3 Technológiák

3.1 Samsung Gear VR

A Gear VR a Samsung Electronics és az Oculus VR együttműködésében fejlesztett, és a Samsung által gyártott virtuális valóság szemüveg. A szemüveg magában nem tartalmaz kijelzőt, csak Samsung okostelefonnal együttesen lehet használni.

A Gear VR előnye a mobilitás, és emiatt is választottam a szoftverem platformjának. A virtuális valóság szemüvegek nagyrésze igényel egy középkategóriás asztali számítógépet, lighthouse modulokat, és többségüknek nincs kábelmentes támogatása. A lighthouse modulok feladata nyomon követni a szemüveget a térben, így a szemüveg funkcionalitása erősen függ ezektől a moduloktól. Olyan terekben, ahol nem lehet lighthouse modult felszerelni, vagy nem látja jól a szemüveget, nem tud működni egy olyan szemüveg, ami ilyen technológián alapszik. Fontos, hogy a felhasználó megőrizze a mobilitását a szemüveg viselése közben is, így a kábeles szemüvegeknél akadályt jelenthet a kábel. Egy középkategóriás asztali számítógép pedig nem található meg minden háztartásban. Így a Gear VR előkészületek, és kényelmetlenségek nélkül használható, bárhol bármikor.

A Gear VR szemüveg lehetőséget ad lighthouse modulok nélkül is a virtuális látótér mozgatására 3 tengelyen, le-fel, jobbra-balra fordítás, jobbra-balra döntés, valamint a szemüveg oldalán lévő panel megérintése helyettesíti a kattintást. Így a fej mozgatásával, és a panel megérintésével tudunk navigálni virtuális térben.

A Gear VR rendelkezik kontrollerrel, de mivel nem szükséges a gyakorlatokhoz, ezért a szoftverben nincs szükség kontroller támogatásra.

3.2 Fejlesztői eszközök

3.2.1 Unity Motor

A Unity a Unity Technologies által fejlesztett és 2005-ben kiadott fizikai motor. A motor több mint 25 platformot támogat, többek között Gear VR-t, használható 3 dimenziós játékok, 2 dimenziós játékok, virtuális valóság, kiterjesztett valóság és szimuláció fejlesztésre is. Videójáték fejlesztésen kívül más iparágakban is használják a Unity motort, például filmiparban. A motort felhasználók széles körére fejlesztették, így teljesen kezdőtől, az erre specializálódott fejlesztőkig mindenki tudja használni.

A Unity projekt létrehozásánál felajánl VR támogatást. Ezek után be kell állítani a platformot Androidra, ezt egyszerűen meg lehet tenni a **File > Build Settings** fül alatt, ahol az Android platform kiválasztása után már is platformot válthatunk.

Szükséges még a Gear VR kompatibilitásához, az Oculus Intergation nevű csomag, amit a Unity Asset Store-ból könnyen le lehet tölteni. Az csomag biztosítja az alap scripteket a szemüveg szenzoraihoz és a kontrollerek működéséhez.

3.2.2 Integrált fejlesztői környezet

Egy szoftver fejlesztésénél fontos a megfelelő fejlesztői környezet kiválasztása. Mivel a Unity motor a C# nyelvet támogatja, és nem használtam még a C# nyelvet, szükségem volt egy új fejleszői környezetre. Mivel a JetBrains fejlesztői környezetei közül az egyiket már alaposan ismerem, és több projektet is elkészítettem benne, valamit gyakran használom tanulmányaimhoz, így egyértelműnek tűnt, hogy a JetBrains Rider fejlesztői környezetet válasszam.

A Rider külön plugin-nal biztosítja az integrációt a Unity-vel. A plugin biztosít Unity specifikus kód generálást, debuggolást, szintaktikai hibák jelölését, lehetőséget ad Unity specifikus tesztek készítésére.

