Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Zetelaki Dániel**
Neptun-kód:	**MR47MG**
Ágazat:	**Infokommunikáció**
E-mail cím:	**zetelaki.d@gmail.com**
Konzulens(ek):	**Dr. Sonkoly Balázs** **Dóka János** **Nagy Bálint György**
E-mail címe(ik):	**sonkoly@tmit.bme.hu** **doka.janos@vik.bme.hu** **nagy.balint.gyorgy@vik.bme.hu**

Téma címe: Többjátékos AR játék készítése

Feladat

Az önálló labor keretein belül, a feladatomnak egy többjátékos AR, azaz Augumented Reality (Kiterjesztett Valóság) alkalmazás készítését határoztuk meg a konzulenseimmel, melynek témájára saját ötlettel áltam elő. Egy kosárlabda témájú játék készítését gondoltam ki, majd valósítottam meg, melyben két játékos a valóságban tetszőleges helyszínen lehelyezhető, virtuális kosárpalánkra tudja dobálni a virtuális kosárlabdákat.

2022/2023. 2. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1 Bevezető

Ez a beszámoló az AR alapú többjátékos kosárlabda játékról szól, amelyet a Unity játékmotor[1] segítségével készítettem, IOS platformra. A játék célja, hogy két játékos virtuálisan kosárlabdázzon egymással valós időben.

Az AR technológia lehetővé teszi, hogy a valódi világban valós időben virtuális objektumok jelenjenek meg kiterjesztett valóságként, és azokkal interakcióba léphessünk. Az AR alkalmazások egyre elterjedtebbek, és számos területen hasznosíthatók, beleértve az oktatást, az ipart, a turizmust, a marketinget és a játékfejlesztést.

A játékom többjátékos részét a MIRROR[2] nevű Unity plugin segítségével valósítottam meg, amely lehetővé teszi a többfelhasználós játékok alapjának egyszerűbb megvalósítását, továbbá a játékosok és virtuális objektumok pozíciójának, valamint interakcióinak szinkronizációját. A játéktér (vagyis a koordinátarendszer, amiben a játékosok mozognak) szinkronizációjához, a konzulenseim által javasolt QR kódos szinkronizációt alkalmaztam, amely biztosítja, hogy a virtuális objektumok azonos pozícióban legyenek a valós világban is, minden játékos számára.

Ezt a feladatot én találtam ki és valósítottam is meg. A célom elsősorban nem is a felhasználói játékélmények maximalizálása volt (mint az a játékoknál megszokott), hanem leginkább a környezet használatának - , és a témakörhöz tartozó készségek elsajátítása volt.

1.2 Elméleti összefoglaló

A projektem felépítés szempontjából a legegyszerűbben négy főbb részre bontható.

Az első rész a projekt felépítése volt Unityben ami egy olyan videójáték-motor, mely többek között lehetőséget ad 3 dimenziós játékok készítésére is. A felépítés alatt az alkalmazás vázának megalkotását értem, melybe beletartoznak például a játékban használatos objektumok elkészítése, a felhasználói felület megtervezése, valamint a Unity környezet helyes beállítása. Mivel én többjátékos AR alkalmazás készítéséhez akartam használni a Unity-t, ezért le kellett töltenem néhány ezt lehetővé tevő csomagot, ami a Unityben egyszerű megoldható.

A projekt Unityben való elkészítése után, szükséges volt az egyjátékos játék logikáját is megalkotni, melyet C# szkriptek segítségével hoztam létre. A kódot a Rider nevű integrált fejlesztési környezetben írtam meg.

Miután elkészült az egyjátékos alkalmazás, a következő feladatom a többfelhasználós mód megvalósítása volt, amihez a már korábban említett Mirror plugin használtam. Ebben a lépésben főleg a meglévő egyjátékos módú kódjaimat kellett kiegészítsem, hogy többjátékos módban is megfelelő működésűek legyenek.

Ezek után az utolsó lépés a projekt build-elése volt. Mivel én IOS-re fejlesztettem az alkalmazásomat, ezért a build-elés után még a Mac OS X operációs rendszeren futó XCode alkalmazást is használatba kellet vegyem, mivel a Unity projektet csak egy XCode projekt létrehozásán és futtatásán keresztül lehet csak IOS-es készülékekre feltelepíteni.

---

[1] A Unity egy platformokon átívelő játékmotor, amely háromdimenziós (3D) és kétdimenziós (2D) játékok, valamint interaktív szimulációk és egyéb élmények létrehozására használható.

[2] A Mirror egy magas szintű hálózati könyvtár a Unity számára, amelyet a könnyű használatra és a siker valószínűségére optimalizáltak.

1.3 A munka állapota, készültségi foka a félév elején

A félév elején már rendelkeztem némi alapszintű ismeretekkel AR alkalmazások fejlesztése terén, köszönhetően az előző félévben, a témalaboratórium keretein belül készített UFO AR játékomnak. Ennek következtében, immár magabiztosabb tudásra tettem szert a Unity használatát illetően, és kellő ismereteket is szereztem az AR alkalmazás alapokat tekintve.

A téma szempontjából, a konkrét feladat maga még nem létezett, ezt, ahogy már korábban megjegyeztem, én találtam ki a félév során.

