

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM
MAROSVÁSÁRHELYI KAR,
INFORMATIKA SZAK**



**SAPIENTIA
ERDÉLYI MAGYAR
TUDOMÁNYEGYETEM**

OpenGL alkalmazás okostelefonon

DIPLOMADOLGOZAT

Témavezető:

Dr. Kovács Lehel István,
Egyetemi adjunktus

Végzős hallgató:

Iuhos Imola

2023

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,
SPECIALIZAREA INFORMATICĂ**



**UNIVERSITATEA
SAPIENTIA**

Aplicație OpenGL pentru smartphone

LUCRARE DE DIPLOMĂ

Coordonator științific:
Dr. Kovács Lehel István,
Lector universitar

Absolvent:
Iuhos Imola

2023

**SAPIENTIA HUNGARIAN UNIVERSITY OF
TRANSYLVANIA**
FACULTY OF TECHNICAL AND HUMAN SCIENCES
COMPUTER SCIENCE SPECIALIZATION



SAPIENTIA
HUNGARIAN UNIVERSITY
OF TRANSYLVANIA

OpenGL application for smartphone

BACHELOR THESIS

Scientific advisor:

Dr. Kovács Lehel István,
Lecturer

Student:

Iuhos Imola

2023

LUCRARE DE DIPLOMĂ

Coordonator științific:
Lect. dr. Kovács Lehel István

Candidat: **Iuhos Imola**
Anul absolvirii: **2022**

a) Tema lucrării de licență:

Proiectarea și dezvoltarea unei aplicații mobile scrisă în OpenGL destinată copiilor pentru a aprofunda cunoștințele biblice

b) Problemele principale tratate:

- Studiu bibliografic privind limbajele OpenGL și C#
- Studiul kitului și bibliotecilor oferite de platforma Unity
- Studiu bibliografic privind desenării virtuale în 3D prin folosirea uneltelor oferite de platformă pentru a vizualiza aplicația
- Proiectarea și realizarea unei aplicații mobile care realizează comunicația între telefon și telecomandă prin cablu USB, distribuirea ecranului pe două lentile pentru folosirea ochelarii virtuale, recepția și prelucrarea comenzielor primite prin telecomandă și vizualizarea grafică a informațiilor: componente jocului (butoane, mișcarea zarului, mutarea marionetelor și afișarea mesajelor)
- Documentarea adecvată a stadiilor de proiectare a aplicațiilor

c) Desene obligatorii:

- Schema bloc a sistemului
- Diagrame de proiectare pentru aplicația mobilă realizată

d) Softuri obligatorii:

- Aplicația este bazată pe tehnologia OpenGL și C# și a fost dezvoltată prin platforma Unity
- Este necesar un telefon mobil echipat cu Gyroscop
- Conectarea unei telecomande/gamepad prin cablu USB la telefonul mobil folosit

e) Bibliografia recomandată:

[1] <https://developers.google.com/cardboard/develop/unity/quickstart>

[2] <https://docs.unity3d.com/Manual/UnityManual.html>

[3] Galambos Adrienn és S. Nagy Katalin: Virtuális valóság

f) Termene obligatorii de consultații:

g) Locul și durata practicii: Universitatea „Sapientia” din Cluj-Napoca, Facultatea de Științe Tehnice și Umaniste din Târgu Mureș, sala / laboratorul 414.

Primit tema la data de: **05.01.2022**

Termen de predare: **30.06.2023**

Semnătura Director Departament

Semnătura responsabilului
programului de studiu

Antal

Semnătura coordonatorului

Semnătura candidatului

Declarație

Subsemnatul/a IUHOS ITOLĂ, absolvent(ă) al/a specializării INFORMATICA, promoția 2022 ... cunoscând prevederile Legii Educației Naționale 1/2011 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, CLUJ-NAPOCĂ

Data: 29.05.2023.

Absolvent

Semnătura.....

Kivonat

Dolgozatom témája egy olyan okostelefonos alkalmazás, amely OpenGL alapú. Az OpenGL (Open Graphics Library) egy részlegesen kidolgozott szabvány, amely olyan API-t takar, amely segítségével egy egyszerű, szabványos felületen keresztül megvalósítható a grafikus kártya kezelése és a két-, háromdimenziós grafika programozása. Az OpenGL egy olyan eszköz, amit a programozók nagy kedvvel használnak, hiszen általa szinte közvetlenül vezérelhető a grafikus kártya, könnyen kirajzolhatóak a 3 dimenziós alakzatok és a királyzolás módja szabályozható.

A jelenlegi világban az emberek egyre vizuálisabbak, figyelmüket az kelti fel, ami a szemnek tetszik, ami érdekes, izgalmas, új, modern. Mindannyian új tapasztalatokat szeretnénk gyűjteni, új élményekkel gazdagodni, modern technológiát hasznáni.

A választásom éppen ezért esett az OpenGL-re, mivel egy olyan alkalmazást szerettem volna megvalósítani, amely a gyerekek számára egy szorakoztató lehetőséget nyújtson a Bibliai ismeretek felmérésére. A tudás elmélyítése mellett fontos cél volt az is, hogy a gyerekek számára egy élményt is nyújtson a játék. Mivel a gyerekek nagyon vizuálisak, így a virtuális valóságot találtam a legideálisabb módszernek ahhoz, hogy a 7 - 12 éves gyerekek érdeklődését meg tudjam ragadni és bibliai tudásukat elmélyítsem, kibővítsem.

Az OpenGL számomra egy egyszerű, élvezetes és gyorsan megvalósítható segítséget nyújtott a céлом eléréséhez.

Jelen dolgozatomban be fogom mutatni röviden a Unity programozási platformot, a Google Cardboard virtuális valóságot megvalósító programot, illetve az általam elkészített alkalmazást.

Rezumat

Subiectul tezei mele este crearea unei aplicații pentru smartphone bazată pe OpenGL. OpenGL (Open Graphics Library) este un standard parțial dezvoltat care acoperă un API cu ajutorul căruia este posibilă gestionarea plăcii grafice și programarea graficelor bi- și tridimensionale printr-o interfață simplă, standard. OpenGL este un instrument pe care programatorii îl folosesc cu mare plăcere, deoarece permite controlul aproape direct al plăcii grafice, formele tridimensionale pot fi desenate cu ușurință, iar metoda de randare poate fi controlată.

În lumea actuală, oamenii sunt din ce în ce mai vizuali, le este atrasă atenția spre ceea ce face plăcere ochiului, ceea ce este interesant, incitant, nou, modern. Cu toții dorim să adunăm experiențe noi, să ne îmbogățim cu noi experiențe și să folosim tehnologia modernă.

Tocmai de aceea am ales OpenGL, deoarece am vrut să implementez o aplicație care să ofere copiilor o oportunitate distractivă de a-și evalua cunoștințele despre Biblie. Pe lângă aprofundarea cunoștințelor, un obiectiv important a fost acela de a oferi copiilor o experiență de joc. Deoarece copiii sunt foarte vizuali, am găsit că realitatea virtuală este cea mai ideală metodă de a capta interesul copiilor de 7-12 ani și de a profunda și extinde cunoștințele biblice.

OpenGL mi-a oferit un ajutor simplu, plăcut și rapid implementabil pentru a-mi atinge obiectivul.

În această disertație, voi prezenta pe scurt platforma de programare Unity, programul de realitate virtuală Google Cardboard și aplicația pe care am creat-o.

Abstract

The topic of my thesis is to create a smartphone application based on OpenGL. OpenGL (Open Graphics Library) is a partially developed standard that covers an API that makes it possible to manage the graphics card and program two- and three-dimensional graphics through a simple, standard interface. OpenGL is a tool that programmers enjoy using because it allows almost direct control of the graphics card, three-dimensional shapes can be easily drawn, and the rendering method can be controlled.

In today's world, people are increasingly visual, their attention is drawn to what pleases the eye, what is interesting, exciting, new, modern. We all want to gather new experiences, enrich ourselves with new experiences and use modern technology.

That's exactly why I chose OpenGL, because I wanted to implement an application that would give children a fun opportunity to assess their knowledge of the Bible. In addition to deepening knowledge, an important goal was to provide children with a play experience. Since children are very visual, we have found virtual reality to be the most ideal way to capture the interest of 7-12 year olds and deepen and expand their Bible knowledge.

OpenGL gave me a simple, nice and quickly implementable help to achieve my goal.