A Rider Unity plugin rengeteg funkciója közül a leghasznosabbnak a **Find Usage** funkciót tálátam, ami egy kódrészlet Rider-ben való kijelölése után Unity-ben kiemeli azokat az asseteket és objektumokat, amik használják a kódrészletet.

3.2.3 Verzió kezelés

Bár a tantárgyainkon keveset hallunk a verziókezelésről, én amióta megtanultam használni a Git-et, azóta minden projektemhez használom. Így ehhez a szoftverhez is létre hoztam egy Github repository-t. Git kliensnek a Github asztali verzióját használom, mivel egy könnyen használható, Windows-ra készített szoftver

4 Megvalósítandó feladatok

4.1 Raycasting implementálás

A Raycasting egy láthatatlan sugár kilövése, hogy megkapjuk a sugár útjába eső testeket. Elengedhetetlen az implementálása, ha tudni akarjuk, hogy mire néz a felhasználó, ennek a szoftvernek a kontextusában pedig, elengedhetetlen, ha tudni akarjuk, hogy megfelelő irányba mozgatja a felhasználó a fejét.

A gyakorlat közben a szoftver objektumokat fog mutatni a felhasználónak, amikre a rá kell néznie. A raycasting ebben fog segíteni, hogy a program tudja mikor néz a felhasználó az objektumra. Ebben egy kis fókuszpont fogja segíteni, ami a szemüveg képernyőjének közepén fog elhelyezkedni.

4.2 Véletlenszerű objektum pozíció generálás

Lesz lehetőség nem előre létrehozott gyakorlatra is, amin a szoftver véletlenszerű pozíciókra generálja az objektumokat. Fontos, hogy a generálás egy előre meghatározott téren belül történjen, ha az objektum a felhasználó látóterén vagy mozgásterén kívül, vagy ha nagyon messze lévő pozícióra generálódik, akkor az megszakíthatja a gyakorlat folyamatosságát. Ezért a generáláshoz előre megszabott tér valójában egy görbített téglalap, aminek minden pontja ugyan olyan távolságra van a felhasználótól. Az oldalainak hosszát pedig a mozgékonysági változó fogja megszabni. A téglalapból ki kell még hagyni a jelenlegi objektum méretének megfelelő részt ott, ahol a jelenlegi objektum van, ez biztosítja, hogy a következő objektum mindenképp a tér egy másik részén fog megjelenni.

Így amikor a felhasználó ránéz az objektumra, az objektum koordinátáit megváltoztatjuk a téglalap egyik véletlenszerű koordinátájára.

4.3 Mozgékonyság változó

A mozgékonysági változó fogja biztosítani, hogy az objektumok a felhasználó mozgásterén belül jelennek meg. Ezt a változót úgy kapjuk meg hogy lemérjük a felhasználó maximális le-fel nézését, és jobbra-balra fej fordítását. Ez után a generáláshoz használt téglalap oldalainak távolságát a téglalap középpontjától csökkentjük vagy növeljük (r \* 𝜋 \* 𝛼)/180-ra, ahol r a téglalap és kamera távolsága, 𝛼 pedig a kamera a téglalap középpontjába haladó sugár és a kamerából a felhasználó maximum fordulása felé haladó sugár által bezárt szög.

4.4 Gyakorlat tervező

Lesz lehetőség gyakorlatokat létrehozni, és elmenteni. A gyakorlat készítés közben fixált nagyságú téglalapra lesz lehetőségük elhelyezni az objektumokat sorrendben a felhasználóknak.

A gyakorlat indításakor a téglalapot a mozgékonyság változó alapján eltorzítjuk, minden irányba a megfelelő értékkel arányosan, a torzítás során az objektumok koordinátáit is megváltoztatjuk, hogy a téglalap referencia középpontjához való távolságuk arányosan változzon a torzulással.

4.5 Felhasználói felület

A felhasználóknak a mozgás során nyugodt virtuális környezetben érdemes lenniük, ami nem tereli el a figyelmüket a feladatról, de eltereli a figyelmüket arról, hogy mozognak.