Matematikai és Informatikai Intézet

Játékfejlesztés Unity keretrendszerben

Készítette
Szabó Márk
programtervező informatikus Bsc.

Témavezető
Troll Ede
tanársegéd

Eger, 2025

Tartalomjegyzék

Bevezetés **4**

1. Technológiai áttekintés **5**
1.1. Enginek 5
1.1.1. Unreal 5
1.1.2. Godot 6
1.1.3. Unity 6
1.2. Többjátékos technológia a játékfejlesztésben 6
1.2.1. Korai megoldások 6
1.2.2. Kliens-Szerver 7
1.2.3. Peer-to-Peer (P2P) 8

2. Unity **9**
2.1. Többjátékos technológiák a Unity-ben 9
2.1.1. Relay 9
2.1.2. Technológia1 9
2.1.3. Technológia2 9
2.2. Modul1 9
2.3. Modul2 9

3. Rendszerterv **10**
3.1. Rendszer célja 10
3.2. Követelmények 10
3.3. Architekturális terv 10
3.4. Használt fejlesztői eszközök 10

4. Saját szoftver megvalósítása **11**
4.1. Texas Hold'Em 11
4.1.1. Bevezetés 11
4.1.2. Szabályok ismertetése 11
4.1.3. Használt kézkiértékelő algoritmus 14
4.2. Többjátékos kapcsolat megvalósítása 18

4.2.1. Kapcsolat kezelése a PC oldalon 18
4.2.2. Kapcsolat kezelése a Mobile oldalon 18
4.3. Játékmenet megvalósítása 18
4.3.1. Modul1 18
4.3.2. Modul2 18

5. Tesztelés **19**
5.1. SharedDLL tesztelése 19
5.2. PC játék tesztelése 19
5.3. Mobile játék tesztelése 19
5.4. Általános tesztelés 19

Összegzés **20**

Irodalomjegyzék **21**

Bevezetés

1. fejezet

Technológiai áttekintés

Manapság a játékfejlesztés legelterjedtebb módja a játékmotorok használata. Alternatív megoldásként azonban még előfordul az API-alapú fejlesztés is, amely során különféle API-k – például grafikai vagy multimédiakezelő API-k – segítségével történik a fejlesztés. A szakdolgozatom elkészítéséhez én a játékmotor-alapú fejlesztést választottam, így ki tudtam használni az általa kínált lehetőségeket és eszközöket.

1.1. Enginek

A játékmotorok olyan fejlesztői környezetek, amelyek célja, hogy egyszerűsítsék és felgyorsítsák a játékok fejlesztésének folyamatát. Egy tipikus játékmotor különböző komponenseket és funkciókat szolgáltat a fejlesztőnek, mint például: grafikai renderelést, fizikai szimulációt, animációkezelést, hangkezelést. A játékmotorok használata megkönnyíti a fejlesztők dolgát, viszont nem ad annyi rugalmasságot, mint az API alapú fejlesztés. [1]

1.1.1. Unreal

Az Unreal Engine egy professzionális, nagy teljesítményű játékmotor, amelyet az Epic Games fejleszt. Főbb előnyei közé tartozik a kiemelkedő grafikája. Az ingyenes hozzáférése és a vizuális programozást lehetővé tevő Blueprint rendszere miatt kisebb fejlesztők körében is népszerűvé vált. Elsősorban C++-ban történő fejlesztésre optimalizált, de a már említett Blueprint-rendszernek köszönhetően programozói tapasztalat nélkül is lehetséges játékot fejleszteni benne. A játékmotorban sok újrahasználható komponens megtalálható, így a gyakran használt dolgokat pl. karakter mozgást, nem kell felépíteni az alapoktól. A játékmotort 3D játékok fejlesztésére találták ki. [2]

1.1.2. Godot

A Godot egy nyílt forráskódú, ingyenesen elérhető játékmotor. Nagy előnye a könnyű használhatóság, a kis erőforrás igénye, valamint a nyílt forráskód nyújtotta rugalmasság és közösségi támogatás. Támogatja a 2D és 3D játékfejlesztést egyaránt, natív scriptnyelve a GDScript (Pythonhoz hasonló nyelv). [3]

