zászlós játékmód nyerte el tetszésem. A változatos és gyors játékmenet, melltett a csapatok együttműködésének is szerepe van a játék menetében. Sőt az ellenfél kicselezése a legfőbb cél. A játék viszonylag könnyen megtanulható, nincs szükség bonyolult szabályokra. Így a játék alkalmas baráti játszmák lebonyolítására, jelentős ráfordított tanulási periódus nélkül is, viszont van lehetőség a fejlődésre. A kezdő és a gyakorlott játékos között érezhető különbségek lesznek. A csapatok együttműködése még ennél is fontosabb, és még több teret ad a versenyszerűségnek.

### A saját elképzelés

A játék megvalósításánál nem a látványra fókuszálok, inkább a játszhatóságra és a versenyszerűségre. Az egyszerű formák nem terelik el a játékos figyelmét. A minimalista összeállítás segít a gyors döntéshozásban. Minden játékos saját csapata színét viseli. A csapatok bázisát és zászlóját szintén a csapatszín jelöli. A csapatszínek jellemzően kiegészítő színpárok, tehát jól elkülönülnek. A játéktér lehetőleg szimmetrikus, így egyik csapat sem jut semmiféle előnyhöz.

## Több játékos, nem csak a pályán

A felvázolt játékot egy időben több játékos játssza, a játéktípus magában, legalább 4 játékost feltételez. A játékosok helyi hálózaton vagy az interneten keresztül kapcsolódnak egy központi kiszolgálóhoz. A játék megvalósítása mellett egyéb funkciókra is szükség lesz. A játék megszervezésére és a játékot megelőző egyeztetésére is alkalmasnak kell lennie a felületnek. Kézenfekvő megoldás a kommunikációra rövid üzenetek cseréje. A játék tehát nem csak magában létezik, egyéb szolgáltatások veszik körül, így egy chat

# Tervezés

## Elvárások a játékkal szemben

A célom, hogy a játékot minél több felhasználó kipróbálhassa, élvezhesse, a már általa megszokott környezetben. Elsősorban asztali számítógépekre készül, de a felsőkategóriás hordozható eszközökön is elvárható a megfelelő működés. Elsődleges a felhasználó kényelme. Telepítés nem szükséges. Egy modern böngésző képes a szükséges rajzolási és hálózati teljesítmény kezelésére. A böngészős megvalósítás nagy előnye, hogy a szabványos megoldásoknak köszönhetően különböző eszközökön, különböző operációs rendszereken is ugyanarra a végeredményre számíthatunk.

Másfelől megközelítve más elvárásokkal találkozunk. Egy technikai szempontból fontosnak tekinthetjük a játék rajzolási teljesítményét vagy hálózati átvitelkésési tényezőjét. Egy közvetlenül az operációs rendszerre írt program nagyobb teljesítményt biztosít, így például csökken a minimális hardverkövetelmény. Ezért viszont kényelmi pontokat veszítünk, hiszen az ilyen programot telepíteni kell a böngészős változattal szemben, ahol csak egy weboldalt kell felkeresnünk. A modern böngészők és az elérhető áru számítógépek világában a legtöbb felhasználónak nem lesz gondja a teljesítménnyel.

## Fejlesztési platform megválasztása

A böngészős alkalmazásunk programozásához több megoldás közül választhatunk. Lehetőségünk van kiegészítőket használni például Adobe Flash, vagy Microsoft Silverlight. Ezeket viszont nem a böngésző tartalmazza, hanem előre telepítendőek. Ezek elavulttá váltak a böngészőkbe épített JavaScript futtatókörnyezet fejlődésével. Ez azt jelenti, hogy csakis egy tetszőleges modern böngészőt például a Mozilla Firefox-ot vagy a Google Chrome-ot kell beszereznünk, ha még nem tettük. Ez tehát a felhasználói oldal.

A kiszolgáló környezet megválasztása már nehezebb kérdés. Sok szempontot kell figyelembe venni és a választék is igen széles. A kiszolgáló oldal tervezésekor lényeges hogy hatékony megoldásokat válasszunk, hiszen a kiszolgáló központi csomópont, minden felhasználó ide csatlakozik, így ha ez nem képes tartani az iramot, azt minden játékos megérzi. Célszerű elterjedt megoldásokat használni a széleskörű támogatottság miatt.

