

**Pannon Egyetem**  
Műszaki Informatikai Kar  
Informatikai Rendszerek és Alkalmazásai Tanszék  
Mérnökinformatikus BSc

# SZAKDOLGOZAT

**3D memóriajáték tervezése mesterséges  
intelligenciával**

**Csesznák Tamás Levente**

Témavezető: Szabó Patrícia

2024

# Tartalomjegyzék

<b>Jelölésjegyzék</b>	<b>2</b>
<b>1. Bevezetés</b>	<b>3</b>
1.1. Projekt célja . . . . .	3
1.2. Projet bemutatása . . . . .	3
1.2.1. Kutatómunka . . . . .	4
1.2.2. Játék megtervezése . . . . .	5
1.2.3. Játék fejlesztése . . . . .	5
1.2.4. Adatgyűjtés és VR támogatás . . . . .	5
1.2.5. AI betanítása . . . . .	5
1.2.6. AI játszatása . . . . .	5
<b>2. Irodalomkutatás</b>	<b>6</b>
<b>3. Felhasznált technológiák</b>	<b>7</b>
3.1. Godot Engine . . . . .	7
3.2. OpenXR . . . . .	8
3.3. GDscript . . . . .	8
<b>4. A játék működése</b>	<b>10</b>
4.1. Játék ismertetése . . . . .	10
4.2. Struktúralis felépítés . . . . .	12
4.2.1. Menü . . . . .	12
4.2.2. Basic Scene . . . . .	14
4.2.3. Deck . . . . .	15
4.2.4. Card . . . . .	16
<b>Irodalomjegyzék</b>	<b>18</b>

## **Jelölésjegyzék**

*2D* Kettő dimenziós

*3D* Hárrom dimenziós

*AI* Artificial Intelligence (Mesterséges Intelligencia)

*IDE* Integrált fejlesztői környezet

*VR* Virtual Reality

## 1. fejezet

### Bevezetés

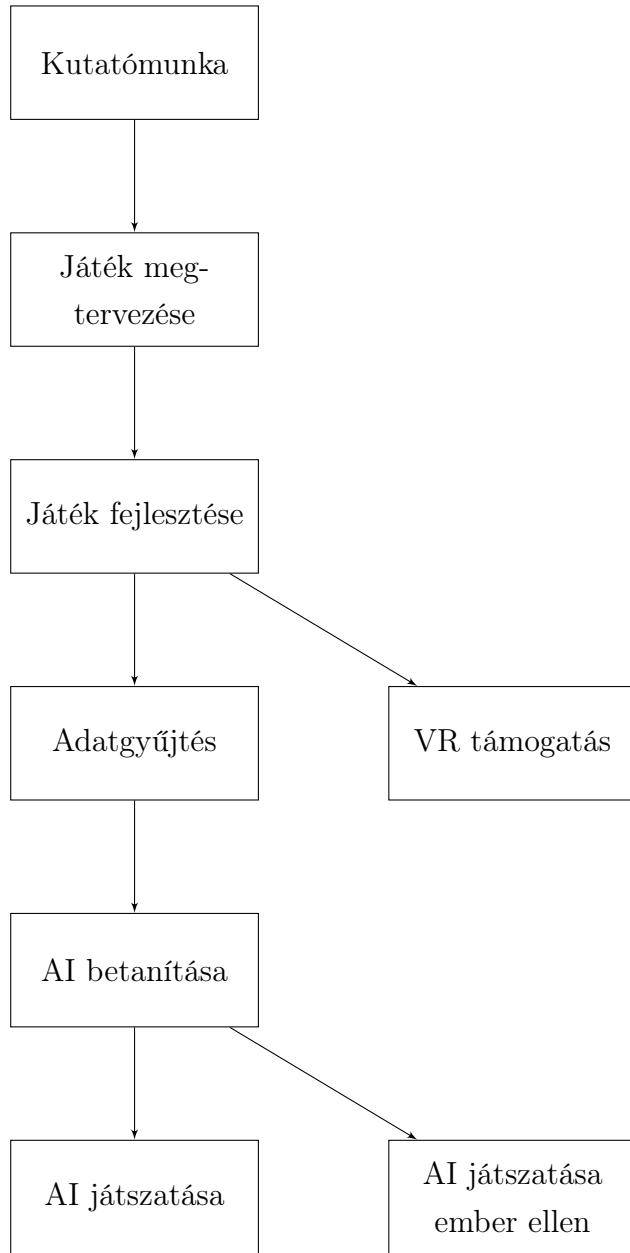
#### 1.1. Projekt célja

A jelenlegi kor társadalmi és technológiai kihívásai közepette egyre fontosabbá válik az emberiség számára az olyan innovatív megoldások keresése, amelyek segíthetnek fejleszteni és támogatni az emberek minden napjai életét. Az Artificial Intelligence (AI), vagyis a Mesterséges Intelligencia, ebben az összefüggésben különösen figyelemre méltó tényezővé vált. Bár sokan aggódnak amiatt, hogy az AI alkalmazása az emberi társadalom hanyatlásához vezethet, én úgy vélem, hogy a megfelelő módon felhasználva az AI lehetőségei elősegíthetik a társadalmi fejlődést és előnyöket hozhatnak az emberi élet számos területén.