In this dissertation, I will briefly introduce the Unity programming platform, the Google Cardboard virtual reality program, and the application I created.

Tartalomjegyzék

1. Bevezető	10
2. Unity platform bemutatása	12
2.1. Unity Hub	12
2.2. Unity Editor	13
2.2.1. GameObject	14
2.2.2. Cameras (Kamerák)	16
2.2.3. Scenes (Jelenetek)	16
2.2.4. Prefabs (Előre elkészített objektumok)	16
2.2.5. Animation	17
2.2.6. Assets (Eszközök)	17
2.2.7. Scripts	17
2.2.8. Multiplayer and Networking	18
2.2.9. Google VR	18
2.2.10. Alkalmazás futtatása okostelefonon	19
3. Bibble	21
3.1. A játék bemutatása	21
3.2. Játék elkészítése	23
3.2.1. Controller	23
3.2.2. Játékmező	24
3.2.3. QuizPanel	26
3.2.4. Játék lefolyása	28
Összefoglaló	29
Köszönetnyilvánítás	30
Ábrák jegyzéke	31
Irodalomjegyzék	32
Függelék	33
F.1. Alkoss virtuális valóságot!	33

1. fejezet

Bevezető

Felmerülhet bennünk a kérdés, hogy mi is az a virtuális valóság és hogy egyáltalán mire jó, vagy hol használhatjuk fel? Röviden összefoglalva a virtuális valóság (Virtual Reality) egy számítógép által létrehozott mesterséges világ, amibe a felhasználó belehelyezkedhet. Úgy is definiálható mint számítógéppel vezérelt multiszenzoros kommunikációs technológia, amely lehetővé teszi az intuitív interakciót az adatokkal, új módon bevonva az emberi érzékelést. Ezt a technológiát azért hozták létre, hogy az emberek könnyebben kezeljék az információt. Vannak olyan virtuális valóságot megvalósító eszközök is, amellyel az ember akár teljesen el tud különülni a valóságtól és egy új világban találhatja magát. Egyik ilyen eszköz az Oculus Meta Quest, amelynek ára mindenki számára elfogadható, hiszen csak £299 (Angol font).



1.1. ábra. Oculus Meta Quest

Viszont vannak drágább, de ugyanakkor performánsabb szemüvegek is, mint az Oculus VR Gaming VRGineers XTAL 3, amelynek ára már £15 430.

Sokan szkeptikusan állnak a virtuális szemüvegekhez, mivel legelterjedtebb oka amiért használják az a szorakozás. Viszont nem csak ezért volt kitalálva, hanem ennél hasznosabb helyzetekre is, mint pl. azoknak is egy kis szorakozást visz az életükbe akik ágyban vannak és nincs lehetőségük fizikailag megtapasztalni dolgokat, de ezen eszközök által lehetőségük van vizuálisan átélni. Emellett hasznos a depresszióra, egyedül létre, minden-napi problémáktól való kiszakadásra és ha esetleg nincs meg az anyagi keret az egzotikus kirándulásokhoz, vagy éppen a tengert kivánjuk látni, de ténél van, könnyen megoldható, hogy ott érezhessük magunkat a szemüveg által.

Fontos viszont azt is kiemelni, hogy káros hatásai is lehetnek a virtuális valóságnak. Egy felmérés alapján a felhasználók 30 százaléka tengeribetegséghez hasonló tüneteket produkáltak, mint szédülés, fejfájás, hideg verítékezés és hánnyinger, ami a szemüveg levétele után sem szünt meg sok esetben. Sokan hallhattuk gyerekkorunkban, hogy ne üljünk túl közel a tévéhez, mert elromlik a szemünk. Nos, ezt akár a VR-ral kapcsolatban is elmondhatnák a szüleink, csak sajnos egy headset esetében nincsen lehetőségünk eltávolodni a képernyőtől. A teóriák szerint már egy órányi VR-használat is negatív hatással lehet érzékszervünkre, aki pedig tovább viseli a headsetet, fokozottan veszélyeztetve lesz. Mindezek mellett talán az egyik legkárosabb hatása a függőség nagymértékben fennállható esélye. [Dzs16]

Igy a virtuális világ könnyen vezethet ártalmatlan szorakozástól a komoly problémákhoz.



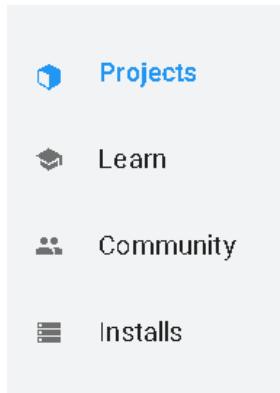
1.2. ábra. A szemüvegen túl

2. fejezet

Unity platform bemutatása

2.1. Unity Hub

A Unity Hub egy felügyeleti eszköz, amellyel az összes Unity projektet és telepítést kezelhetjük. [Ope99] A Hub segítségével kezelhetjük a Unity Editor több telepítését a hozzájuk tartozó összetevőkkel együtt, új projekteket hozhatunk létre, és megnyithatjuk a meglévő projekteket. [Ian01]



2.1. ábra. Unity Hub fontosabb részei

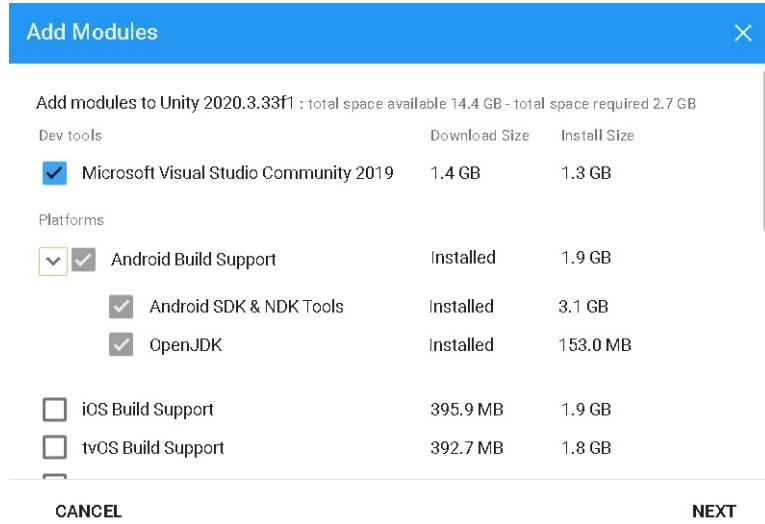
A "Projects" részben található minden olyan projekt amelyen dolgoztunk. Az "Installs" gombra kattintva, megtekinthetjük milyen verziók vannak feltelepítve a rendszeren.

A "Learn" gombra kattintva pedig olyan alap leckéket tanulhatunk, amivel beleterkinthetünk, hogy mit is csinál egy programozó, vagy akár megtanulhatjuk felépíteni az első játékunkat.

A "Community" részben olyan híreket, fórumokat, akár segítséget is kaphatsz élőben egy unity szakértőtől.

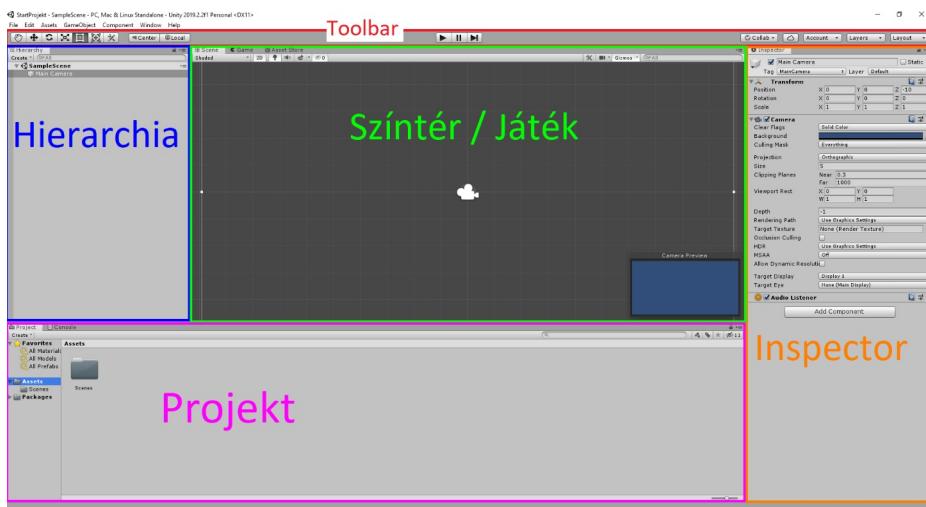
Ahhoz, hogy egy projektet futtathassunk mindenkorral telepítenünk kell egy unity editor verziót az "Installs" részben, másképp nem tudjuk futtatni a projektünket. Arra azonban vigyázni kell, hogy bizonyos projektek csak bizonyos unity editor verziókkal futnak optimálisan, ha más unity editor verzióban szeretnénk futtatni akkor előfordulhat, hogy bizonyos modulok nem töltődnek be, vagy bizonyos kód részletek nem futnak le. A unity editor telepítésekor jól meg kell gondoljuk milyen modulokra lesz szükségünk, mivel a tárhely igénye egy ilyen modulnak elég nagy és a telepítési ideje is. [Man]

2.2. Unity Editor



2.2. ábra. Unity Editor csomagjai és méretei

A Unity szerkesztő az a vizuális komponens, amely lehetővé teszi, hogy játékokat építse a következő elven: „amit látsz, azt kapsz”. A Unity elindításakor az első felület amit látunk az Editor ablak. Itt tudjuk összeállítani a programunkat, és testre szabni annak részeit.