## Használt technológiák

## Irányelvek

A JavaScript nyelvben lehetőség van az objektumorientált programozásra. Ez nagyon ajánlott, de nem kötelezettség. A probléma részekre bontása segíti a megértést. Egymással szoros kapcsolatban lévő blokkokat alakíthatunk ki. Egy probléma objektumorientált megoldásénak elkészítése jellemzően több munkát és tervezés igényel, de ez a befektetés könnyen megtérül hosszútávon. A később eszközölt változtatások viszont nagyságrendekkel egyszerűbben elvégezhetőek.

## Szerveroldal

### Node.js

A Node.js egy szerveroldali JavaScript alapú futtatókörnyezet. A Chrome V8 JavaScript motorjára épült, amely már évek óta a legjobban teljesítő JavaScript-motor. Sikerét jórészt annak köszönheti, hogy a programot közvetlenül gépi kódra fordítja. Így nagyságrendekkel javul a teljesítmény az értelmezett, vagy a bájt kódra fordított megoldásokkal összehasonlítva.

Ezen kívül esemény vezérelt, ez annyit jelent, hogy eseményeket készíthetünk és ezek bekövetkezéséhez logikát vagy agy további eseményeket rendelhetünk. A Node.js jelenlegi verziója már C++ kiegészítéseket is kezel. Így akár C++ könyvtárakat is használhatunk vagy esetleg a közvetlen memóriaelérés is megoldható a még nagyobb teljesítmény érdekében. A Node.js híres még a méretezhetőségéről, különösen jól teljesít kis, független feladatok elvégzésében. Több nagy cég is alkalmazza, így a Yahoo a PayPal és az eBay. Mindez bizonyítja, hogy alkalmas valós idejű kiszolgálók megvalósítására. Fontos jellemző még, hogy mind Windows, Linux és Macintosh rendszereken is használható.

### Websocket

Valósidejű alkalmazásokhoz elengedhetetlen a gyors kétirányú adatkapcsolat. A böngészők eredetileg, http kéréseket küldtek, majd az erre kapott válasz után lezárták a kapcsolatot. Ez dokumentumok betöltésére hatékony, de folyamatos oda-vissza kommunikációra kényelmetlen és lassú.

## Kliensoldal

### HTML5 Canvas

A böngészők régóta képesek grafikai elemek megjelenítésére. Ez az elmúlt években ezen a téren jelentős előrelépések voltak mind funkcionalitásban mind teljesítményben. A HTML5-ös szabványos vászonra JavaScript-el rajzolhatunk alakzatokat, így téglalapot, kört, szöveget vagy akár raszter, sőt vektorgrafikus képeket is. A canvas alapú megjelenítéssel együtt jár annak könnyű beillesztése a felületbe, hiszen a böngésző saját stílusbeállításait használhatjuk.

## Felületi tervek

# Megvalósítás előkészületei

## Futtatókörnyezet telepítése

A választott futtatókörnyezet Node.js, ezt Windowson telepíthetjük a nodejs.org-ról letölthető Windows Installer-el. Linux rendszereken legkönnyebben parancssorból telepíthetjük az „*apt-get install nodejs*” paranccsal. De akár portable (hordozható) változatot is beszerezhetünk, ebben az esetben nincs szükség rendszergazda jóváhagyására, de számolnunk kell apróbb kellemetlenségekkel.

A teszteléshez, és a felület megvalósításához Mozilla Firefox-ot és Google Chrome-ot is használok Az internetezők több mint 60%-a e két böngésző egyikét használja. Így biztosítható, hogy mindkét böngészőben minden zökkenőmentesen működik.