Szakdolgozatom központi célja az, hogy az AI alkalmazásával támogassam embertársaim rövidtávú memóriájának fejlesztését. Ehhez egy saját fejlesztésű virtuális valóság alapú, három dimenziós memória játékot tervezek létrehozni, amely segítségével interaktív és hatékony módon lehet fejleszteni a játékosok kognitív képességeit. "Thus, VR technology is ideally suited to aid self-improvement, which is about ending negative behaviors, and promote personal-development, which is about learning, growing, expanding awareness, and developing one's full potential." [1]

#### 1.2. Projekt bemutatása

A projektem több feladatból állt, melyet egy folyamatdiagram (1.1 ábra) szemléltet. A folyamat egyes lépései a következő részben fejteném ki részletesebben.



1.1. ábra. A projekt folyamatábrája

### 1.2.1. Kutatómunka

A kutatómunka során elsősorban azt vizsgáltam, hogy melyik tanító algoritmussal érhetem el a kívánt eredményt. Különböző irodalmakat tanulmányoztam, valamint áttekintettem mások munkáit a témaban. A kutatómunka végeztével összegeztem a talált eredményeket.

### **1.2.2. Játék megtervezése**

A kutatómunka után el kellett döntenem, hogy milyen játékot fejlesztek, amely elég bonyolult ahhoz, hogy kihívást jelentsen a játékosok számára, ugyanakkor elég egyszerű ahhoz, hogy az AI betanítása belátható időn belül megtörténjen. Ezen a ponton meg kellett azt is határoznom, hogy milyen technológiát alkalmazok, valamint hogy mely területekre összpontosítok a fejlesztés folyamán. A Játék elkészítéséhez a Godot engine-n-t választottam.

### **1.2.3. Játék fejlesztése**

A megfelelő tervezés után lefejlesztettem a választott fejlesztői környezetben a játékot. A fejlesztés során két fontos szempontot tartottam szem előtt: a játékot lehetővé kell tenni virtuális valóságban és asztali számítógépen egyaránt, valamint biztosítanom kell, hogy az AI képes legyen kezelni a játékot csupán a játék metainformációinak ismeretében.

### **1.2.4. Adatgyűjtés és VR támogatás**

Miután elkészült a játék, több különböző korosztállyal játszattam azt annak érdekében, hogy elegendő adatom legyen az AI betanításához. Ebben az időszakban foglalkoztam a játék VR támogatásának fejlesztésével is.

### **1.2.5. AI betanítása**

A gyűjtött adatokat felhasználva betanítottam az AI-t egy tanító algoritmus segítségével.

### **1.2.6. AI játszatása**

A játékhoz létrehoztam egy interfészt, amely lehetővé tette az AI számára, hogy játszhasson vele. Miután ez sikeresen működött, lehetőséget teremtettem arra is, hogy az emberi játékos a gép ellen is játszhassa a játékot.

## 2. fejezet

### Irodalomkutatás

Kutatásom során első sorban azt akartam megvizsgálni, hogy mások milyen AI tanítási formákat használtak különböző játékok játszatásához.

Sokáig úgy tartották, hogy az egyetlen játék, amit nem tudunk megtanítani a mesterséges intelligenciának, akkor az a Go. Azonban, Training Deep Convolutional Neural Networks to Play Go tanulmánya [2] bemutatja, hogy egy neurális hálózattal ez is lehetséges. A bemenethez az aktuális állapotot az aktuális állást használják, a kimenethez pedig a valószínűsége az összes rácspontnak a táblán, ahova az AI lerakja a követ.

A neurális hálózatot használhatjuk különböző játékelméleti döntésekhez is, ahogy azt A recurrent neural network for game theoretic decision making [3] cikkben is olvashatjuk.

A Modular Neural Networks for Learning Context-Dependent Game Strategies [4] tanulmányban már 1991 -ben írtak arról, hogyan lehet megfigyelt tanítással kontextus függő játékokat játszatni a neurális hálózattal, például amőba, vagy Backgammon-t. A Backgammonhoz moduláris és monolithic hálózatot is használnak.