2.3. ábra. Unity Editor részei

Az ablak központi, legnagyobb részét a Színtér/Játék nézet terület foglalja el. minden Unity-s játék úgynevezett színterekből áll, amik a játék egy-egy összefüggő részét adják. A színtér nézet a játékunk tervező felülete, ahol ezeknek a színtereknek az elemeit össze tudjuk állítani, a játék nézetben pedig meg tudjuk nézni, hogy hogyan is néz ki játék végleges formájában.

A Toolbar az Editor felső részén helyezkedik el. Segítségével váltani tudunk a szerkesztő, tervező eszközök között, és módosítani tudjuk a tervező nézet kinézetét. A Toolbar

közepén el is tudjuk indítani, vagy meg tudjuk állítani a játékot. Ilyenkor a nézet automatikusan átvált a Játék nézetbe. A játék nézetben azt látjuk, amit a játékos látna, itt módosítani nem lehet a játékon, csak élvezhetjük a játékmenetet.

A játékaink kisebb elemekből, úgynevezett GameObject-ekből állnak. Az Inspector ablakban ezeknek a paramétereit tudjuk beállítani, és testre szabni.

A játékban a játék elemek között hierarchiát építhetünk. Ha például készítünk egy autót, annak lehet négy kereke. A kerekek bár lehetnek külön játékelemek, attól függetlenül az autóhoz tartoznak. Ezt úgy jelöljük, a Hierarchia ablakban az autó gyermekéive tesszük a kerekeket. A hierarchia ablakban nem csak az elemek egymáshoz való viszonyát adhatjuk meg. Lista formában itt meg tudjuk találni a játékunk minden elemét, így az elemek közötti keresgélésben is hasznos lehet.

A projekt ablakban találhatjuk meg a játékunk lehetséges építőkockáit. minden amit bele építhetünk a játékba (képek, zenék, animációk, programkódok, stb.) itt helyezkednek el. Innen tudjuk az egyes elemeket a színtérbe beépíteni.

2.2.1. GameObject

A Unity-ben van egy nagyon fontos konstrukció, amit ismernünk kell, mielőtt belefognánk a munkába.

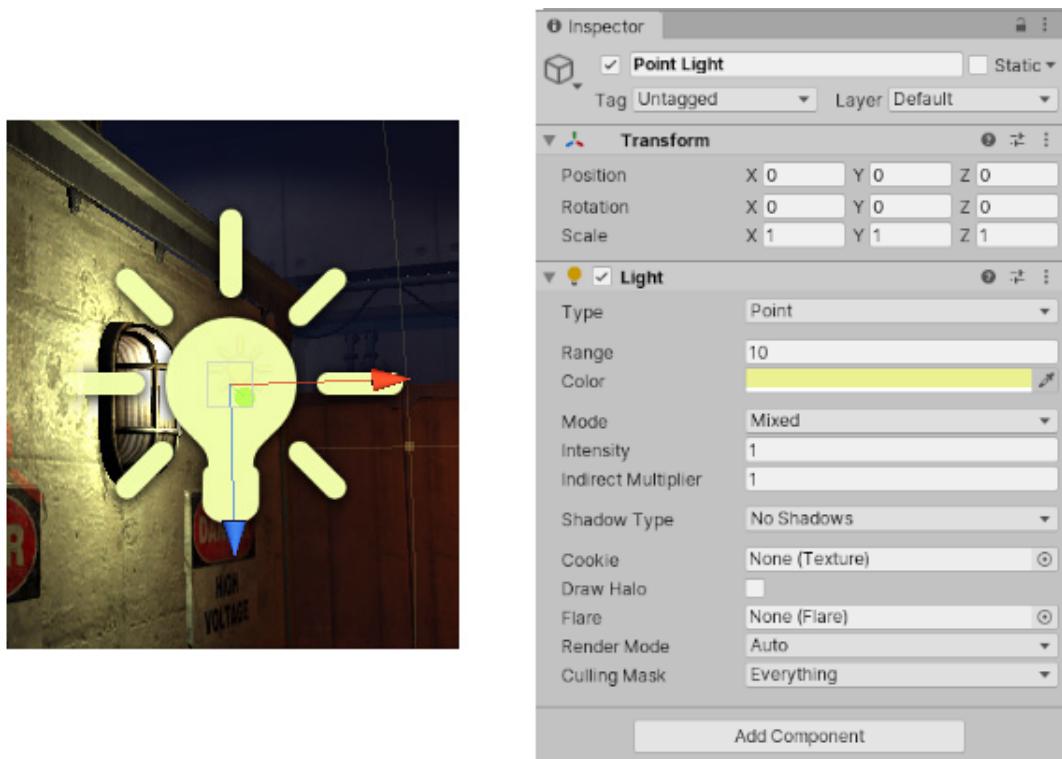
Ahogy korábban említettük a játékok színterekből épülnek fel. A színterek a játék összefüggő részei, amikre most akár tekinthetünk úgy is, mint egy-egy pályára.

A színterekben különböző játékelemeket helyezhetünk el. Ezeket nevezzük GameObject-nek (játékobjektumnak). Fontos, hogy a játékban minden egy-egy GameObject, vagyis az egyes elemek alapvetően nem különböznek egymástól.

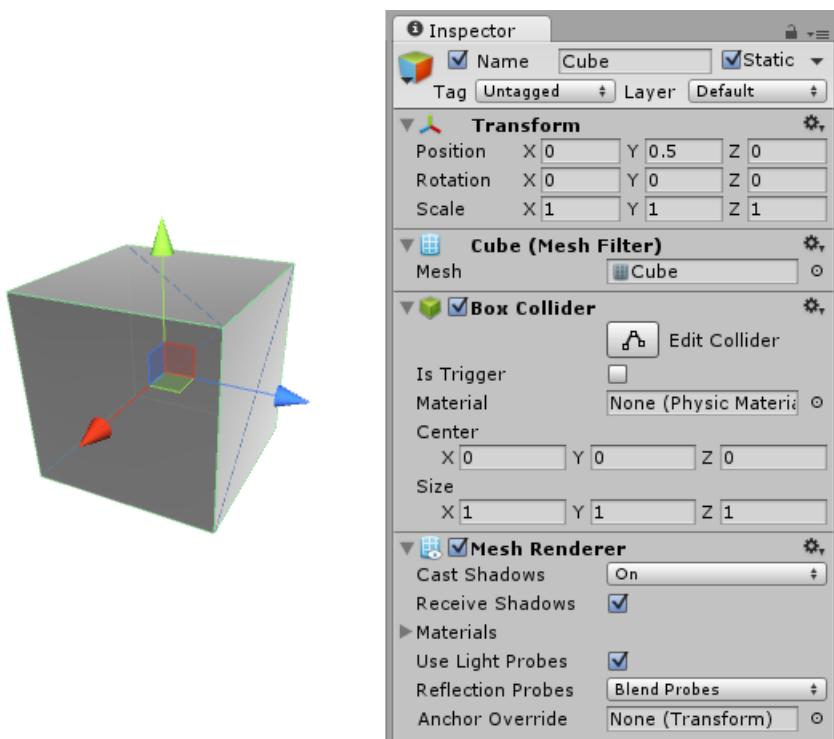
Önmagukban semmit sem csinálnak, hanem olyanok, mint a konténerek, amelyekhez a többi komponens rögzítve van, így alakítva ki az objektum végső funkcionálitását. Az összetevők olyan szkriptek, amelyek meghatározzák a GameObjects viselkedését, azok mozgását, helyzetét, elforgatását, méretét és bármilyen más egyéni viselkedést. Így például a Light Object (2.4. ábra) úgy születik meg, hogy a Light komponenst hozzáadjuk egy GameObjecthez.

