VR JÁTÉKFEJLESZTÉS REHABILITÁCIÓS CÉLRA

KERTÉSZ DOMONKOS

TÉMAVEZETŐ: DR. GUZSVINECZ TIBOR

CÉL

- Nyak tornáztató VR alkalmazás elkészítése
- Hordozható eszközre optimalizálva
- Gyakorlat készítő mód, és gyakorlatok megosztása eszközök között
- Multiplatform (PC/VR)

TECHNOLÓGIÁK

- Samsung GearVR – Könnyű, Android alapú, nem igényel PC-t
- Unity Engine – Multiplatform build opció
- Jet Brains Rider – IntelliSense, Unity integráció
- GitHub

KIHÍVÁSOK

VR platformból adódó kihívások

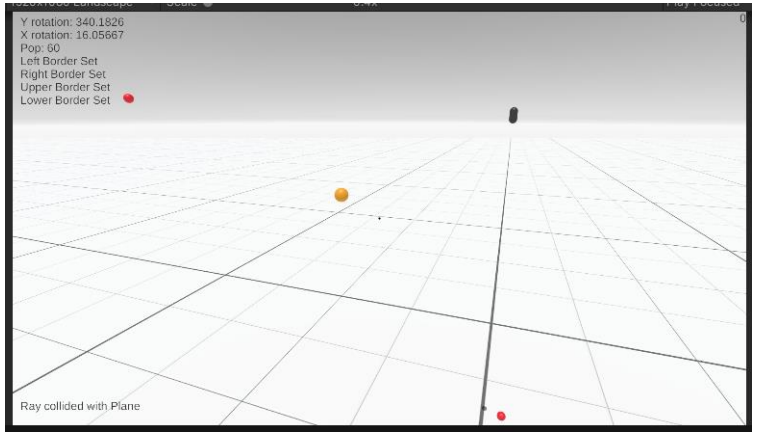
- Nincs kurzor, nem tudunk a UI elemeire kattintani
- Figyelembe kell vennünk a felhasználó fizikai képességeit
- Fontos az optimális futás, mert az akadozások rosszulléthez vezethetnek

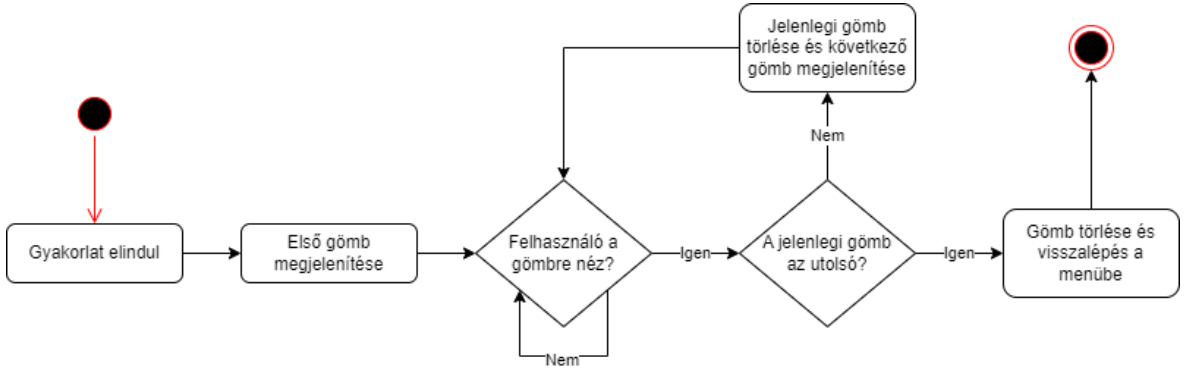
Választott technológiákból adódó kihívások

- Unity nagyon erőforrás igényes
- A választott hardver limitált erőforrásokkal rendelkezik
- Limitált input, nincsenek VR kontrollerek, csak a szemüveg forgásszöge és az érintőpanel a bemenet

