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 indoklása

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
- Stabil és megbízható hálózati kapcsolatot biztosít
- Vizuálisan vonzó és reszponzív grafikus felülettel rendelkezik

1.3. A dolgozat felépítése

A dolgozat a következő főbb fejezetekre tagolódik. A felhasználói dokumentáció részletesen bemutatja a program telepítését és használatát, kitérve minden játékmódra és funkcióra. A fejlesztői dokumentáció ismerteti a program architektúráját, a főbb tervezési döntéseket és azok indoklását, valamint részletesen tárgyalja az online játék és a mesterséges intelligencia megvalósítását. A dokumentáció kitér a tesztelési folyamatokra és eredményekre is. Végül az összefoglalásban értékeljük az elért eredményeket és felvázoljuk a lehetséges továbbfejlesztési irányokat.

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 ellentétes irányban
5. Ha az utolsó kő a saját gyűjtőgödörbe kerül, a játékos újra léphet
6. Ha az utolsó kő egy üres gödörbe kerül a saját oldalon, a játékos megszerzi ezt a követ és az átellenes gödör köveit is
7. A játék akkor ér véget, ha az egyik játékos oldalán minden gödör kiürül

2.2. A program használata

2.2.1. Főmenü

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

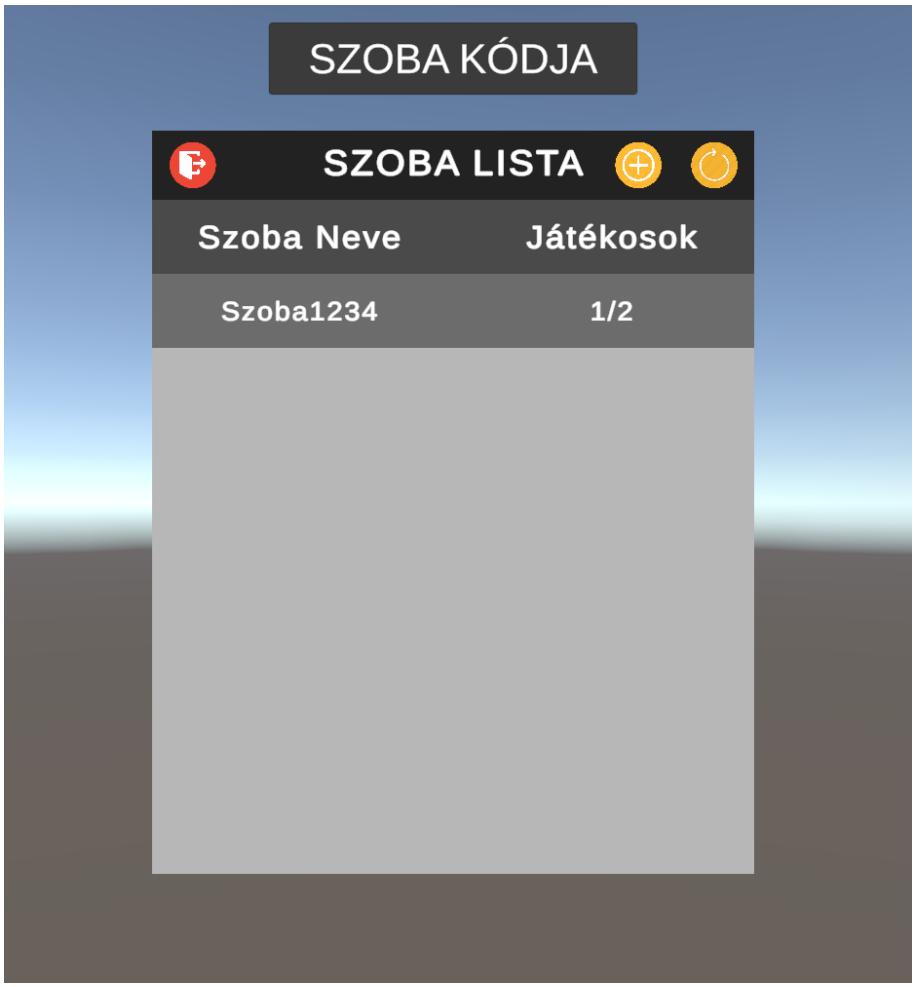
- **Többjátékos mód:** Online játék más játékosok ellen
- **Lokális játék:** Helyi játék két játékos részére egy számítógépen
- **AI elleni játék:** Játék a mesterséges intelligencia ellen