A szilárd kocka objektumnak (2.5. ábra) van egy Mesh Filter és Mesh Renderer komponense a kocka felületének megrajzolásához, valamint egy Box Collider komponens, amely az objektum szilárd térfogatát fizikailag reprezentálja.

A GameObjecthez minden van egy Transform komponens csatolva (a pozíciót és a tájolást ábrázolja), és ezt nem lehet eltávolítani. Az objektum funkcionálitását biztosító többi komponens hozzáadható a szerkesztő Komponens menüből vagy egy szkriptből. Számos hasznos előre elkészített objektum (primitív alakzatok, kamerák stb.) is elérhető a GameObject > 3D Object menüben, lásd: Primitív objektumok.



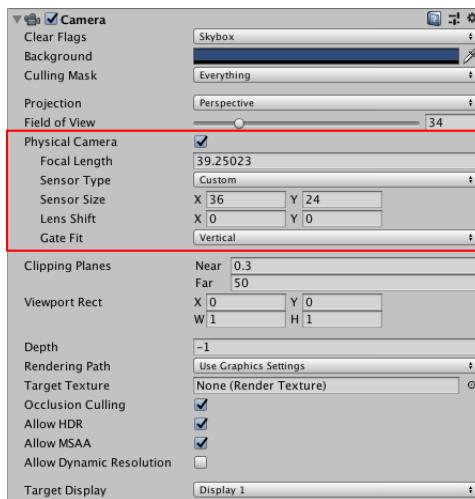
2.4. ábra. Fény GameObject



2.5. ábra. Kocka GameObject

2.2.2. Cameras (Kamerák)

Ahogy a filmekben kamerákat használnak a történet megjelenítésére a közönség előtt, úgy a kamerák is a Unityben a játék világának megjelenítésére szolgálnak a játékos számára. Mindig lesz legalább egy kamera egy jelenetben, de több is lehet. A több kamera kétjátékos osztott képernyőt biztosít, vagy speciális egyéni effektusokat hozhat létre. A kamerákat animálhatjuk, vagy fizikával vezérelheti. Gyakorlatilag minden elképzelhető a kamerákkal, és használhatunk tipikus vagy egyedi kamerákat, hogy illeszkedjen a játék stílusához.



2.6. ábra. Kamera

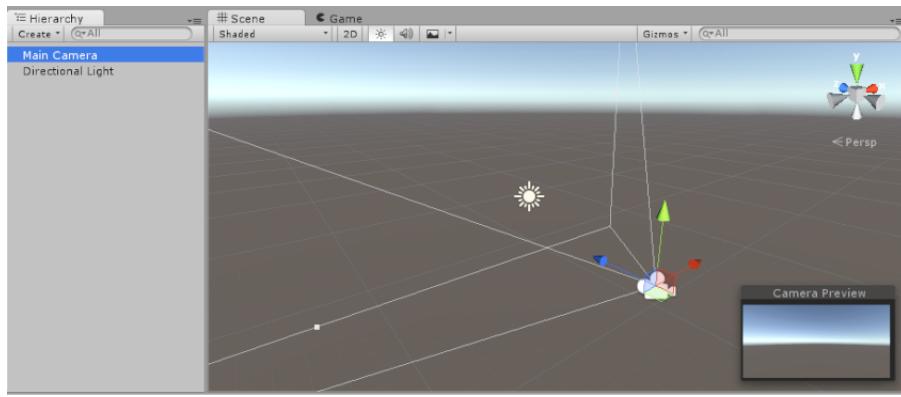
2.2.3. Scenes (Jelenetek)

A jelenetek azok, ahol tartalommal dolgozunk a Unityben. Olyan eszközök, amelyek egy játékot vagy alkalmazást vagy azok egy részét tartalmazzák. Például létrehozhatunk egy egyszerű játékot egyetlen jelenetben, míg egy összetettebb játékhoz használhatunk egy jelenetet szintenként, mindegyik saját környezettel, karakterekkel, akadályokkal, dekorációkkal és felhasználói felülettel. Egy projektben tetszőleges számú jelenet hozható létre.

Amikor létrehozunk egy új projektet, és először nyitjuk meg, a Unity megnyit egy minta jelenetet, amely csak egy kamerát tartalmaz és egy Fény objektumot (2.7. ábra). A New SCene dialog akkor nyílik meg, ha új jelenetet hoz létre a Fájl menüből: (Fájl > Új jelenet) vagy a Ctrl/Cmd + n billentyűparancs segítségével. Ahhoz, hogy egy új jelenetet hozzunk létre a New Scene dialogon, kiválasztunk egy sablont a sablonlistából, majd rákattintunk a Létrehozás gombra.

2.2.4. Prefabs (Előre elkészített objektumok)

A Unity Prefab rendszere lehetővé teszi GameObject létrehozását, konfigurálását és tárolását minden összetevőjével, tulajdonságértékével és gyermek GameObject-jével együtt újrafelhasználható eszközöként. Az előregyártott eszköz sablonként működik, amely-



2.7. ábra. Scene

ből új előregyártott példányokat hozhat létre a jelenetben. Ezt úgy kell elképzelni, hogy mi megírunk egy dolgot egyszer és azután fénymásoljuk ahányszor szükséges.

2.2.5. Animation

Unity különlegesség, hogy objektumainkra animációkat rakhatunk. Ezek az animációk lehetnek nagyon egyszerűek, mint például az ugrás, és lehetnek nagyon bonyolultak, mint például az ember mozgását jelképező animáció. Az animáció egy bizonyos triggerre (ravaszra) hívódik meg. Ha az adott esemény bekövetkezik, akkor beállíthatjuk azt, hogy mennyi idő múlva induljon el az animációnk, illetve azt is beállíthatjuk, hogy mennyi ideig fusson le az adott animáció.

2.2.6. Assets (Eszközök)

Egy nagyon hasznos és fontos megvalósítása a Unitynek, hogy egy olyan weboldalt hoztak létre ([Unity Asset Store \[Sto\]](#)), ahonnan bizonyos eszközöket letölthetünk és projektünkbe importálhatunk. Az eszközök megjeleníthetnek vizuális vagy audio elemeket a projektben, például 3D modelleket, textúrákat, sprite-okat, hangeffektusokat vagy zenét. Az eszközök absztraktabb elemeket is képviselhetnek, például színátmeneteket, animációs maszkokat vagy tetszőleges szöveges vagy numerikus adatokat bármilyen felhasználásra. Egy eszköz származhat a Unity-n kívül létrehozott fájlból, például 3D-modellból, hangfájlból vagy képből.

2.2.7. Scripts

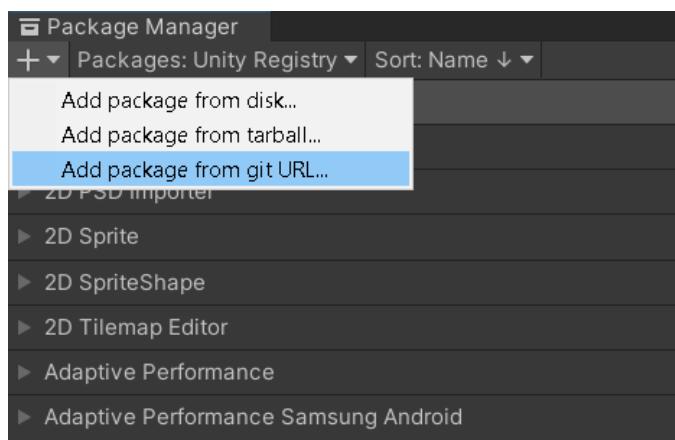
A szkriptelés elengedhetetlen összetevője az összes Unity alkalmazásnak. A legtöbb alkalmazáshoz szkriptekre van szükség reagálni a játékos bemeneteire, és megszervezni, hogy a játék során az események akkor történjenek meg, amikor kell. Ezen túlmenően a szkriptek felhasználhatók grafikus effektusok létrehozására, az objektumok fizikai viselkedésének szabályozására, vagy akár egyéni mesterséges intelligencia rendszer megvalósítására a játékban szereplő karakterek számára.