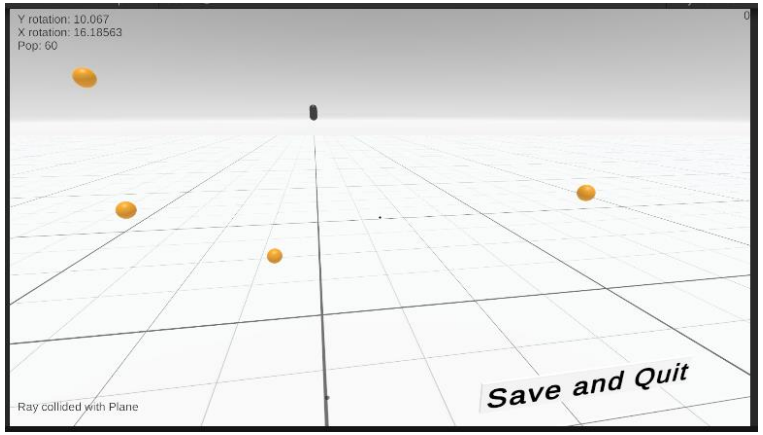
JÁTÉK MENETE

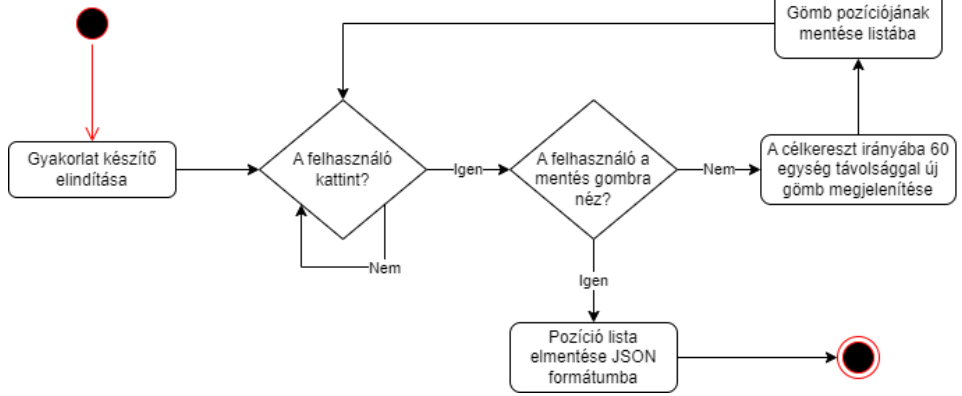
Játék mód



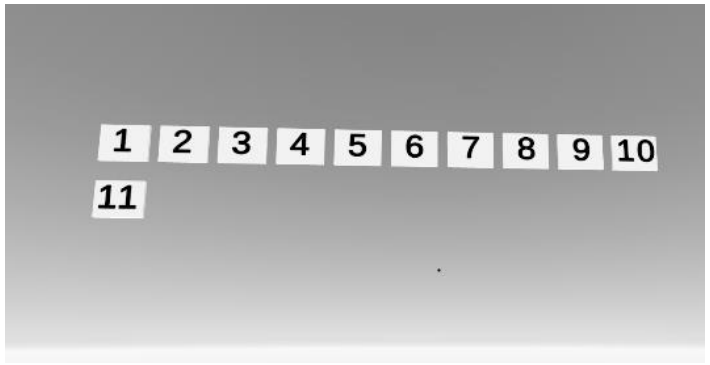


Gyakorlat készítő mód



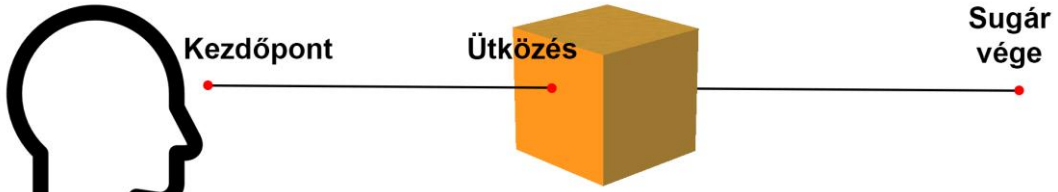


JÁTÉKOBJEKTUMOK KEZELÉSE



- Dinamikusan hozzuk őket létre és töröljük őket, létrehozáshoz tudnunk kell a pozíciót és, hogy mit akarunk létrehozni. Törléshez tudnunk kell mit akarunk törölni.
- A lehető legkevesebb objektum létezik egyszerre
- A menü nem a UI része, térbeli objektumokból épül fel
- Térben lévő objektumok, raycasting segítségével kattintunk

RAYCASTING



- A látótér középpontját kiszámoljunk mint térbeli koordináta, és innen lövünk a látótér irányvektorának segítségével egy sugarat. Meg tuzdjuk vizsgálni milyen objektumokkal ütközik a sugar
- Így kezeli le az alkalmazás a menü elemekre, és térbeli objektumokra való kattintást