A Cognitive Learning and the Multimodal Memory Game: Toward Human-Level Machine Learning [1] egy próbálkozás, amelyben egy memória játékot próbálnak egy AI-nak megtanítani olyan elméletet használva, ahogyan az emberek is tanulnak.

### 3. fejezet

#### Felhasznált technológiák

Munkám során törekedtem arra, hogy a felhasznált technológiákat lehetőleg minimáljam. Figyelembe vettetem továbbá azt is, hogy nyílt forráskódú, és multiplatform eszközöket válasszak. Ezen döntések lehetővé tették számomra a kellő flexibilitást, és elősegítették a munkámat.

##### 3.1. Godot Engine

A Godot egy nyílt forráskódú, ingyenesen elérhető játékmotor és fejlesztői környezet, amelyet a játékok, interaktív tartalmak és egyéb multimédiás alkalmazások létrehozására terveztek. A motorot Juan Linietsky, Ariel Manzur és George Marques alapította 2014-ben, és azóta folyamatos fejlesztés alatt áll, számos kiadott verzióval és fejlesztői közösséggel.

A Godot kiemelkedik sokoldalúsága és könnyűsége miatt. Az egyik legfontosabb jellemzője az integrált fejlesztői környezet (IDE), amely segítségével a fejlesztők egyetlen alkalmazásban végezhetik el a játékterv készítését, a kódolást, a grafika létrehozását és a játéktesztek futtatását. Az IDE rendelkezik számos funkcióval, mint például kódszerkesztő, jelenet szerkesztő, animációkészítő, fizikai motor, hangkezelő, és még sok más, amelyek egyszerűsítik és gyorsítják a fejlesztési folyamatot.

A Godot támogatja a kettő dimenziós (2D) és a három dimenziós (3D) játékfejlesztést is, és számos előre elkészített funkciót és sablont kínál minden típushoz. A motor különösen erős a vizuális effektek, az animációk és a szkriptelés terén, és lehetővé teszi a fejlesztők számára, hogy rugalmasan alkalmazzák saját ötleteiket és terveiket a játék készítése során.

A Godotot széles körben használják különböző projektekben, beleértve az indie játékokat, oktatási alkalmazásokat, interaktív médiaalkotásokat és még sok mást.

A motor aktív és elkötelezett fejlesztői közösséggel rendelkezik, amely folyamatosan hozzájárul az új funkciók, javítások és dokumentációk fejlesztéséhez.

Azért a Godot mellett döntöttem, mivel úgynevezett Assesst Libary formályában, lehetőségem volt könnyedén integrálni a projektembe a VR eszközök natív támogatását. Valamint hobbimból kifolyólag van ismeretem a program használatában.

### 3.2. OpenXR

Az OpenXR egy nyílt szabványú API (Application Programming Interface), amelyet a virtuális valóság és kiterjesztett valóság (AR) alkalmazások fejlesztésére terveztek.

Az OpenXR-t az OpenXR Working Group hozta létre azért, amelyben olyan nagy szereplők vesznek részt, mint az Oculus, a Valve, az Epic Games és a Google.

Az OpenXR célja, hogy egy általános API-t nyújtson, amely lehetővé teszi a fejlesztők számára, hogy alkalmazásaikat egységes kód alapján futtathassák az összes támogatott VR/AR eszközön, függetlenül azok gyártójától vagy típusától. Ezáltal nincs szükség külön-külön optimalizálniuk alkalmazásokat minden egyes VR/AR platformra, hanem egyszerűen használhatják az OpenXR-t, hogy egyetlen kódázból több platformon is futtatható legyen a kész termékük.

Az API lehetővé teszi a fejlesztők számára, hogy közvetlen hozzáférést kapjanak a VR/AR eszközök hardveres funkcióihoz és jellemzőihez, mint például a képminőség beállítás és a mozgásérzékelés.

Az OpenXR API széles körben támogatott a VR/AR iparágban, és egyre több eszköz és platform támogatja az OpenXR specifikációkat, mint például a Godot.

Ennek az API-nak hála, lényegében egy gombnyomásra kitudtam exportálni a Meta Quest 3 VR szemüvegemre a kész játékot, és futtatni tudtam rajta azt azonnal.

### 3.3. GDscript

A GDScript a Godot engine saját szkriptelési nyelve, amelyet a játékfejlesztéshez terveztek. Könnyen tanulható és használható nyelv, amelyet kifejezetten a Godothoz optimalizáltak, így tökéletesen illeszkedik a motor által nyújtott funkciókhoz és struktúrához.

A GDScript egy dinamikus típusú script nyelv, ezáltal egyszerűbb és rugalmasabb

kódolási stílust tesz lehetővé, amely könnyen alkalmazható a játékfejlesztés során.