2.2.8. Multiplayer and Networking

A Unity Multiplayer fejlesztés alatt áll, az UNet pedig egy elavult megoldás, amely más megoldásokhoz képest nem ajánlott. Emiatt a játékom csak egy eszközről használható két játék számára egyidőben. A többjátékos biztosítását kódszinten biztosítottam, azáltal, hogy két Playert inicializáltam és a játék során a program számon tartja a játkosok pontjait, ahhoz, hogy a végén jelezni tudja, hogy melyik játékos nyert.

2.2.9. Google VR

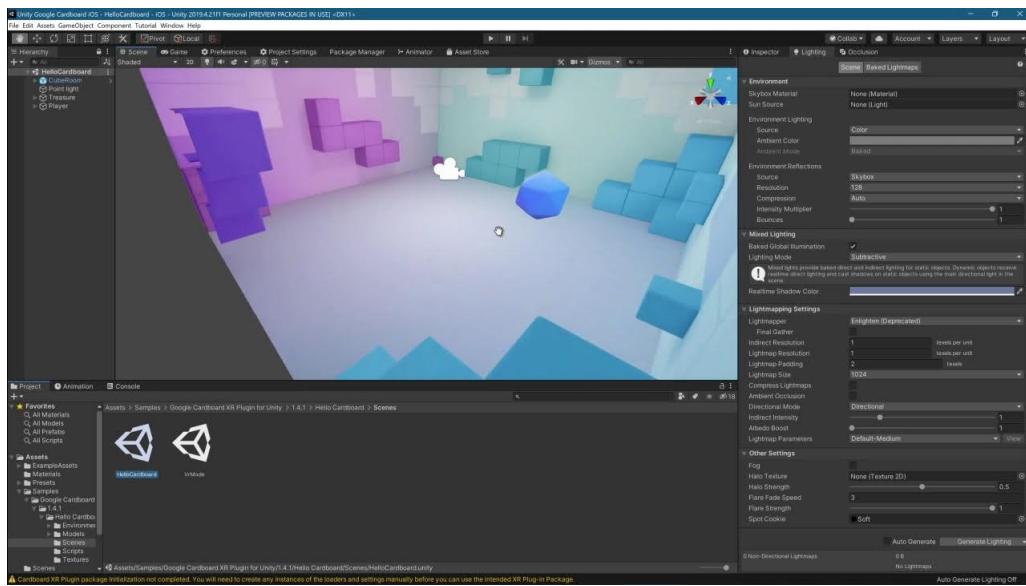
Egy virtuális alkalmazás elkészítéséhez nem elégseges az android modul telepítése, hanem még bizonyos csomagokat és beállításokat el végezni. A legfontosabb csomag, ami lehetővé teszi a VR alkalmazás az a Google Cardboard csomag, amit a Git Hubról kell beimportálni a projektbe. A csomagokat a Window->Package Manager gombra kattintva találjuk meg, innen kiválaszthatjuk, hogy git csomagot szeretnénk beimportálni (lásd a 2.8. ábrát).



2.8. ábra. Git csomag importálása

Beillesztjük az megjelenő üres sávba a következő linket: <https://github.com/googlevr/cardboard-xr-plugin> és az add gombra kattintva, már telepíti is nekünk a csomagot. A csomag telepítése után egy alap játék jelenik meg, ami által meg tudjuk figyelni, hogyan van felépítve egy virtuális játék, milyen programok futnak a háttérben, mire van szükségünk egy scene felépítéséhez, illetve erre az alapjátékra könnyedén lehet ráépíteni a saját játékunkat is. A 2.9.ábrán látható a szoba, amely megjelnik a csomag letöltése után. A játék lényege, hogy a játékos a szobát végignézve, meg kell keresse a jutalmat és miután megtalálta, az adott alakzat eltűnik és egy másik helyen jelenik meg.

A kód ami a játék jelentős részét vezérli C# -ban van megírva. A 2.10. ábrán látható a CameraPointer osztály. Az Update() függvény az minden framenel meghívódik egyszer és annyiszor fog lefutni egy másodperc alatt, ahány fpsen (frame per second) fut. Ha például egy játék 20 fpsen fut, akkor az alábbi kód 20-szor fog lefutni egy másodperc alatt. A CameraPointer osztály mindenkor figyeli, hogy a látkép közepén vagy nem távolabb mint 10 egységnyire, van-e olyan objektum amivel a felhasználó interakcióba tudna lépni. Ha talált egy olyan objektumot, akkor a program "OnPointerEnter" üzenetet küld



2.9. ábra. Google VR sablon szobája

tovább, amikor pedig kilép akkor "OnPointerExit" üzenetet közvetít, ahhoz hogy tudjuk a felhasználó mire is néz pontosan, mivel lép interakcióba vagy talált-e valamit. Ha a felhasználó egy olyan objektumot talált, amellyel interakcióba léphet (a mi esetünkben a "Jatalom" alakzattal), akkor azt ki is választhatja és "OnPointerClick" üzenetet küld. A fent említett események közül a legfontosabb az OnPointerClick, mivel ezáltal a felhasználó tudatja az alkalmazásnak, hogy megtalálta az alakzatot és ki is választotta. Miután ezt kiválasztottuk, az alakzat egy új, véletlenül kiválasztott pozícióra költözik a szobában.

2.2.10. Alkalmazás futtatása okostelefonon

Az alkalmazás okostelenen való futtatása egy hosszabb folyamat és több lépésből, konfigurációból áll. A szükséges lépések megtalálhatóak a következő linken [Quick-start for Google Cardboard for Unity \[Qui\]](#). Az alkalmazás letöltéséhez az okostelefonunkra szükséges, hogy be legyen kapcsolva az USB debugging, ahhoz hogy az editor és a számítógép észleljé az eszközünket.

```

1  using System.Collections;
2  using UnityEngine;
3
4  /// <summary>
5  /// Sends messages to gazed GameObject.
6  /// </summary>
7  public class CameraPointer : MonoBehaviour
8  {
9      private const float _maxDistance = 10;
10     private GameObject _gazedAtObject = null;
11
12     /// <summary>
13     /// Update is called once per frame.
14     /// </summary>
15     public void Update()
16     {
17         // Casts ray towards camera's forward direction, to detect if a
18         // GameObject is being gazed
19         RaycastHit hit;
20         if (Physics.Raycast(transform.position, transform.forward, out hit,
21             _maxDistance))
22         {
23             // GameObject detected in front of the camera.
24             if (_gazedAtObject != hit.transform.gameObject)
25             {
26                 // New GameObject.
27                 _gazedAtObject?.SendMessage("OnPointerExit");
28                 _gazedAtObject = hit.transform.gameObject;
29                 _gazedAtObject.SendMessage("OnPointerEnter");
30             }
31         }
32         else
33         {
34             // No GameObject detected in front of the camera.
35             _gazedAtObject?.SendMessage("OnPointerExit");
36             _gazedAtObject = null;
37         }
38
39         // Checks for screen touches.
40         if (Google.XR.Cardboard.Api.IsTriggerPressed)
41         {
42             _gazedAtObject?.SendMessage("OnPointerClick");
43         }
44     }
}

```

2.10. ábra. C# programozási nyelvben írt CameraPointer osztály

3. fejezet

Bibble

3.1. A játék bemutatása

A játékom főként a gyerekek számára volt kitalálva és létrehozva, de a magasabb korosztályba tartozó személyek is élvezhetik és tanulhatnak általa. Mivel kevés olyan játék található ami segít a gyerekek bibliai ismeretét felmérni és elmélyíteni, így úgy döntöttem, hogy kitalálok és létrehozom én a Bibble nevű játékot, ami egy társasjáték quiz kérdésekkel nehezítve. Amint elindítjuk a programot, először egy "előszobába" kerülünk (lásd 3.1. ábra), ahol elolvashatjuk a játékszabálokat, majd ezután a "JÁTÉK INDÍTÁSA" gombra kattintva a betöltödik a játékmező (lásd 3.2. ábra). [ZB02]



