

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

## SZAKDOLGOZAT

3D memóriajáték tervezése mesterséges  
intelligenciával

Csesznák Tamás Levente

Témavezető: Szabó Patrícia

Külső/belső konzulens: «**külső konzulens neve**», «**intézménye**»

2024



# PANNON EGYETEM

## MŰSZAKI INFORMATIKAI KAR

### Programtervező informatikus BSc szak

Veszprém, 2022. március 23.

#### SZAKDOLGOZAT TÉMAKIÍRÁS

##### **Hallgató neve**

Programtervező informatikus BSc szakos hallgató részére

##### **Szakdolgozat címe**

Témavezető: Témavezető neve, beosztása

##### **A feladat leírása:**

...

##### **Feladatkiírás:**

- Dolgozza fel a témaival kapcsolatos eddig hazai és külföldi irodalmat!
- Határozza meg a szoftverrel szemben támasztott követelményeket!
- Tervezze meg a játékhoz szükséges adatstruktúrát, algoritmusokat!
- ...

Dr. Szakfelelős Oktató  
egyetemi docens  
szakfelelős

Témavezető Oktató  
egyetemi docens  
téma vezető

---

## Hallgatói nyilatkozat

Alulírott Csesznák Tamás Levente hallgató kijelentem, hogy a dolgozatot a Pannon Egyetem Informatikai Rendszerek és Alkalmazásai Tanszékén készítettem a mérnökinformatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozatban lévő érdemi rész saját munkám eredménye, az érdemi részen kívül csak a hivatkozott forrásokat (szakirodalom, eszközök stb.) használtam fel.

Tudomásul veszem, hogy a dolgozatban foglalt eredményeket a Pannon Egyetem, valamint a feladatot kiíró szervezeti egység saját céljaira szabadon felhasználhatja.

Dátum: Veszprém , 2024.05.01

.....  
Csesznák Tamás Levente

---

## Témavezetői nyilatkozat

Alulírott Szabó Patrícia témavezető kijelentem, hogy a dolgozatot Csesznák Tamás Levente a Pannon Egyetem Informatikai Rendszerek és Alkalmazásai Tanszékén készítette a mérnökinformatikus végzettség megszerzése érdekében.

Kijelentem, hogy a dolgozat védésre bocsátását engedélyezem.

Dátum: Veszprém , 2024.05.01

.....  
Szabó Patrícia

## **Tartalomjegyzék**

## **Jelölésjegyzék**

*2D* Kettő dimenziós

*3D* Hárrom dimenziós

*AI* Artificial Intelligence (Mesterséges Intelligencia)