1.1.3. Unity

A Unity az egyik legismertebb és legelterjedtebb játékmotor a piacon, amely támogatja a mobil, asztali, konzolos, valamint webes játékok fejlesztését 2D és 3D játékokhoz egyaránt. Jelentős előnye a könnyen tanulható felhasználói felület és a jól dokumentáltság. Elsősorban a C# nyelvet használja scriptelésre, de támogatja a vizuális programozást is. A játékmotorhoz tartozik több különböző előfizetési szint is, de van ingyenesen használható verziója is. Mivel a C# közel áll hozzám, tapasztalatom is van már ezzel a játékmotorral és maga a játékmotor adottságai pont megfeleltek a szakdolgozatom követelményeihez, ezért erre a játékmotorra esett a választásom. Erről a játékmotorról a 2. fejezetben bővebben írok. [4]

1.2. Többjátékos technológia a játékfejlesztésben

A mai megjelenő játékok nagy része már tartalmaz valamiféle többjátékos módot. Ez a játékfejlesztés elején még csak helyi megoldásokat jelentett, de ma már hatalmas online terekben játszódó játékokról is beszélhetünk.

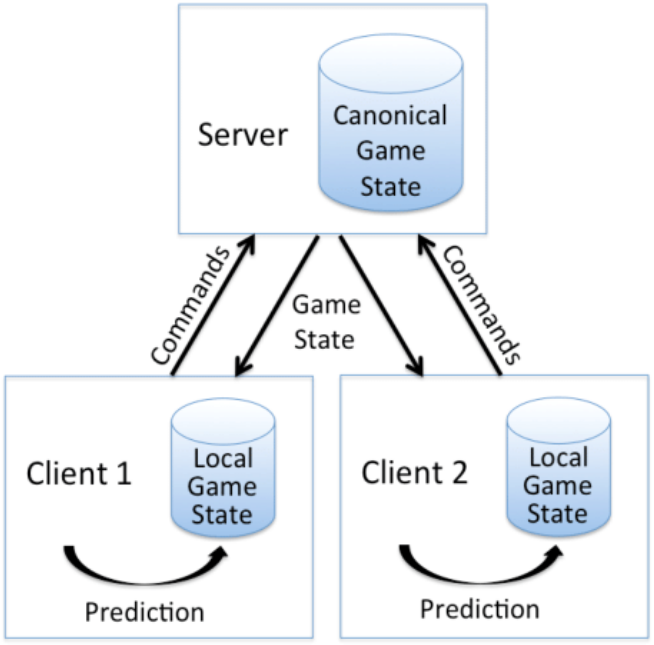
1.2.1. Korai megoldások

Többjátékossal játszható játék először stratégia kör alapú játékokban jelent meg, de ez még nem egyidejű játékot jelentett. Ezek után egyidejű többjátékos játék lehetősége először egy 2 játékossal egyszerre egy helyen játszható játékkal jött létre, ez a játék volt a jól ismert Pong. A Pong után kezdtek egyre jobban elterjedni a helyi többjátékos játékok, ahol az egymás elleni játék helyett, egymással együtt működve is tudtunk játszani. Egy népszerű módszer a helyi többjátékos játékok megvalósításánál az osztott képernyő, ahol általában el van választva a kijelző több egyenlő részre, ahol 1-1 részt 1-1 játékos kezelhet. A 2000-es évek elején pedig már megjelent az online többjátékos játékok lehetősége is, ahol már nem volt szükség egy térben tartózkodni a többi játékossal. [5]

1.2.2. Kliens-Szerver

A Kliens-Szerver architektúra az egyik legtöbbet használt módszer az online többjátékos játékok megvalósításához. Az architektúrában 2 szerep van:

- Szerver:
  - Felelős a játék logikájának feldolgozásáért, az interakciók kezelésért, az erőforrások kezeléséért és a kliensek közötti információk szinkronizálásáért.
  - A szerver lehet egyetlen gépen elhelyezett (dedikált szerver) vagy a játékosok egyik gépén futó (hosztolt szerver).
  - A szervernek egyszerre több kliens kéréseit kell kezelnie, ami azt jelenti, hogy skálázhatónak és hatékonynak kell lennie.
- Kliens:
  - A játékos játékpéldánya. Bemeneti parancsokat küld a szervernek, és frissítéseket kap a játék állapotáról.
  - Felelősek a játékvilág renderelésért és a helyi interakciók kezeléséért.