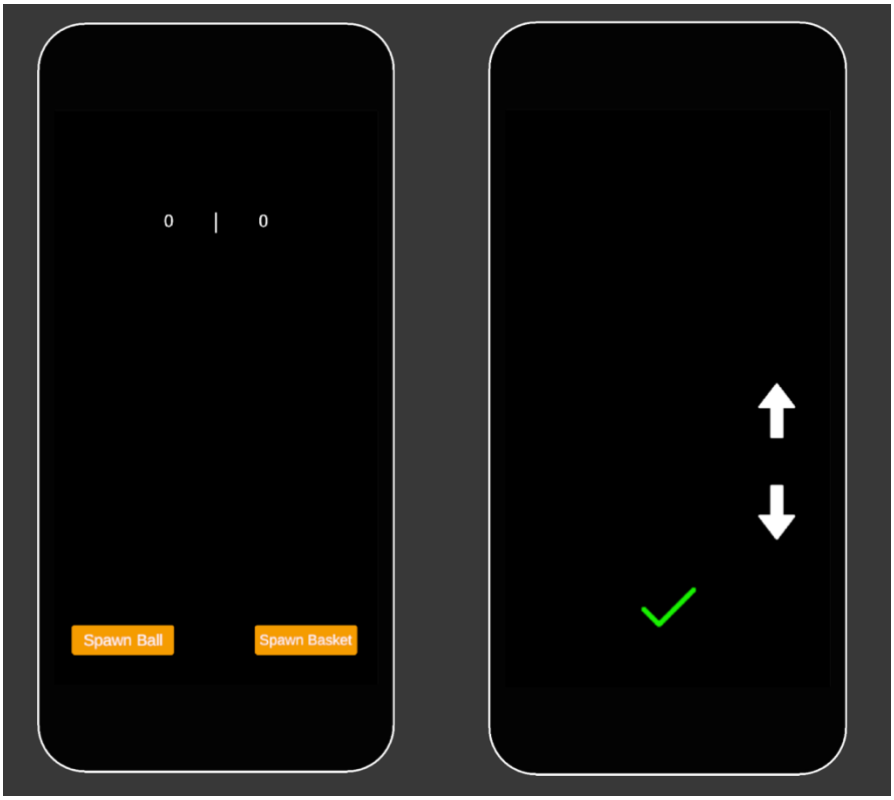
2. Az elvégzett munka és eredmények ismertetése

2.1 Egyjátékos mód elkészítése

A feladatom kitalálása után, az első mérföldkő az egyjátékos módú játék elkészítése volt.

Először is létrehoztam a Unity projektet, majd beállítottam a megfelelő paramétereket az IOS-re történő fejlesztéshez szükséges, és megteremtettem a szükséges AR környezet az AR kamera, és az AR munkamenet létrehozásával.

Ezeket után a felhasználói felület (UI) kialakítása következett. A cél egy egyszerű, letisztult, de könnyen érthető dizájn megalkotása volt.

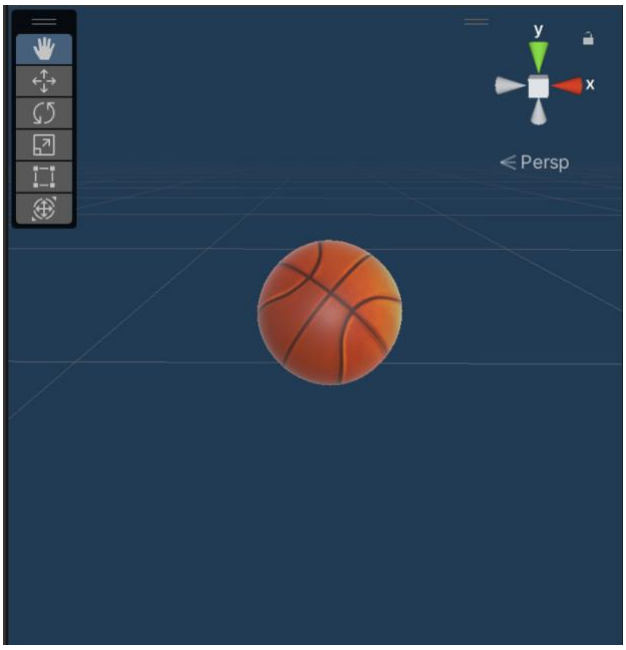


1. ábra: felhasználói felület

Ahogy az *1. ábra* ábrán látható, a bal oldali telefonon található két gomb, nevezetesen a *Spawn Ball* és a *Spawn Basket*, melyek a kosárlabdák, és a kosárpalánk lehelyezésére szolgálnak. Ezenfelül még megjelenítésre kerül egy eredményjelző tábla is, mely a két játékos által szerzett pontok számát mutatják.