*IDE* Integrált fejlesztői környezet

*VR* Virtual Reality

## 1. fejezet

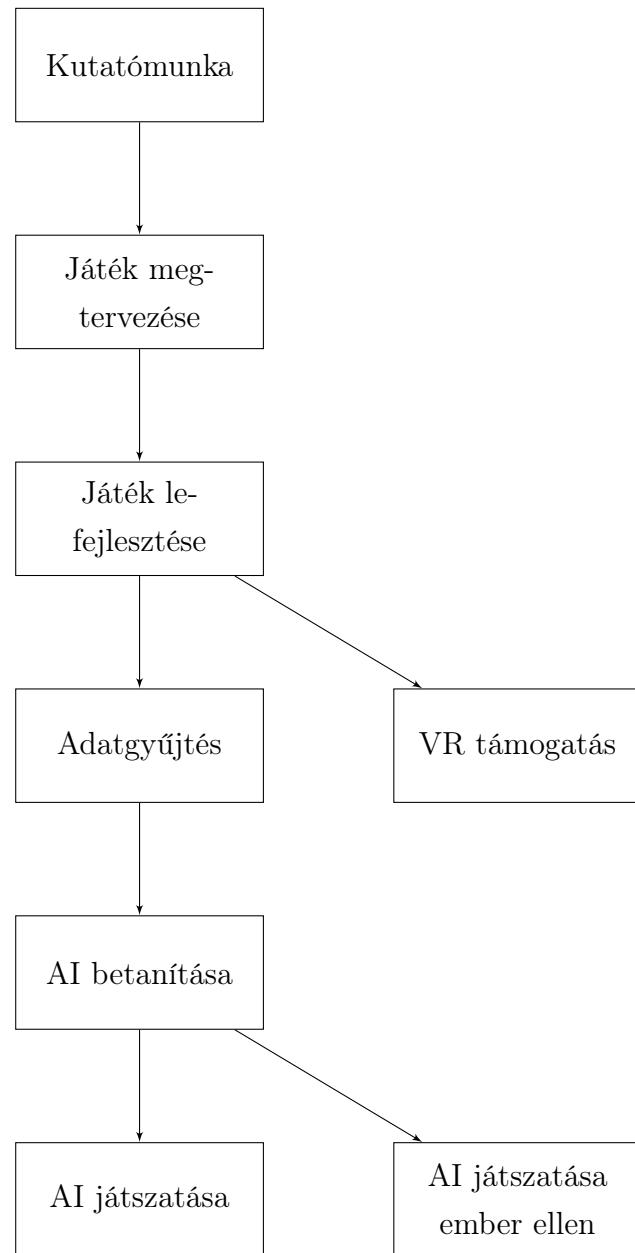
### Bevezetés

#### 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 memóriajátékot tervezek létrehozni, amely segítségevel interaktív és hatékony módon lehet fejleszteni a játékosok kognitív képességeit.

## 1.2. Projekt bemutatása



1.1. ábra. Projekt folyamatábrája

projektem több feladatból állt, melyet egy folyamatdiagram (?? ábra) szemléltet.

### **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.

### **1.2.3. Játék lefejlesztése**

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 kezelní 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

### 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.

#### 2.1. 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.

## 2.2. 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 kódba-zisból 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.

## 2.3. GDscript

A GDScript [?] a Godot engine [?] saját szkriptelési nyelve, amelyet a játékfejlesztéshez terveztek. Könnyen tanulható és használható nyelv, amelyet kifejezetten a Godot-hoz optimalizáltak, így tökéletesen illeszkedik a motor által nyújtott funkcióhoz 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.

### 3. fejezet

## A játék működése

### 3.1. 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 ???. ábrán is látható. A kártyák előlapján betűk találhatók. 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.

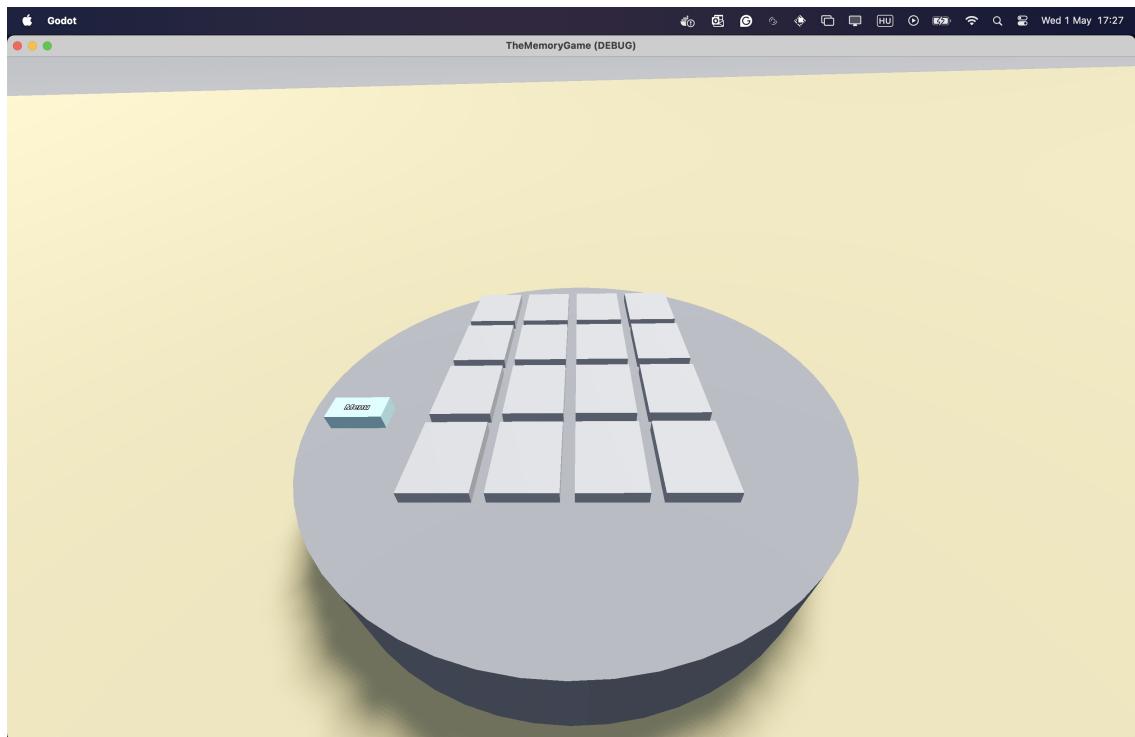
Ahhoz, hogy egy kártyát megfordítson, a játékosnak rá kell kattintania. Ekkor láthatóvá válik, mely betűhöz tartozik a memória elemhez (???. ábra). A megfordított kártyához választani kell egy másikat. A játékosnak törekednie kell, hogy korábbi ismeretei alapján, a következőre a választott kártya előlapján 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 (???. ábra).