Támogatja az objektumorientált programozás alapvető elveit, mint például az osztályok, az öröklődés és a polimorfizmus. Emellett rendelkezik számos beépített funkcióval és osztállyal, melyek jelentősen megkönnyítik a játékprogram elkészültét.

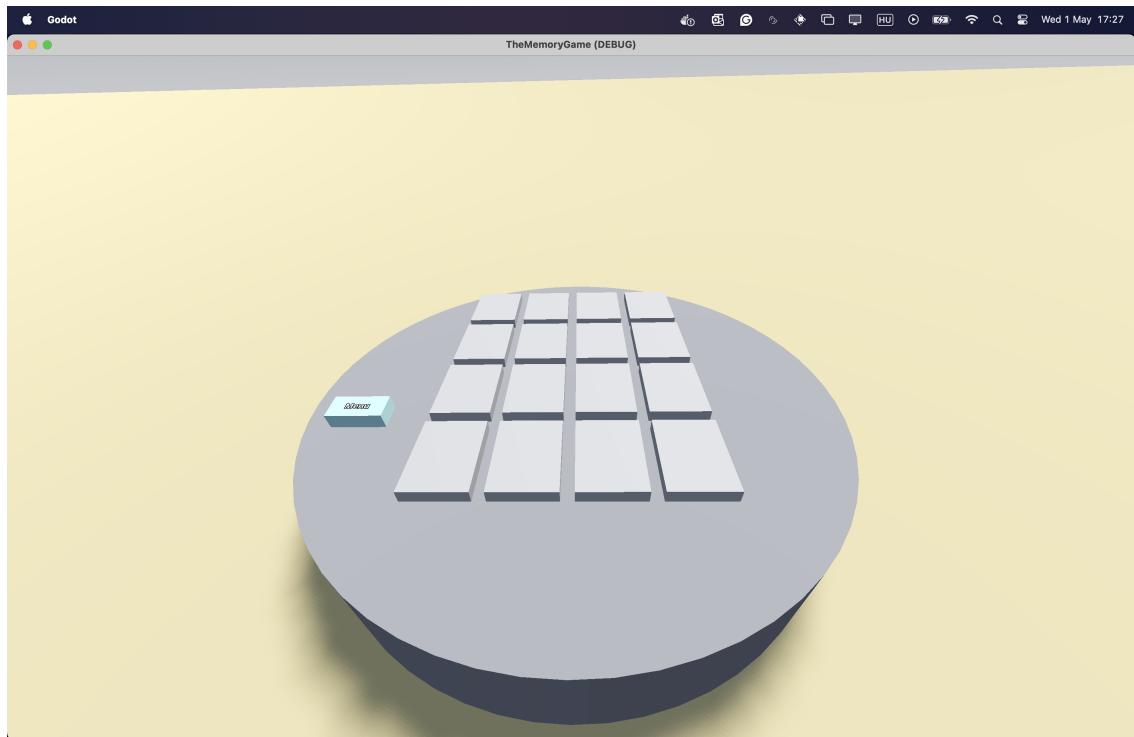
## 4. fejezet

### A játék működése

#### 4.1. Játék ismertetése

A játék melyet lefeljlesztem, a közismert memória játék. A játékot lehet egyedül, vagy akár többen is játszani.

A játékban, egy asztalon meghatározott számú kártya pár található, képpel lefelé fordítva ahogyan az a 4.1. ábrán is látható. A kártyák előlapján betűk találhatók.

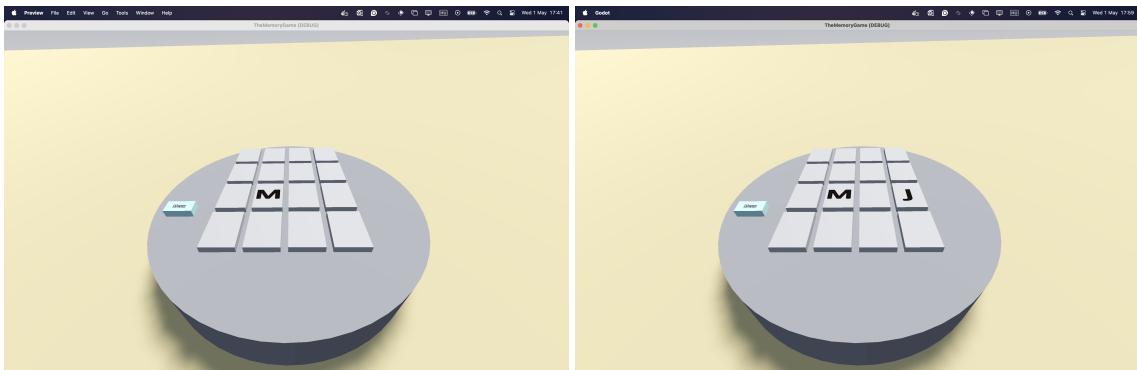


4.1. ábra. 4x4-es memóriajáték kezdő állapota

Egyjátékos esetben a játékos célja, hogy minél kevesebb kártyapár megfordításából megtalálja az összes memória párt. Többjátékos esetben, hogy ő szerezze a legtöbb

pontot, vagyis több kártyapárt fordítson fel, mint az ellenfelei.