1.1. ábra. Kliens-Szerver architektúra

A kliens és a szerver megbízható hálózati protokoll (pl. TCP vagy UDP) segítségével kommunikál, és olyan csomagokat küld, amelyek a játékosok interakcióiról, játékeseményekről és állapotfrissítésekről tartalmaznak információkat.

Végső sorban szerver ellenőriz minden interakciót és csak ő tud dönteni a változásokról. Ez azért jó, mert így a csalásokkal szemben is védekezik a játék.

Mivel mindig várni kell a szerver által küldött információra egy interakció esetén, így késleltetés léphet fel, főleg lassabb internet kapcsolat esetén. Erre egy módszer amit alkalmazni szoktak a klienseknél az úgynevezett predikció, ahol a helyi interakció alapján a kliens megpróbálja kitalálni mi fog ténylegesen történni. [6]

1.2.3. Peer-to-Peer (P2P)

2. fejezet

Unity

2.1. Többjátékos technológiák a Unity-ben

2.1.1. Relay

2.1.2. Technológia1

2.1.3. Technológia2

2.2. Modul1

2.3. Modul2

3. fejezet

Rendszerterv

Azok az elemek, amiket tanultatok RFT-n, azok kerülnek ide

3.1. Rendszer célja

3.2. Követelmények

3.3. Architekturális terv

PC unity, Mobile unity, SharedDLL

3.4. Használt fejlesztői eszközök

4. fejezet

Saját szoftver megvalósítása

4.1. Texas Hold'Em

4.1.1. Bevezetés

A póker a világ egyik legismertebb kártyajátéka, amelyben a játékosok célja, hogy a saját lapjaikból a lehető legjobb kombinációt kialakítva megszerezzék az asztalon lévő kasszát. A póker számos különböző szabályrendszerrel rendelkező változatban létezik.

A *Texas Hold'Em* a közösségi pókerjátékok legnépszerűbb formája, amelyet jellemzően 2 és 10 játékos között játszanak. Ez egy viszonylag zárt struktúrájú játék, ahol a licitálás menete állandó szabályok szerint zajlik.

4.1.2. Szabályok ismertetése

Ebben a fejezetben a póker legelterjedtebb, hivatalos versenyeken is alkalmazott szabályrendszere kerül bemutatásra. Ami közös az összes variációban, hogy a játékot 52 lapos francia kártyával játsszák dzsókerek nélkül.

Játék menete

A játék során három fontos szerep forog körbe a játékosok között, amit „gombokkal" jelölünk. Ezek a szerepek az osztó, kis vak és nagy vak. Az osztótól balra ülő játékos lesz a kis vak, a kis vaktól balra ülő pedig a nagy vak. Az osztót pedig több különböző módon választhatjuk meg a játék elején.

Az osztó keveri és osztja ki a lapokat a szabályok szerint. A vakok pedig még osztás előtt kötelesek betenni a vak téteket, ahol a kis vak tét általában a nagy vak tét fele. [8]

1. *Osztás*

- Az osztó először megkeveri a paklit. A kiosztás előtt a kis vak és a nagy vak beteszik a kötelező tételeket. Ezt követően az osztó balról kezdve, két körben, egyesével oszt minden játékosnak egy-egy zárt lapot.

2. *Pre-Flop (első licitkör)*
   - A licitálás a nagy vaktól balra ülő első játékossal kezdődik, aki az alábbi lehetőségek közül választhat:
     - Tartás – megadja az aktuális tétet.
     - Emelés – növeli a tétet a limitszabályok szerint.
     - Dobás – eldobja a lapjait, ezzel kiszáll a játékból.
   - A licitálás az óramutató járásával megegyező irányban halad tovább.