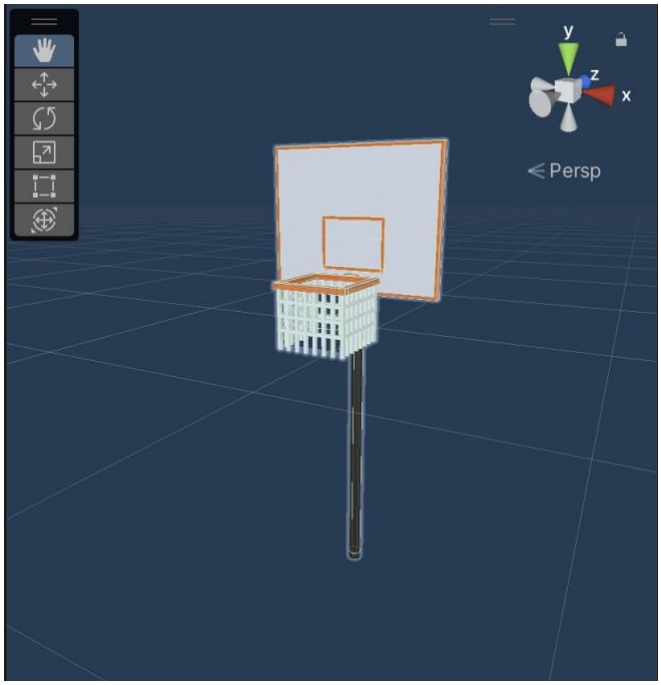
A jobb oldali telefonon egy pipa alakú gomb, valamint fel-, és lefele mutató nyilak helyezkednek el. A nyilak funkciója a kosárpalánk könnyű fel-, és lefele történő pozícionálása, a pipa pedig egy „OK” gombnak felel meg, mellyel véglegesíthetjük a változtatásainkat.

A felhasználói felület megalkotása után mindenekelőtt létre kellett hozzak egy kosárlabda-, és egy kosárpalánk objektumot.



2. ábra: a virtuális kosárlabda

A *2. ábra*ábrán a játékban használt kosárlabda látható, mely egy egyszerű háromdimenziós gömb objektum, amire ráhelyeztem egy internetről, kép formájában letöltött kosárlabda textúrát. Az objektumhoz a textúrán kívül még létrehoztam egy úgynevezett *Physics Material*-t, ami a megfelelő beállítások után, felelős az kosárlabda valósághű funkcionalitásáért, tehát a rugalmasságáért és a pattogásáért.



3. ábra: a virtuális kosárpalánk

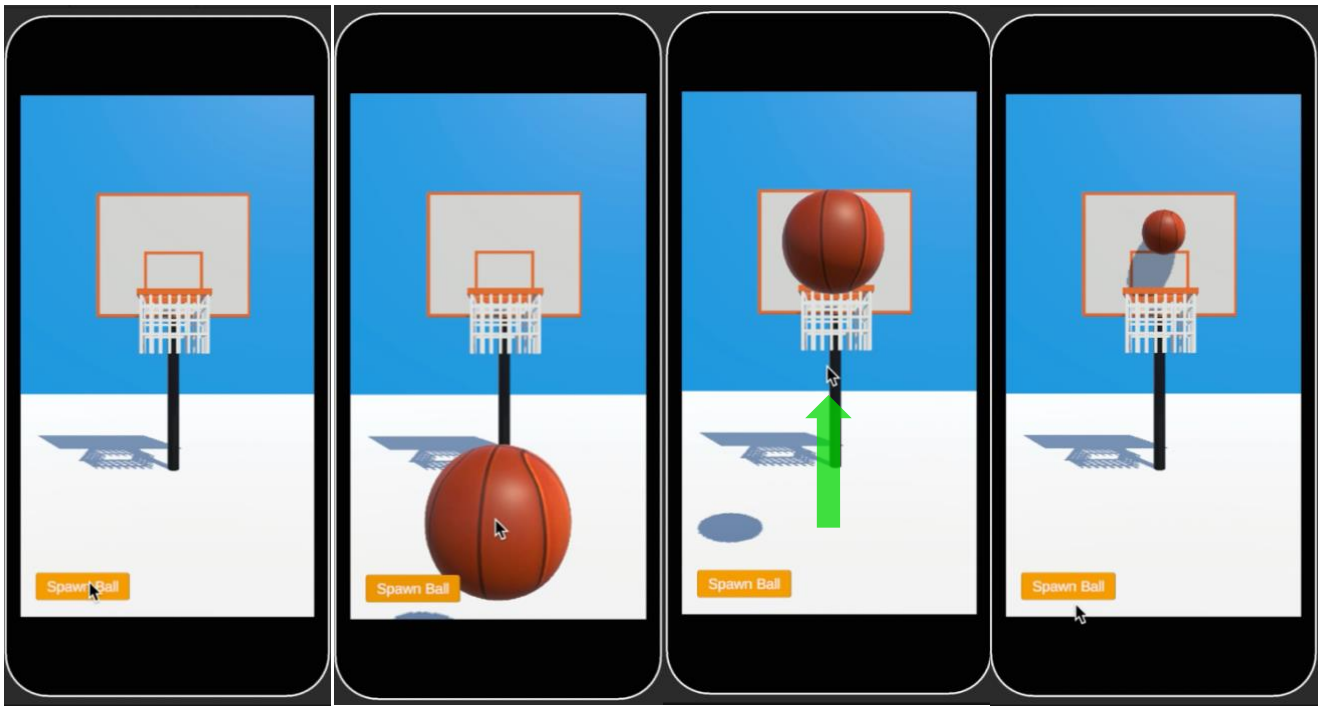
A *3. ábra*ábrán a kosárpalánk objektum jelenik meg, melyet a Unity-ben alapértelmezetten

használható objektumokból, vagyis kockákból, síkokból és hengerekből építettem fel. Ennek az egyszerű dizájnnak köszönhetően, a labdák ütközése a kosárpalánkkal sokkal kevesebb számítást vesz igénybe (a lényegesen egyszerűbb alakzatú „ütközők” által), mint ha egy bonyolultabb alakú, kerek gyűrűvel rendelkező kosárpalánk objektumot használtam volna az internetről. A kevesebb számítás, pedig végeredményben kevesebb üzenetet és kisebb késleltetési időt jelent a többfelhasználós módú játéknál.