Ha a felfordított kártyák nem alkotnak párt, akkor a kártyák maguktól visszafor dulnak pár másodperc elteltével. Ez után egyszemélyes játék esetén esetén végre hajtunk egy újabb fordítást. Többjátékos esetén a következő játékos végezheti el a körét.

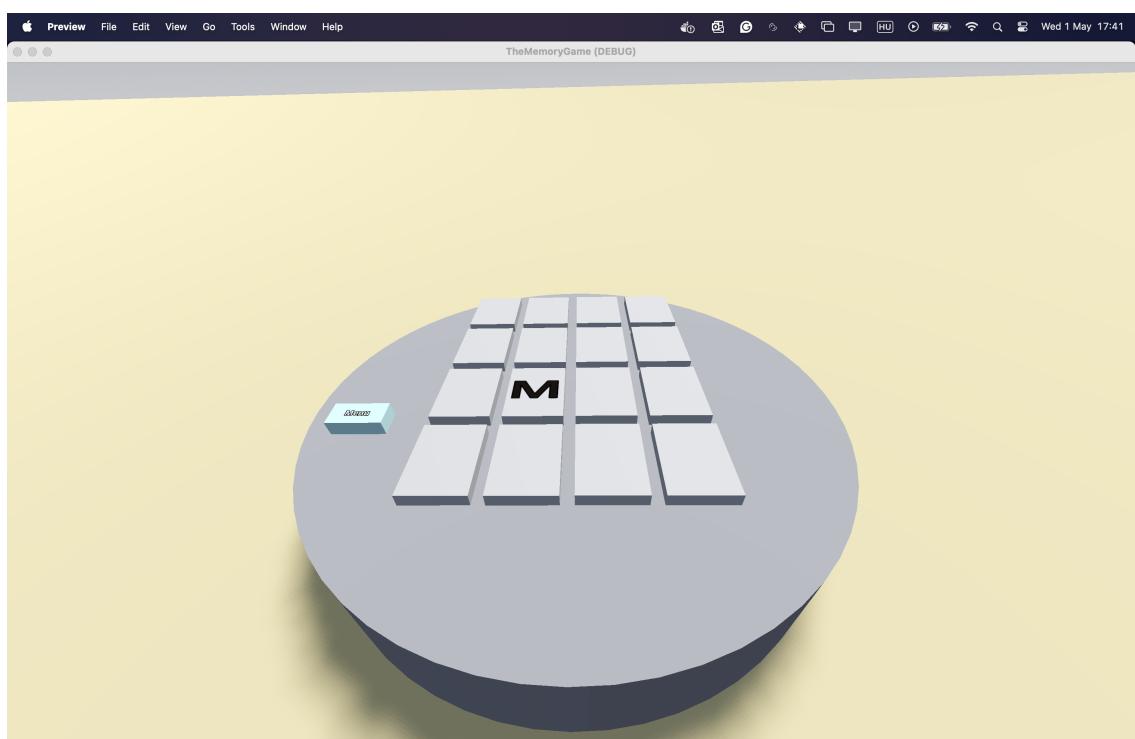
Ha párt alkotnak (???. ábra), akkor a kártyák eltűnnek a játékmezőről (???. ábra). Többjátékos esetben a felfordított játékos kap egy pontot, és egy újabb fordítással folytatja a körét, mindenkor, míg egy nem párt fordít.