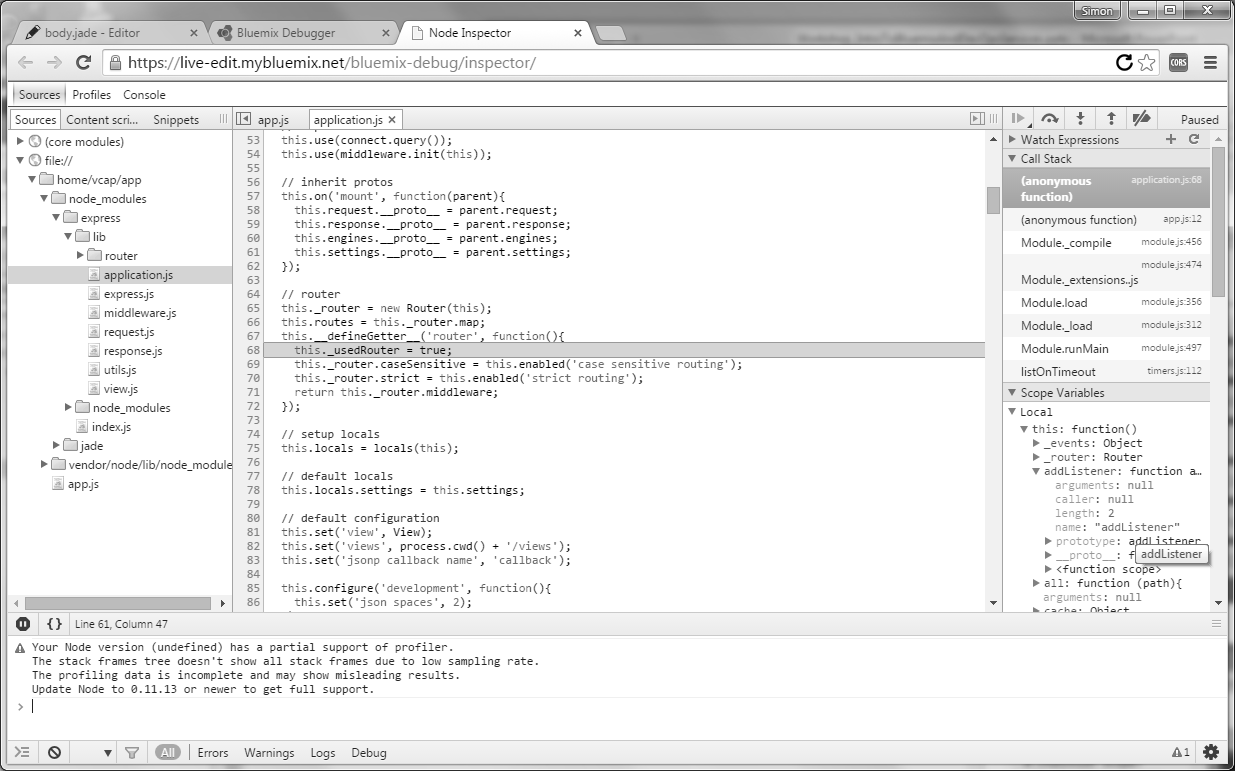
## Fejlesztőkörnyezet megválasztása

A web fejlesztők többsége egyszerű szövegszerkesztőt használ a fejlesztőmunka során. Windowson a legelterjedtebb a könnyen használható Notepad++. Támogatja a szintakszis kiemelés és a forrásfájlok automatikus formázására is van lehetőség. Viszont csak kezdetleges szókiegészítésre képes Természetesen a feladat megoldására teljesen alkalmas.

A szoftverfejlesztés során viszont nem csak a forrásfájlok at szerkesztünk. A program logikájának megvalósítását menet közben találjuk ki. Az így megírt kódblokkok tesztelése természetes és szüksége. Előfordulnak azonban összetett részfeladatok, amelyek elkészítése nem skierül egyből. A hibásan megírt program javítását hibakeresésnek vagy debugolásnak hívjuk.

A hibakeresés során speciális hibakereső szofvert használun. Amely lehetővé teszi a program lépésenkéinti futtatását. Így képet kaphatunk a progra futásának bármely pillanatáról. Viszgálhatjuk a változók értékei, és a programot soronként léptethetjük, ezzel a hibák sokkal könnyebben felderíthetőek, mintha csak egy hibás eredmény vagy hibaüzenetet látnánk a program lefutása után.

Az általam választott platform is rendelkezik ilyen eszközökkel. Legelterjedtebb a node-inspector. Ez webes felületen engedi programunk vizsgálatát.



## Verziókövetés

A szakdolgozat készítési folyamat egésze alatt verziókövetést használok. Erre a legalkalmasabb a Git nevű verzókövető szoftver. Egyaránt képes kis és nagy projektek kezelésére. A verziókövetés segít átlátni a fejlesztések haladását, sorrendjét. A verziókövetés néhány egyszerű műveletből áll. A szerkesztett fájlokat először kiválasztjuk szakszóval stage-eljük majd amint elvégeztük a kívánt módosításokat és úgy gondoljuk, hogy érdemes rögzíteni haladásunkat, véglegesítjük másszóval commit-oljuk az új verziót. Lehetőség van még új haladási szálak létrehozására. Az így keletkezett ágakat a változtatások összesítésével egyesíthetjük, így új verzió alakítható. A kényelmi funkcionalitás sem elhanyagolható. Egy parancs kiadásával továbbíthatjuk a helyi változtatásokat egy központi tárolónak. A verziókezelés egyszemélyes projekteknél nem kulcsfontosságú, de felgyorsítja és átláthatóvá teszi a programfejlesztést.