A játékhoz szükséges objektumok létrehozása után a játék logikájának megalkotása következett.

Elsőként a labda eldobásának mechanikáját készítettem el, melyhez inspirációt Alexander Zotov játékfejlesztő youtube videójából [I] merítettem. A szkript működése két részre bontható. Az egyik, a labda eldobásának erősségét szabályozza annak függvényében, hogy mennyi ideig tartjuk lenyomva a képernyőt, a másik rész pedig, a labda repülésének irányáért felelős, amely az ujjunkkal a képernyőn húzott vektor irányát használja fel az eldobási irány kiszámításához. Ezen felül még tartozik a labdához egy egyszerű szkript, ami csupán azért felel, hogy ha a *Spawn Ball* gombot lenyomjuk, akkor a játékos (valóstérbeni telefonja) elé lehelyezi a labdát.

A labda lehelyezését és eldobását a ***4.*** ábra ábrán látható képsorozat demonstrálja.



4. ábra: labda eldobása

Ezt követően a palánk lehelyezéséért felelős szkriptet írtam meg, melynek megoldására szintén egy youtube videóból találtam ötleteket [II]. A szkriptem lényege, hogy ha a játékos rányom a *Spawn Basket* gombra, akkor utána tetszőleges helyre lehelyezhesse a palánkot. A működés első lépése a játékos környezetének, illetve elsősorban az őt körülvevő sík területek feltérképezése (plane detection), melyet egy beépített AR alapú Unity függvény hajt végre. Ha már rendelkezésre áll feltérképezett terület, akkor erre nyomva a telefon kijelzőjén, az alkalmazás a megcélzott pontra helyezi a palánkot egy, szintén AR alapú Unity osztály az ARRaycastManager segítségével[3]. Miután lehelyezésre került, először az ujjunkkal tudjuk a síkkal merőleges irányba

[3] Az ARRaycastManager lehetővé teszi, hogy meghatározzuk, hol metszi egy sugár (amelyet a képernyőből kifele, a kamera irányába mutat) a követhető tárgyat vagy síkot.

mozgatni a palánkot, majd ezután függőleges irányba a korábban említett fel-, és lefele mutató nyilak segítségével. A palánkhoz a lehelyezési és mozgatási szkripteken kívül még írtam egy pontszámító szkriptet is, mely érzékeli, ha a gyűrűn áthalad egy objektum, vagyis beleesik a kosárba, azonban ezen a pontom még nem implementáltam a pontszám megjelenítését, mivel ez csak a kétjátékos módban találtam relevánsnak.

2.2 Mirror megismerése

Az egyjátékos mód elkészítése után a következő kihívás a többjátékos mód megvalósítása volt, a Mirror könyvtár használatával. Mivel még nem volt semmilyen tapasztalom többfelhasználós játék készítésében, így pedig a Mirror használatában sem, ezért mindenekelőtt meg kellett ismerjem ezt a környezetet. Ezt elsősoron a Mirror dokumentációjának [III] végig olvasásával tettem, de ezen felül néztem még néhány youtube tutorialt [IV] [V] is, hogy könnyebben megértsem a működését.

A Mirror egy olyan nyílt forráskódú hálózati megoldás, amely lehetővé teszi, a többjátékos mód egyszerű megvalósítását az alkalmazásokban. Számos osztályt és komponenst tartalmaz, amik közül most az alkalmazásomhoz elengedhetetleneket fogom részletezni.

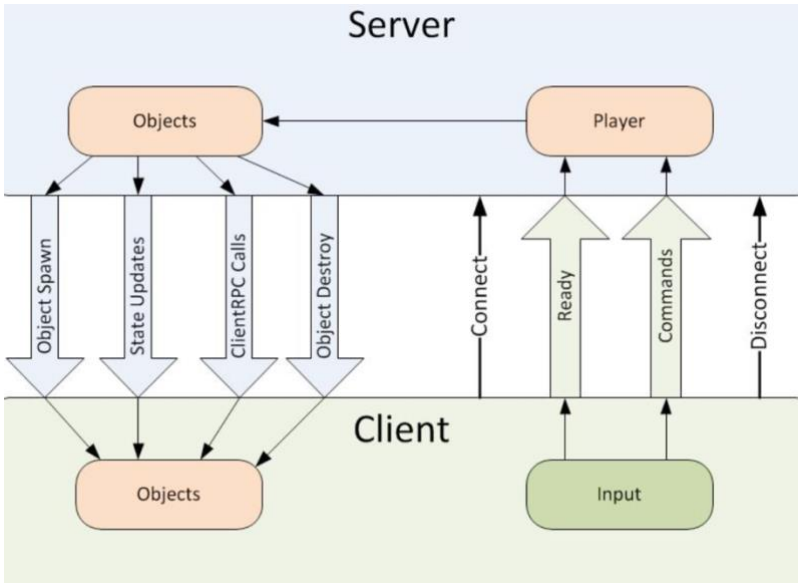
Az egyik legfontosabb komponens, a **Network Manager** (azaz hálózati menedzser). Ez kezeli a többjátékos mód hálózati aspektusait. Rengeteg hasznos funkciót foglal egy helyre, mint például a játék-állapot kezelés, objektumok lehelyezése a szerveren, a szerver és a kliensek kialakítása, valamint az ezek közötti üzenetváltások lebonyolítása.