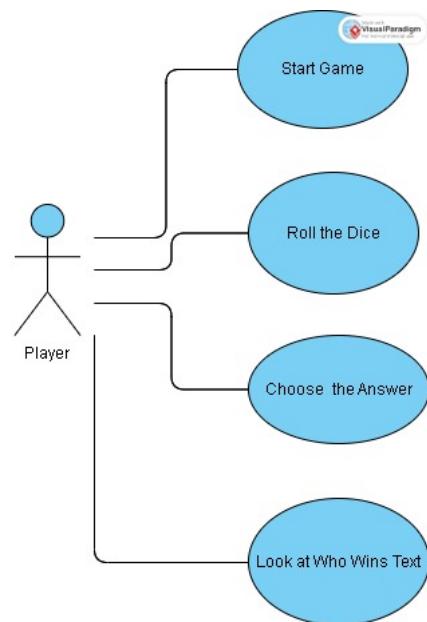
3.1. ábra. Játék előszobája

A játékot egyszerre két személy tudja játszani ugyanazt az okostelefont használva. A játékosok a kockára kattintva indítják el saját körüket amelyben egy újabb ablak ugrik fel, egy kérdés és a hozzá tartozó négy választási lehetőséggel. minden kérdésnek csak egy helyes válasza van és ha az illető játékos helyesen válaszol akkor a megjelölt válasz színe zöldre változik, ellenkező esetben pedig pirosra. Helyes válasz esetén a játékos bábuja annyit lép a táblán ahányast a játékos a kör elején dobott. Viszont ha a válasz helytelen volt akkor a bábu helyben marad és a másik játékos köre következik. Azt, hogy mikor ki van a sor könnyen követni tudjuk, mivel a játékmező szélén megjelenik a két bábu képe és felettük írja, hogy mikor melyiken van a sor.



3.2. ábra. Játékmező

A játék könnyebb átlátása érdekében segítségünkre van egy Use Case diagram is, amely bemutatja mi a játékos feladata.



3.3. ábra. Use Case Diagram

3.2. Játék elkészítése

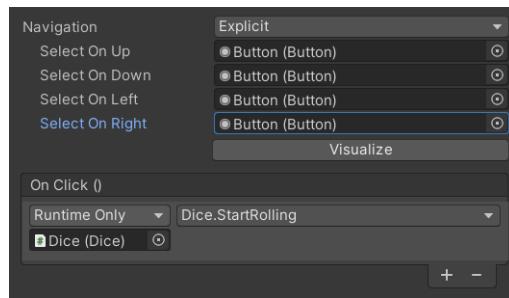
3.2.1. Controller

Mivel az okostelefont, amin a játék található be kell helyezzük egy virtuális szemüvegbe, ezért egy megoldást kellett találnom, mivel a kezem nem használhatom a gombok megnyomására a képernyőn. Ezért egy Rii márkJú távirányítót vásároltam (lásd 3.3. ábra), amit USB-vel a telefonhoz tudok csatlakoztatni és bizonyos gombokat beállítottam, hogy interakcionálni tudjak a játékkal (például az enter gomb a kiválasztáshoz, a nyilakat pedig navigációhoz a válaszok között).



3.4. ábra. Controller

Ehhez egy új csomagot kellett letöltenem a Unityn belül, az Input System-et és minden esetben ahol kommunikáció történik a program egy eleme és a felhasználó között be kellett állítanom a Navigación keresztül, hogy mikor, hogyan és milyen gombakra kell reagálnia (lásd 3.4. ábra).



3.5. ábra. Gombok navigálása

A dobókocka gurításához viszont még nincs ilyen bővitmény kitalálva a Unityn belül, így ehhez egy "Dice.cs" elnevezésű szkriptet (lásd 3.5. ábra).

```

1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.Events;
4
5  public class Dice : MonoBehaviour {
6
7      private Sprite[] diceSides;
8      private SpriteRenderer rend;
9      private int whosTurn = 1;
10     private bool coroutineAllowed = true;
11     [SerializeField] internal GameControl control;
12     [SerializeField] Transform quizPanel;
13     [SerializeField] QuizManager quizManager;
14     private GameControl player;
15     // public static GameObject player2;
16
17     // Use this for initialization
18     public void StartRolling () {
19         rend = GetComponent<SpriteRenderer>();
20         diceSides = Resources.LoadAll<Sprite>("DiceSides/");
21         rend.sprite = diceSides[5];
22         if (!GameControl.gameOver && coroutineAllowed)
23             StartCoroutine("RollTheDice");
24     }
25
26     public void OnPointerDown(PointerEventData eventData)
27     {
28         if (!GameControl.gameOver && coroutineAllowed)
29             StartCoroutine("RollTheDice");
30
31     }
32
33 }
```

A Dice osztályon belül vezéreljük a dobókocka körül történő eseményeket. Az OnPointerDown függvényen keresztül figyeljük, ha történt gomblenyomás és ha igen, akkor egy új korutint indít, meghívja a RollTheDice függvényt a kocka megforgatásához. A dobókocka hat oldalból áll és a program véletlenszerűen választja ki, hogy melyik oldalát mutassa. Amelyik oldalára került a kocka, a játékos annyi pontot fog kapni ha helyesen válaszol majd a kérdésre. Ezután betöltődik a Quiz kérdés ablaka, majd meghívódik a QuizManager osztály.

3.2.2. Játékmező

Mivel a gyerekek figyelmét szerettem volna inkább megragadni a játék kinézetével, így egy színesebb, gyerekesebb, érdekesebb játékmezőt szerettem volna megalkotni (lásd 3.2. ábra). A játékmező alapjában véve egy megszerkeztetett kép, amin négy kép látható. Ez a négy kép az isteni szeretet négy legnyilvánvalóbb szeretet kinyilvánítását ábrázolja: amikor megeremtette az embert, amikor Jézust leküldte a földre, Jézus kereszthalála bűneinkért és amikor majd személyesen találkozhatunk Vele. A bábuk Ádámot és Évát ábrázolják, akik egy hosszú utat tesznek meg a teremtéstől fogva a Mennybe való bejutásig. A bábukat nem kell a játékosnak mozgatnia, mivel a quiz kérdésre való válaszolás

```

34     Unity message + 0 references
35     private void OnMouseDown()
36     {
37         if (!GameControl.gameOver && coroutineAllowed)
38             StartCoroutine("RollTheDice");
39     }
40 
41     0 references
42     private IEnumerator RollTheDice()
43     {
44         coroutineAllowed = false;
45         int randomDiceSide = 0;
46         for (int i = 0; i <= 20; i++)
47         {
48             randomDiceSide = Random.Range(0, 6);
49             rend.sprite = diceSides[randomDiceSide];
50             yield return new WaitForSeconds(0.05f);
51         }
52 
53         GameControl.diceSideThrown = randomDiceSide + 1;
54 
55         quizPanel.gameObject.SetActive(true);
56 
57         quizManager.generateQuestion();
58 
59         //quizManager.options[valasztottOpcio].GetComponent<AnswerScript>().isCorrect == true
60         while(quizManager.kivalasztottOpcio == -1)
61         {
62             yield return new WaitForSeconds(0.05f);
63         }
64 
65         Debug.Log("Kivalasztott opcio = " + quizManager.kivalasztottOpcio);
66 
67         yield return new WaitForSeconds(1f);
68 
69         quizPanel.gameObject.SetActive(false);
70     }

```

```

71     if (quizManager.options[quizManager.kivalasztottOpcio].GetComponent<AnswerScript>().isCorrect)
72     {
73 
74         if (whosTurn == 1)
75         {
76             //GameControl.MovePlayer(1);
77             control.MovePlayer(1);
78         }
79         else if (whosTurn == -1)
80         {
81             //GameControl.MovePlayer(2);
82             control.MovePlayer(2);
83         }
84     }
85 
86     quizManager.kivalasztottOpcio = -1;
87 
88     whosTurn *= -1;
89     coroutineAllowed = true;
90 }
91 
92 }
93 
```

3.6. ábra. C# programozási nyelvben megírt Dice osztály

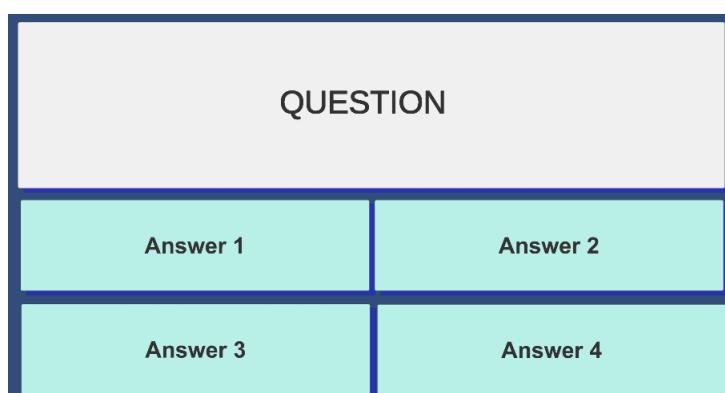
után a program maga dönti el, hogy a játékos megérdemli a pontot vagy sem a válaszának helyessége alapján. Maga a játékmezőn a játékosnak igazából csak az a feladata, hogy figyelje mikor van rajta a sor (a játék ezt is jelzi) és guritson. A tábla szélén jobb oldalon szintén megjelenik a két játékos bábuja és felettünk minden jelzi egy szöveg "TE SOROD", hogy éppen melyik felhasználó következik. Ha az egyik játékos beér a célba, akkor a játék leáll és a képernyőn megjelenik egy utolsó üzenet "1/2. JÁTÉKOS NYERT".