# Megvalósítás

A program írása közben Igyekeztem előre gondolkodni, és minél szakszerűbb, könnyen megérthető megoldásokat produkálni. Ennek a tervezési folyamatnak során határoztam el, hogy külön fájlokat fogok létrehozni a programegységeknek. Ez segíti az átláthatóságot és így a szerkesztés folyamatát.

## Modulokra bontás

A Node.js támogatja a moduláris programozást, tehát tartalmaz egy modul betöltő rendszert. A modulok készítése egyszerű csak az exports változóba kell írnunk tetszőleges néven az elérni kívánt függvényt, vagy értéket. Amennyiben egyetlen objektumot szeretnénk exportálni, használhatjuk a module.exports változót. Így nem kell jellemzőnkként felépíteni az objektumot. Modulok betöltésére a require fúggvényt használhatjuk, melynek első paramétereként a betölteni kívánt fájl nevét kell megadnunk. Ez mind a Node.js jellegzetessége, tehát a szabványos JavaScriptnek így a V8-nek sem szerves része.

## Kapcsolatkezelő

A kiszolgálóarhiterktúra szélső eleme. Minden beérkező csomag ezen halad át. Folyamatosan hallgat a megadott TCP porton. Új kapcsolat vagy üzenet esetén gondoskodik arról, hogy a kiszolgáló egyéb érintett részei tudjanak a bejövő információról. Fontos szerepe van még kapcsolatok lezárásában.

## Parancsértelmező

A parancsértelmező a központi kiszolgáló és a kapcsolatkezelő közt helyezkedik el. Minden olyan üzenet, ami a kapcsolatkezelőn túljut, a parancsértelmezőbe jut. A parancsértelmező feladata az üzenetek feldolgozása, majd a döntés meghozatala. Parancsnak számít minden olyan üzenet, ami per (/) jellel kezdődik. Ezek értelmezve vannak. Minden egyéb üzenet felhasználói üzenetként jelenik meg.

## Csoportkezelő

A csoportvezérlő a kiszolgáló magját képezi. A csatlakozó felhasználók alapértelmezetten egy speciális csoportba kerülnek. Innen indulva készíthetnek saját csoportot, vagy csatlakozhatnak egy már létezőhöz. A csoportok jelszóval védhetőek. Minden csoportnak van egy tulajdonosa, aki jogosult azt átnevezni vagy a csoport jelszavát megváltoztatni. Csoportot viszont a csoport tulajdonosa sem szüntetheti meg. A csoport csak abban az esetben törlődik, ha azt minden felhasználó elhagyta. Amennyiben a csoportot tulajdonosa elhagyja, a csoport új tulajdonosa az első csatlakozott játékos lesz.

## Játékvezérlő

A Játékvezérlő a programnak azon része, amely az egyes Játékok lebonyolítását végzi. Egy-egy játékvezérlő rendelhető a mindegyik csoporthoz. A játékvezérlő már valós idejű technológiát alkalmaz. Így például képes követni minden egyes játékos nyomva tartott billentyűit, és erre közvetlen választ is képes küldeni. A játékvezérlő szintén modulokból áll. A leglényegesebb a csalkozó vagy távozó játékosok kezelője, a játéklogika, és a kliensek kiszolgálásáért felelős részek.

## A vezérlők közti kommunikáció

A vezérlők közti kommunikáció megtervezései igazi kihívásnak bizonyult. Szem előtt tartottam a modulok hierarchiáját, és ehhez illesztettem a kapcsolati sémát is. Viszont bizonyos esetekben megspóroltam egy-egy felesleges lépést. Ennek értelmében a vezérlők bizonyos műveleteket egy közös objektumon keresztül bonyolíthatják le. Ez sok extra munkát spórol meg.

# A játékmotor

## A játékmotor célja

## Rugalmas ütközése

## Ütközés fallal

## Ütközés játékossal

## Pontszerzés és annak logikája