Ahhoz, hogy egy kártyát megfordítson, a játékosnak rá kell kattintania. Ekkor láthatóvá válik, mely betűhöz tartozik a memória elemhez (4.2a. ábra). A megfordított kártyához választani kell egy másikat. A játékosnak törekednie kell, hogy korábbi ismeretei alapján, a következőre a választott kártya előlapján ugyanaz a betű szerepeljen, mint a már felfordított memória lapon, vagyis egy párt fordítson fel. Értelemszerűen ez az első felfordításkor nem lehetséges, hiszen nincs korábbi ismerete a játékról (4.2b. ábra). Ha a felfordított kártyák nem alkotnak párt, akkor



(a) Egy kártya ki van választva

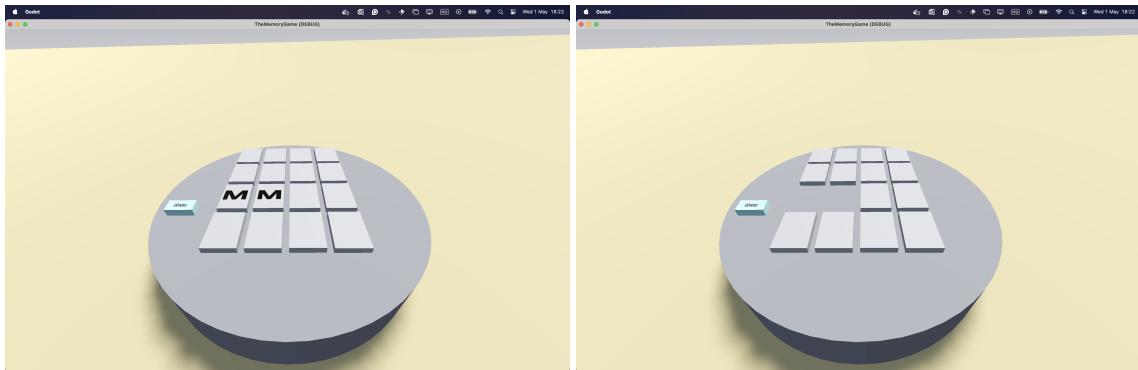
(b) Mivel a betűk nem azonosak, ezér ez nem egy pár, visszafordítjuk a kártyákat.

4.2. ábra. 4x4-es játék, példa kör, amikor nem egy párt húzunk fel

a kártyák maguktól visszafordulnak pár másodperc elteltével. Ez után egyszemélyes játék esetén esetén végrehajtunk egy újabb fordítást. Többjátékos esetén a következő játékos végezheti el a körét.

Ha párt alkotnak (4.3a. ábra), akkor a kártyák eltűnnek a játékmezőről (4.3b. ábra). Többjátékos esetben a felfordított játékos kap egy pontot, és egy újabb fordítással folytatja a körét, mindaddig, míg egy nem párt fordít.

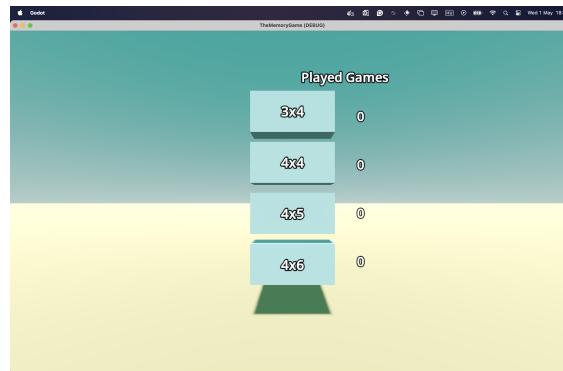
## A játék működése



(b) Eltűnik a pár az asztalról

4.3. ábra. 4x4-es játék, példa kör, amikor egy párt húzunk fel

Amint az összes kártya eltűnik az asztalról, a játék véget ér, és visszakerülünk a menübe. A játékba több nehézségi szintet tettünk, melyet a menüből érhetünk el (4.4. ábra). A különböző menüpontok, a kártyák számának elhelyezkedését jelölik.