3.7. ábra. Játék vége kiírás

3.2.3. QuizPanel

A játékban a kviz kérdéssel felugró ablakot neveztem el QuizPanelnek. Ez az ablak felel a kviz kérdés és választási lehetőségek kiíratásáért, illetve elődönti, hogy a játékos által kiválasztott opción helyes-e. Amint a a 3.8.ábrán is látható, a játékosnak négy válasz közül kell kiválasztania egyet. Ha a válasza helyes akkor zöldre változik a négyzet színe, ha pedig helytelen akkor piros lesz. Ezután újra a Játékmező kerül előtérbe, hogy figyelni tudjuk a soros következő lépést. A játékban összesen 33 darab kérdés van előkészítve és a kérdések nem ismétlődnek.



3.8. ábra. QuizPanel kinézete

A QuizPanelben történő eseménykért a QuizManager osztály (lásd 3.9.ábra) felelős. A kérdések a válaszaikkal együtt vannak eltárolva. Amikor meghívódik egy adott kérdés,

akkor meghívódnak vele együtt a hozzá tartozó válaszok és a helyes válasz száma is, ezt hajtja végre a generateQuestion függvény. A válaszok sorban vannak eltárolva, illetve a CorrectAnswer változóban pedig a helyes válasz indexe, ami szerint azonosítani tudja a program a megadott válasz helyességét. Ezt a műveletet a SetAnswers függvény végzi el, veszi a felhasználó által kiválasztott opció indexét és összehasonlítja a tényleges helyes válasz indexével. Ha a két szám megegyezik, akkor a gomb színét zöldre változtatja, ellenkező esetben pedig pirosra változik a gomb színe.

```

1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using System.IO;
5  using UnityEngine;
6  using UnityEngine.UI;
7
8  public class QuizManager : MonoBehaviour
9  {
10    public int k = 0;
11    public List<QnA> QnA;
12    public GameObject[] options;
13    public int currentQuestion;
14
15    public Text QuestionTxt;
16
17    public int kivalasztottOpcion = -1;
18    private Button theButton, theButton2;
19    private ColorBlock theColor;
20
21    private void Start()
22    {
23        generateQuestion();
24    }
25
26    public void correct()
27    {
28        //QnA.RemoveAt(currentQuestion);
29        generateQuestion();
30    }
31
32

```

```

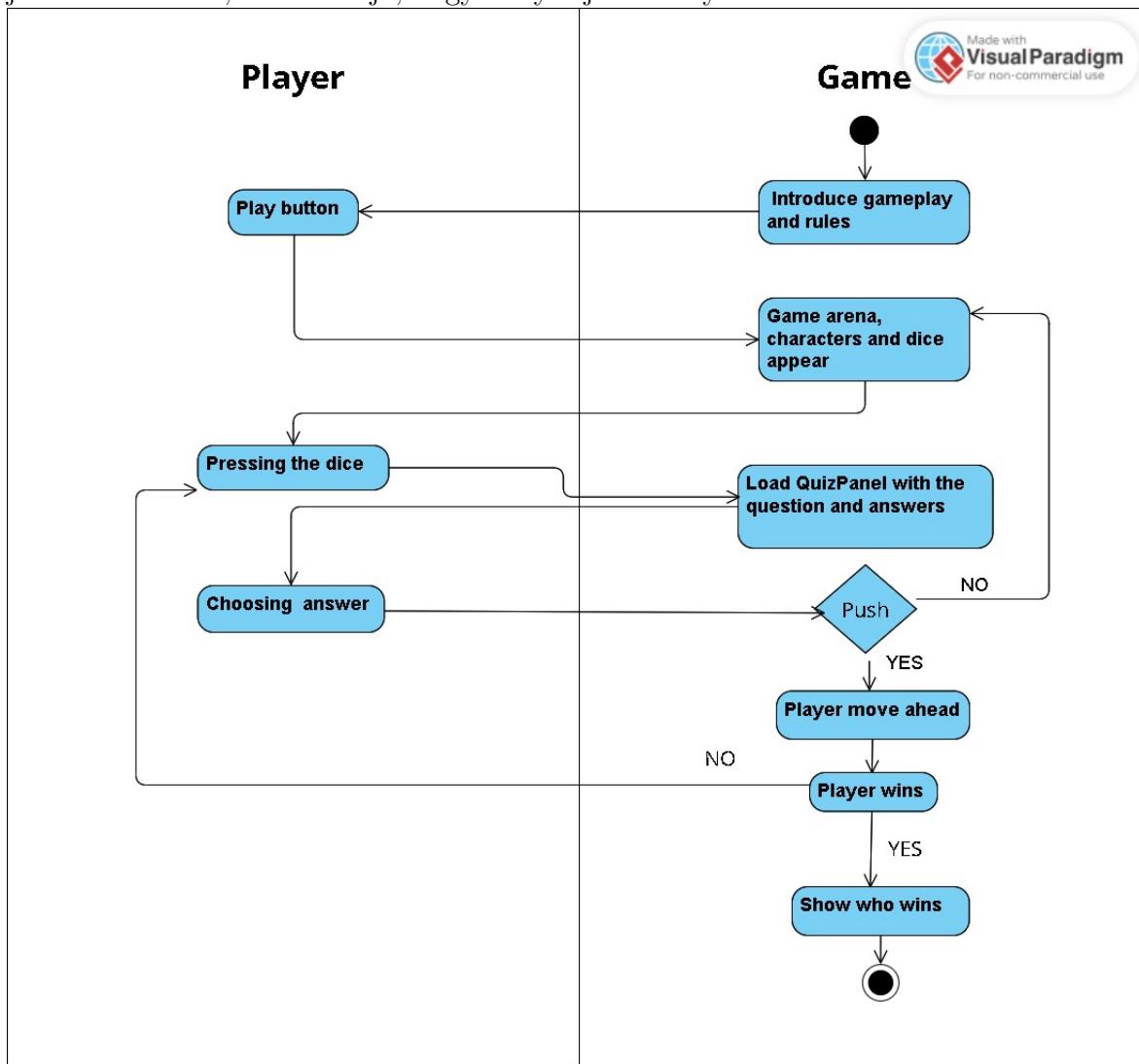
33    void SetAnswers()
34    {
35        for(int i=0; i<options.Length; i++)
36        {
37
38            options[i].GetComponent<AnswerScript>().isCorrect = false;
39            options[i].transform.GetChild(0).GetComponent<Text>().text = QnA[currentQuestion].Answers[i];
40            //options[i].GetComponent<AnswerScript>().answerID = i;
41            theButton = options[i].GetComponent<Button>();
42            theColor = options[i].GetComponent<Button>().colors;
43            theColor.pressedColor = Color.red;
44
45            if (QnA[currentQuestion].CorrectAnswer == i + 1)
46            {
47                options[i].GetComponent<AnswerScript>().isCorrect = true;
48                theColor.pressedColor = Color.green;
49                theButton.colors = theColor;
50            }
51            else
52            {
53                theColor.pressedColor = Color.red;
54                theButton.colors = theColor;
55            }
56
57        }
58
59    }
60
61    public void generateQuestion()
62    {
63        currentQuestion = k;
64        //currentQuestion = UnityEngine.Random.Range(0, QnA.Count);
65
66        QuestionTxt.text = QnA[currentQuestion].Question;
67
68        SetAnswers();
69        k += k + 1;
70
71    }
72

```

3.9. ábra. C# programozási nyelvben megírt QuizManager osztály

3.2.4. Játék lefolyása

A játék legjobb működése érdekében több program megírásához is szükség volt, aminek könyebb megírásához Victor Brusca [Bru16] Advanced Unity Game Development című könyve segített. Az Activity diagrammon (lásd 3.10.ábra) pontosabban is látható, hogy milyen sorrendben hívódnak meg a függvények ahhoz, hogy a szükséges funkciókat véghez vigyék. Vannak olyan funkciók amelyekhez a játékos közreműködéséhez is szükségünk van. Ilyen például az indító gomb megnyomása, a dobókocka dobása és a szerinte helyes válasz kiválasztása. Emellett vannak olyan funkciók is amelyeket a program maga végez ez, mint amikor betölti a játék előszobáját, betölti a játékmezőt a bábukkal és dobókockával együtt, betölti a kviz ablakot a kérdéssel és opciókkal együtt, a háttérben a program leellenőrzi ha a kiválasztott opció helyes és ha igen akkor tovább léptetni a bábut, ellenkező esetben csak újra betölteni a játékmezőt és ha a játék a végéhez ért, azaz ha az egyik játékos célba ért, akkor kiírja, hogy melyik játékos nyert.