A következő komponens a **Network Transform**. Ez felel a hálózatba kapcsolt játékobjektumok pozíciójának, elforgatásának és egyéb tulajdonságainak szinkronizációjáért a hálózaton keresztül.

A Network Transform megfelelő működéséhez azonban elengedhetetlen a **Network Identity** (azaz hálózati azonosító). Ez a komponens a Unity hálózati magas szintű APIjának középpontjában áll. Ez vezérli a játékobjektum egyedi identitását a hálózaton, és ezt az identitást használja arra, hogy a hálózati rendszer tudjon a játékobjektumról, valamint megtudja különböztetni más objektumoktól.

Egy másik elengedhetetlen komponens a **Network Behaviour**. Ez egy osztály, melyből, ha leszármaztatjuk saját szkriptünket, akkor rendkívül könnyen kezelhetjük a Network Identityvel rendelkező objektumokat a játékban. Ez az osztály olyan attribútumokat állít rendelkezésünkre, melyekkel megjelölhetjük, hogy egy adott műveletet hol szeretnénk végrehajtani. Az ilyen típusú magas szintű műveleteket néha **Remote Procedure Call**-nak (**RPC**, azaz távoli eljáráshívás) nevezik. A hálózati rendszerben kétféle RPC létezik: a Command-ok (parancs) - amelyeket a kliensről hívnak és a kiszolgálón, vagyis a szerveren futnak; és a ClientRpc hívások - amelyeket a szerveren hívnak és az klienseken futnak.

A Mirror dokumentációjában található ***5.*** ábra ábra mutatja be a távoli műveletek irányait.



5. ábra: directions of remote actions

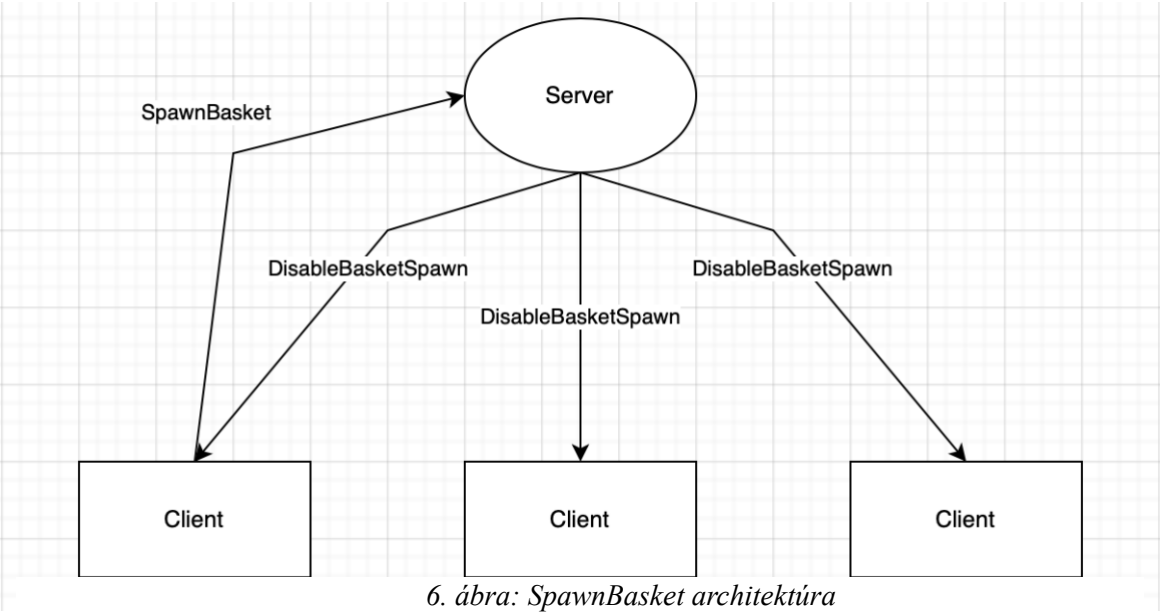
Az RPC-ken kívül még lehetőség van saját, alacsony szintű hálózati üzenetek létrehozására is. Ehhez létezik egy **NetworkMessage** nevű nyilvános interfész, amelynek kiterjesztésével szerializálható hálózati üzenetstruktúrákat hozhatunk létre.

2.3 Többjátékos mód elkészítése

A Mirror megismerése után a többjátékos mód elkészítése következett. Az egyszerűség kedvéért az alkalmazást kétjátékos módra készítettem, de ez nem okoz változást a többjátékos módú logikában, így bármikor kiterjeszthető kettőnél több felhasználós játékká.