4.4. ábra. A játék menüje

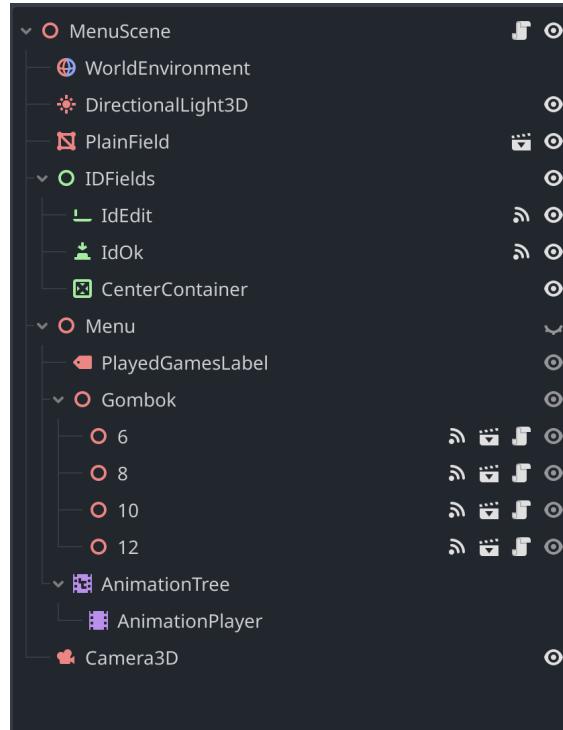
## 4.2. Struktúrális felépítés

A memória játék a Godot elveinek megfelelően Node-okból és Scene-ekből [5] áll. A Scene-ek struktúrája a következő.

#### 4.2.1. Menü

A Játék menüje (4.5. ábra), a következő módon épül fel. A Gombok olyan MeshInstance3D Node-ok [6], melyekre ha a játékos rákattint, akkor emittálnak egy

`button_pressed()` signal-t (4.6. ábra). A MenuScene kódjában hallgatózunk erre külön külön a gombokra. A megfelelő gomb megnyomásával beállítjuk a `Constant.CARD_PAIR_NUMBER` globális változót, mely segítségével létrehozzuk a `basic_scene`-t.



4.5. ábra. A játék menü Scene-jének struktúrája

```

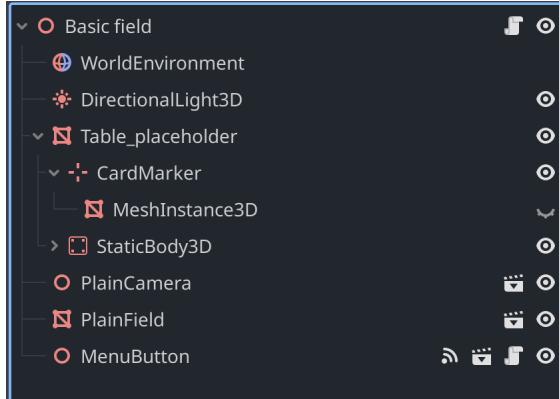
1 func _on_area_3d_input_event(camera, event, position,
normal, shape_idx):
2     if event is InputEventMouseButton:
3         if (event.button_index == MOUSE_BUTTON_LEFT
&& event.pressed == true):
4             emit_signal("button_pressed")
5
6 func set_number_label_text(new_label_text: String):
7     number_label.text = new_label_text;
8

```

4.6. ábra. A menü gombja `button_pressed` signal-t emittál

### 4.2.2. Basic Scene

A Basic Scene (4.7. ábra) struktúrája dinamikusan épül fel a Card Scene-ekből, melyet a Deck globális objektum ad oda a Basic Scene -nek.



4.7. ábra. A játéktér Scene struktúrális felépítése

A scene kiszámolja, az előre beállított kártya szélesség, magasság és margó konstansok alapján, hogy a megkapott kártyák számát, a CardMarker -höz képest hova kell lerakni (4.8. ábra)

```
1     func _calculate_coordinate(i, j):
2         return Vector3(
3             TABLE.position.x + (((CARD_WIDTH + MARGO) *
4 CARD_SCALE) * i ) - (((CARD_WIDTH * CARD_ROW) - (
5 CARD_WIDTH) + (MARGO*(CARD_ROW-1)))*CARD_SCALE / 2),
6             TABLE.position.y,
7             TABLE.position.z + (((CARD_HEIGHT + MARGO) *
8 CARD_SCALE) * j ) - (((CARD_HEIGHT * CARD_COLUMN) - (
9 CARD_HEIGHT) + (MARGO*(CARD_COLUMN-1)))*CARD_SCALE /
10 2)
11     )
```

4.8. ábra. Kártyák koordinátájának kiszámítása

Amint a Deck objektum elküldi a `cards_empty` signal-t, vagyis a játék végét, vagy ha a játékos megnvomja a Menü gombot, a játék visszatér a menübe.

Más feladata nincs.

#### 4.2.3. Deck

A `Deck` egy olyan globális objektum, mely a program futása során bármikor elérhető.

