Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozáselmélet és Szoftvertechnológiai Tanszék

Kétszemélyes társasjáték applikáció

Témavezető:
Tichler Krisztián
egyetemi adjunktus

Szerző:
Nagy Gergely Máté
programtervező informatikus BSc

Budapest, 2025

Tartalomjegyzék

1. fejezet

Bevezetés

A digitális technológia fejlődésével és az internet széles körű elterjedésével a klasszikus társasjátékok is kezdenek új formát ölteni. A hagyományos, asztali játékok digitális adaptációi nem csupán megőrzik az eredeti játékok szellemiségét, hanem új lehetőségekkel is gazdagítják azokat. A digitalizáció lehetővé teszi, hogy kedvenc játékainkat térben és időben egymástól távol lévő játékosok is élvezhessék, valamint hogy a mesterséges intelligencia segítségével egyedül is gyakorolhassunk és fejlődhessünk.

A kalaha, más néven mancala, az egyik legősibb táblajáték, amelynek eredete több ezer évre nyúlik vissza. Ez a stratégiai játék Afrikából származik, és az évszázadok során számos változata alakult ki. A játék egyszerű szabályai mögött komplex stratégiai döntések húzódnak meg, ami ideálissá teszi mind kezdő, mind haladó játékosok számára. A kalaha különlegessége, hogy míg szabályai könnyen elsajátíthatók, a győztes stratégiák kidolgozása és alkalmazása jelentős gyakorlást és gondolkodást igényel.

1.1. Témaválasztás

A témaválasztást több tényező is motiválta. Elsősorban a klasszikus társasjátékok digitalizálásának növekvő igénye, hiszen a modern életvitel mellett egyre kevesebb lehetőség nyílik a személyes találkozásra és játékra. Egy online platform lehetővé teszi, hogy a játékosok akkor is játszhassanak egymással, amikor fizikailag nem tudnak egy asztalhoz ülni. Másrészt a kalaha játék matematikai és stratégiai mélysége kivá-

ló alapot nyújt egy mesterséges intelligencia alapú ellenféljátékos megvalósításához, ami új kihívást jelenthet a fejlesztésben.

A Unity játékmotor választása mellett szólt annak széleskörű támogatottsága, fejlett hálózati képességei és a platformfüggetlen fejlesztés lehetősége. A C# programozási nyelv használata pedig lehetővé teszi a tiszta, jól strukturált és könnyen karbantartható kód írását.

1.2. Célkitűzések

A szakdolgozat célja egy olyan modern, felhasználóbarát kalaha játék implementálása, amely:

- Lehetővé teszi két játékos számára az online játékot
- Tartalmaz különböző nehézségi szintű mesterséges intelligencia ellenfeleket
- Biztosítja a játék szabályainak pontos betartását
- Kellemes felhasználói élményt nyújt intuitív kezelőfelülettel
- Vizuálisan vonzó és reszponzív grafikus felülettel rendelkezik

2. fejezet

Felhasználói dokumentáció

2.1. A szoftver általános bemutatása

A kalaha játék applikáció egy klasszikus táblajáték modern, digitális adaptációja. A program három különböző játékmódot kínál: helyi többjátékos módot, online többjátékos módot, valamint AI elleni játék lehetőségét. Az alkalmazás célja, hogy a játékosok számára könnyen hozzáférhető és élvezetes módon tegye elérhetővé ezt az ősi stratégiai játékot.

2.1.1. Rendszerkövetelmények

A program futtatásához az alábbi minimális rendszerkövetelmények szükségesek:

- Operációs rendszer: Windows 10 vagy újabb
- Processzor: Intel Core i3 vagy jobb
- Memória: 4 GB RAM
- Grafikus kártya: DirectX 11 kompatibilis
- Internet kapcsolat: Online játékhoz stabil internetkapcsolat szükséges
- Tárhely: 500 MB szabad terület

2.1.2. A játék szabályai

A kalaha egy két személyes stratégiai játék, amelyben a játékosok célja minél több kő összegyűjtése. A játék szabályai:

1. A játéktábla 2x6 kisebb gödörből és 2 nagyobb gyűjtőgödörből áll
2. Minden kis gödörben kezdetben 4 kő található
3. A játékosok felváltva lépnek
4. Egy lépés során a játékos kiválaszt egy gödröt a saját oldaláról, és az összes követ egyesével szétosztja az óramutató járásával megegyező irányban
5. Ha az utolsó kő a saját gyűjtőgödörébe kerül, a játékos újra léphet
6. Ha az utolsó kő egy üres gödörbe kerül a saját oldalán, a játékos megszerzi ezt a követ és az átellenes gödör köveit is
7. A játék véget ér, ha az egyik játékos oldalán minden gödör kiürül, ekkor a másik játékos összegyűjti a maradék köveit. A győztes az aki több követ tudott összegyűjteni.