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 (???. ábra). A különböző menüpontok, a kártyák számának elhelyezkedését jelölik.

## A játék működése

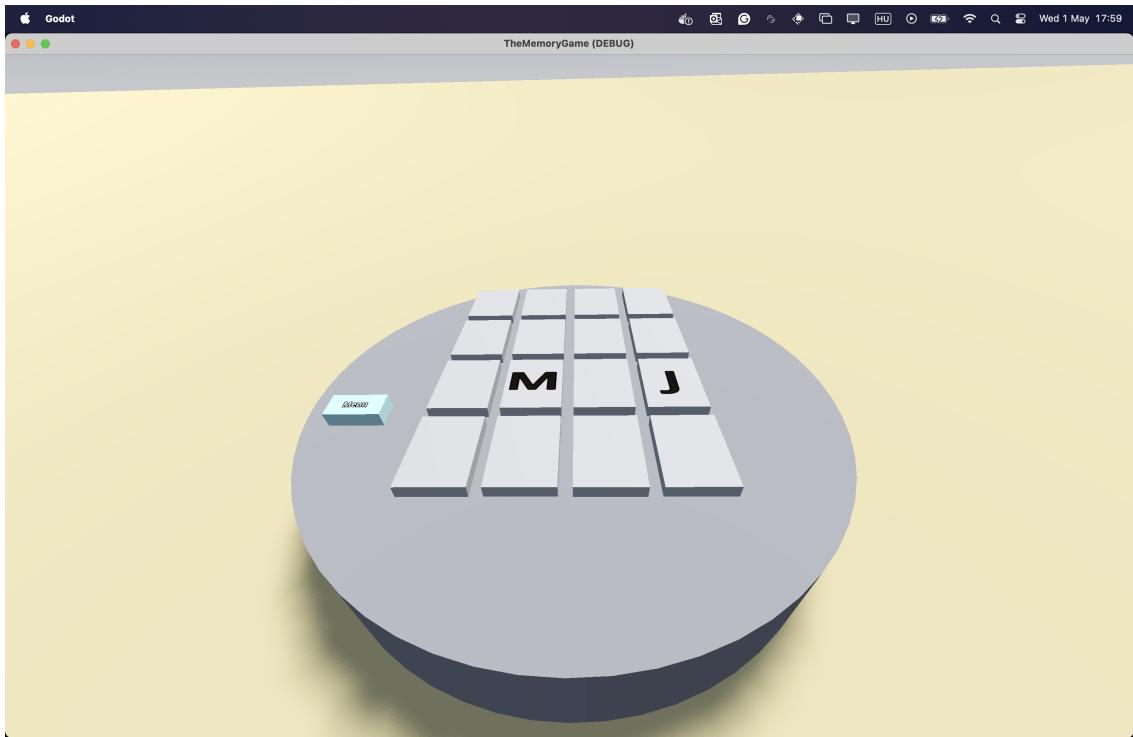


3.1. ábra. 4x4-es memóriajáték kezdő állapota

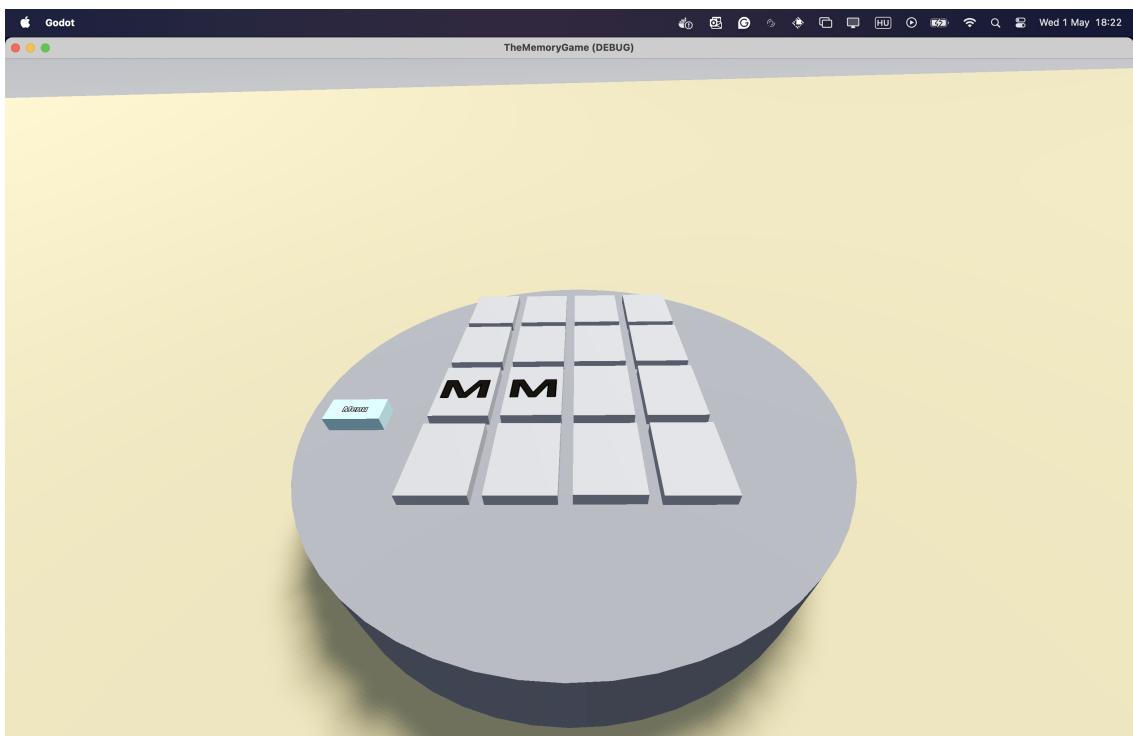


3.2. ábra. 4x4-es memóriajáték egy kártya ki van választva

### A játék működése

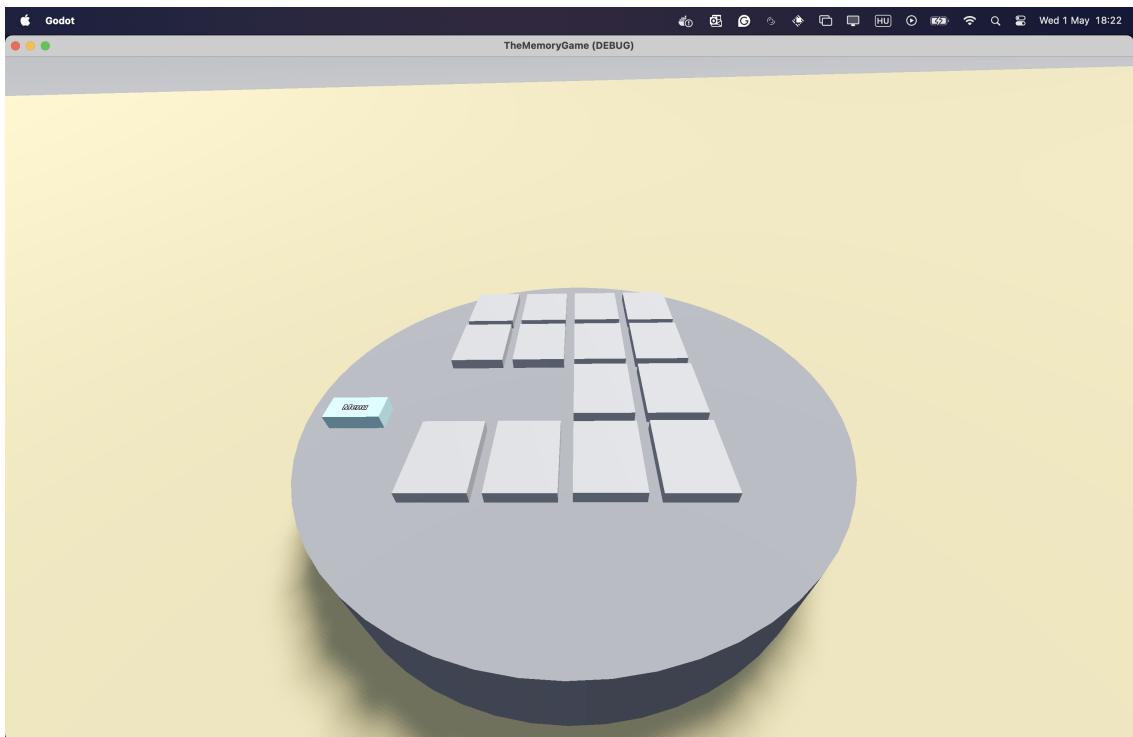