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

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



2.1. ábra. A szobák listázása és létrehozása

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 aktuális játékos jelzése
- A gödrökben lévő kövek száma
- A gyűjtőgödrökben összegyűjtött 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 animációja nem megfelelően jelenik meg

2.3. ábra. A játéktábla játék közben

- Egyes esetekben a játékos státusza nem frissül azonnal a szobában

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

3. fejezet

Fejlesztői dokumentáció

3.1. Architektúra és tervezési döntések

3.1.1. Technológiai stack

A játék fejlesztéséhez az alábbi technológiákat választottam:

- **Unity játékmotor:** A Unity választását több tényező indokolta:
  - Beépített fizikai motor a kövek mozgásának szimulálásához
  - Fejlett hálózati képességek az online játékhoz
  - Hatékony 3D renderelés
  - Platformfüggetlen fejlesztés lehetősége
- **C# programozási nyelv:** A Unity natív nyelve, amely modern és tiszta kód írását teszi lehetővé
- **Unity Netcode for GameObjects:** Az online többjátékos funkciók megvalósításához
- **Unity Relay:** A játékosok közötti közvetlen kapcsolat biztosításához

3.1.2. Architektúrális döntések

A program architektúrája a Manager alapú tervezési mintát követi, ahol minden főbb funkcionalitásért egy dedikált Manager osztály felel. Ez az architektúra több előnnyel is jár:

- Tiszta felelősségi körök
- Könnyű bővíthetőség
- Jól tesztelhető komponensek
- Egyszerű függőség kezelés

3.1.3. Főbb komponensek

GameManager

A GameManager a játék központi vezérlő komponense, amely:

- Kezeli a játék állapotait (lobby, játék, szünet, játék vége)
- Koordinálja a többi manager működését
- Biztosítja a játékmódok közötti váltást
- Kezeli a játékosok csatlakozását és leválását

```
1  public class GameManager : NetworkBehaviour
2  {
3    public static GameManager Instance;
4    private GameMode currentGameMode;
5    public GameState currentGameState { get; set; }
6
7    private IPlayerController player1;
8    private IPlayerController player2;
9
10   public void StartOnlineGame(string sceneName, LoadSceneMode
       loadSceneMode,
11   List<ulong> clientsCompleted, List<ulong> clientsTimedOut)
12   {
13     currentGameMode = GameMode.Online;
14     currentGameState = GameState.Running;
15     bool shouldFlip = UnityEngine.Random.value > 0.5f;
16     bool isPlayer1Starting = UnityEngine.Random.value > 0.5f;
17
18     if (NetworkManager.Singleton.IsHost)
19     {
20       FlipDecisionServerRpc(shouldFlip, isPlayer1Starting);
```

```
21      }
22    }
23  }
```

BoardManager

A BoardManager felelős a játéktábla állapotának kezeléséért:

- Tárolja a játéktábla aktuális állapotát
- Validálja és végrehajtja a lépéseket
- Ellenőrzi a játék szabályait
- Nyilvántartja a pontszámokat

```
1   public class BoardManager
2   {
3     private int[] table;
4     private IPlayerController currentPlayer;
5     private bool isGameOver;
6
7     public void MakeMove(int index)
8     {
9       bool isPlayer1Turn = currentPlayer == player1;
10      if (!IsMoveValid(index))
11      {
12        currentPlayer.RequestMove();
13        return;
14      }
15
16      int rocksToPlace = table[index];
17      table[index] = 0;
18      int currentIndex = index;
19
20      for (int i = 0; i < rocksToPlace; i++)
21      {
22        currentIndex = GetNextValidIndex(currentIndex,
              isPlayer1Turn);
23        table[currentIndex]++;
24      }
```

```
25
26      HandleSpecialRules(currentIndex, isPlayer1Turn);
27
28      if (!ShouldPlayerGetExtraTurn(currentIndex, isPlayer1Turn))
29      {
30        SwitchCurrentPlayer();
31      }
32    }
33  }
```