2.2. A program használata

2.2.1. Főmenü

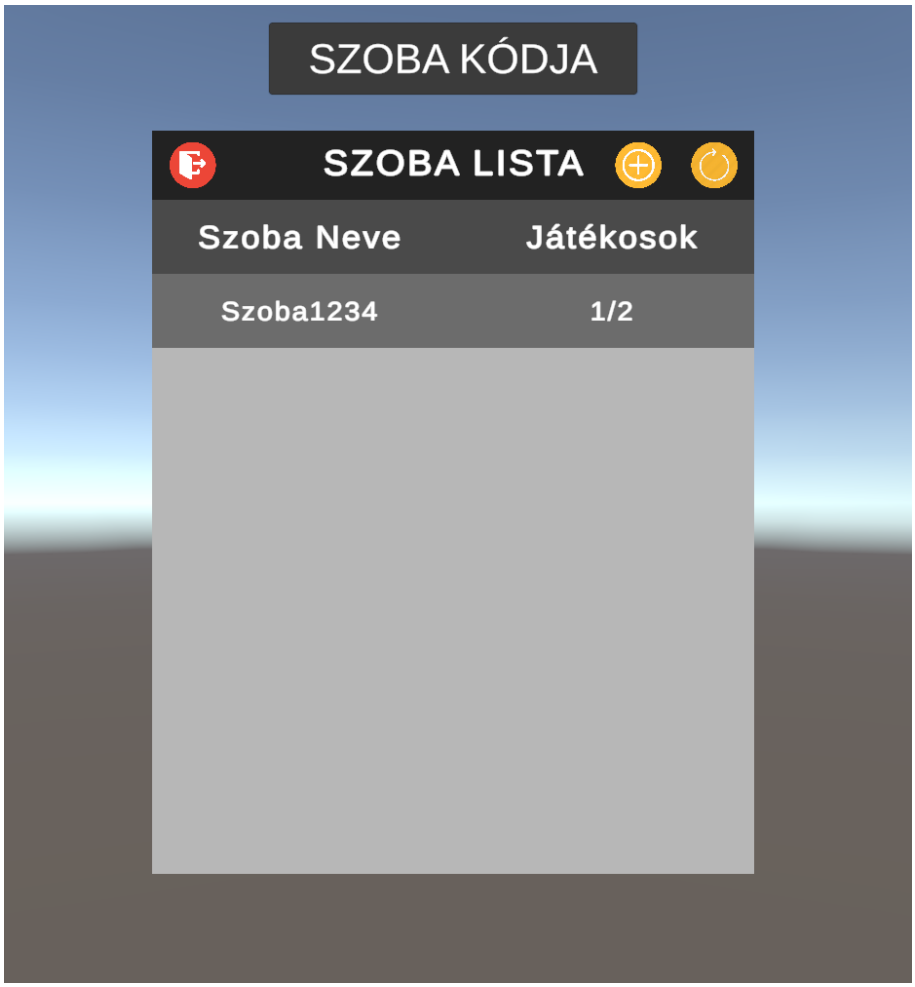
A program indításakor a főmenü jelenik meg, ahol három játékmód közül választhatunk:

- **Többjátékos mód:** Online játék más játékosok ellen
- **Lokális játék:** Helyi játék két játékos részére egy számítógépen
- **AI elleni játék:** Játék a mesterséges intelligencia ellen

2.2.2. Online játék

Az online játékmód választása esetén a szoba kezelő felület jelenik meg. Itt lehetőség van:

- Meglévő szobák böngészésére és csatlakozásra
- Új szoba létrehozására
- Csatlakozásra szobakód alapján



2.1. ábra. A szobák listázása és létrehozása

A szobában a játékosok láthatják egymás nevét és státuszát (host/kliens). A host játékos indíthatja el a játékot, amikor mindkét játékos csatlakozott.

2.2.3. Játékmenet

A játék során a játéktábla 3D nézetben jelenik meg. A játékosok felváltva választhatnak gödröt a saját oldalukon kattintással. A program automatikusan végrehajtja a lépést, animálva a kövek mozgását.