3.3. ábra. 4x4-es memóriajáték. Mivel a betűk nem azonosak, ezér ez nem egy pár, visszafordítjuk a kártyákat.

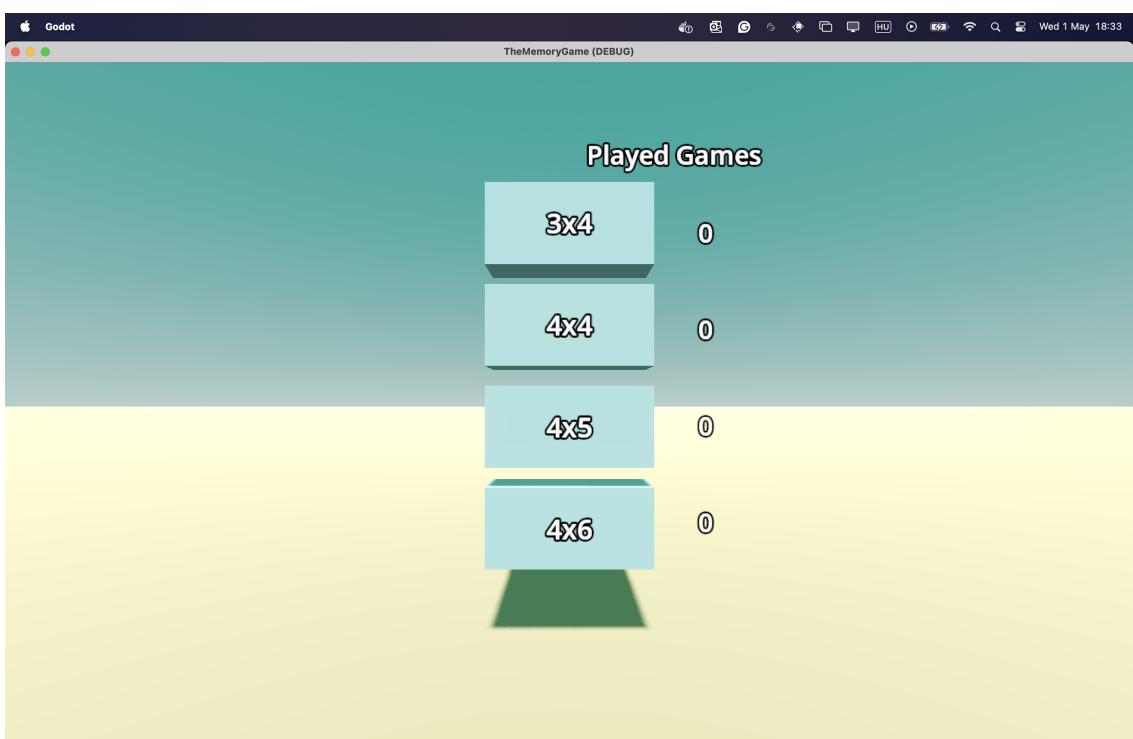


3.4. ábra. A kiválasztott kártyák párt alkotnak

## A játék működése



3.5. ábra. Eltűnik a pár az asztalról



3.6. ábra. A játék menüje

```

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

```

3.7. ábra. A menü gombja button\_pressed signal-t emittál

### 3.2. Struktúrális felépítés

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

#### 3.2.1. Menü

A Játék menüje (???. ábra), a következő módon épül fel. A Gombok olyan MeshInstance3D Node-ok [?], melyekre ha a játékos rákattint, akkor emittálnak egy button\_pressed() signal-t (???. 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.