A fő kihívás, itt az objektumok szinkronizálása volt. Először is át kellett írjam a meglévő szkriptjeimet, hogy több játékos esetén is rendeltetésszerűen működjön a játék.

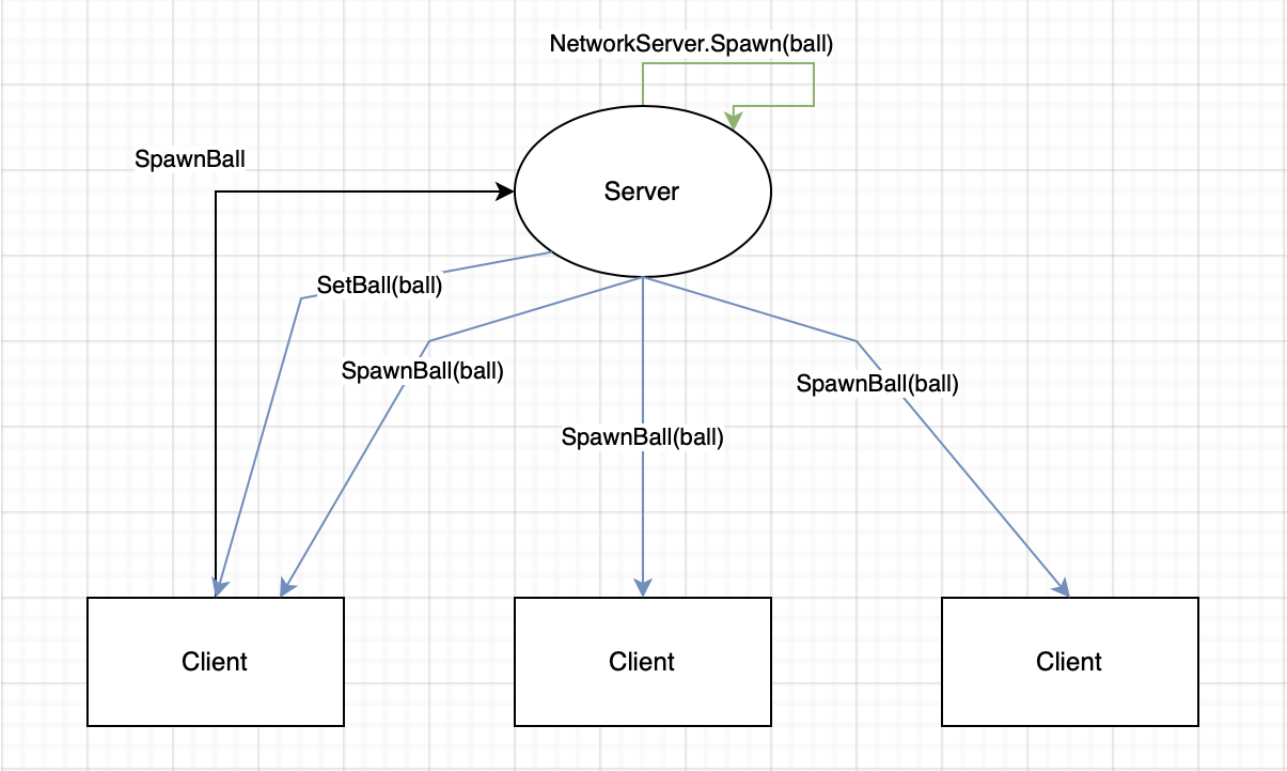
Az első probléma a kosárpalánk volt, mivel ebből csak egyet lehet lehelyezni a játék során, így ha az egyik játékos már rányomott a gombra, a többinek meg kell hogy szűnjön ez a lehetőség. A ***6.*** ábra ábrán látszik az architektúra, ami ezért a működésért felelős. Leolvasható, hogy ha a



6. ábra: SpawnBasket architektúra

szervernek érkezik egy SpawnBasket üzenet, akkor minden kliensnek üzenetet kell küldjön, hogy már nem lehet többé palánkot lehelyezni (a küldőnek is érkezik, hisz többet ő sem helyezhet le).

Hasonlóképpen működik a labda lehelyezése itt, ahogyan az látszik a *7.* ábra ábrán.



7. ábra: SpawnBall architektúra

Ha egy játékos labdát akar leheleyzni, arról egy (már korábban említett) Command attribútummal ellátott művelet segítségével értesíti a szervert, ami pedig lehelyezi (NetworkServer.Spawn() metódus) magánál a labdát, kicsivel a küldő kliens pozíciója elé. Erről ezek után üzenetet küld minden kliensnek, hogy ők is láthassák a labdát. A továbbiakban a Network Transform segítségével a szerver folyamatosan frissítés üzeneteket fog küldeni a labdáról a klienseknek, hogy amennyiben változás történik a labdával, az az ő kijelzőjükön is látszódjon. Mivel az volt a cél, hogy minden kliens csak a saját labdáját dobhassa el, ezért a szerver még visszahív a küldő kliensre egy ClientRPC hívással, amiben átadja a lehelyezett labdát, hogy majd a kliens ezt el tudja dobni.