3. *Flop (második licitkör)*
   - Az osztó egy lapot félretesz égető lapként, majd három közös lapot felfordítva az asztal közepére helyez.
   - A licitálást az osztógombtól balra ülő első aktív játékos kezdi, és az alábbi lehetőségek közül választhat:
     - Passzolás – nem emel, de marad a játékban.
     - Nyitás – tétet tesz be a limitszabályok szerint.
     - Dobás – eldobja a lapjait és kiszáll a körből.
   - Ha valaki nyit, a többiek dönthetnek:
     - Tartás – megadják a tétet.
     - Emelés – növelik a tétet.
     - Dobás – kiszállnak a körből.

4. *Turn (harmadik licitkör)*
   - Az osztó ismét éget egy lapot, majd egy negyedik közös lapot felfordítva az asztalra helyez.
   - A harmadik licitkör a második licitkörhöz hasonlóan zajlik.

5. *River (negyedik licitkör)*
   - Az osztó még egy égető lapot félretesz, majd kiosztja az utolsó, ötödik közös lapot.
   - Minden játékos hét lapból próbálja a lehető legjobb ötlapos kombinációt kialakítani.
   - Az utolsó licitkör a második és a harmadik licitkörhöz hasonlóan zajlik.

6. *Showdown (lapok felfedése)*

   – Ha az utolsó licitkör után egynél több játékos marad, akkor a játékosok megmutatják a lapjaikat választásuk szerint.

   – A kasszát a legerősebb pókerkezet birtokló játékos nyeri el.

Pókerkezek

Az alábbi felsorolás a lehetséges pókerkezeket mutatja be, amelyeket erősségük szerint rendeztem el, a legerősebbtől a leggyengébbig. A lista tetején található kéz a pókerben elérhető legmagasabb értékű kombináció, és innen lefelé haladva egyre gyengébb kezek következnek. Az alábbi sorrendet megtekinthetjük a 4.1. ábrán is. [10, 9. oldal]

1. *Royal flös (royal flush)*

   – A legerősebb lapkombináció. Egyszínű 10-es, bubi, dáma, király, ász lapokból áll. Ha két ilyen találkozik, akkor döntetlen[1] van.

2. *Színsor (straight flush)*

   – Öt egyszínű sorba rendezhető lapból áll. Ha két ilyen találkozik, a legmagasabb lap dönt. Ha egyforma, akkor döntetlen van.

3. *Póker (four of a kind)*

   – Négy ugyanolyan számozású vagy jelű lapból és egy akármilyen másik lapból áll. Ha két ilyen találkozik, a magasabb póker nyer.

4. *Full (full house)*

   – Három ugyanolyan számozású vagy jelű lapból és két másik ugyanolyan számozású vagy jelű lapból áll. Ha két ilyen találkozik, a magasabb drill nyer. Ha egyforma, a magasabb pár nyer.

5. *Szín (flush)*

   – Öt ugyanolyan színű lapból áll. Ha két ilyen találkozik, a legmagasabb lap dönt. Ha egyforma, a második legmagasabb dönt, és így tovább...

6. *Sor (straight)*

   – Öt sorba rendezhető lapból áll. Ha két ilyen találkozik, a legmagasabb lap dönt. Ha egyforma, a színerősség dönt.

7. *Drill (three of a kind)*

[1] Osztozás történik a nyeremény között.

– Három ugyanolyan számozású vagy jelű lapból és két akármilyen másik lapból áll. Ha két ilyen találkozik, a magasabb drill nyer. Ha egyforma, a magasabb semleges lap, majd az alacsonyabb dönt.