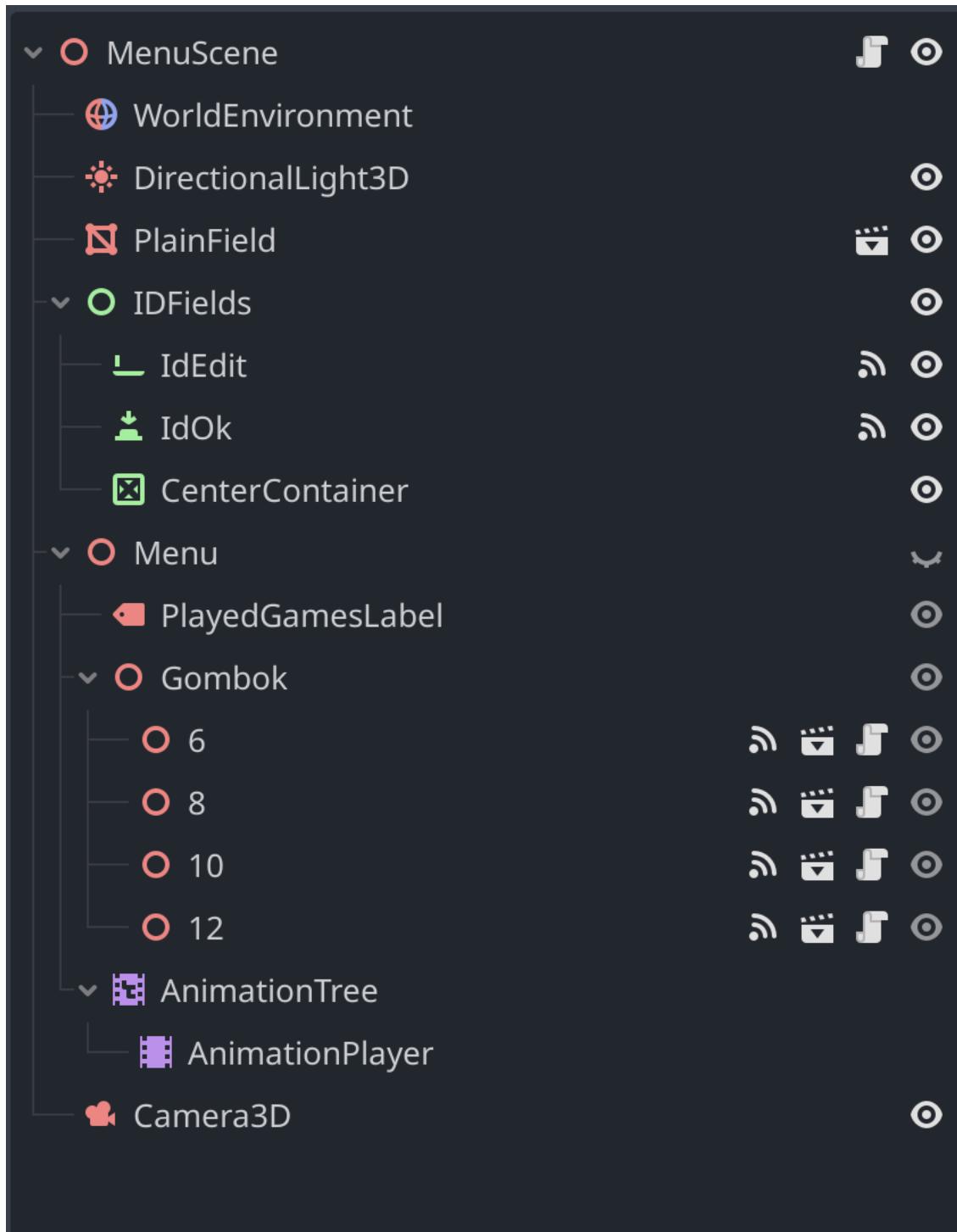
#### 3.2.2. Basic Scene

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

Amint a Deck objektum elküldi a cards\_empty signal-t, vagyis a játék végét, vagy ha a játékos megnyomja a Menü gombot, a játék visszatér a menübe.