StoneManager

A StoneManager a játék vizuális megjelenítéséért felelős:

- Kezeli a kövek fizikai reprezentációját
- Animálja a kövek mozgását
- Szinkronizálja a vizuális megjelenítést a játék logikai állapotával

```
1   public class StoneManager : MonoBehaviour
2   {
3     [SerializeField] private GameObject[] rockPrefabs;
4     [SerializeField] private GameObject[] rockSpawners;
5     private List<List<GameObject>> rocks;
6     private bool isStoneMoving = false;
7
8     private IEnumerator MoveStonesCoroutine(int index, int
          numberOfStones,
9     bool isPlayer1Turn, bool isGameOver, int player1Score, int
          player2Score)
10    {
11      isStoneMoving = true;
12      OnStoneMoveStart?.Invoke();
13
14      List<GameObject> stonesToMove = CollectStonesFromPit(index,
            numberOfStones);
15      yield return StartCoroutine(LiftStonesFromPit(stonesToMove,
            index));
16
17      yield return StartCoroutine(DistributeStones(stonesToMove,
            index, isPlayer1Turn));
```

```
18
19      yield return StartCoroutine(HandleCapture(lastIndex,
            isPlayer1Turn));
20
21      isStoneMoving = false;
22      OnStoneMoveComplete?.Invoke();
23    }
24  }
```

3.2. Online játék megvalósítása

3.2.1. Hálózati architektúra

Az online játék peer-to-peer alapú, Unity Relay szolgáltatás használatával:

- A Relay szerver biztosítja a kezdeti kapcsolódást
- A játékosok közötti kommunikáció közvetlen
- A host játékos validálja a lépéseket

3.2.2. Lobby rendszer

A lobby rendszer az Unity Lobby szolgáltatására épül:

- Szobák létrehozása és kezelése
- Játékosok csatlakozásának kezelése
- Szobák listázása és szűrése

```
1   public class OnlineLobby : MonoBehaviour
2   {
3     public static OnlineLobby Instance;
4     private Lobby hostLobby;
5     private Lobby joinedLobby;
6
7     public async void CreateLobby(string lobbyName, bool isPrivate)
8     {
9       try
10      {
```

```
11          int maxPlayers = 2;
12          string joinCode = await RelayConnection.Instance.
                CreateRelay();
13
14          CreateLobbyOptions options = new CreateLobbyOptions
15          {
16            IsPrivate = isPrivate,
17            Player = GetPlayer(),
18            Data = new Dictionary<string, DataObject>
19            {
20              { "Start", new DataObject(DataObject.VisibilityOptions.
                    Member, joinCode) }
21            }
22          };
23
24          Lobby lobby = await LobbyService.Instance.CreateLobbyAsync(
                lobbyName, maxPlayers, options);
25          hostLobby = lobby;
26          joinedLobby = hostLobby;
27        }
28        catch (LobbyServiceException e)
29        {
30          Debug.Log($"Failed to create lobby - {e}");
31        }
32      }
33    }
```

3.3. Mesterséges intelligencia

Az AI játékos implementációja a következő komponensekre épül:

3.3.1. AI stratégia

Az AI játékos több szintű döntési mechanizmust használ:

- Alapvető szabályok követése
- Lépések előnyösségének értékelése
- Több lépéses kombinációk keresése

```
public class AIPlayer : MonoBehaviour, IPlayerController
{
  private int GetAIMove(int[] availableMoves)
  {
    if (availableMoves.Length == 0)
    {
      return -1;
    }

    System.Random r = new System.Random();
    int randomIndex = r.Next(0, availableMoves.Length);
    return availableMoves[randomIndex];
  }

  private IEnumerator SimulateThinking()
  {
    int[] availableMoves = GetAvailableMoves();
    int moveCount = availableMoves.Length;

    float thinkTime;
    if (moveCount >= 5) thinkTime = 2.5f;
    else if (moveCount >= 3) thinkTime = 1.7f;
    else thinkTime = 1f;

    yield return new WaitForSeconds(thinkTime);

    int aiMove = GetAIMove(availableMoves);
    IPlayerController.RaiseMoveSelected(aiMove);
  }
}
```

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