8. *Két pár (two pairs)*

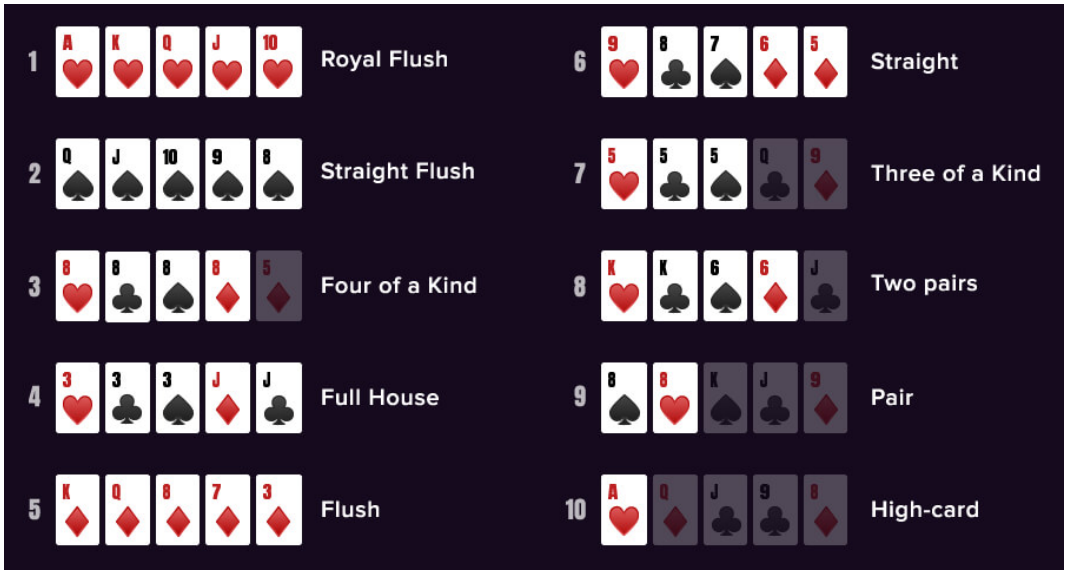
   – Kétszer két ugyanolyan számozású vagy jelű lapból áll. Ha két ilyen találkozik, a magasabb pár, majd az alacsonyabb, majd a semleges lap erőssége dönt.

9. *Egy pár (one pair)*

   – Két ugyanolyan számozású vagy jelű lapból és három akármilyen másik lapból áll. Ha két ilyen találkozik, a magasabb pár nyer. Ha egyforma, a semleges lapok döntenek.

10. *Magas lap (high card)*

    – Bármilyen lap, abból is a legmagasabb értékkel rendelkező.



4.1. ábra. Lehetséges pókerkezek

4.1.3. Használt kézkiértékelő algoritmus

A legelterjedtebb kézkiértékelő algoritmusok azon alapulnak, hogy a kezek értékeit egy előre kiszámított kéz értékeket tároló táblából keressük ki. Ez az egyik leggyorsabb módszer, viszont a módszer egy nagy hátránya a nagy méretű tábla tárolása, amely tárhely igényes. A játékomban egy bit matematikán alapuló algoritmust használtam, aminek alapjait a [9] blog adta. Ez a módszer lassabb, mint a táblás, viszont a tárhely igényes probléma itt megszűnik.

Algoritmus

Az alábbi lépéseket kell végrehajtanunk a kézkiértékeléshez:

1. Minden kártyáról el kell tárolnunk a rangját és a színét. Az inputunk 5 darab ilyen kártyából fog állni.

2. 2 különböző bitmező létrehozása a kártyák alapján

   – Az első mező a kártyák rangjának előfordulásáról tartalmaz 15 biten információt. Az egyes biteken azt fogjuk jelölni, hogy az adott rangból van-e az 5 adott kártya között (1 ha van, 0 ha nincs). A 4.1 táblázatban és a 4.2 táblázatban látható 1-1 példa az elő állított bitmezőről.

0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
A	K	Q	J	T	9	8	7	6	5	4	3	2		

4.1. táblázat. [9, 10, J, Q, K] kártyákhoz tartozó első bitmező

0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
A	K	Q	J	T	9	8	7	6	5	4	3	2		

4.2. táblázat. [9, 9, 9, K, K] kártyákhoz tartozó első bitmező

   – A második mező a rangok előfordulásának számáról tárol információt 60 biten. Minden ranghoz rendelünk 4 bitet, ahol annyit jelölünk 1-el, amennyi darabunk van az adott rangból. A 4.3 táblázatban és a 4.4 táblázatban látható 1-1 példa az elő állított bitmezőről.