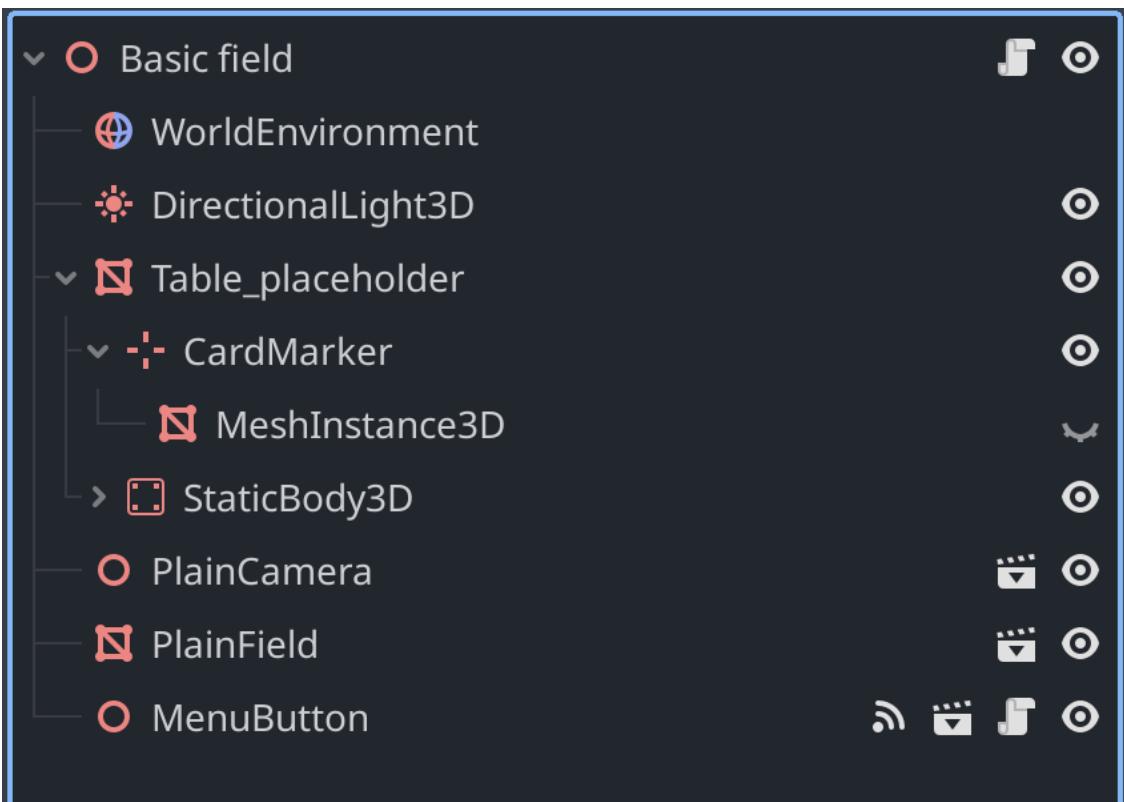
Más feladata nincs.

## A játék működése



3.8. ábra. A játék menü Scene-jének struktúrája

## A játék működése



3.9. ábra. A játéktér Scene struktúrális felépítése

```
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 / 2)
10        )
```

3.10. ábra. Kártyák koordinátájának kiszámítása

```

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;
19

```

3.11. ábra. Létrehozzuk a kártyákat

### 3.2.3. Deck

A **Deck** egy olyan globális objektum, mely a program futása során bármikor elérhető.  
A feladatai:

1. Létrehozni a kártyákat, a játék kezdetekor (???. 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 (???. 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.

### 3.2.4. Card

## A játék működése

```
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
```

3.12. ábra. Figyeljük a találatot

## Irodalomjegyzék

- [1] G. Engine, „Godot Engine - Free and open source 2D and 3D game engine — godotengine.org.” <https://godotengine.org/>. [Accessed 01-05-2024].
- [2] „OpenXR - High-performance access to AR and VR —collectively known as XR— platforms and devices — khronos.org.” <https://www.khronos.org/OpenXR/>. [Accessed 01-05-2024].
- [3] „GDScript reference — docs.godotengine.org.” [https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript\\_basics.html](https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html). [Accessed 01-05-2024].
- [4] „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].
- [5] „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

## Táblázatok jegyzéke