A labdák és a palánk szinkronizációját követően szükségem volt még egyéb célt szolgáló üzenetek küldésére is, amire a Network Message interfészt tudtam használni. Először is létrehoztam egy nagyon egyszerű, saját típusú üzenetet, amit az alábbi C# kódrészlet mutat.

```
public struct Notification : NetworkMessage
{
    public int value;
}
```

A saját üzenettípusom a Network Message-ből származik, és csupán egy integer típusú változója van. Az ilyen üzenetet, ha a szerveren hívjuk, a NetworkServer SendToAll metódusával lehet kiküldeni minden kliensre, és itt kell meg az üzenet értékét is.

Különböző értékekre különböző működést implementáltam, így elértem, hogy egy típusú üzenettel tudom a klienseken állítani a *SpawnBasket* és *SpawnBall* gombok láthatóságát, illetve a

játékosok pontszámának változását is. Erre a lentebb látható szintén C# kódban mutatok példát.

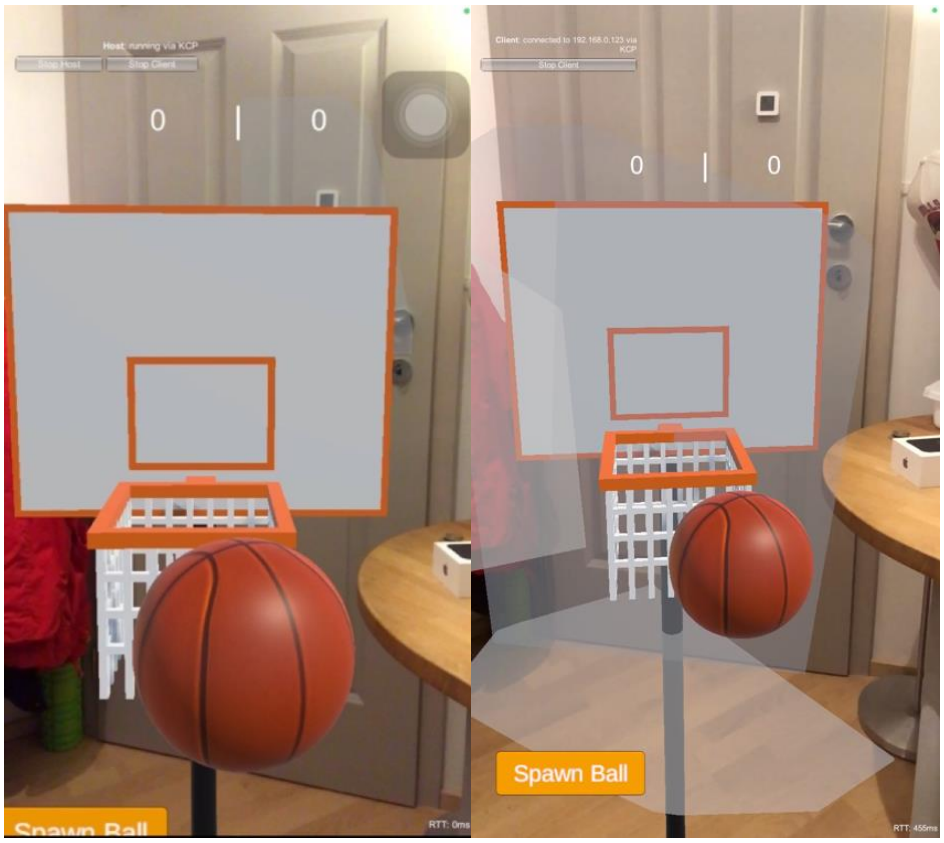
```
private void OnNotification(Notification msg)
{
    if (msg.value == netId)
    {
        score += 1;
        scoreText.text = score.ToString();
    }
    //...
}
```