A feladatai:

1. Létrehozni a kártyákat, a játék kezdetekor (4.9. ábra)
2. Folyamatosan figyeli, hogy mely `Card`-ok vannak még játékban a `lstinline|cards|` tömbben.
3. Kezeli a kártyák felfordítását. Ha két kártya azonos, akkor azokat kiveszi a listájából (4.10. ábra).
4. Lementeni a játékos minden lépését a data tömbbe a memóriába.
5. Figyeli a játék végét.
6. Lementeni a data tömböt egy JSON file-ba a játék végével.

```

1  func make_deck(card_pair_number: int, card_scene:
2      PackedScene):
3      data.card_pair_number = card_pair_number;
4      cards.clear();
5      for i in range(0, card_pair_number):
6          var word = "";
7          while ABC.find(word) > - 1 or word == "":
8              word = generate_word(
9                  abcdefghijklmnopqrstuvwxyz', 1).to_upper();
10             ABC.push_back(word);
11             var card = card_scene.instantiate();
12             #call_deferred("add_child", card);
13             if card.has_method("set_label"):
14                 card.set_label(word);
15             cards.push_back(card);
16             cards.push_back(card.duplicate());
17             cards.shuffle();
18             data.card_labels = ABC;

```

4.9. ábra. Létrehozzuk a kártyákat

```

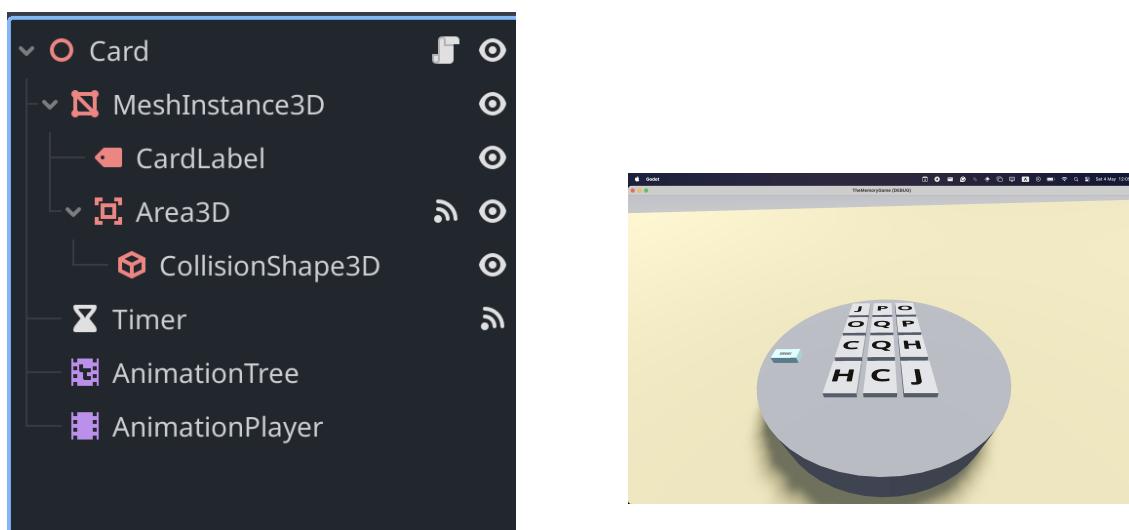
1 func talalat():
2     if (chosenA.get_label() == chosenB.get_label()):
3         cards.remove_at(cards.find(chosenA));
4         cards.remove_at(cards.find(chosenB));
5         chosenA.queue_free();
6         chosenB.queue_free();
7     else:
8         chosenA.play_card_reflip_animation();
9         chosenB.play_card_reflip_animation();
10    chosenA = null;
11    chosenB = null;
12    can_flip = true;
13    if cards.size() == 0 :
14        cards_empty.emit()
15        save_data()
16

```

4.10. ábra. Figyeljük a találatot

#### 4.2.4. Card

A kartyák vagyis Card scene (4.11. ábra) rendelkezik egy `labellel` amelyet a Deck add neki, a kártya létrehozásakor. Ez mondja meg, hogy milyen kártya. Egy játékmezőn pontosan két egyforma labellel rendelkező kártya szerepel(4.12. ábra).