0000	0001	0001	0001	0001	0001	0000	0000	0000	0000	0000	0000	0000	0000	0000
A	K	Q	J	T	9	8	7	6	5	4	3	2		

4.3. táblázat. [9, 10, J, Q, K] kártyákhoz tartozó második bitmező

0000	0011	0000	0000	0000	0111	0000	0000	0000	0000	0000	0000	0000	0000	0000
A	K	Q	J	T	9	8	7	6	5	4	3	2		

4.4. táblázat. [9, 9, 9, K, K] kártyákhoz tartozó második bitmező

3. kéz értékelés a második bitmező segítségével

   – A második bitmező 15-el való maradékos osztásának segítségével kitudunk értékelni 6 különböző pókerkezet. Először átszámoljuk a második bitmezőt 10-es számrendszerbe, majd utána végezzük el a 15-el való maradékos osztást. Pár példa erre:

- Póker (four of a kind): [9, 9, 9, 9, K] kártyákhoz tartozó második bináris mező 10-es számrendszerbe átszámolva: 4504630419521536. Ez az érték maradékosan osztva 15-el egyenlő lesz 1-el. Minden lehetséges ilyen pókerkézhez tartozó érték 1-et fog visszaadni erre az osztásra.
- Full (full house): Mindig 10-et fog visszaadni az osztás eredményeképpen.
- Drill (three of a kind): Mindig 9-et fog visszaadni az osztás eredményeképpen.
- Két pár (two pairs): Mindig 7-et fog visszaadni az osztás eredményeképpen.
- Egy pár (one pair): Mindig 6-ot fog visszaadni az osztás eredményeképpen.
- Magas lap (high card): Mindig 5-öt fog visszaadni az osztás eredményeképpen.

A varázslat az, hogy mind a hat pókerkéz a fenti módszerrel történő számolás mindig ugyanazt az eredményt adja, függetlenül attól, hogy mi a tényleges öt lap. Amit a maradékos osztás 15-el csinál, az egyenértékű az egyes 4 bites kártyák értékének összegzésével. Tehát a [2, 2, 2, 3, 3] kártyák esetében ez pl. 0111 + 0011 amit binárisan összeadva, majd átszámolva 10-et kapunk. És ez mind a 6 pókerkéznél így fog működni.

4. sorok ellenőrzése

   – A *sorok* ellenőrzéséhez az első bitmezőt fogjuk felhasználni. Az első bitmezőn bitenkénti ÉS műveletet alkalmazunk önmagának negatívjával. Ennek eredményeképpen megkapjuk az LSB-t, ami a bináris szám ábrázolásának legjobb oldali bitjére utal. Majd ezután, ha elosztjuk az eredeti bitmezőt az LSB-vel, akkor megtudjuk állapítani, hogy a pókerkéz *sor*-t tartalmaz-e. Ha az osztás eredménye pontosan egyenlő 11111-el, azaz decimálisan 31-el, akkor *sorunk* van, ellenkező esetben pedig nincs *sorunk*. Ez a módszer minden *sorra* működni fog, kivéve az *ász-alacsony sorra*, amit egy külön ellenőrzéssel megtudunk állapítani. Ha az első bitmezőnk pontosan egyenlő az alábbi bitmezővel: 100000000111100, akkor *ász-alacsony sorunk* van.

5. flush ellenőrzése

   – A *flush* ellenőrzésénél nincs szükségünk bit matematikára, szimplán csak azt kell megnéznünk, hogy a kapott 5 input kártya közül, mindnek megegyezik a színe. Ha mindegyik ugyanolyan színű, akkor a pókerkéz *flush*, ellenkező esetben pedig nem. Ha *flush* kezünk van és az előző ellenőrzés

szerint *sorunk* is van, akkor *straight flush* kezünk van. Az első bitmező segítségével pedig tudjuk ellenőrzini a *royal flush*-t is. Hasonló módon mint az *ász-alacsony sornál* itt is az első bitmezőt kell ellenőriznünk. Ha az első bitmezőnk pontosan egyenlő az alábbi bitmezővel: 111110000000000, akkor *royal flush* pókerkezünk van.