2.4 Koordinátarendszer szinkronizáció

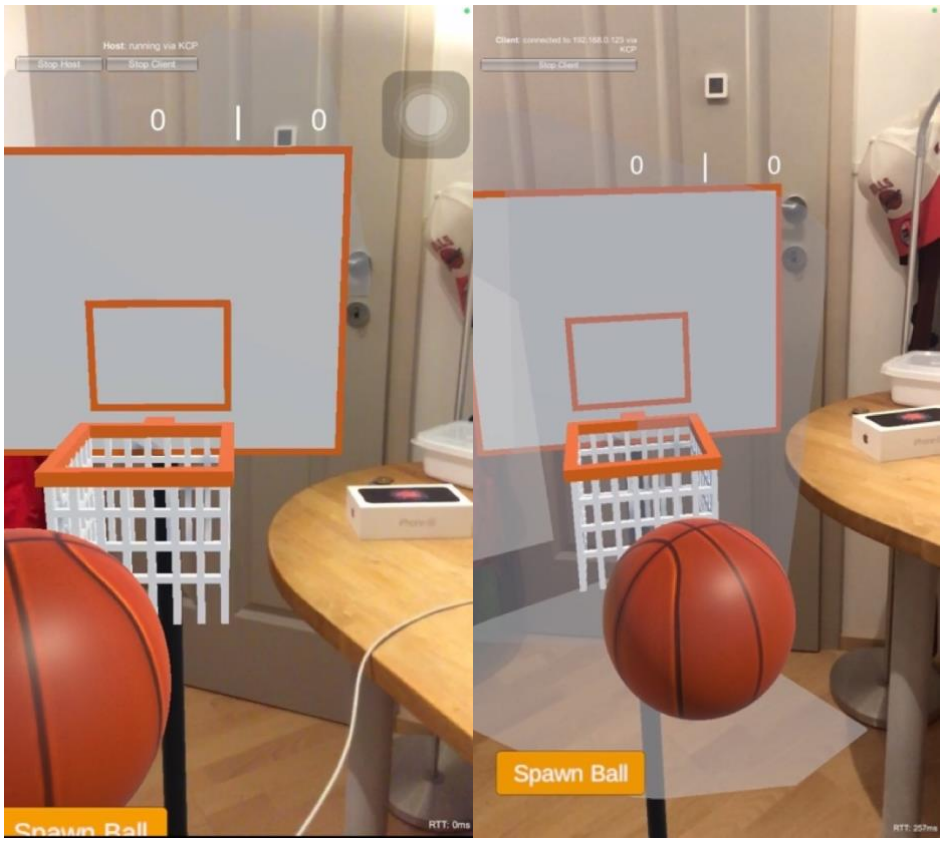
Végezetül már csak egy dolog hiányzott, a játékosok koordinátarendszerének szinkronizációja. Ez azt jelenti, hogy például a játékosok a kosárpalánkot ne egymástól elcsúszva, hanem pontosan azonos helyen lássák a telefonjuk kijelzőjén, ezzel is növelve a játékélményt. Azt nagyon nehéz és kényelmetlen lenne megoldani, hogy a két játékos ugyanazon az adott helyszínen indítsa el a játékot, így más módot kellett találnom erre a problémára.

A konzulenseimmel erre azt a megoldást találtuk, hogy egy QR kód lehelyezésével, és beolvasásával, használva a Unity által biztosított AR Tracked Image Manager-t, a játékosok képesek hozzászinkronizálni a játékbeli koordinátarendszerüket, egy valóságbéli, fizikai ponthoz.

Ezek után el is készült a végleges játék, melybe betekintést mutat a ***8.*** ábra és a 9. ábra ábra. Mindkét képen a baloldali telefon a host (kliens és szerver egyben), a bal oldali pedig egy kliens.



8. ábra: Host labdája



9. ábra: Kliens labdája

2.5 Játék tesztelése

Mivel egy telefonos AR alkalmazást készítettem, ezért a munka elengedhetetlen része volt a tesztelés, próbálgatás. Természetesen a játék fejlesztése során sajnos nem sikerült minden elsőre, így nagyon sokszor csak akkor vettem észre egy-egy kisebb nagyobb hibát, mikor feltelepítettem a telefonokra az alkalmazást és kipróbáltam. Ez az eljárás sajnos nagyon időigényes volt, mivel minden változtatásnál újra kellett buildelnem a Unity- , és az XCode projektet is, valamint a telefonokra is újra kellett telepíteni az alkalmazást, viszont ez elengedhetetlen része a játékfejlesztésnek.

2.2 Összefoglalás

Ebben a félévben a önálló labornak köszönhetően mégjobban elmélyülhettem a Unity, valamint az AR használatában, valamint megismerkedhettem a Mirror és a többjátékos módú játékfejlesztés világával, és a sok szembejövő nehézség ellenére, úgy érzem, hogy sokat tudtam tanulni, valamint, hogy sikerült elérnem a kitűzött célt.

A továbbfejlesztését illetően (a sok, játékélményt javító, funkcionalitás szempontjából azonban elhanyagolható, esztétikai tényezők hozzáadásával most nem számolva) még esélyt látok többféle koordináta szinkronizáció használatára is, mivel az többféle lehetőséggel is találkoztam a félév során. Úgy gondolom, hogy érdekes lenne ezeket mind kipróbálni, és összehasonlítani, hogy melyikkel lehet maximalizálni a felhasználói játékélményt.

Összeségében számomra ez egy nagyon élvezetes projekt volt, és remélem, hogy a félév alatt szerzett tudást majd minél előbb újra hasznosíthatom.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

[I] Alexander Zotov: *How to Add Force To Gameobjects with Swipe to Throw It In 3D Android Unity Game? Simple Tutorial.*
https://www.youtube.com/watch?v=7O9bAFyGvH8

[II] Dinesh Punni: *Unity AR Foundation Tutorial - Tap to Place Objects in AR*
https://www.youtube.com/watch?v=xguiSueY1Lw&t=64s&pp=ygUeYXIgcGxhY2Ugb2JqZWN0IHR1dG9yaWFsIHVuaXR6

[III] Mirror dokumentáció
https://mirror-networking.gitbook.io/docs/
https://github.com/MirrorNetworking/Mirror

[IV] Dapper Dino*: Mirror Multiplayer*
- Tutorials that explain how to implement different features using Mirror.
https://www.youtube.com/playlist?list=PLS6sInD7ThM1aUDj8lZrF4b4lpvejB2uB

[V] Shrine Wars: *Unity + Mirror Networking Tutorials*
- Explains the basics about Networking with Unity and Mirror
https://www.youtube.com/playlist?list=PLXEG2omgKgCapAmGe20XBgd87rmxFdKhK