A játék állapotát jelző információk folyamatosan láthatók:

- Az soron következő játékos jelzése
- A gödrökben lévő kövek száma

2.2. ábra. Részletek a játékszobában

2.3. Hibaelhárítás

2.3.1. Kapcsolódási problémák

Ha problémák merülnek fel az online játék során:

- Ellenőrizze az internetkapcsolatot
- Próbáljon újracsatlakozni a szobához
- Ha a probléma továbbra is fennáll, indítsa újra az alkalmazást

2.3.2. Ismert hibák

- Ritkán előfordulhat, hogy a kövek mozgatásánál egy kő kiesik/két tároló közé esik.

2.3. ábra. A játéktábla játék kezdetén

- Egyes esetekben a játékos neve nem frissül ha a hoszt elindítja a játékot míg a név változtatás nem történt meg.

Ezen hibák általában nem befolyásolják a játék működését, és a következő körre vagy lépés után automatikusan javulnak.

3. fejezet

Fejlesztői dokumentáció

3.1. Tervezési fázis

3.1.1. Követelmények elemzése

A tervezési fázis első lépéseként azonosítottam a kulcsfontosságú követelményeket:

- A játéknak támogatnia kell három különböző játékmódot: helyi többjátékos, online többjátékos és AI elleni játék
- A játékmenetnek gördülékenynek kell lennie, vizuálisan vonzó animációkkal
- Az AI-nak skálázható nehézségi szintekkel kell rendelkeznie
- Az online játéknak megbízhatóan kell működnie különböző hálózati körülmények között
- A felhasználói felületnek intuitívnak és reszponzívnak kell lennie

3.1.2. Architektúrális tervezés

A követelmények alapján egy MVC (Model-View-Controller) architektúra mellett döntöttem, amely jól illeszkedik a Unity játékmotor működéséhez és a projekt igényeihez:

- **Model:** A játék üzleti logikája és adatmodellje (BoardManager)
  - Tábla állapotának kezelése

  - Játékszabályok implementálása
  - Játékállapot validáció

  Copy

- **View:** Unity scene-ek, prefabok és UI elemek
  - 3D játéktábla megjelenítése
  - Felhasználói felület elemei
  - Vizuális visszajelzések és animációk
- **Controller:** Játékosok és játékmenet vezérlése
  - Különböző játékostípusok (HumanPlayer, AIPlayer, NetworkPlayer)
  - Bemenet kezelése
  - Játékállapot-változások koordinálása

Az architektúra erősen eseményvezérelt, ami lehetővé teszi a komponensek laza csatolását és a játék állapotváltozásainak hatékony kezelését. Például amikor egy játékos lép, az eseményrendszer biztosítja, hogy minden érintett komponens (animációk, UI elemek, hálózati szinkronizáció) megfelelően reagáljon.

3.1.3. Technológiai stack kiválasztása

A technológiák kiválasztásánál több szempontot is figyelembe vettem:

- **Unity játékmotor:** A döntés mellett szólt:
  - Kiforrott fizikai motor, ami elengedhetetlen a kövek mozgásának realisztikus szimulációjához
  - Erős közösségi támogatás és dokumentáció
  - Beépített networking megoldások
  - Platformfüggetlen fejlesztés lehetősége
  - Hatékony asset management

  Copy

- **Unity Netcode for GameObjects:** Az online többjátékos mód implementálásához:
  - Host-Client modell támogatása
  - Automatikus állapotszinkronizáció
  - Beépített biztonsági megoldások
- **Unity Relay:** A játékosok közötti kapcsolat biztosításához:
  - NAT átjárás problémák kezelése
  - Biztonságos kapcsolat
  - Alacsony látencia
- **Python az AI implementációhoz:** A döntés okai:
  - Gyors prototípus készítés lehetősége
  - Hatékony algoritmus implementáció
  - Könnyű debuggolás és tesztelés
  - Gazdag matematikai és algoritmus könyvtárak

3.2. Rendszerarchitektúra

3.2.1. Főbb komponensek és felelősségeik

GameManager (Controller)

A játék központi vezérlő komponense, amely:

- Kezeli a különböző játékállapotokat
- Koordinálja a játékmódok közötti váltást
- Menedzseli a játékosok létrehozását és inicializálását
- Eseményeket közvetít a komponensek között

```
public class GameManager : NetworkBehaviour
{
  public void StartAIGame(int strength)
  {
    currentGameMode = GameMode.VsAI;
    currentGameState = GameState.Running;
    Copy    player1Object = new GameObject();
    HumanPlayer player1 = player1Object.AddComponent<HumanPlayer
       >();
    player1.Id = PlayerIdGenerator.GetNextId();

    player2Object = new GameObject();
    AIPlayer aiPlayer = player2Object.AddComponent<AIPlayer>();
    aiPlayer.strength = strength;
    aiPlayer.Id = PlayerIdGenerator.GetNextId();

    this.player1 = player1;
    this.player2 = aiPlayer;

    SceneManager.sceneLoaded += AIGameLoaded;
    SceneManager.LoadScene("GameScene");
  }
}
```

BoardManager (Model)

A játék logikai motorja, amely:

- Tárolja és kezeli a játéktábla állapotát
- Validálja a játékosok lépéseit
- Ellenőrzi a játékszabályok betartását
- Eseményeket generál a játékállapot változásairól

StoneManager (View)

A játék vizuális megjelenítéséért felelős komponens:

- Kezeli a kövek fizikai reprezentációját

- Animálja a kövek mozgását
- Szinkronizálja a vizuális állapotot a játéklogikával
- Események alapján frissíti a megjelenítést

3.3. AI implementáció

Az AI modul a minimax algoritmus egy optimalizált implementációját használja alpha-beta vágással. Az implementáció Python nyelven történt, ami lehetővé tette a gyors prototípuskészítést és hatékony tesztelést.

3.3.1. Az algoritmus működése

A minimax algoritmus rekurzívan építi fel a lehetséges lépések fáját:

- A fa minden csomópontja egy játékállást reprezentál
- A levelek értékelése a játékos szempontjából történik
- Az alpha-beta vágás jelentősen csökkenti az átvizsgálandó csomópontok számát
- A keresési mélység dinamikusan változik az AI szintjétől függően

A döntéshozatal folyamata:

1. Az aktuális játékállás kiértékelése
2. Lehetséges lépések generálása
3. Minden lépésre rekurzív kiértékelés
4. A legjobb értékelésű lépés kiválasztása

3.3.2. Nehézségi szintek implementációja

A különböző AI szintek megvalósítása a keresési mélység és a kiértékelési függvény módosításával történik:

```
def calculate_kalaha_move(board_state, is_player_1, ai_level):

depth = min(ai_level + 1, 6)  # Base depth from AI level

eval_score, selected_move = minimax(
  board_state,
  depth,
  float('-inf'),
  float('inf'),
  is_player_1,
  True
)

return selected_move
```

3.4. Tesztelés és teljesítménymérés

3.4.1. AI teljesítmény elemzése

Az AI különböző nehézségi szintjeinek tesztelése egy dedikált tesztkörnyezetben történt. A tesztek során az AI random stratégiát alkalmazó ellenfelek ellen játszott. Minden nehézségi szinten 100 játszmát futattunk le, az eredmények:

- 1-es szint: 59
- 2-es szint: 73
- 3-as szint: 77
- 4-es szint: 82
- 5-ös szint: 90

Az eredmények alapján látható, hogy:

- Az AI már az 1-es szinten is képes a random stratégiánál jobb teljesítményre
- A nehézségi szintek között jelentős teljesítménykülönbség van
- Az 5-ös szintű AI már közel optimális játékot játszik

A teljesítménynövekedés magyarázata:

- Magasabb nehézségi szinteken mélyebb keresés
- Jobb pozícióértékelés
- Hatékonyabb alpha-beta vágás

3.4.2. Hálózati teljesítmény

A hálózati komponensek tesztelése során vizsgáltuk:

- A lobby rendszer stabilitását
- A játékszinkronizáció pontosságát
- A különböző hálózati körülmények hatását

3.5. További fejlesztési lehetőségek

- Az AI továbbfejlesztése gépi tanulási módszerekkel
- Játékmenet rögzítése és visszajátszási lehetőség
- Részletes statisztikai rendszer és ranglisták
- Több játékos egyidejű megfigyelési lehetősége
- Mobil platformra való portolás

4. fejezet

Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

Köszönetnyilvánítás

Amennyiben a szakdolgozati / diplomamunka projekted pénzügyi támogatást kapott egy projektből vagy az egyetemtől, jellemzően kötelező feltüntetni a dolgozatban is. A dolgozat elkészítéséhez segítséget nyújtó oktatók, hallgatótársak, kollégák felé is nyilvánítható külön köszönet.

A. függelék

Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus,

sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

Ábrák jegyzéke

Táblázatok jegyzéke

Algoritmusjegyzék

Forráskódjegyzék