SZERIALIZÁLÁS

```
public string SerializeVectorListToJson(List<Vector3> vectorList)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("{\"vectors\":[");

    for (int i = 0; i < vectorList.Count; i++)
    {
        Vector3 v = vectorList[i];
        sb.Append("{\"x\":");
        sb.Append(v.x.ToString(format: "F3"));
        sb.Append(",\"y\":");
        sb.Append(v.y.ToString(format: "F3"));
        sb.Append(",\"z\":");
        sb.Append(v.z.ToString(format: "F3"));
        sb.Append("}");

        if (i < vectorList.Count - 1)
        {
            sb.Append(",");
        }
    }

    sb.Append("]}");

    return sb.ToString();
```

- Futás közben egy List<List<Vector3>> tárolja a tornákat, egy Singleton osztály tárolja a listát és a segéd függvényeket
- Tornák tárolása JSON formátumban történik két futás között
- Saját megoldás szerializálásra és deszerializálásra a job teljesítmény eléréséhez

DESZERIALIZÁLÁS

```
public List<Vector3> DeserializeJsonToVectorList(string json)
{
    int startIndex = json.IndexOf("[{") + 1;
    int endIndex = json.LastIndexOf("}]");

    string[] vectorJsonStrings = json.Substring(startIndex, length: endIndex - startIndex) // string
        .Split( separator: new string[] { "},{" }, StringSplitOptions.RemoveEmptyEntries);

    return (from vectorJson :string in vectorJsonStrings
        select vectorJson.Replace( oldValue: "{",  newValue: "")
            .Replace( oldValue: "}",  newValue: "") // string
            .Split( separator: new char[] { ',' }, StringSplitOptions.RemoveEmptyEntries) // string[]
        into components :string[]
        let x :float  = float.Parse(components[0].Split( separator: ':')[1])
        let y :float  = float.Parse(components[1].Split( separator: ':')[1])
        let z :float  = float.Parse(components[2].Split( separator: ':')[1])
        select new Vector3(x, y, z)).ToList();
}
```

- JSON-ból Vektor Listává alakítás LINQ segítségével

JÁTÉKTÉR KISZÁMÍTÁSA

```
Vector3 positionNormalizedToPlane = new Vector3(positions[i].x, y:60, positions[i].z);
Vector3 playerBaseObjectVector = positionNormalizedToPlane - player;

float r = coords.SpawnDistanceFromPlayer;
float beta = Vector3.Angle(playerOriginVector, playerBaseObjectVector);
float ro = (gamma * (beta / alpha));

ro = 90 - ro;
double roRads = (Math.PI / 180) * ro;

double x = r * Math.Cos(roRads);
double z = r * Math.Sin(roRads);

Vector3 newVector = new Vector3((float)x, positions[i].y, (float)z);
positions[i] = newVector;
```

- Csak olyan mozgást szeretnénk kérni a felhasználótól mit meg is tud csinálni, sérülések elkerülése végett
- Induláskor megkérjük a felhasználót, hogy fordítsa a fejét balra, jobbra, felfelé, és lefelé amennyire csak tudja
- Ezekből a forgásszögekből kiszámoljuk a játékteret, és a megnyitott gyakorlat objektumait dinamikusan átmozgatjuk, hoyg a térben legyenek
- Számolással horizontálisan majd vertikálisan átszámolunk minden pozíciót, úgy hogy az egymástól való távolságuk arányát megtartsák
- Polárkoordináta számolással, két dimenziós koordinátaként számoljuk az új koordinátát (mivel a harmadik koordináta változatlan marad)

JÁTÉKTÉR KISZÁMÍTÁSA II.

- Mivel a számolás nem pontos ezért "szépítés" céljából a újraszámoljuk, hogy a távolságuk a felhasználótól megegyezzen

```
private List<Vector3> NormalizePositions(List<Vector3> positions)
{
    return positions.Select(
        position :Vector3 =>
            transform.position + Vector3.Normalize(position - transform.position) * ObjectCoordinates.Instance.SpawnDistanceFromPlayer)
        .ToList(); // List<Vector3>
}
```

EREDMÉNY

- Android GearVR eszközön használható szoftver
- PC-n használható gyakorlatok készítésére

fin