3.10. ábra. Játék Activity Diagrammja

Összefoglaló

Dolgozatomban egy OpenGL alapú okostelefonos alkalmazás megalkotásával foglalkoztam, amelyet különböző programozási technológiák segítségével valósítottam meg. Először ismertettem a Unity Hub felület szerepét és ennek részeit, illetve hogy milyen eszközök állnak rendelkezésünkre és hogyan tudunk bizonyos csomagokat importálni git-hubról a projektbe ahhoz, hogy egy virtuális szemüvegre tudjunk játékot írni. Megfigyelhettük, hogy mennyire megkönnyítik ezek az eszközök a programozó munkáját, illetve mennyire különlegesebbé teszik a programot és bepillantást nyerhettünk hogy hogyan is néz ki egy színhely. Miután elméletben minden szükséges eszközt megvizsgáltunk és importáltunk, rátérhettünk a játék bemutatására. Ismertettem célomat a program létrehozásával kapcsolatban, kinek és miért lett kitalálva ez a játék. Ezután bemutattam a játék felépítését, melyet négy nagyobb alapegységre bontottam a könnyebb átláthatóság érdekében: Controller, Játékmező, QuizPanel, Játék lefolyása, ahhoz hogy átláthatóbb legyen mindegyik rész funkciója és működése. Megfigyelhettük ezek hogyan kapcsolódnak egymáshoz kódszinten is és hogyan alkotnak együtt egy működőképes és használható játékot.

Jövőbeli terveimet illetően szeretnék jobban elmerülni a virtuális játékok fejlesztésében és játékomat tovább fejleszteni további funkciókkal, mint újraindítás gomb, előt pontok megtekintése, illetve bevezetni a játékba az osztott rendszereket is ahhoz, hogy az alkalmazást a játékosok két különböző okostelefonról is tudják egyidejűleg használni. Továbbá érdemesnek tartom a játék feltöltését olyan platformra, ahol bárki elérheti és ingyenesen használhatja.

<https://github.com/juhosimola21/Allamvizsga.git>

Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőnek, Dr. Kovács Lehel Istvánnak, aki észrevételeivel és tanácsaival segítette munkámat, valamint rendelkezésemre bocsátotta a dolgozat elkészüléséhez szükséges szakirodalmat. Köszönöm szüleimnek, öcsémnek és húgomnak és drága férjemnek, hogy szeretetükkel támogattak egyetemi tanulmányaim alatt, valamint köszönöm türelmüket, segítségszüket és a pozitív hozzáállásukat. Köszönöm a barátaimnak, Anitának, Ingridnek és Krisztinának, hogy együtt éltük át az egyetem jó, bár néha nehéz napjait. Végül de nem utolsó sorban szeretném hálásan megköszönni a Marosvásárhelyi Sapientia Erdélyi Magyar Tudományegyetem Informatika Kara valamennyi oktatójának, dolgozójának azt a lelkismeretes munkáját, amivel a hallgatók képzését, tanulását és a versenyképes tudás megszerzését támogatják.

Ábrák jegyzéke

1.1. Oculus Meta Quest	10
1.2. A szemüvegen túl	11
2.1. Unity Hub fontosabb részei	12
2.2. Unity Editor csomagjai és méretei	13
2.3. Unity Editor részei	13
2.4. Fény GameObject	15
2.5. Kocka GameObject	15
2.6. Kamera	16
2.7. Scene	17
2.8. Git csomag importálása	18
2.9. Google VR sablon szobája	19
2.10. C# programozási nyelvben írt CameraPointer osztály	20
3.1. Játék előszobája	21
3.2. Játékmező	22
3.3. Use Case Diagram	22
3.4. Controller	23
3.5. Gombok navigálása	23
3.6. C# programozási nyelvben megírt Dice osztály	25
3.7. Játék vége kiírás	26
3.8. QuizPanel kinézete	26
3.9. C# programozási nyelvben megírt QuizManager osztály	27
3.10. Játék Activity Diagrammja	28
F.1.1 Alkoss virtuális valóságot!	33

Irodalomjegyzék

- [AK97] Galambos Adrienn and S Nagy Katalin. Virtuális valóság. 1997.
- [Bru16] Victor G Brusca. Advanced unity game development, 2016.
- [Dzs16] Dzséjt. A virtuális valóság káros hatásai. 2016.
- [Ian01] Parberry Ian. Introduction to computer game programming with directx 8.0. 2001.
- [Man] Unity user manual.
- [Ope99] Opengl programming guide: The official guide to learning opengl, version 1.2 (3rd edition). 1999.
- [Qui] Quickstart for google cardboard for unity.
- [Sto] Unity Asset Store.
- [ZB02] G. Jakab. Z. Balogh. Terep, karakterek, és effektusok számítógépes játékokban. bme iit, tdk dolgozat. 2002.

Függelék

F.1. Alkoss virtuális valóságot!



F.1.1. ábra. Alkoss virtuális valóságot!

Az ember egyetlen cselekvésterületet sem hódított meg annyira, mint amire a VR lehetőséget kínál. Persze ezek a lehetőségek nem aktualizálhatóak, mert lényegében azon a félreértésen alapulnak, hogy a VR téridejében szokásos események és magának a VR-nek a konstrukciója két "valóságban", téridőben történik, ám ez korántsem így van. E fikció szerint a teremtő ember (már ez is blaszfémikus) először maga kialakítja az általa "belakni" kívánt virtuális világot, majd elmerül benne, mint egy "álomban", és ott a (bel-ső) téridőben az adott "világ" játékszabályainak megfelelően cselekszik. Csakhogy maguk ezek a leírások sem felelnek meg igazán másnak, mint egy teljesen hétköznapi "objektív" megfigyelőnek, azaz ez egy egyszerű, "evilágbeli" fantázia leírása tudományos technológiai szavakkal. Magának az interaktív médiának a leírása sem lehet egyértelmű a valóságfogalom fenti értelmezésében. A konstruktivizmusban az igazság és az objektív megismerés kérdése zárójelbe kerül, és helyettük a kogníció (kör)folyamata és a valóságmodellek konstruktív szabályai kerülnek az érdeklődés középpontjába, "valamint a szubjektumoknak és a társadalmi csoportoknak a valóságkonstrukció mögött álló szükségletei, érdekei és motivációi". (Hauptmeier, Rusch, 17. old.) A megértést pedig két dolog, a szocializáció és a konvenciók teszik lehetővé, más szóval a szubjektumok belső struktúrájának homológiája. A kogníció folyamata műveletileg zárt, műveletei pedig maga a tapasztalás. (u. o. 29.

old.) Megközelíthetjük eszerint a VR-t mint valami olyat, amelybe az egyes szubjektumok "belefejlődhetnek", kialakulhat a technikának és a modern értelemben vett embernek egy olyan összefonódása, amely előállítja magának a virtuális teret és időt. De nyilvánvaló, hogy ha meg is őrizzük a személy(iség) fogalmát, vagy a biológiai és társadalmi emberét, akkor is szükségszerűen beszélnünk kell az így létrejövő VR-személyiség autonómiájáról. Azaz mindaddig csak korlátos autonómiájáról, amíg e személyiség legalábbis részben le van horgonyozva abban a társadalmi "világban", amelyben megszületett, más világ pedig számunkra egyszerűen elképzelhetetlen. (Stone, 298. old.) - [\[AK97\]](#)