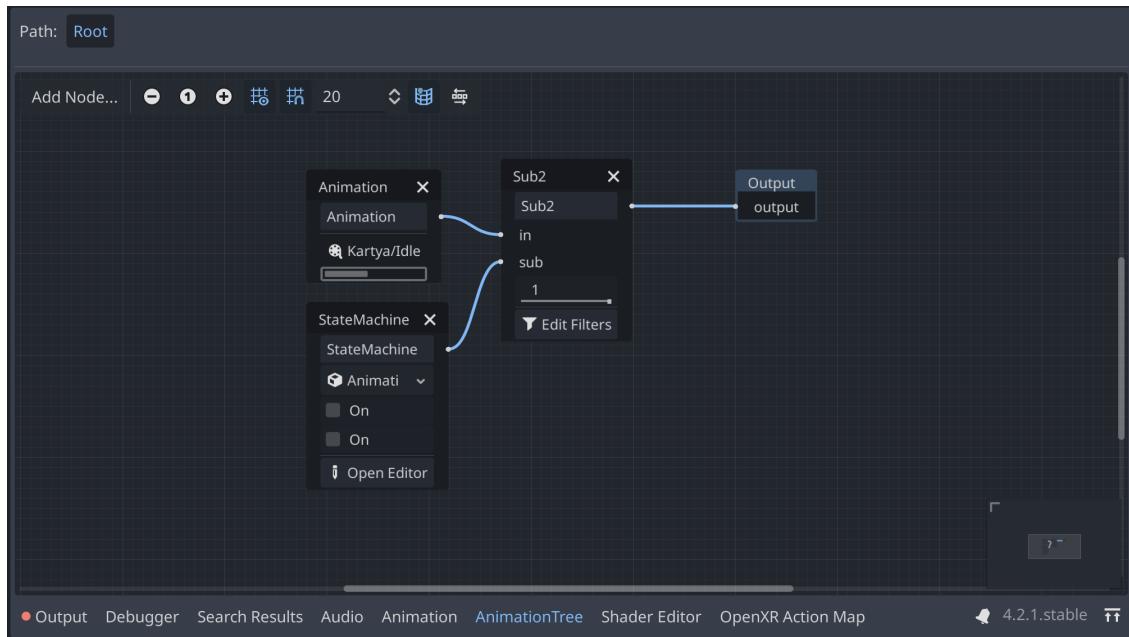
4.11. ábra. A Card Scene struktúrája

4.12. ábra. 4x4 kártya. minden betűből csak egy pár szerepel

A kártya rendelkezik animációval is, melyet az AnimationTree (4.13. ábra) kezel.

## A játék működése

Ha rányomunk a kártyára, akkor lefut a `card_flip_animation` function, vagyis az AnimationTree State Machine-nek odaadjuk azt az információt, mely szerint meg kell fordítani a kártyát. Az objektum elküldi önmagát a Deck-nek, mint kiválasztott kártya.



4.13. ábra. A Card animáció fája

## Irodalomjegyzék

- [1] B.-T. Zhang, „Cognitive learning and the multimodal memory game: Toward human-level machine learning,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 3261–3267, 2008.
- [2] C. Clark and A. Storkey, „Training deep convolutional neural networks to play go,” in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1766–1774, PMLR, 07–09 Jul 2015.
- [3] S. Bhatia and R. Golman, „A recurrent neural network for game theoretic decision making,” in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 36, 2014.
- [4] J. A. Boyan, *Modular neural networks for learning context-dependent game strategies*. University of Cambridge. Computer Laboratory, 1992.
- [5] „Nodes and Scenes — docs.godotengine.org.” [https://docs.godotengine.org/en/stable/getting\\_started/step\\_by\\_step/nodes\\_and\\_scenes.html](https://docs.godotengine.org/en/stable/getting_started/step_by_step/nodes_and_scenes.html). [Accessed 01-05-2024].
- [6] „MeshInstance3D — docs.godotengine.org.” [https://docs.godotengine.org/en/stable/classes/class\\_meshinstance3d.html](https://docs.godotengine.org/en/stable/classes/class_meshinstance3d.html). [Accessed 01-05-2024].

## Ábrák jegyzéke

1.1. A projekt folyamatábrája . . . . .	4
4.1. 4x4-es memória játék kezdő állapota . . . . .	10
4.2. 4x4-es játék, példa kör, amikor nem egy párt húzunk fel . . . . .	11
4.3. 4x4-es játék, példa kör, amikor egy párt húzunk fel . . . . .	12
4.4. A játék menüje . . . . .	12
4.5. A játék menü Scene-jének struktúrája . . . . .	13
4.6. A menü gombja <code>button_pressed</code> signal-t emittál . . . . .	13
4.7. A játéktér Scene struktúrális felépítése . . . . .	14
4.8. Kártyák koordinátájának kiszámítása . . . . .	14
4.9. Létrehozzuk a kártyákat . . . . .	15
4.10. Figyeljük a találatot . . . . .	16
4.11. A Card Scene struktúrája . . . . .	16
4.12. 4x4 kártya. minden betűből csak egy pár szerepel . . . . .	16
4.13. A Card animáció fája . . . . .	17

## Táblázatok jegyzéke