6. döntetlenek eldöntése

   – Abban az esetben, ha két pókerkéz azonos típusú, a döntetleneket úgy döntjük el, hogy az 5 kártyát először az előfordulás sorrendje, majd a rangja szerint rendezzük, és bitek eltolásával összehasonlítható pontszámot hozunk létre az alábbiak szerint:

     - Az első kártya értékét kettes számrendszerben eltoljuk balra 16-al
     - A második kártya értékét kettes számrendszerben eltoljuk balra 12-vel
     - A harmadik kártya értékét kettes számrendszerben eltoljuk balra 8-cel
     - A negyedik kártya értékét kettes számrendszerben eltoljuk balra 4-el
     - Az ötödik kártya értékét nem toljuk el

     Ezután kombináljuk mind az öt létrejött számot bitenkénti VAGY művelettel és ennek az eredménye lesz a döntetleneket eldöntő pontszáma a pókerkéznek, ahol minél magasabb ez a pontszám, annál jobb kezünk van.

A teljes algoritmus tehát röviden így működik:

1. Eltároljuk az 5 kártyánkról a szükséges szín és rang adatokat.
2. A második bitmezőn maradékos osztást használva ellenőrizzük, hogy póker, full, drill, két pár, egy pár vagy magas lap pókerkezünk van-e.
3. Az első bitmező LSB-vel történő osztásával ellenőrizzük a sorokat, majd elvégezzük az ász-alacsony sorok extra ellenőrzését.
4. Ellenőrizzük, hogy van-e öt azonos színű lap a flush-höz, beleértve a royal flush extra ellenőrzését is.
5. A döntetleneket felbontjuk a pókerkezekhez kiszámolt bizonyos pontszámok összehasonlításával.

4.2. Többjátékos kapcsolat megvalósítása

4.2.1. Kapcsolat kezelése a PC oldalon

4.2.2. Kapcsolat kezelése a Mobile oldalon

4.3. Játékmenet megvalósítása

4.3.1. Modul1

4.3.2. Modul2

5. fejezet

Tesztelés

5.1. SharedDLL tesztelése

5.2. PC játék tesztelése

5.3. Mobile játék tesztelése

5.4. Általános tesztelés

Összegzés

Irodalomjegyzék

[1] Wikipedia: *Game engine*, Wikipedia, az online enciklopédia, elérhető: `https://en.wikipedia.org/wiki/Game_engine` [Letöltve: 2025-02-28]

[2] Wikipedia: *Unreal Engine*, Wikipedia, az online enciklopédia, elérhető: `https://en.wikipedia.org/wiki/Unreal_Engine` [Letöltve: 2025-02-28]

[3] GitHub: *Godot*: `https://github.com/godotengine/godot)` [Letöltve: 2025-02-28]

[4] Wikipedia: *Unity (game engine)*, Wikipedia, az online enciklopédia, elérhető: `https://en.wikipedia.org/wiki/Unity_(game_engine)` [Letöltve: 2025-02-28]

[5] Jamal Aladdin: *The Evolution of Multiplayer Gaming: A Journey Through Time*, Jamal Aladdin blogja, elérhető: `https://medium.com/@Jamal_Aladdin/the-evolution-of-multiplayer-gaming-a-journey-through-time-e34ef59294c2` [Letöltve: 2025-02-28]

[6] Lem Apperson: *Beginning Game Development: Client-Server Architecture*, Lem Apperson blogja, elérhető: `https://medium.com/@lemapp09/beginning-game-development-client-server-architecture-1b7676d80dea` [Letöltve: 2025-02-28]

[7] Wikipedia: *Póker*, Wikipedia, az online enciklopédia, elérhető: `https://hu.wikipedia.org/wiki/P%C3%B3ker` [Letöltve: 2024-11-12]

[8] Wikipedia: *Texas Hold'Em*, Wikipedia, az online enciklopédia, elérhető: `https://hu.wikipedia.org/wiki/Texas_Hold%E2%80%99Em` [Letöltve: 2024-11-13]

[9] Jonathan Hsiao: *Evaluating Poker Hands with Bit Math*, Jonathan Hsiao blogja, elérhető: `https://jonathanhsiao.com/blog/evaluating-poker-hands-with-bit-math` [Letöltve: 2025-02-25]

[10] Szurdi András: *Pókerkönyv. Kezdőknek és haladóknak*, Ciceró, Budapest, 1995.

[11] Varga Ervin: *Póker alapkönyv*, Vagabund, Kecskemét, 2008.